



AVEVA™ Plant SCADA

2023 R2

© 2015-2025 AVEVA Group Limited and its subsidiaries. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, photocopying, recording, or otherwise, without the prior written permission of AVEVA Group Limited. No liability is assumed with respect to the use of the information contained herein.

Although precaution has been taken in the preparation of this documentation, AVEVA assumes no responsibility for errors or omissions. The information in this documentation is subject to change without notice and does not represent a commitment on the part of AVEVA. The software described in this documentation is furnished under a license agreement. This software may be used or copied only in accordance with the terms of such license agreement. AVEVA, the AVEVA logo and logotype, OSIsoft, the OSIsoft logo and logotype, ArchestrA, Avantis, Citect, DYNSIM, eDNA, EYESIM, InBatch, InduSoft, InStep, IntelTrac, InTouch, Managed PI, OASyS, OSIsoft Advanced Services, OSIsoft Cloud Services, OSIsoft Connected Services, OSIsoft EDS, PIPEPHASE, PI ACE, PI Advanced Computing Engine, PI AF SDK, PI API, PI Asset Framework, PI Audit Viewer, PI Builder, PI Cloud Connect, PI Connectors, PI Data Archive, PI DataLink, PI DataLink Server, PI Developers Club, PI Integrator for Business Analytics, PI Interfaces, PI JDBC Driver, PI Manual Logger, PI Notifications, PI ODBC Driver, PI OLEDB Enterprise, PI OLEDB Provider, PI OPC DA Server, PI OPC HDA Server, PI ProcessBook, PI SDK, PI Server, PI Square, PI System, PI System Access, PI Vision, PI Visualization Suite, PI Web API, PI WebParts, PI Web Services, PRISM, PRO/II, PROVISION, ROMeo, RLINK, RtReports, SIM4ME, SimCentral, SimSci, Skelta, SmartGlance, Spiral Software, WindowMaker, WindowViewer, and Wonderware are trademarks of AVEVA and/or its subsidiaries. All other brands may be trademarks of their respective owners.

#### U.S. GOVERNMENT RIGHTS

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the license agreement with AVEVA Group Limited or its subsidiaries and as provided in DFARS 227.7202, DFARS 252.227-7013, FAR 12-212, FAR 52.227-19, or their successors, as applicable.

AVEVA Legal Resources: <https://www.aveva.com/en/legal/>

AVEVA Third Party Software Notices and Licenses: <https://www.aveva.com/en/legal/third-party-software-license/>

# Contents

<b>Welcome to AVEVA Plant SCADA .....</b>	<b>129</b>
<b>Legal information .....</b>	<b>129</b>
<b>Contact information .....</b>	<b>130</b>
<b>Safety Information .....</b>	<b>130</b>
<b>Security Guidelines for Industrial Control Systems .....</b>	<b>132</b>
General considerations for security .....	132
Introduction .....	133
Secure the host .....	134
General guidelines for securing the host .....	134
Windows updates .....	134
ICS software updates .....	135
Scanning the Host .....	135
Protect the applications and content on the host .....	135
Configure encryption in SQL server .....	137
Secure the network .....	139
Segment the ICS network .....	139
Manage network services and ports .....	140
Secure communication between the client and server .....	141
Cloud-based systems .....	142
Secure systems through authentication and authorization .....	142
Manage users and groups through windows .....	143
Manage users and groups through ICS software .....	144
Plan contingency .....	144
Audit and log .....	145
Plan business continuity .....	145
Plan disaster recovery .....	145
Conclusion .....	146
Security Configuration for Plant SCADA .....	146
<b>Information and Support .....</b>	<b>147</b>
<b>AVEVA™ Help .....</b>	<b>148</b>
<b>AVEVA™ Knowledge &amp; Support Center .....</b>	<b>149</b>
 <b>What's New in Plant SCADA .....</b>	 <b>153</b>
<b>Support for Plant SCADA on AVEVA Integration Studio .....</b>	<b>153</b>
<b>Double Point Status Alarms .....</b>	<b>153</b>
<b>Kernel Access Improvements .....</b>	<b>154</b>
<b>New Installation Framework .....</b>	<b>155</b>
<b>AVEVA™ Help .....</b>	<b>155</b>
<b>Setup Wizard supports HMI Computer Role .....</b>	<b>156</b>
<b>Unique IDs for Equipment .....</b>	<b>156</b>
<b>Equipment Update Improvements .....</b>	<b>156</b>
<b>Project DBF AddIn supports 64-bit Excel .....</b>	<b>156</b>

<b>Citect.ini Parameters in Plant SCADA 2023 R2 .....</b>	<b>157</b>
<b>Cicode Functions in Plant SCADA 2023 R2 .....</b>	<b>158</b>
<b>What's New - Previous Releases .....</b>	<b>159</b>
What's New in Plant SCADA 2023 .....	159
AVEVA™ Enterprise Licensing .....	160
Configuration Parameters Database for Equipment Types .....	160
Add Web Content to a Graphics Page .....	161
Tag Writes Enabled for Industrial Graphics Applications .....	161
Tag Writes Enabled for OPC UA Client Applications .....	161
Alarm Filtering Performance Improvements .....	162
Unique IDs for Variables .....	162
Unique IDs for Projects .....	162
Debug Cicode without Compiling .....	162
Citect.ini Parameters in Plant SCADA 2023 .....	163
Cicode Functions in Plant SCADA 2023 .....	165
What's New in Plant SCADA 2020 R2 .....	167
Rebranding to Plant SCADA .....	167
Integration of AVEVA™ Industrial Graphics .....	168
Integrated OPC UA Server Functionality .....	168
Graphics Usage .....	168
Graphics Editing Enhancements .....	168
IPv6 Support .....	168
Driver Classifications .....	169
Citect.ini Parameters in Plant SCADA 2020 R2 .....	169
What's New in Citect SCADA 2018 R2 .....	170
Graphics Browsing .....	170
Size and Position Objects .....	171
Encrypted Connections .....	171
Publish Plant SCADA Data to the Cloud .....	171
Situational Awareness Improvements .....	171
Citect.ini Parameters in Citect SCADA 2018 R2 .....	172
Cicode Functions in Citect SCADA 2018 R2 .....	173
What's New in Citect SCADA 2018 .....	174
Situational Awareness Starter Project .....	175
Additional Example Project .....	175
Alarm Management .....	176
Graphics Improvements .....	177
Screen Profiles .....	179
Equipment References .....	179
Composite Genies .....	179
Refresh Super Genie Associations .....	180
Grid Column Resizing in Plant SCADA Studio .....	180
Display Name for Equipment .....	180
Menu Configuration Enhancements .....	180
Array Functions .....	181
Indirect Tag Value Lookups in Foreground Code .....	181
Citect.ini Parameters in Citect SCADA 2018 .....	181
Cicode Functions in Citect SCADA 2018 .....	183
What's New in Citect SCADA 2016 .....	190

Plant SCADA Studio .....	190
Topology .....	192
Deployment .....	193
Calculated Variables .....	194
Schedule Integration .....	194
Citect.ini Parameters in Citect SCADA 2016 .....	194
Cicode Functions in Citect SCADA 2016 .....	196
What's New in Citect SCADA 2015 SP1 .....	196
Cicode Functions in Citect SCADA 2015 SP1 .....	197
What's New in Citect SCADA 2015 .....	198
Citect.ini Parameters in Citect SCADA 2015 .....	199
Cicode Functions in Citect SCADA 2015 .....	201
What's New in Citect SCADA 7.40 SP1 .....	203
Citect.ini Parameters in 7.40 SP1 .....	203
Cicode Functions in 7.40 SP1 .....	204
What's New in Citect SCADA 7.40 .....	205
Citect.ini Parameters in 7.40 .....	205
Cicode Functions in 7.40 .....	206
CtAPI Functions in 7.40 .....	209
What's New in Citect SCADA 7.30 .....	210
Citect.ini Parameters in 7.30 .....	210
Cicode Functions in 7.30 .....	214
CtAPI Functions in 7.30 .....	225
What's New in Citect SCADA 7.20 .....	226
Citect.ini Parameters in 7.20 .....	227
Cicode Functions in 7.20 .....	234
CtAPI Functions in 7.20 .....	242
What's New in Citect SCADA 7.10 .....	243
Citect.ini Parameters in 7.10 .....	243
Cicode Functions in 7.10 .....	245
What's New in Citect SCADA 7.0 .....	247
Citect.ini Parameters in version 7.0 .....	247
Cicode Functions in 7.0 .....	257
CtAPI Functions in 7.0 .....	266
Kernel Commands in Version 7.0 .....	267
<b>About Plant SCADA .....</b>	<b>268</b>
<b>Working with Plant SCADA .....</b>	<b>268</b>
Configure Plant SCADA Projects .....	268
Define a Topology .....	269
Build a System Model .....	270
Create Graphics Pages .....	271
Secure Your System .....	272
Compile and Run a Project .....	273
Deploy a Project .....	273
<b>Typical System Scenarios .....</b>	<b>274</b>
Standalone System .....	276
Distributed I/O System .....	276

Client-Server System .....	277
Redundant Server System .....	278
Clustered Control System .....	279
Redundant and Distributed Control System .....	280
Load Sharing System .....	281
<b>Licensing .....</b>	<b>282</b>
AVEVA™ Enterprise Licensing .....	284
AVEVA™ Enterprise Licensing Topologies .....	285
AVEVA™ Enterprise Licensing Workflow .....	288
Allocating AVEVA™ Enterprise Licenses .....	288
Launch Enterprise License Manager from Plant SCADA Studio .....	289
Update a Sentinel Key with CiUSAFFE .....	290
Dynamic Point Count .....	292
Specify the Required Point Count for a Computer .....	293
Demo Mode .....	293
<b>Install Plant SCADA .....</b>	<b>294</b>
Installation Information .....	295
What Should I Install? .....	300
Configurator .....	302
View the Configurator Messages List .....	304
Install a Development Workstation .....	304
Development Workstation Installation Requirements .....	306
Development Workstation Installation Procedure .....	307
Use Configurator to Set Up a Development Workstation .....	309
Set Up a System Management Server .....	311
Configure a System Management Server .....	312
Connect a Computer to a System Management Server .....	314
Install a Runtime Server .....	316
Runtime Server Installation Requirements .....	317
Runtime Server Installation Procedure .....	318
Use Configurator to Set Up a Runtime Server .....	320
Install a Deployment Server .....	323
Deployment Server Installation Requirements .....	325
Deployment Server Installation Procedure .....	326
Install an Industrial Graphics Server .....	328
Industrial Graphics Server Functionality Limitations .....	329
Industrial Graphics Server Installation Requirements .....	330
Industrial Graphics Server Installation Procedure .....	331
Use Configurator to Set Up an Industrial Graphics Server .....	333
Confirm the Settings for an Industrial Graphics Server .....	334
Install an OPC UA Server .....	335
OPC UA Server Installation Requirements .....	337
OPC UA Server Installation Procedure .....	338
Use Configurator to Set Up an OPC UA Server .....	340
Configure Client Certificates for an OPC UA Server .....	341
Confirm the Settings for an OPC UA Server .....	344
OPC UA Client Driver .....	345
Install a Runtime-only Client .....	346
Runtime Only Client Installation Requirements .....	346

Runtime Only Client Installation Procedure .....	348
Use Configurator to Set Up a Runtime-only Client .....	349
AVEVA™ Enterprise Licensing .....	351
AVEVA™ Enterprise Licensing Topologies .....	352
AVEVA™ Enterprise Licensing Workflow .....	355
Enterprise License Server Installation Requirements .....	355
Enterprise License Server Installation Procedure .....	356
Use Configurator to Set Up an Enterprise License Server .....	358
Install OPC Factory Server .....	359
Installation Troubleshooting .....	360
Unattended Silent Installation .....	361
Customize a Response File .....	363
Add Configurator Settings to a Response File .....	366
Uninstall or Change Components .....	371
Manage Updates .....	372
Silently Install an Update .....	375
<b>Upgrade Plant SCADA .....</b>	<b>376</b>
Upgrade Method .....	377
Upgrade Path .....	377
Offline Upgrade .....	378
Migrate to Production .....	383
Troubleshooting an Offline Upgrade .....	385
Online Upgrade .....	385
Pre-requisites for Online Upgrade .....	386
Upgrading from v2020 R2 .....	387
Upgrading from v2023 .....	388
Troubleshooting an Online Upgrade .....	389
Upgrade Deployment from Plant SCADA 2020 R2 .....	390
Project Migration Tool .....	392
Using the Project Migration Tool .....	393
Remove Obsolete Memory and Alarm Devices .....	395
Memory Devices .....	396
Alarm Devices .....	396
Converting Memory Variables .....	396
Inserting New Local Variables .....	397
Deleting Variable Tags .....	397
Deleting Obsolete I/O Devices .....	398
Creation of Roles for Existing Users .....	398
Migrate Included Projects .....	398
Default Scale .....	399
<b>Plant SCADA Studio .....</b>	<b>400</b>
<b>About Plant SCADA Studio .....</b>	<b>400</b>
Plant SCADA Activities .....	401
Activity Bar .....	402
Activity Menu .....	404
Command Bar .....	405
Compile Messages Area .....	407

The System Menu .....	408
Plant SCADA Studio Options .....	408
The Grid Editor .....	411
Change the Default Sorting .....	413
Customize Grid Editor Columns .....	414
Filtering .....	416
Add Rows to the Grid .....	418
Edit Values in the Grid Editor .....	418
Delete Rows from the Grid .....	419
Cut/Copy and Paste Rows and Columns .....	419
Save Changes .....	420
Cascade Changes .....	421
Export to Microsoft™ Excel .....	423
Import from Microsoft Excel .....	423
Property Grid .....	424
Edit Values in the Property Grid .....	426
Edit Multiple Values .....	426
Search for a Property .....	427
Using Find and Replace .....	428
The Find and Replace Window .....	429
Specify Search Coverage .....	430
Use the Results List .....	431
Remove Results .....	432
Export Results .....	432
Jumping to a Result (Go To) .....	433
Replace Results .....	433
Find and Replace Error Messages .....	435
Troubleshooting Searches .....	437
Projects .....	437
Project Types .....	440
Create a Project Using a Starter Project .....	445
Create a Standard Project .....	447
Make Active .....	450
Add an Included Project .....	450
Back Up a Project .....	451
Run a Backup from the Command Line .....	454
Restore a Project .....	456
Link to an Existing Project .....	460
Link to Project Hierarchy .....	461
Pack a Project .....	462
Repair a Project's Unique IDs .....	462
Copy a Project .....	463
View Projects .....	465
Delete a Project .....	467
Filter Projects .....	467
Edit a Project's Properties .....	468
View a Project's Folders .....	471
Analyze a Project .....	472
Project Specifications .....	472

Run a Project Using the Setup Wizard .....	474
Run the Setup Wizard .....	474
Project Configuration .....	475
Runtime-only Environment .....	476
Profile Setup .....	477
Screen Setup .....	477
Computer Role Configuration .....	477
Network Model .....	480
Reports Configuration .....	482
Trends Configuration .....	482
Alarm Server Configuration .....	483
CPU Configuration .....	483
Events Configuration .....	484
Startup Functions Configuration .....	485
Cluster Connections Configuration .....	486
Server Password Configuration .....	487
Server User Configuration .....	487
Control Menu Security Configuration .....	488
Keyboard Security Configuration .....	489
Miscellaneous Security Configuration .....	489
General Options Setup .....	489
Finish .....	491
<b>Topology .....</b>	<b>491</b>
Computers .....	492
Add a Computer .....	493
View Computers .....	494
Add Network Addresses .....	495
Clusters .....	497
Add a Cluster .....	499
View Clusters .....	499
Assigning Graphics Page Objects to a Cluster at Runtime .....	500
Server Processes .....	501
Add an Alarm Server Process .....	503
Add a Report Server Process .....	506
Add a Trend Server Process .....	507
Add an I/O Server Process .....	508
I/O Devices .....	510
Configure I/O Device Communications .....	513
Add a Board .....	514
Add a Port .....	516
Add an I/O Device .....	517
Using a COM Port .....	528
COMx Driver Special Options Reference .....	528
Using a Serial Board .....	530
Using Ethernet .....	532
TCP/IP Driver Special Options Reference .....	533
Using a Proprietary Board .....	534
Working With Device Drivers .....	535
Determining Which Driver to Use .....	535

Installing a Driver Pack.....	536
The Driver Reference Help .....	537
Using the Device Communications Wizard .....	537
Device Communications Wizard - Introduction.....	538
Device Communications Wizard - Server Selection.....	538
Device Communications Wizard - Device Selection .....	538
Device Communications Wizard - I/O Device Type .....	539
Device Communications Wizard - I/O Device Communications Selection .....	539
Device Communications Wizard - TCP/IP Address .....	539
Device Communications Wizard - I/O Device Address .....	540
Device Communications Wizard - I/O Device Connection Schedule .....	540
Device Communications Wizard - Caller ID and Commands.....	541
Device Communications Wizard - Link to External Database .....	542
Device Communications Wizard - Serial Device .....	544
Device Communications Wizard - Summary .....	544
Retrieving Time-stamped Data from I/O Devices .....	544
Configure a Disk I/O Device .....	545
Disk I/O Device Setup .....	546
Redundant Disk I/O Devices .....	548
Disk I/O Device Error Messages .....	550
Using Memory Mode .....	550
Using Persisted I/O Memory Mode .....	551
Link an I/O Device to an External Data Source .....	552
Refresh the Tags for a Linked I/O Device .....	552
Breaking the Link to the External Data Source .....	553
Import Variable Tags from an External Data Source .....	553
Unity Link Tag Import .....	557
Additional Tag Generation Configuration .....	558
OPC Data Access Server Tag Browser .....	558
OPC Factory Server (OFS) Tag Import .....	558
OPC Factory Server (OFS) Tag Import Limitations .....	559
TagGen XML Template .....	562
Referencing Variables .....	564
Referencing Arrays .....	564
XML Template Tags .....	564
Pattern Matching .....	567
Built-In Functions .....	568
Sample XML Templates .....	568
Export Variable Tags to an External Data Source .....	571
External Data Source .....	572
Format File .....	573
Format File Layout .....	574
[General] Section .....	575
[Columns] Section .....	575
[ImportFilterMap] and [ExportFilterMap] Sections .....	576
Concatenation .....	577
Field Conversion .....	577
Recognizing Format Files .....	582
Communicating with Remote Devices via Modems .....	583

Modems at the I/O Server .....	584
Modems at the I/O Device .....	585
Configure a Modem .....	585
I/O Device Constraints for Multi-dropping .....	587
Configuring Multidrop Remote I/O Devices .....	588
Example Configurations for Modems at the I/O Server .....	589
I/O Server Redundancy for Dial-up Remote I/O Devices .....	593
Troubleshooting Dial-up Remote I/O Device Communications .....	594
Alternative (backward compatibility) Method of Persistent Connection .....	595
Scheduled Communications .....	595
Specifying a Schedule .....	595
Writing to the Scheduled I/O Device .....	596
Reading from the scheduled I/O Device .....	597
Keeping Data Up-to-Date during Prolonged connections .....	597
Avoiding Unnecessary Multiple Reads of I/O Device Data .....	598
Communication Performance Considerations .....	598
Caching Data .....	599
Grouping Registers .....	599
Remapping Variables in an I/O Device .....	601
Advanced Driver Information .....	604
Variable (Digital) Limitations .....	604
Validating Distributed Project Data for Tag-based Drivers .....	605
Write Delay Effects .....	605
Troubleshooting Device Communications .....	606
Gathering Communications Information .....	607
Debugging a COMx Driver .....	608
Debugging a TCP/IP Driver .....	609
Debugging a Protocol Driver Using Serial Communications .....	611
Debugging Proprietary Board Drivers .....	612
File Formats .....	612
Serial Port Loop-Back Test .....	614
Test Setup .....	614
Serial Port Loop-back Cable .....	615
Error Messages .....	616
Protocol Generic Errors .....	616
Protocol-Specific Errors .....	629
Standard Driver Errors .....	633
Profiles .....	636
Add a Profile .....	637
Profile Wizard .....	637
Run the Profile Wizard .....	637
Profile Setup .....	638
Add a New Profile .....	638
Select a Profile .....	639
Select a Node .....	639
Select a Screen Profile .....	639
Configure the Computer Role .....	640
Configure the Network Model .....	642
Configure Reports .....	644

Configure Trends .....	645
Configure Alarms .....	645
Configure CPU .....	646
Configure Events .....	647
Configure Startup Functions .....	647
Configure Cluster Connections .....	648
Configure the Server Password .....	649
Configure Server User .....	650
Configure Control Menu Security .....	650
Configure Keyboard Security .....	651
Configure Miscellaneous Security .....	651
Configure General Options .....	652
Finish .....	653
Redundancy .....	653
I/O Server Redundancy .....	654
I/O Device Promotion .....	655
Redundancy and Persistence .....	656
Data Path Redundancy .....	658
Multiple Device Redundancy (Standby Data Paths) .....	659
Network Redundancy .....	661
Configuring Network Redundancy .....	662
Alarms Server Redundancy .....	663
Alarm Server Side Synchronization .....	664
Reports and Trends Server Redundancy .....	666
Reports Server Redundancy .....	667
Trends Server Redundancy .....	668
File Server Redundancy .....	668
Redundancy of Standalone Systems .....	669
<b>System Model .....</b>	<b>669</b>
Equipment .....	669
Equipment Hierarchy .....	670
Equipment Types .....	672
Items .....	673
Equipment Instances .....	674
Equipment States .....	674
Equipment References .....	675
Equipment Editor .....	675
Equipment Updates .....	676
Equipment Property Referencing .....	677
Configuration Parameters .....	679
Configure Equipment Types Using Equipment Editor .....	682
Add an Equipment Type .....	683
Open an Equipment Type .....	684
Duplicate an Equipment Type .....	684
Copy an Equipment Type to a User-created Project .....	685
Move an Equipment Type to an Included Project .....	686
Enable Editing for an Equipment Type in a System Project .....	687
Edit Equipment Types .....	688
Delete an Equipment Type .....	689

Add a State to an Equipment Type .....	690
Add an Item to an Equipment Type .....	691
Add an Element to an Item .....	692
Add an Accumulator to an Equipment Type .....	694
Edit the Fields for an Element or Accumulator .....	695
Add Configuration Parameters to an Equipment Type .....	697
Link a Genie to an Equipment Type .....	699
Update Equipment in Equipment Editor .....	702
Configure Equipment Instances Using Equipment Editor .....	702
Add an Equipment Instance .....	703
Open an Equipment Instance .....	705
Add an Equipment Instance to a New Hierarchy Level .....	705
Delete an Equipment Instance .....	707
Duplicate an Equipment Instance .....	707
Edit the Field Values for an Equipment Instance .....	707
Paste a Linked Genie on a Graphics Page .....	708
Configure Equipment in Plant SCADA Studio .....	710
Define Equipment in Plant SCADA Studio .....	711
Define Equipment Types in Plant SCADA Studio .....	716
Define Equipment States in Plant SCADA Studio .....	717
Define Equipment Configuration Parameters in Plant SCADA Studio .....	720
Define Equipment References .....	721
Define Equipment Runtime Parameters .....	723
Define an Equipment Association for a Tag .....	725
Update Equipment in Plant SCADA Studio .....	726
Use the Project DBF Add-In to Define Equipment Associations .....	727
Import Equipment Using XML Templates .....	727
Equipment XML Template .....	728
Equipment Templates: Header .....	729
Equipment Templates: Parameters .....	729
Equipment Templates: Input .....	730
Equipment Templates: Outputs .....	731
Equipment Templates: Output Node .....	732
Equipment Templates: Output Field .....	732
Equipment Templates: Output Common Fields .....	733
Equipment Templates: Output Database .....	734
Equipment Templates: Footer .....	735
Calculator Examples .....	736
Troubleshooting - Equipment XML Templates .....	741
Migrate Custom Parameters to the Configuration Parameters Database .....	741
Variable Tags .....	742
Tag Names .....	743
Tag Name Syntax .....	744
Structured Tag Names .....	744
Reserved Words .....	745
Tag Data Types .....	747
Tag Elements .....	749
The Quality Item .....	750
Tag Extensions .....	750

Arrays .....	752
Local Variables .....	754
Calculated Variables .....	754
Tag Data Persistence .....	757
XML DataSource Schema .....	758
Configure Variable Tags .....	760
Add a Variable Tag .....	760
Unique IDs for Variable Tags and Equipment .....	767
Format Specifiers .....	768
Add a Calculated Variable .....	770
Add a Local Variable .....	772
Refer to Variable Tags .....	774
Refer to Array Elements .....	774
Refer to Tag Extensions .....	775
Read a Tag Value .....	775
Write to a Tag Value .....	777
Set a Tag to Control Inhibit Mode .....	778
Set a Tag to Override Mode .....	779
Determine Tag Status .....	783
Use an 'Equipment.Item' Reference .....	783
Configure Tag Browsing .....	784
The Browsing Session .....	785
Open a Browsing Session .....	786
Browsing Filter .....	786
Specify Browsing Field .....	786
Specify a Sort Order .....	787
Browsing Clusters .....	787
Link Tags to an External Data Source .....	788
Unity Support Matrixes .....	789
Imported Tags .....	789
Exported Tags .....	790
Linking Tags .....	792
Alarms .....	793
Defined Alarm Types .....	795
Hardware Alarms .....	799
Alarm Server Process .....	800
Viewing Alarms .....	801
Alarm Pages .....	801
Alarm Banner .....	805
Alarm Equipment Tree .....	806
Events .....	808
Manage Alarms .....	809
Prioritize Alarms .....	809
Categorize Alarms .....	810
Provide Cause and Response Information to Operators .....	811
Use Alarm Indicators .....	812
Alarm States .....	813
Create a Custom Flag for an Alarm Indicator .....	814
Set the Alarm Scan Rate .....	817

Configure Alarms .....	817
Use Equipment Editor to Configure Alarms .....	817
Use Plant SCADA Studio to Configure Alarms .....	819
Name an Alarm Tag .....	820
Use an Alarm Name to Describe an Alarm Tag .....	821
Assign an Alarm to an Alarm Category .....	821
Assign an Alarm to a Cluster .....	822
Associate an Alarm with Equipment .....	823
Add a Digital Alarm .....	824
Add a Time Stamped Alarm .....	828
Add an Analog Alarm .....	834
Add an Advanced Alarm .....	841
Add a Multi-Digital Alarm .....	846
Add a Time Stamped Digital Alarm .....	852
Add a Time Stamped Analog Alarm .....	856
Add a Double Point Status Alarm .....	864
Add an Alarm Category .....	870
Use Alarm Delay .....	878
Customize Alarm Pages .....	879
Confirm that Alarms are Updated for All Clusters .....	880
Format an Alarm Display .....	881
Format Alarm Pages in a Starter Project .....	881
Format Alarms in a Standard Project .....	882
Alarm Format Fields .....	883
Alarm Display Fields .....	883
Alarm SOE Fields .....	888
Alarm Summary Fields .....	894
Create Priority and State Symbols for an Alarms List .....	896
Add Variable Data to Alarm Messages .....	898
Alarms List Default Sort Order .....	899
Action Alarms at Runtime .....	900
Acknowledge Alarms .....	900
Disable Alarms .....	904
Enable a Disabled Alarm .....	907
Shelve Alarms .....	908
Display Cause and Response Information .....	913
Filter Alarms .....	914
Sort Alarms .....	917
Add a Column to an Alarms List .....	918
Save and Restore an Alarms List View .....	919
Print an Alarms List .....	920
Export an Alarms List .....	921
Add a Comment to the SOE Page .....	922
Add an Event to the SOE Page .....	923
Archive the Alarms History .....	923
Configure the Archiving Parameters .....	924
Specify a Destination for Archived Events .....	925
Initiate Archiving with a Cicode Function .....	926
Configure Automatic Archiving for an Alarm Server .....	926

Using Scheduler to Trigger Archiving .....	927
View Archived Event Data .....	928
Using Custom Alarm Filters .....	928
Implementing Queries that Use Custom Alarm Filters .....	929
Named Filters .....	930
Implementing Alarm Filters Using Cicode .....	932
Converting Legacy AlarmSetQuery() Functions .....	934
Using Alarm Properties as Tags .....	935
Supported Alarm Properties .....	937
Writing to Alarm Properties .....	942
Setting Up Alarm Properties .....	944
Alarm Server Database Specification .....	944
CDBAlarmSummary .....	945
CDBEventJournal .....	949
CiAdvancedAlarm .....	951
CiAlarmObject .....	963
CiAnalogAlarm .....	971
CiDigitalAlarm .....	983
CiMultiStateDigAlarm .....	995
CiTimestampedAlarm .....	1007
Trends .....	1020
View Trends .....	1020
Configure Trends .....	1021
Add a Trend Tag .....	1022
Configure Event Trends for a DRI Driver .....	1032
Create Pages for Trend Graphs .....	1033
Trend Interpolation .....	1033
Print Trend Data .....	1034
Export Trend Data .....	1035
Use Trend History Files .....	1035
Storage Method .....	1037
Calculating Disk Storage .....	1037
Reconfiguring History Files .....	1038
Use a Path Substitution .....	1039
Debugging Trending .....	1040
Trends - Frequently Asked Questions .....	1040
Accumulators .....	1041
Add an Accumulator .....	1042
Statistical Process Control (SPC) .....	1044
Process Variation .....	1045
Statistical Control .....	1046
Process Capability .....	1046
XRS Control Charts .....	1047
Capability Charts .....	1048
Pareto Charts .....	1049
Using Statistical Process Control (SPC) .....	1049
Add an SPC Tag .....	1050
Configure an SPC Chart .....	1058
SPC Control Charts .....	1059

Configure an XRS Control Chart .....	1059
Configure a Capability Chart .....	1059
Configure a Pareto Chart .....	1060
SPC Alarms .....	1060
SPC Formulas and Constants .....	1062
Control Chart Line Constants .....	1065
<b>Visualization .....</b>	<b>1067</b>
Menu Configuration .....	1068
Configure a Standard Plant SCADA Menu .....	1068
Configure a Menu for Industrial Graphics Pages .....	1079
Pages .....	1082
Browse Pages in Plant SCADA Studio .....	1084
View and Edit Page Properties in Plant SCADA Studio .....	1090
Browse Industrial Graphics Pages in Plant SCADA Studio .....	1094
Add an Industrial Graphics Page in Plant SCADA Studio .....	1099
Libraries .....	1101
Browse Libraries in Plant SCADA Studio .....	1102
Browse Industrial Graphics Libraries in Plant SCADA Studio .....	1113
Add an Industrial Graphics Library in Plant SCADA Studio .....	1121
Add an Industrial Graphics Symbol in Plant SCADA Studio .....	1123
Embed an Industrial Graphics Symbol .....	1124
Content Types .....	1128
Configure Content Types .....	1129
Keyboard Commands .....	1130
Define System Keyboard Commands .....	1130
Define Key Sequences for Commands .....	1131
Keyboards .....	1132
Using a Hot Key .....	1133
Using Variable Data Input .....	1133
Passing Multiple Arguments .....	1135
Passing Keyboard Arguments to Functions .....	1136
Reports .....	1136
Configure a Report .....	1137
Format a Report .....	1140
Run a Report .....	1142
Handling Communication Errors in Reports .....	1144
<b>Security .....</b>	<b>1145</b>
Encrypted Communications .....	1145
Configure a System Management Server .....	1147
Connect a Computer to a System Management Server .....	1149
Enable Encryption .....	1151
Configure a Runtime Computer for Encryption .....	1153
Advanced Configuration for a System Management Server .....	1154
Use SMS Certificates with Web Applications .....	1155
Use Externally Provided Certificates for Encryption .....	1157
Use Externally Provided Certificates with Web Applications .....	1160
Personal Certificates Requirements .....	1161
Running Products without a System Management Server .....	1162
Troubleshooting Certificate Error Messages .....	1162

Encrypt Plant SCADA Folders using SMB3 .....	1164
<b>Security Roles .....</b>	<b>1166</b>
Modifying the Members of a Security Role .....	1168
Add the Required Users to the Runtime Users Role .....	1171
<b>Directory Security .....</b>	<b>1172</b>
Configure Directory Security for Modified Folder Locations .....	1173
<b>Runtime System Security .....</b>	<b>1174</b>
<b>Areas .....</b>	<b>1174</b>
Use a Label to Name an Area .....	1176
Create an Area Group .....	1177
Configure View-only Access to an Area .....	1178
<b>Privileges .....</b>	<b>1179</b>
Privilege and Area Combinations .....	1180
<b>Roles .....</b>	<b>1182</b>
<b>Users .....</b>	<b>1186</b>
Add a Plant SCADA User .....	1187
Reserved User Names .....	1189
User Records and Project Restoration .....	1189
<b>Integrate Windows™ User Groups .....</b>	<b>1190</b>
Windows™ Security Usage Scenarios .....	1191
<b>Securing Runtime Computers .....</b>	<b>1191</b>
Client Startup Restrictions .....	1192
Running a Display Client as a Shell .....	1192
Disabling Windows Keyboard Commands .....	1192
Disabling Control Menu Commands .....	1193
Removing the Cancel Button .....	1193
<b>Standards .....</b>	<b>1193</b>
<b>Labels .....</b>	<b>1193</b>
Define a Label .....	1196
Use Arguments in Labels .....	1197
Convert Values into Strings .....	1198
<b>Styles .....</b>	<b>1199</b>
Configure Style Overrides for Quality and Status Elements .....	1201
Configure Style Overrides for Element Styles .....	1203
Configure Format Styles for Value Display Animations .....	1205
Copy Styles to Another Project .....	1206
<b>Groups .....</b>	<b>1207</b>
Add a Group .....	1207
<b>Fonts .....</b>	<b>1208</b>
Add a System Font .....	1209
Configuring Custom Color Fonts .....	1211
<b>Setup .....</b>	<b>1211</b>
<b>Alarming .....</b>	<b>1212</b>
Add an Alarm Category .....	1212
Configure Display Properties for an Alarm Priority .....	1219
Configure Display Properties for an Alarm Mode .....	1222
Add Cause and Response Information to Alarms .....	1223
Project Database Parameters .....	1225
Screen Profiles .....	1226

Devices .....	1229
Add a Device .....	1232
Configure SQL Devices .....	1236
Configure a Device Group .....	1237
Format Data in the Device .....	1238
Command Fields .....	1240
Use a Database Device .....	1241
Use Device History Files .....	1243
Predefined Devices .....	1245
Print Management .....	1246
Triggering Events .....	1247
Add a Triggering Event .....	1247
Specify Event Times and Periods .....	1249
Use Event Triggers .....	1250
Languages .....	1251
Configure Languages for a Standard Project .....	1251
Prepare a Project for Multi-language Support .....	1252
Define the Languages Supported by a Project .....	1252
Languages Properties .....	1254
Officially Supported Languages .....	1255
Define an Unsupported Language .....	1258
Translate a Local Language Database .....	1259
Mark Text for Translation .....	1259
Mark Alarm Text for TransltionP .....	1260
Set the Local Language Used at Runtime .....	1262
Logging Data in Different Languages .....	1262
Change the Local Language at Runtime .....	1263
Alarm Data Localization .....	1263
ASCII and ANSI Character Sets .....	1264
OEM Character Sets .....	1264
Configure Languages for Industrial Graphics Applications .....	1265
Working with Industrial Graphics Languages in Included Projects .....	1268
Keyboard Keys .....	1268
Define Keyboard Keys .....	1269
Custom Files .....	1270
Include Custom Files .....	1271
<b>Compile .....</b>	<b>1273</b>
Compile a Project .....	1274
Debug a Compilation .....	1275
Compile Messages .....	1276
Compile Error Messages .....	1277
Compile Fatal Messages .....	1298
Compile Warning Messages .....	1304
Incremental Compilation .....	1313
Compilation Options .....	1313
Post Compile Commands .....	1314
Distribute the Project .....	1314
<b>Deployment .....</b>	<b>1317</b>
Prepare Your System for Deployment .....	1319

Configure a Deployment Server .....	1320
Configure a Deployment Client .....	1322
Configure a Runtime Computer for Deployment .....	1324
Provide Deployment Access to Additional Users .....	1327
Connect Plant SCADA Studio to a Deployment Server .....	1327
Deployment in Plant SCADA Studio .....	1328
Computers .....	1329
Versions .....	1332
Settings .....	1334
Using the Deployment Activity .....	1335
Add a Deployment Server .....	1336
Make a Deployment Server Active .....	1337
Remove a Deployment Server .....	1337
Create a Computer Group .....	1337
Rename a Computer Group .....	1338
Remove a Computer Group .....	1339
Remove a Deployment Client .....	1339
Add a Version .....	1339
Deploy a Version .....	1340
Remove a Version .....	1342
Specify the Update Method .....	1343
Create a Custom Deployment Prompt .....	1344
Using Citect Anywhere to View Deployed Projects .....	1344
Deployment Troubleshooting .....	1345
<b>Runtime .....</b>	<b>1352</b>
Run a Project .....	1352
Runtime Manager .....	1353
Runtime Manager Interface .....	1355
Launch Runtime Manager from Plant SCADA Studio .....	1356
Monitor Runtime Processes .....	1356
Start a Process .....	1357
Stop a Process .....	1358
Restart a Process .....	1358
Cancel a Process .....	1359
Reload a Process .....	1359
Use the Kernel with a Selected Process .....	1360
Hide Runtime Manager .....	1361
Shut Down Runtime Manager .....	1361
Operate Runtime Manager in Service Mode .....	1362
Online Changes .....	1363
Client Side Online Changes .....	1364
Handling Cicode Changes during Runtime .....	1364
Server Side Online Changes .....	1365
Restart an Alarms Server .....	1366
Restart a Trends Server .....	1367
Restart a Reports Server .....	1368
Impact of a Server Reload on Historical Records .....	1368
Restarting the System Online .....	1372
Restarting a Networked System Online .....	1373

Using Multiple Projects .....	1374
Initiating the Online Restart .....	1375
Using a Callback Function .....	1375
Firewall Settings and Plant SCADA .....	1376
Startup and Runtime Configuration .....	1378
Server Redirection Using Address Forwarding .....	1378
Virtualization Host Support .....	1379
Anti-virus Software Setup .....	1380
Using an Alternative INI File .....	1380
Monitor Runtime .....	1381
Log Files .....	1381
Log File Time Stamping .....	1383
Log File Locations .....	1384
Perform Log File Calculations .....	1384
Configure Logging .....	1385
Adjust Logging During Runtime .....	1386
The Crash Handler .....	1387
Debug Runtime .....	1388
The Kernel .....	1389
Display the Kernel Window .....	1390
Display the Kernel from the Control Menu .....	1391
Define a Runtime Command .....	1391
Close the Kernel Window .....	1392
Inside the Kernel .....	1392
Use the Kernel to Commission a System .....	1394
The Cicode Profiler .....	1394
Kernel Commands .....	1395
Cache .....	1397
Cicode .....	1398
Cls .....	1399
Ctapi .....	1400
Debug .....	1400
Diag Cicodestack .....	1402
DriverTrace .....	1402
Exit .....	1405
Help .....	1405
Log .....	1406
Page General .....	1406
Page Driver .....	1412
Page Memory .....	1417
Page Queue .....	1417
Page RDB .....	1419
Page Table .....	1419
Page Table Accum.ReloadError .....	1420
Page Table Alarm.ReloadError .....	1421
Page Table Broadcast .....	1422
Page Table Cicode .....	1422
Page Table CSAt PSI.Subs .....	1422
Page Table Data.Connection .....	1423

Page Table Data.Recordset .....	1423
Page Table OPCServerConnections .....	1424
Page Table Page .....	1425
Page Table PerformanceCounter .....	1425
Page Table Platform.Sessions .....	1426
Page Table Profile .....	1427
Page Table Report.ReloadError .....	1428
Page Table SQL .....	1429
Page Table Stats .....	1429
Page Table Tran .....	1430
Page Table Trend.ReloadError .....	1434
Page Table Users .....	1435
Page Unit .....	1435
Pause .....	1439
Shell .....	1439
Stats .....	1440
Tran .....	1440
Write Log .....	1441
Write Modules .....	1442
System Tuning .....	1442
Cache Tuning .....	1442
Log Viewer .....	1443
Running the System as a Windows Service .....	1444
Configure a System to Run as a Windows Service .....	1445
Running as a Service Under a Specific User Account .....	1445
Running an OPC DA Server as a Service .....	1446
Time Synchronization .....	1447
Runtime - Frequently Asked Questions .....	1448
 <b>Runtime Client Tools .....</b>	 <b>1450</b>
<b>Process Analyst .....</b>	<b>1450</b>
The Process Analyst Interface .....	1450
The Chart View .....	1452
The Object View .....	1453
The Navigation Toolbar .....	1454
The Main Toolbar .....	1455
Process Analyst Properties Dialog Box .....	1456
Using the Property Tree Right-click Menu .....	1457
Pens .....	1458
Pen Layout .....	1460
Pen Selection .....	1460
Data Compaction .....	1461
Interpolated Samples .....	1462
Request Modes .....	1462
Data Quality .....	1462
Cursors .....	1463
Using Cursor Labels .....	1465
Mouse Pointers .....	1466

Interacting with the Process Analyst .....	1467
Add Pens .....	1468
Delete Pens .....	1469
Lock/Unlock Pens .....	1470
Specify a Start Time and End Time .....	1470
Shift and Fit Time Units .....	1471
Scroll the Chart .....	1472
Scale the Chart .....	1472
Lock/Unlock the Time Span .....	1473
Navigate Time .....	1473
Synchronize to Now .....	1473
Toggle Autoscrolling .....	1474
Zoom In/Zoom Out .....	1474
Undo Last Zoom .....	1475
Toggle Box Zoom .....	1475
Set a Nonstandard Time Span .....	1476
Edit Vertical Scale .....	1476
Reset to Default Span .....	1477
Use the Object View .....	1477
Using Instant Trends with Process Analyst .....	1478
Configure the Process Analyst at Runtime .....	1478
Configure Chart-wide Properties .....	1479
Configure General Properties .....	1479
Configure Server Paths .....	1481
Configure Chart Panes .....	1482
Configure Pens .....	1483
Configure Pen Appearance .....	1483
Configure Analog and Digital Pens .....	1484
Configure Alarm Pens .....	1485
Configure Pen Gridlines .....	1486
Configure Pen Axes .....	1486
Configure Pen Quality .....	1488
Configure the Pen Data Connection .....	1489
Configure Cursor Labels .....	1491
Configure Cursors .....	1492
Configure Defaults .....	1492
Configuring Toolbars .....	1492
Adding or Removing Toolbar Commands .....	1493
Changing the Order of Toolbar Commands .....	1494
Configuring the Object View .....	1494
Object View Properties Page .....	1495
Working with Views .....	1496
Saving a View .....	1496
Load a View .....	1497
Operator Command Reference .....	1497
View Commands .....	1498
Zoom Commands .....	1498
Navigation Commands .....	1499
Export Commands .....	1500

Interface Commands .....	1501
General Commands .....	1502
Printing and Exporting .....	1503
Process Analyst Reports .....	1503
Configure Process Analyst Report Options .....	1504
Setting up Report Legends .....	1505
Setting up Report Options .....	1506
Exporting Pen Data .....	1507
Copying Data to the Clipboard .....	1508
Copying Data to File .....	1508
Customize Process Analyst .....	1509
Configuring the Process Analyst Control from Graphics Builder .....	1509
Security and Permissions .....	1510
Administration Privilege .....	1510
Command Privilege .....	1511
Write Privilege .....	1511
Multi-language Support .....	1511
About Process Analyst Resources .....	1511
Creating Your Own Process Analyst Resource.dll .....	1512
Localizing the Process Analyst Resource DLL .....	1513
Using Plant SCADA to Switch the Process Analyst Language .....	1514
Manually Switching Languages using IProcessAnalyst.Language .....	1515
Changing the Input Language .....	1515
Persistence .....	1515
Saving while Using Graphics Builder .....	1515
Using the Save View Toolbar Button .....	1516
Using the SaveToFile Automation Method .....	1516
Saving between Plant SCADA Page Transitions (Runtime) .....	1516
Resetting Back to the Default State .....	1517
Backing up Projects .....	1517
Configuring Design Time Properties .....	1517
Adding New Commands .....	1518
Editing Existing Custom Commands .....	1519
Creating or Editing Object View Columns .....	1519
Process Analyst View Synchronization .....	1520
Using the Process Analyst Command System .....	1522
Command System Overview .....	1522
Custom Commands .....	1522
CommandExecuted .....	1522
UpdateCommand .....	1522
Icons .....	1523
Automation Model .....	1523
Execution Results .....	1524
Enumerations .....	1525
AlarmType [Enumeration] .....	1525
AxisLabelType [Enumeration] .....	1526
DisplaySize[Enumeration] .....	1527
ErrorNotifyCode [Enumeration] .....	1528
FileLocation [Enumeration] .....	1529

HatchStyle [Enumeration] .....	1529
LineStyle [Enumeration] .....	1530
LineType [Enumeration] .....	1531
PenNameMode [Enumeration] .....	1531
PenType [Enumeration] .....	1532
PointType [Enumeration] .....	1532
QualityCompactionType [Enumeration] .....	1533
QualityType [Enumeration] .....	1534
RequestMode [Enumeration] .....	1534
ToolbarButtonType [Enumeration] .....	1535
<b>Events .....</b>	<b>1536</b>
CommandExecuted [Event] .....	1536
CursorMoved [Event] .....	1537
Error [Event] .....	1538
HorizontalAxisChanged [Event] .....	1539
MouseClick [Event] .....	1539
MouseDoubleClick [Event] .....	1540
OVColumnAdded [Event] .....	1541
OVColumnRemoved [Event] .....	1541
OVIItemAdded [Event] .....	1542
OVIItemChecked [Event] .....	1543
OVIItemRemoved [Event] .....	1543
OVIItemSelected [Event] .....	1544
PenCreated [Event] .....	1544
PenDeleted [Event] .....	1545
PenRenamed [Event] .....	1546
PenSelectionChanged [Event] .....	1546
PropertyChanged [Event] .....	1547
UpdateCommand [Event] .....	1548
VerticalAxisChanged [Event] .....	1549
ViewLoadedSaved [Event] .....	1549
<b>Interfaces .....</b>	<b>1550</b>
<b>IAlarmPen Interface .....</b>	<b>1551</b>
IAlarmPen.AlarmType [Property][Get/Set] .....	1551
IAlarmPen.GetFillColor [Method] .....	1552
IAlarmPen.GetHatchColor [Method] .....	1553
IAlarmPen.GetHatchStyle [Method] .....	1554
IAlarmPen.LineColor [Property][Get/Set] .....	1555
IAlarmPen.LineWidth [Property][Get/Set] .....	1555
IAlarmPen.SetFillColor [Method] .....	1556
IAlarmPen.SetHatchColor [Method] .....	1557
IAlarmPen.SetHatchStyle [Method] .....	1558
<b>IAnalogPen Interface .....</b>	<b>1559</b>
IAnalogPen.LineColor [Property][Get/Set] .....	1559
IAnalogPen.LineInterpolation [Property][Get/Set] .....	1560
IAnalogPen.LineWidth [Property][Get/Set] .....	1561
<b>ICommand Interface .....</b>	<b>1562</b>
ICommand.Tooltip [Property][Get] .....	1564
ICommand.ButtonType [Property][Get] .....	1565

ICommand.CommandId [Property][Get] .....	1566
ICommand.Enabled [Property][Get/Set] .....	1567
ICommand.Pressed [Property][Get/Set] .....	1567
ICommand.Privilege [Property][Get] .....	1568
<b>ICommandSystem Interface</b> .....	<b>1569</b>
ICommandSystem._NewEnum [Property][Get] .....	1570
ICommandSystem.Count [Property][Get] .....	1570
ICommandSystem.Create [Method] .....	1571
ICommandSystem.Execute [Method] .....	1572
ICommandSystem.Item [Property][Get] .....	1573
ICommandSystem.ItemById [Property][Get] .....	1574
ICommandSystem.Remove [Method] .....	1574
<b>ICursors Interface</b> .....	<b>1575</b>
ICursors._NewEnum [Property][Get] .....	1575
ICursors.Count [Property][Get] .....	1576
ICursors.Create [Method] .....	1577
ICursors.Item [Property][Get] .....	1578
ICursors.ItemByName [Property][Get] .....	1578
ICursors.RemoveAll [Method] .....	1579
<b>IDigitalPen Interface</b> .....	<b>1580</b>
IDigitalPen.Fill [Property][Get/Set] .....	1580
IDigitalPen.FillColor [Property][Get/Set] .....	1581
IDigitalPen.LineColor [Property][Get/Set] .....	1582
IDigitalPen.LineWidth [Property][Get/Set] .....	1583
<b>IOBJECTVIEW Interface</b> .....	<b>1584</b>
IOBJECTVIEW.BackgroundColor [Property][Get/Set] .....	1584
IOBJECTVIEW.Columns [Property][Get] .....	1585
IOBJECTVIEW.ForeColor [Property][Get/Set] .....	1586
IOBJECTVIEW.Height [Property][Get/Set] .....	1587
IOBJECTVIEW.Items [Property][Get] .....	1587
IOBJECTVIEW.SelectedItem [Property][Get] .....	1588
IOBJECTVIEW.Visible [Property][Get/Set] .....	1589
<b>IOBJECTVIEWCOLUMN Interface</b> .....	<b>1590</b>
IOBJECTVIEWCOLUMN.Name [Property][Get] .....	1590
IOBJECTVIEWCOLUMN.Text [Property][Get] .....	1591
IOBJECTVIEWCOLUMN.Width [Property][Get/Set] .....	1592
<b>IOBJECTVIEWCOLUMNS Interface</b> .....	<b>1592</b>
IOBJECTVIEWCOLUMNS._NewEnum [Property][Get] .....	1593
IOBJECTVIEWCOLUMNS.Add [Method] .....	1593
IOBJECTVIEWCOLUMNS.Count [Property][Get] .....	1594
IOBJECTVIEWCOLUMNS.Hide [Method] .....	1595
IOBJECTVIEWCOLUMNS.Item [Property][Get] .....	1596
IOBJECTVIEWCOLUMNS.ItemByName [Property][Get] .....	1596
IOBJECTVIEWCOLUMNS.Remove [Method] .....	1597
IOBJECTVIEWCOLUMNS.Show [Method] .....	1598
<b>IOBJECTVIEWITEM Interface</b> .....	<b>1599</b>
IOBJECTVIEWITEM.Expanded [Property][Get/Set] .....	1599
IOBJECTVIEWITEM.GetField [Method] .....	1600
IOBJECTVIEWITEM.Items [Property][Get] .....	1601

IObjectViewItem.PutField [Method] .....	1602
IObjectViewItem.Tag [Property][Get/Set] .....	1603
IObjectViewItems Interface .....	1604
IObjectViewItems._NewEnum [Property][Get] .....	1604
IObjectViewItems.Count [Property][Get] .....	1604
IObjectViewItems.Item [Property][Get] .....	1605
IObjectViewPenItem Interface .....	1606
IObjectViewPenItem.BlockColor [Property][Get] .....	1606
IObjectViewPenItem.Checked [Property][Get/Set] .....	1607
IObjectViewPenItem.Selected [Property][Get] .....	1608
IPane Interface .....	1609
IPane.BackgroundColor [Property][Get/Set] .....	1609
IPane.Collection [Property][Get] .....	1610
IPane.Delete [Method] .....	1611
IPane.FixedHeight [Property][Get/Set] .....	1611
IPane.Height [Property][Get/Set] .....	1612
IPane.Name [Property][Get/Set] .....	1613
IPane.Pens [Property][Get] .....	1614
IPanes Interface .....	1615
IPanes._NewEnum [Property][Get] .....	1615
IPanes.Count [Property][Get] .....	1616
IPanes.Create [Method] .....	1616
IPanes.Item [Property][Get] .....	1617
IPanes.ItemByName [Property][Get] .....	1618
IPanes.RemoveAll [Method] .....	1619
IPen Interface .....	1619
IPen.AddSample .....	1621
IPen.AxisBackgroundColor [Property][Get/Set] .....	1623
IPen.BlockRepaint [Property][Get/Set] .....	1623
IPen.Clear [Method] .....	1624
IPen.ClearOnResolutionChange [Property][Get/Set] .....	1625
IPen.Collection [Property][Get] .....	1626
IPen.DataPoint [Property][Get/Set] .....	1627
IPen.DataServer [Property][Get/Set] .....	1628
IPen.Delete [Method] .....	1629
IPen.GetDefaultSpan [Method] .....	1629
IPen.GetHorizontalAxisTimeSpan [Method] .....	1631
IPen.GetInformation [Method] .....	1632
IPen.GetStatistic [Method] .....	1634
IPen.GetVerticalAxisSpan [Method] .....	1635
IPen.GoToNow [Method] .....	1636
IPen.Height [Property][Get/Set] .....	1636
IPen.HorizontalAxisColor [Property][Get/Set] .....	1637
IPen.HorizontalAxisResize [Property][Get/Set] .....	1638
IPen.HorizontalAxisScroll [Property][Get/Set] .....	1639
IPen.HorizontalAxisWidth [Property][Get/Set] .....	1640
IPen.HorizontalGridlinesColor [Property][Get/Set] .....	1641
IPen.HorizontalGridlinesStyle [Property][Get/Set] .....	1642
IPen.HorizontalGridlinesWidth [Property][Get/Set] .....	1643

IPen.HorizontalMinorGridlinesColor [Property][Get/Set] .....	1644
IPen.HorizontalMinorGridlinesStyle [Property][Get/Set] .....	1645
IPen.HorizontalScrollBy [Method] .....	1646
IPen.HorizontalZoom [Method] .....	1647
IPen.InstantTrend [Property][Get/Set] .....	1648
IPen.IsDeleted [Property][Get] .....	1648
IPen.isSelected [Property][Get] .....	1649
IPen.LocalTime [Property][Get/Set] .....	1650
IPen.Name [Property][Get/Set] .....	1650
IPen.PointsVisible [Property][Get/Set] .....	1651
IPen.PutHorizontalAxisTimeSpan [Method] .....	1652
IPen.PutVerticalAxisSpan [Method] .....	1654
IPen.RefreshData [Method] .....	1655
IPen.RequestMode [Property][Get/Set] .....	1655
IPen.ResetToDefaultSpan [Method] .....	1656
IPen.SamplePeriod [Property][Get/Set] .....	1657
IPen.Select [Method] .....	1658
IPen.SetDefaultSpan [Method] .....	1658
IPen.SetQualityCompactionPointType [Method] .....	1659
IPen.SetQualityLineStyle [Method] .....	1660
IPen.SetVerticalAxisLabelValue [Method] .....	1661
IPen.Stacked [Property][Get/Set] .....	1662
IPen.TrendCursorLabelFillColor [Property][Get/Set] .....	1663
IPen.TrendCursorLabelLineColor [Property][Get/Set] .....	1664
IPen.TrendCursorLabelTextColor [Property][Get/Set] .....	1665
IPen.VerticalAxisAutoscale [Property][Get/Set] .....	1666
IPen.VerticalAxisColor [Property][Get/Set] .....	1667
IPen.VerticalAxisLabelType [Property][Get/Set] .....	1667
IPen.VerticalAxisResize [Property][Get/Set] .....	1668
IPen.VerticalAxisScroll [Property][Get/Set] .....	1669
IPen.VerticalAxisWidth [Property][Get/Set] .....	1670
IPen.VerticalGridlinesColor [Property][Get/Set] .....	1671
IPen.VerticalGridlinesStyle [Property][Get/Set] .....	1672
IPen.VerticalGridlinesWidth [Property][Get/Set] .....	1673
IPen.VerticalMinorGridlinesColor [Property][Get/Set] .....	1674
IPen.VerticalMinorGridlinesStyle [Property][Get/Set] .....	1675
IPen.VerticalScrollBy [Method] .....	1676
IPen.VerticalZoom [Method] .....	1677
IPen.Visible [Property][Get/Set] .....	1678
IPens Interface .....	1679
IPens._NewEnum [Property][Get] .....	1679
IPens.Count [Property][Get] .....	1679
IPens.Create [Method] .....	1680
IPens.Item [Property][Get] .....	1681
IPens.ItemByName [Property][Get] .....	1682
IPens.Pane [Property][Get] .....	1683
IPens.RemoveAll [Method] .....	1683
IProcessAnalyst Interface .....	1684
IProcessAnalyst.BlockUpdates [Method] .....	1685

IProcessAnalyst.UnBlockUpdates [Method] .....	1686
IProcessAnalyst.CopyToClipboard [Method] .....	1687
IProcessAnalyst.CopyToFile [Method] .....	1688
IProcessAnalyst.FreezeEvent [Method] .....	1689
IProcessAnalyst.LoadFromFile [Method] .....	1690
IProcessAnalyst.PrintAll [Method] .....	1691
IProcessAnalyst.SaveToFile [Method] .....	1691
IProcessAnalyst.ShowProperties [Method] .....	1692
IProcessAnalyst.SubscribeForPropertyChange [Method] .....	1693
IProcessAnalyst.SynchroniseToNow [Method] .....	1694
IProcessAnalyst.UnsubscribePropertyChange [Method] .....	1695
IProcessAnalyst.AdminPrivilegeLevel [Property] [Get] .....	1696
IProcessAnalyst.AutoScroll [Property][Get/Set] .....	1697
IProcessAnalyst.BackgroundColor [Property][Get/Set] .....	1698
IProcessAnalyst.CommandSystem [Property][Get] .....	1699
IProcessAnalyst.ContextMenu [Property][Get/Set] .....	1700
IProcessAnalyst.Cursors [Property][Get] .....	1701
IProcessAnalyst.DataRequestRate [Property][Get/Set] .....	1701
IProcessAnalyst.DisplayRefreshRate [Property][Get/Set] .....	1702
IProcessAnalyst.DisplaySize[Property][Get/Set] .....	1704
IProcessAnalyst.Language [Property] [Get/Set] .....	1704
IProcessAnalyst.LastSelectedPen [Property][Get] .....	1705
IProcessAnalyst.LockedPens [Property][Get/Set] .....	1706
IProcessAnalyst.NumberofSamples[Property][Get/Set] .....	1707
IProcessAnalyst.ObjectView [Property][Get] .....	1709
IProcessAnalyst.Panes [Property][Get] .....	1710
IProcessAnalyst.PrimaryPath [Property][Get/Set] .....	1710
IProcessAnalyst.SecondaryPath [Property][Get/Set] .....	1711
IProcessAnalyst.Toolbars [Property][Get] .....	1712
IProcessAnalyst.ViewLocation [Property][Get] .....	1713
IProcessAnalyst.WritePrivilegeLevel [Property][Get] .....	1714
IProcessAnalyst.ZoomMode [Property][Get/Set] .....	1715
IToolbar Interface .....	1716
IToolbar.Buttons [Property][Get] .....	1716
IToolbar.Visible [Property][Get/Set] .....	1717
IToolbars Interface .....	1718
IToolbars.Item [Property][Get] .....	1718
IToolbars._NewEnum [Property][Get] .....	1719
IToolbars.Count [Property][Get] .....	1719
IToolBarButton Interface .....	1720
IToolBarButton.CommandId [Property][Get] .....	1720
IToolbarButtons Interface .....	1721
IToolbarButtons._NewEnum [Property][Get] .....	1721
IToolbarButtons.Add [Method] .....	1722
IToolbarButtons.Count [Property][Get] .....	1723
IToolbarButtons.Item [Property][Get] .....	1723
IToolbarButtons.Remove [Method] .....	1724
IToolbarButtons.RemoveAll [Method] .....	1725
ITrendCursor Interface .....	1725

ITrendCursor.Collection [Property][Get] .....	1726
ITrendCursor.Color [Property][Get/Set] .....	1727
ITrendCursor.Delete [Method] .....	1728
ITrendCursor.GetValue [Method] .....	1728
ITrendCursor.LabelsLocked [Property][Get/Set] .....	1729
ITrendCursor.Name [Property][Get/Set] .....	1730
ITrendCursor.PenLabelHeight [Property][Get/Set] .....	1731
ITrendCursor.PenLabelVisible [Property][Get/Set] .....	1732
ITrendCursor.PenLabelWidth [Property][Get/Set] .....	1733
ITrendCursor.PenLabelX [Property][Get/Set] .....	1734
ITrendCursor.PenLabelY [Property][Get/Set] .....	1735
ITrendCursor.Position [Property][Get/Set] .....	1736
ITrendCursor.Visible [Property][Get/Set] .....	1737
ITrendCursor.Width [Property][Get/Set] .....	1738
Automation Examples .....	1739
Handling an Event .....	1739
Enumerating Collections .....	1740
Implementing a Custom Command .....	1741
Enabling and Disabling a Command .....	1743
Implementing a Custom Column .....	1744
The Update Function .....	1744
Event Handlers .....	1744
<b>Schedules .....</b>	<b>1747</b>
Scheduler .....	1750
Equipment Tree .....	1751
Calendar .....	1752
Special Days .....	1754
Schedule Inheritance .....	1756
Schedule Priorities .....	1758
Override Mode .....	1759
Prepare Your System for Scheduling .....	1761
Configure Equipment for Scheduling .....	1761
Configure a Demand and Response Solution .....	1762
Add the Scheduler ActiveX Object to a Page .....	1763
Use ActiveX Properties to Control Scheduler's Current Selection .....	1763
Back Up Schedules .....	1766
Manage Security and Permissions .....	1766
Configure Scheduler Redundancy .....	1766
Integrate BACnet Schedules into Scheduler .....	1767
Specify a Data Type for a BACnet Schedule .....	1768
Operate Scheduler at Runtime .....	1770
Add a Schedule Entry .....	1770
Add a Recurring Schedule Entry .....	1771
Edit a Schedule Entry .....	1773
Enable Override Mode .....	1774
Manually Set a State Change .....	1775
Configure Special Days .....	1776
Manage Daylight Saving Transitions .....	1778
View and Edit BACnet Schedules at Runtime .....	1779

Monitor Schedules .....	1780
Kernel Trace Messages .....	1780
Tracelog Trace Messages .....	1781
<b>Graphics .....</b>	<b>1783</b>
<b>Graphics Builder .....</b>	<b>1783</b>
Graphics Builder Settings .....	1784
Set Graphics Builder Options .....	1784
Customize Graphics Builder .....	1786
Display the Drawing Toolbox .....	1786
Use a Grid to Align Objects .....	1788
Use Guidelines to Align Objects .....	1788
Display the Zoom Dialog Box .....	1790
Use a Cross Hair Cursor .....	1790
Colors .....	1790
Color Picker .....	1791
Edit Favorite Colors Dialog Box .....	1791
Swap Color Dialog Box .....	1794
Adjust Colors Dialog Box .....	1796
Page Templates .....	1797
Screen Resolution .....	1799
Sizing the Runtime Window .....	1800
Configure Graphics Pages .....	1803
Create a Graphics Page .....	1803
Open or Save a Graphics Page .....	1805
Locate a Graphics Page .....	1807
Edit the Properties for a Page .....	1807
Page Properties - General .....	1808
Page Properties - Appearance .....	1809
Page Properties - Keyboard Commands .....	1811
Page Properties - Events .....	1813
Page Properties - Environment .....	1815
Page Properties - Associations .....	1816
Edit the Default Page Settings .....	1818
Use a Browse Sequence .....	1819
Configure a Startup Page and Splash Screen .....	1820
Secure the Window Title Bar .....	1821
Display Tags on a Page .....	1821
Create a Template .....	1822
Link a Page to a Template .....	1823
The Bitmap Editor .....	1823
Update Pages .....	1825
Pack Libraries .....	1825
Graphics Objects .....	1826
Graphics Object Types .....	1827
Object Groups .....	1828
Configure Graphics Objects .....	1829
Add a Free Hand Line .....	1829

Add a Straight Line .....	1831
Add a Rectangle .....	1832
Add an Ellipse .....	1836
Add a Polygon .....	1841
Animate a Polygon at Runtime .....	1845
Add a Pipe .....	1849
Add Text .....	1850
Text Properties - Appearance Display Value (On/Off) .....	1853
Text Properties - Appearance Display Value (Multi-state) .....	1854
Text Properties - Appearance Display Value (Array) .....	1855
Text Properties - Appearance Display Value (Numeric) .....	1856
Text Properties - Appearance Display Value (String) .....	1856
Add a Button .....	1857
Add Numbers .....	1860
Add a Symbol Set .....	1860
Symbol Set Properties - Appearance General (On/Off) .....	1862
Symbol Set Properties - Appearance General (Multi-state) .....	1863
Symbol Set Properties - Appearance General (Array) .....	1864
Symbol Set Properties - Appearance General (Animated) .....	1865
Add a Trend .....	1866
Insert Trend Dialog Box .....	1869
Add a Cicode Object .....	1869
Paste a Symbol .....	1871
Symbol Properties - Appearance (General) .....	1872
Add a Genie .....	1873
Insert a Composite Genie .....	1873
Add a Web Content Object .....	1875
Add an ActiveX Object .....	1876
Tag Association .....	1877
Managing Associated Data Sources .....	1880
Object Identification .....	1881
Add a Database Exchange Control .....	1882
Import Graphics .....	1882
Edit Common Object Properties .....	1883
Appearance .....	1884
3D Effects .....	1884
Visibility .....	1887
Movement .....	1888
Object Properties - Movement (Horizontal) .....	1891
Object Properties - Movement (Vertical) .....	1893
Object Properties - Movement (Rotational) .....	1895
Scaling .....	1897
Object Properties - Scaling (Horizontal) .....	1898
Object Properties - Scaling (Vertical) .....	1901
Fill .....	1904
Fill Color .....	1904
Object Properties - Fill Color (On/Off) .....	1905
Object Properties - Fill Color (Multi-state) .....	1907
Object Properties - Fill Color (Array) .....	1909

Object Properties - Fill Color (Threshold) .....	1912
Object Properties - Fill Color (Gradient) .....	1914
Fill Level.....	1917
Sliders .....	1921
Object Properties - Slider (Horizontal) .....	1921
Object Properties - Slider (Vertical).....	1923
Object Properties - Slider (Rotational) .....	1924
Input .....	1926
Object Touch Commands.....	1926
Object Keyboard Commands.....	1929
Access .....	1932
General Access to Objects .....	1933
Disable Access to Objects .....	1936
Metadata .....	1937
Using Metadata .....	1938
Passing Animation Point Metadata as Super Genie Associations .....	1945
Alarm Indicator .....	1946
Manipulate Graphics Objects .....	1949
Select Objects.....	1949
Move Objects .....	1950
Resize Objects .....	1950
Reshape Objects.....	1952
Delete Objects .....	1952
Lock/Unlock Objects .....	1952
Group Objects .....	1953
Copy and Paste Objects .....	1953
Change the Overlap of Objects .....	1954
Align Objects .....	1955
Rotate Objects .....	1956
Mirror Objects .....	1957
Locate an Object .....	1957
Copy an Object to the Library .....	1959
Adjust Position and Size Dialog .....	1959
Symbols .....	1960
Create a Symbol .....	1961
Open an Existing Symbol.....	1962
Save a Symbol to a Library .....	1962
Genies .....	1963
Create a New Genie .....	1964
Define Genie Substitutions .....	1965
Use Genie Substitutions in Templates .....	1966
Use Equipment.Item References with Genies .....	1967
Use Structured Tags with Genies .....	1968
Use the IFDEF Macro .....	1969
Genie Forms .....	1970
Genie Definition and Title .....	1971
Record Definition .....	1972
Form Definition .....	1973
Paste a Genie .....	1974

Genie Parameters Dialog Box .....	1974
Configure a Genie .....	1977
Dynamically Instantiate a Genie at Runtime .....	1979
Composite Genies .....	1980
Insert a Composite Genie .....	1981
Edit/Update a Composite Genie .....	1983
Delete Unused Composite Genie Instances .....	1985
Delete a Composite Genie .....	1986
Composite Genie Templates .....	1987
Visual Template .....	1987
Parameters .....	1988
Alarm Indicators .....	1993
Content Items .....	1995
Compositions .....	1996
Prerequisites .....	1996
Container .....	1997
More About Margins and Alignment .....	2002
More About Layouts .....	2003
More About Alarm Indicators .....	2006
Conditions .....	2007
Alarm Indicators and Alarm Flags .....	2010
Super Genies .....	2011
Create a New Super Genie .....	2012
Super Genie Library Objects and the Page Properties Associations Tab .....	2013
Attach a Super Genie to a Genie .....	2013
Edit a Super Genie Library Object .....	2013
Edit a Super Genie Page .....	2014
Web Content .....	2015
Securing Web Content User Data .....	2017
Dynamic Associations .....	2018
Define Dynamic Associations .....	2019
Use Equipment References with Dynamic Associations .....	2020
Use Structured Tags with Dynamic Associations .....	2021
Link Dynamic Associations .....	2021
Dynamic Associations and Areas .....	2022
Use Constants with Dynamic Associations .....	2022
Use Arrays and Array Offsets with Dynamic Associations .....	2023
Create a Dynamic Association Page .....	2024
Display a Dynamic Association Page .....	2025
Animation Points .....	2026
Naming Graphic Objects .....	2028
Interaction with Graphics Pages at Runtime .....	2028
Touch Commands .....	2029
Slider Controls .....	2029
Configuring Commands - Frequently Asked Questions .....	2029
<b>AVEVA™ Industrial Graphics .....</b>	<b>2030</b>
Design an Industrial Graphics Application .....	2033
Enable Tag Writes for Industrial Graphics Applications .....	2035
Create Industrial Graphics .....	2036

The Industrial Graphic Editor .....	2037
Tools Panel .....	2039
Elements List .....	2039
Properties Editor .....	2040
Animations Summary .....	2041
Canvas .....	2042
Elements .....	2042
Basic Elements .....	2043
Status Element .....	2044
Windows Common Controls .....	2045
Groups .....	2047
Path Graphics .....	2048
Properties .....	2049
Predefined Properties .....	2049
Custom Properties .....	2049
Properties of Groups .....	2050
Changing/Renaming a Group Name .....	2051
Changing the Position of a Group .....	2051
Changing the Size of a Group .....	2051
Changing the Orientation of a Group .....	2052
Changing the Transparency of a Group .....	2052
Locking the Group .....	2053
Run-Time Properties of a Group .....	2053
Renaming a Group or its Elements .....	2053
Animations .....	2054
Animation Types .....	2054
Data Sources for Animations .....	2056
Animation Capabilities of Groups .....	2057
Animation States .....	2057
Data Type Animation State .....	2057
Truth Table Animation State .....	2059
Embedding Graphics .....	2060
Appearance of Embedded Graphics .....	2061
Size Propagation and Anchor Points .....	2061
Creating Multiple Configurations of a Graphic .....	2063
Understanding Visual and Functional Graphic Configurations .....	2063
Visual Graphic Configurations .....	2063
Functional Graphic Configurations .....	2064
Using the Industrial Graphic Editor .....	2064
Showing, Hiding and Adjusting Panels .....	2064
Panning and Zooming the Canvas .....	2065
Panning .....	2065
Using the Pan and Zoom Window to Pan .....	2065
Using the Hand Tool to Pan .....	2066
Using the Mouse Scroll Wheel to Pan .....	2066
Zooming .....	2067
Zooming In to a Specified Point .....	2068
Zooming Out from a Specified Point .....	2068
Zooming to the Default Zoom Value .....	2068

Zooming a Selected Element .....	2069
Zooming a Specified Area .....	2069
Selecting or Specifying a Zoom Value .....	2069
Using the Pan and Zoom Window to Change the Zoom .....	2070
Using the Mouse Scroll Wheel for Zooming .....	2070
Configuring Designer Preferences .....	2070
Using the Symbol Wizard Editor .....	2072
Working with Graphic Elements .....	2074
About Graphic Elements .....	2075
Drawing and Dragging Elements .....	2076
Drawing Rectangles, Rounded Rectangles, Ellipses, and Lines .....	2076
Drawing Polylines, Polygons, Curves, and Closed Curves .....	2077
Drawing 2-Point Arcs, 2-Point Pies and 2-Point Chords .....	2077
Drawing 3-Point Arcs, 3-Point Pies, and 3-Point Chords .....	2077
Placing and Importing Images .....	2078
Drawing Buttons .....	2078
Placing Text .....	2079
Drawing Text Boxes .....	2079
Drawing Status Elements .....	2079
Drawing User Interface Common Controls .....	2080
Dragging Elements .....	2080
Import an SVG as an Industrial Graphic .....	2080
SVG Limitations .....	2082
Editing Element Properties .....	2086
Selecting Elements .....	2087
Selecting Elements by Mouse Click .....	2089
Selecting Elements by Lasso .....	2089
Selecting All Elements .....	2089
Selecting Elements Using the Elements List .....	2090
Unselecting Elements .....	2090
Inline Editing .....	2090
Copying, Cutting, and Pasting Elements .....	2091
Copying Elements .....	2092
Cutting or Deleting Elements .....	2092
Duplicating Elements .....	2093
Moving Elements .....	2094
Aligning Elements .....	2095
Aligning Elements Horizontally .....	2095
Aligning Elements Vertically .....	2097
Aligning Elements by their Center Points .....	2098
Aligning Elements by their Points of Origin .....	2099
Adjusting the Spacing between Elements .....	2099
Distributing Elements .....	2100
Making Space between Elements Equal .....	2100
Increasing Space between Elements .....	2101
Decreasing Space between Elements .....	2101
Removing All Space between Elements .....	2102
Resizing Elements .....	2102
Resizing a Single Element with the Mouse .....	2103

Resizing Elements by Changing Size Properties .....	2103
Resizing Elements Proportionally .....	2104
Making Elements the Same Width, Height, or Size .....	2104
Adjusting the z-Order of Elements .....	2105
Rotating Elements .....	2107
Rotating Elements with the Mouse .....	2107
Rotating Elements by Changing the Angle Property .....	2108
Rotating Elements by 90 Degrees .....	2108
Moving the Origin of an Element .....	2109
Changing Points of Origin with the Mouse .....	2109
Changing Points of Origin in the Properties Editor .....	2109
Add Connectors Between Graphic Elements .....	2110
Draw a Connector .....	2111
Adding Connection Points .....	2113
Change Connector Properties .....	2113
Change the Type of Connector .....	2115
Change the Length of a Connector .....	2116
Change the Shape of a Connector .....	2117
Delete a Control Point .....	2118
Flipping Elements .....	2119
Locking and Unlocking Elements .....	2120
Making Changes Using Undo and Redo .....	2120
Working with Groups of Elements .....	2121
Creating a Group of Elements .....	2122
Ungrouping .....	2122
Adding Elements to Existing Groups .....	2123
Removing Elements from Groups .....	2123
Editing Components within a Group .....	2124
Using Path Graphics .....	2124
Creating a Path Graphic .....	2125
Breaking the Path of a Path Graphic .....	2127
Changing a Path Graphic .....	2127
Moving Elements in a Path Graphic .....	2128
Resizing Elements in a Path Graphic .....	2128
Editing Start and Sweep Angles of Elements in a Path Graphic .....	2129
Editing Element Control Points in a Path Graphic .....	2129
Swapping the End Points of an Element in a Path Graphic .....	2130
Changing the Z-order of an Element in a Path Graphic .....	2131
Adding Elements to an Existing Path Graphic .....	2132
Removing Elements from a Path Graphic .....	2133
Editing Common Properties of Elements and Graphics .....	2134
Editing the Name of an Element .....	2134
Editing the Fill Properties of an Element .....	2135
Setting Fill Style .....	2135
Setting Unfilled Style .....	2136
Setting Fill Orientation .....	2137
Setting Fill Behavior .....	2137
Setting Horizontal Fill Direction and Percentage .....	2138
Setting Vertical Fill Direction and Percentage .....	2138

Editing the Line Properties of an Element .....	2139
Setting Start or End Points of a Line .....	2139
Setting the Line Weight .....	2140
Setting the Line Pattern .....	2140
Setting the Line Style .....	2140
Setting the Text Properties of an Element .....	2141
Setting the Displayed Text .....	2142
Setting the Text Display Format .....	2142
Setting the Text Font .....	2142
Setting the Text Color .....	2143
Setting the Text Alignment .....	2143
Substituting Strings .....	2144
Setting Style .....	2145
Setting a Solid Color .....	2146
Setting a Solid Color from the Standard Palette .....	2147
Setting a Solid Color from the Color Disc and Bar .....	2147
Setting a Solid Color with the Value Input Boxes .....	2148
Setting a Solid Color with the Color Picker .....	2148
Setting a Solid Color from the Custom Palette .....	2149
Adding and Removing Colors in the Custom Palette .....	2149
Saving and Loading the Custom Palette .....	2150
Setting a Gradient .....	2150
Setting the Number of Colors for a Gradient .....	2151
Setting the Direction of the Gradient .....	2152
Changing the Variant of a Gradient .....	2153
Setting the Color Distribution Shape .....	2153
Setting the Focus Scales of a Gradient .....	2154
Setting a Pattern .....	2155
Setting a Texture .....	2156
Setting the Style to No Fill .....	2156
Setting the Transparency of a Style .....	2157
Setting the Transparency Level of an Element .....	2157
Adjusting the Colors and Transparency of a Gradient .....	2158
Enabling and Disabling Elements for Run-Time Interaction .....	2158
Changing the Visibility of Elements .....	2159
Editing the Tab Order of an Element .....	2159
Using the Format Painter to Format Elements .....	2160
Editing the General Properties of a Graphic .....	2162
Editing Graphic-Specific and Element-Specific Properties .....	2162
About Graphic- and Element-Specific Properties .....	2163
Setting the Radius of Rounded Rectangles .....	2163
Setting Line End Shape and Size .....	2164
Setting Auto Scaling and Word Wrapping for a Text Box .....	2165
Using Images .....	2166
Placing an Image on the Canvas .....	2166
Setting the Image Display Mode .....	2167
Setting the Image Alignment .....	2167
Setting the Image Color Transparency .....	2168
Editing the Image .....	2169

Setting the Image Editing Application .....	2169
Selecting a Different Image .....	2169
Using Buttons .....	2170
Automatically Scaling Text in Buttons .....	2170
Wrapping Text in Buttons .....	2171
Configuring Buttons with Images .....	2171
Editing Control Points .....	2172
Moving Control Points .....	2172
Adding and Removing Control Points .....	2172
Changing the Tension of Curves and Closed Curves .....	2173
Changing Angles of Arcs, Pies and Chords .....	2173
Utilizing Sweep Angle Run-Time Properties .....	2174
Monitoring and Showing Quality and Status .....	2175
Using Status Elements .....	2175
Using Windows Common Controls .....	2175
Changing Background Color and Text Color of Windows Common Controls .....	2176
Reading and Writing the Selected Value at Run Time .....	2177
Configuring Radio Button Group Controls .....	2178
Setting the 3D appearance of a Radio Button Group Control .....	2178
Setting the Layout of the Radio Button Group Options .....	2179
Using Radio Button Group-Specific Properties at Run Time .....	2179
Configuring Check Box Controls .....	2180
Setting the Default State of a Check Box Control .....	2180
Setting the Caption Text of a Check Box Control .....	2180
Setting the 3D appearance of a Check Box Control .....	2181
Configuring Edit Box Controls .....	2181
Setting the Default Text in an Edit Box Control .....	2182
Configuring the Text to Wrap in an Edit Box Control .....	2182
Configuring the Text to be Read-Only in an Edit Box Control .....	2182
Configuring Combo Box Controls .....	2183
Setting the Type of Combo Box Control .....	2183
Setting the Width of the Drop-Down List .....	2184
Avoiding Clipping of Items in the Simple Combo Box Control .....	2184
Setting the Maximum Number of Items to Appear in the Combo Box Drop-Down List .....	2185
Using Combo Box-Specific Properties at Run Time .....	2185
Configuring Calendar Controls .....	2186
Setting the Number of Calendar Month Sheets .....	2186
Setting the First Day of the Week .....	2187
Showing or Hiding Today's Date on a Calendar Control .....	2188
Setting Title Fill Color and Text Color on a Calendar Control .....	2188
Setting the Text Color for Trailing Dates in a Calendar Control .....	2189
Setting the Default Value of the Calendar Control .....	2190
Configuring DateTime Picker Controls .....	2190
Configuring List Box Controls .....	2192
Avoiding Clipping of Items in the List Box Control List .....	2192
Using a Horizontal Scroll Bar in a List Box Control .....	2193
Using List Box-Specific Properties at Run Time .....	2193
Using Custom Properties .....	2194

About Custom Properties .....	2194
Managing Custom Properties .....	2194
Adding and Deleting Custom Properties .....	2195
Configuring Custom Properties .....	2196
Validating Custom Properties .....	2197
Clearing the Configuration of Custom Properties .....	2198
Renaming Custom Properties .....	2198
Linking Custom Properties to External Sources .....	2198
Overriding Custom Properties .....	2198
Reverting to Original Custom Property Values .....	2199
Owning Object and Relative Referencing .....	2199
Working with Element Styles .....	2202
Visual Properties Defined by Element Styles .....	2202
Applying Element Styles to Elements .....	2203
Using the Element Style List .....	2203
Using the Properties Grid .....	2204
Using Format Painter .....	2205
Clearing an Element Style .....	2206
Selecting an Element Style as a Default for a Canvas .....	2206
Applying Element Styles to Groups of Elements .....	2206
Setting a Group's Run-time Behavior to TreatAsIcon .....	2206
Understanding Element Style Behavior with a Group of Elements .....	2207
Configuring an Animation Using Element Styles .....	2207
Configuring a Boolean Animation Using Element Styles .....	2208
Configuring a Truth Table Animation with Element Styles .....	2208
Deleting a Condition from an Animation Truth Table .....	2209
Changing the Processing Order of Element Styles in a Truth Table Animation .....	2210
Animating Graphic Elements .....	2210
About Animations .....	2210
Adding an Animation to an Element .....	2211
Reviewing which Animations are Assigned to an Element .....	2211
Showing and Hiding the Animation List .....	2212
Removing Animations from an Element .....	2212
Enabling and Disabling Animations .....	2213
Validating the Configuration of an Animation .....	2214
Clearing the Configuration from an Animation .....	2214
Managing Animations .....	2214
Organizing the Animation List .....	2215
Switching between Animations .....	2215
Configuring Common Types of Animations .....	2215
Configuring a Visibility Animation .....	2217
Configuring a Fill Style Animation .....	2217
Configuring a Boolean Fill Style Animation .....	2218
Configuring a Truth Table Fill Style Animation .....	2218
Configuring a Line Style Animation .....	2220
Configuring a Boolean Line Style Animation .....	2220
Configuring a Truth Table Line Style Animation .....	2221
Configuring a Text Style Animation .....	2223
Configuring a Boolean Text Style Animation .....	2223

Configuring a Truth Table Text Style Animation .....	2224
Configuring a Blink Animation .....	2225
Configuring an Alarm Border Animation .....	2226
Understanding Requirements of Alarm Border Animations .....	2227
Understanding the Behavior of Alarm Border Animations .....	2227
Configuring Alarm Border Animation .....	2230
Configuring Optional Alarm Border Animation Characteristics .....	2232
Changing Alarm Border Indicator Icons .....	2232
Modify Alarm Border Animation Element Styles .....	2234
Configuring a Percent Fill Horizontal Animation .....	2235
Configuring a Percent Fill Vertical Animation .....	2236
Configuring a Horizontal Location Animation .....	2238
Configuring a Vertical Location Animation .....	2238
Configuring a Width Animation .....	2239
Configuring a Height Animation .....	2240
Configuring a Point Animation .....	2240
Configuring an Orientation Animation .....	2241
Configuring a Value Display Animation .....	2243
Configuring a Boolean Value Display Animation .....	2244
Configuring an Analog Value Display Animation .....	2244
Configuring Value Display Animation with the FormatStyle Property ..	2245
Configuring a String Value Display Animation .....	2246
Configuring a Time Value Display Animation .....	2246
Configuring a Tooltip Animation .....	2248
Configuring a Disable Animation .....	2249
Configuring a User Input Animation .....	2249
Configuring a User Input Animation for a Discrete Value .....	2250
Configuring a User Input Animation for an Analog Value .....	2251
Configuring a User Input Animation for a String Value .....	2252
Configuring a User Input Animation for a Time Value .....	2253
Configuring a User Input Animation for an Elapsed Time Value .....	2255
Configuring a Horizontal Slider Animation .....	2256
Configuring a Vertical Slider Animation .....	2256
Configuring a Pushbutton Animation .....	2257
Configuring a Pushbutton Animation for a Boolean Value .....	2258
Configuring a PushButton Animation for an Analog Value .....	2259
Configuring a PushButton Animation for a String Value .....	2260
Configuring an Action Script Animation .....	2260
Configuring an Action Script Animation with a "Mouse-Down" Event Trigger ..	2262
Configuring a Show Symbol Animation .....	2262
Configuring a Hide Symbol Animation .....	2270
Configuring a Hyperlink Animation .....	2271
Configuring Element-Specific Animations .....	2271
Configuring Animation for a Status Element .....	2272
Restrictions of the Status Element .....	2273
Configuring a Radio Button Group Animation .....	2273
Configuring a Static Radio Button Group Animation .....	2274
Configuring an Enum Radio Button Group Animation .....	2274
Configuring a Check Box Animation .....	2275

Configuring an Edit Box Animation .....	2275
Configuring a Combo Box Animation .....	2276
Configuring a Static Combo Box Animation .....	2276
Configuring an Array Combo Box Animation .....	2277
Configuring an Enum Combo Box Animation .....	2278
Configuring a Calendar Control Animation .....	2278
Configuring a DateTime Picker Animation .....	2279
Configuring a List Box Animation .....	2281
Configuring a Static List Box Animation .....	2281
Configuring an Array List Box Animation .....	2282
Configuring an Enum List Box Animation .....	2283
Configuring a Trend Pen .....	2283
Understanding the Types of Trend Plots .....	2284
Understanding the Types of Trend Pen Periods .....	2284
Submitting the Value Changes .....	2285
Format Strings in Element-Specific Animations .....	2286
Numbers .....	2286
Dates .....	2287
Enumerations .....	2289
Format String Examples .....	2289
Cutting, Copying and Pasting Animations .....	2290
Substituting References in Elements .....	2290
Adding and Maintaining Graphic Scripts .....	2292
About Graphic Scripts .....	2292
Predefined and Named Scripts .....	2292
Execution Order of Graphic Scripts .....	2293
Graphic Script Time outs .....	2293
Error Handling .....	2294
Configuring the Predefined Scripts of a Graphic .....	2294
Ensuring Proper OnShow Script Execution .....	2295
Adding Named Scripts to a Graphic .....	2296
Editing Graphic Scripts .....	2297
Renaming Scripts in a Graphic .....	2297
Removing Scripts from a Graphic .....	2298
Substituting Equipment.item References in Scripts .....	2298
Using Methods in Scripting .....	2298
Configuring Edit Box Methods .....	2299
Configuring Combo Box and List Box Methods .....	2299
Adding and Inserting Items into a List .....	2300
Deleting Items from a List .....	2301
Finding an Item in a List .....	2302
Reading the Caption of a Selected Item in a List .....	2302
Associating Items with Values in a List .....	2302
Loading and Saving Item Lists .....	2303
Embedding Graphics within Graphics .....	2304
Embedding Graphics .....	2304
Editing the Embedded Graphic .....	2305
Overriding Custom Properties of the Source Graphic .....	2306
Restoring an Embedded Graphic to the Original Size of its Source Graphic .....	2307

Converting an Embedded Graphic to a Group .....	2307
Detecting the Source Graphic of an Embedded Graphic .....	2308
Editing the Source of an Embedded Graphic .....	2308
Controlling Size Propagation of Embedded Graphics .....	2309
Setting the Anchor Point of a Source Graphic .....	2309
Showing or Hiding the Anchor Points of Embedded Graphics .....	2310
Enabling or Disabling Dynamic Size Change of Embedded Graphics .....	2310
Working with Symbol Wizards .....	2312
Introduction .....	2312
Understanding the Symbol Wizard Editor .....	2313
Understanding Choice Groups and Choices .....	2314
Understanding Symbol Wizard Layers .....	2315
Defining Graphic Configuration Rules .....	2316
Examples of Graphic Configuration Rules .....	2317
Designing a Symbol Wizard .....	2317
Creating Graphic Choice Groups, Choices, and Options .....	2318
Assigning Graphic Configuration Rules .....	2319
Updating Graphic Layers .....	2319
Associating Configuration Elements to Graphic Layers .....	2321
Associating Graphic Elements to Graphic Layers .....	2321
Using Shortcut Menu Commands to Edit Graphic Layer Graphic Elements .....	2322
Associating Custom Properties to Graphic Layers .....	2324
Associating Named Scripts to Graphic Layers .....	2325
Verifying Graphic Configurations .....	2325
Using Symbol Wizards in an Application .....	2326
Embedding Symbol Wizards .....	2327
Symbol Wizard Tips and Examples .....	2329
Creating Visual Configurations of an Industrial Graphic .....	2329
Planning Symbol Wizard Configurations .....	2329
Identify Graphic Elements .....	2333
Build a Visual Representation of a Symbol Wizard .....	2336
Assign Graphic Elements, Named Scripts, and Custom Properties to Graphic Layers .....	2338
Specify Rules to select Graphic Layers .....	2338
List of Element Properties .....	2342
Alphabetical List of Properties .....	2342
List by Functional Area .....	2366
Graphic Category Properties .....	2367
Appearance Category Properties .....	2368
Fill Style Group Properties .....	2380
Line Style Group Properties .....	2383
Text Style Group Properties .....	2385
Runtime Behavior Group Properties .....	2387
Custom Properties Group Properties .....	2390
Order of Precedence for Property Styles .....	2391
Switching Languages for Graphic Elements .....	2391
Selecting a Language for a Graphic .....	2392
Removing a Language from a Graphic .....	2392
Creating Elements When Multiple Languages are Defined .....	2393

How Fonts are Applied at Design Time .....	2393
Language Switching for Embedded Graphics .....	2394
String Substitutions and Language Switching .....	2394
Translating String Custom Properties .....	2395
Support for Empty Strings .....	2396
Language Switching Example .....	2396
Overriding Translated Strings for Industrial Graphics .....	2398
Language Switching at Run Time .....	2399
Windows Common Control List Methods .....	2399
QuickScript References .....	2402
Script Functions .....	2402
Graphic Client Functions .....	2403
GetCPQuality() .....	2403
GetCPTimeStamp() .....	2403
HideSelf() .....	2404
Math Functions .....	2405
Abs() .....	2406
ArcCos() .....	2407
ArcSin() .....	2407
ArcTan() .....	2408
Cos() .....	2409
Exp() .....	2410
Int() .....	2410
Log() .....	2411
Log10() .....	2412
LogN() .....	2412
Pi() .....	2413
Round() .....	2414
Sgn() .....	2414
Sin() .....	2415
Sqrt() .....	2416
Tan() .....	2417
Trunc() .....	2417
Miscellaneous Functions .....	2418
DateTimeGMT() .....	2419
IsBad() .....	2419
IsGood() .....	2420
IsInitializing() .....	2421
IsUncertain() .....	2422
IsUsable() .....	2422
LogCustom() .....	2423
LogDataChangeEvent() .....	2424
LogError() .....	2425
LogMessage() .....	2426
LogTrace() .....	2427
LogWarning() .....	2428
String Functions .....	2429
DText() .....	2430
StringASCII() .....	2430

StringChar()	2431
StringCompare()	2432
StringCompareNoCase()	2433
StringFromGMTTimeToLocal()	2434
StringFromIntg()	2435
StringFromReal()	2436
StringFromTime()	2437
StringFromTimeLocal()	2438
StringInString()	2439
StringLeft()	2440
StringLen()	2441
StringLower()	2442
StringMid()	2443
StringReplace()	2444
StringRight()	2445
StringSpace()	2446
StringTest()	2447
StringToIntg()	2448
StringToReal()	2449
StringTrim()	2450
StringUpper()	2451
Text()	2452
System Functions	2453
CreateObject()	2453
Now()	2454
QuickScript .NET Variables	2454
Numbers and Strings	2456
QuickScript .NET Control Structures	2457
IF ...THEN ... ELSEIF ... ELSE ... ENDIF	2458
IF... THEN ... ELSEIF ... ELSE ... ENDIF and Equipment.item Quality	2459
FOR ... TO ... STEP... NEXT Loop	2460
FOR EACH ... IN ... NEXT	2461
TRY ... CATCH	2462
WHILE Loop	2463
QuickScript .NET Operators	2464
Parentheses ( )	2466
Negation ( - )	2466
Complement ( ~ )	2466
Power ( ** )	2466
Multiplication ( * ), Division ( / ), Addition ( + ),Subtraction ( - )	2467
Modulo (MOD)	2467
Shift Left (SHL), Shift Right (SHR)	2467
Bitwise AND ( & )	2468
Exclusive OR (^) and Inclusive OR (   )	2468
Assignment ( = )	2468
Comparisons ( <, >, <=, >=, ==, <> )	2469
AND, OR, and NOT	2469
Use the Industrial Graphics Web Client	2469
View Application Graphics in a Web Browser	2472

Application Graphics and Browser Resizing .....	2473
Pan and Zoom Support .....	2473
Customize the Web Client .....	2474
Manage the Industrial Graphics Web Client .....	2474
Secure the Industrial Graphics Web Client .....	2474
Configure an Industrial Graphics Server using a Secure Gateway .....	2476
Configure User Access for an Industrial Graphics Web Client .....	2481
Log On to the Industrial Graphics Web Client .....	2482
Licensing for Industrial Graphics Applications in a Web Browser .....	2482
Single Session Mode .....	2483
Grace Period .....	2483
Supported Graphical Elements and Known Limitations .....	2484
Known Limitations .....	2484
Properties Supported by All Graphical Elements .....	2485
Properties Supported by All Graphic Elements With Some Limitations .....	2486
Supported Graphic Elements and Additional Properties .....	2487
Supported Animations .....	2489
Troubleshooting the Industrial Graphics Web Client .....	2491
Industrial Graphics Web Client Error Handling .....	2493
Browser and Mobile Support .....	2493
<b>Situational Awareness Projects .....</b>	<b>2494</b>
Key Components of a Situational Awareness Project .....	2495
Autofill .....	2497
Default Layout .....	2502
Header Bar .....	2503
Active Alarms Zone .....	2506
Faceplate Zone .....	2508
Information Zone .....	2510
Navigation Zone .....	2515
Default Alarm Pages .....	2517
Default Trend Pages .....	2521
Alarms .....	2525
Libraries .....	2529
Interlocks .....	2532
Getting Started with a Situational Awareness Project .....	2537
Create a Situational Awareness Project .....	2537
Set Up a Screen Profile for Multiple Monitors .....	2538
Run the Computer Setup Wizard .....	2539
Configure Your Project .....	2539
Add Equipment Using Equipment Editor .....	2540
Equipment Hierarchy Example .....	2542
Generate Variable Tags .....	2545
Create Content Pages .....	2546
Page Levels .....	2548
Assign a Content Type to a Page .....	2549
Prepare the Navigation Menu .....	2550
Enable Navigation Zone Alarm Counts .....	2556
Configure Panes on a Master Page .....	2557
Set the Context Mode for a Workspace .....	2561

Configure Interlocks .....	2561
Configure a Bypass for an Interlock .....	2564
Add a Control Link .....	2565
Create a New Faceplate .....	2566
Configure Equipment Selection for Group Objects .....	2569
Customize a Situational Awareness Project .....	2572
Configure a Project that Supports Multiple Screen Resolutions .....	2572
Create a Master Page for a Customized Workspace .....	2574
Configure Display Properties for a Fourth Alarm Priority .....	2579
Add an Additional Alarm Count to the Navigation Zone .....	2581
Add a Fourth Alarm Priority Filter to Alarm Pages .....	2583
Format an Alarm List for a Specific Screen Resolution .....	2584
Apply a Customized Font to Selected Alarms .....	2585
Customize the Logo on the Header Bar .....	2586
Add a Tree View to a Page .....	2587
Add an Alarms List to a Page .....	2593
Add a Generic List to a Page .....	2597
Add an Alarm Filter Button .....	2603
Create a Set of Tabs for a Pane .....	2605
Use Nested Panes .....	2608
Create a Pop-up Window with Panes .....	2611
Use Navigational Genies on Content Pages .....	2613
Apply a Color Theme to a Workspace .....	2617
Use a Scheduler Control in a Situational Awareness Project .....	2617
Defining a Right Click Action on Equipment .....	2618
 <b>System Projects .....</b>	 <b>2620</b>
<b>Situational Awareness System Projects .....</b>	<b>2620</b>
Situational Awareness Include Project .....	2621
Public Functions .....	2621
Debug_SetTraceLevel .....	2623
Debug_Trace .....	2624
Equipment_GetContent .....	2624
Equipment_GetContext .....	2625
EquipmentStatus_Drive_GetValue .....	2626
EquipmentStatus_Meter_GetValue .....	2626
EquipmentStatus_Valve_GetValue .....	2627
GetPrivEx .....	2627
Help_DisplayFloating .....	2629
Help_DisplayInPane .....	2630
Navigation_ShowHomePage .....	2631
Navigation_ShowTargetPage .....	2631
NavigationLink_IsHidden .....	2632
TabBar_AddTab .....	2633
TabBar_GetTabCount .....	2634
TabBar_GetOpenItemsCount .....	2634
TabBar_ShowIcon .....	2635
Theme_Control_GetColorIndex .....	2636

Theme_Control_GetForegroundColorIndex .....	2637
Theme_Header_GetBackgroundColorIndex .....	2638
Theme_Header_GetColorIndex .....	2638
Theme_Header_GetForegroundColorIndex .....	2639
Theme_Header_GetMidColorIndex .....	2640
Theme_SetTheme .....	2641
Theme_ToggleButton_GetForegroundColorIndex .....	2642
Workspace_GetContext .....	2643
Workspace_GetEquipmentCluster .....	2643
Workspace_GetPageNameWithRes .....	2644
Workspace_GetResolution .....	2645
Workspace_GetSecondaryContext .....	2645
Workspace_GetSelContext .....	2646
Workspace_GetSelSecondaryContext .....	2647
Workspace_GetWindow .....	2647
Workspace_GetWorkspaceFromName .....	2648
Workspace_GetWorkspaceFromWindow .....	2649
Workspace_Init .....	2649
Workspace_IsSelContext .....	2650
Workspace_IsSystemInitialised .....	2651
Workspace_RightClickEquipment .....	2651
Workspace_SelectEquipment .....	2652
Workspace_SetContext .....	2653
Workspace_SetSelContext .....	2654
Workspace_ShowContent .....	2655
SA_Include Genie Libraries .....	2656
Alarm Filter Genie Library .....	2656
Item Acknowledged Genie (HD1080) .....	2657
Item Acknowledged Genie (UHD4K) .....	2658
Item Base Genie .....	2659
Item Priority Genie (HD1080) .....	2660
Item Priority Genie (UHD4K) .....	2661
Item Shelved Genie (HD1080) .....	2663
Item Shelved Genie (UHD4K) .....	2664
Item State Genie (HD1080) .....	2665
Item State Genie (UHD4K) .....	2666
Item Unacknowledged Genie (HD1080) .....	2667
Item Unacknowledged Genie (UHD4K) .....	2668
Controls Genie Library .....	2669
Navigation Zone Button Genie (HD1080) .....	2669
Navigation Zone Button Genie (UHD4K) .....	2670
Navigation Zone - 4 Priority Button Genie (HD1080) .....	2671
Navigation Zone - 4 Priority Button Genie (UHD4K) .....	2672
Navigation Zone Tab Genie (HD1080) .....	2673
Navigation Zone Tab Genie (UHD4K) .....	2674
Navigation Zone - 4 Priority Tab Genie (HD1080) .....	2675
Navigation Zone - 4 Priority Tab Genie (UHD4K) .....	2675
Tab Bar Genie .....	2676
Tree View Genie .....	2677

Tree View Item Genie .....	2683
Common Controls Genie Library .....	2684
Tab Base Genie .....	2684
Tab Genie .....	2685
Navigation Genie Library .....	2686
Link Down Genie .....	2686
Link Left Genie .....	2687
Link Right Genie .....	2689
Link Up Genie .....	2691
Static Link Left Genie .....	2692
Static Link Right Genie .....	2693
Workspace Genie Library .....	2694
Pane Genie .....	2694
Situational Awareness Controls Project .....	2698
Controls Genie Library .....	2698
Alarm List Genie .....	2699
Alarm List Vertical Genie .....	2700
Generic List Genie .....	2701
Generic List Vertical Genie .....	2705
List Row Genie .....	2708
Common Controls Genie Library .....	2709
Button Base Genie .....	2710
Button Label Genie .....	2711
Check Box Genie .....	2712
Scroll Bar Horizontal Genie .....	2712
Scroll Bar Vertical Genie .....	2713
Toolbar Button Base Genie .....	2714
Toggle Button Genie .....	2715
Priorities Genie Library .....	2716
Disabled Normal Genie .....	2717
Disabled Small Genie .....	2717
Priority 1 Normal Genie .....	2717
Priority 1 Small Genie .....	2718
Priority 2 Normal Genie .....	2718
Priority 2 Small Genie .....	2719
Priority 3 Normal Genie .....	2719
Priority 3 Small Genie .....	2719
Priority 4 Normal Genie .....	2720
Priority 4 Small Genie .....	2720
Situational Awareness Library Project .....	2721
Common Object Elements .....	2722
Status Indicators .....	2723
Output (OP) Bar .....	2727
Equipment Running State Indicators .....	2729
Equipment Running States .....	2731
Equipment Configuration Parameters .....	2733
Show/Hide Settings .....	2735
Alarm Limits .....	2735
Configure PLC Limits with Equipment .....	2736

Meters .....	2739
Meter Orientations .....	2740
Meter Common Elements .....	2742
Control Meters - Common Elements .....	2746
Meter Special Elements .....	2747
Analyzer Meter .....	2749
Deviation Meter .....	2753
Dual Level Meter .....	2758
Flow Meter .....	2762
Level Meter .....	2767
Miscellaneous Meter .....	2772
Pressure Meter .....	2777
Target Meter .....	2781
Temperature Meter .....	2786
Multiple Meter .....	2790
Drive, Belt and Mining Objects .....	2794
Drive Objects .....	2798
Agitator/Rake .....	2798
Blower .....	2802
Bore Pump .....	2806
Brake .....	2809
Compressor .....	2813
Electric Heater .....	2817
Fan .....	2820
Motor .....	2824
Pump .....	2828
Rotary Valve .....	2831
Shell and Tube Heat Exchanger .....	2835
Sump Pump .....	2839
Turbine .....	2842
Belt Objects .....	2846
Apron .....	2846
Belt Feeder .....	2850
Conveyor .....	2854
Mining Objects .....	2857
Crusher .....	2858
Diverter Gate .....	2861
Dust Collector .....	2865
Feeder Gate .....	2869
Magnet .....	2872
Metal Detector .....	2876
Primary Cutter .....	2879
Screen .....	2883
Secondary Cutter .....	2887
Slider Gate .....	2890
Sprayer .....	2894
Vezin Cutter .....	2897
Vibrating Feeder .....	2901
Valves .....	2905

Block Valve .....	2910
Control Valve .....	2913
Damper .....	2916
Hand Valve .....	2919
Valve Orientations .....	2922
Valve Versions .....	2922
Trend Objects .....	2924
Trend Tail .....	2925
Full Trend .....	2926
Four-Pen Trend .....	2926
Additional Objects .....	2928
Alarm Light .....	2928
Dump Light Genie .....	2929
Handswitch Selector .....	2930
Numeric Input .....	2931
Polar Star .....	2933
Status Light Genie .....	2936
Text Input .....	2937
Wind Compass .....	2938
Wind Compass Elements .....	2940
XY Bar Graph .....	2941
Faceplates .....	2943
Create a New Faceplate .....	2944
Meter Faceplates .....	2946
Analog Controller .....	2947
Analog Indicator .....	2950
Drive Faceplates .....	2951
Single and Multiple DOL Drive .....	2952
Single DOL Drive with Forward/Reverse Capability .....	2956
Single and Multiple VSD Drive - Single Direction .....	2960
Single VSD Drive with Forward/Reverse Capability .....	2964
Valve Faceplates .....	2968
Simple Valve Faceplate .....	2969
Complex Valve .....	2971
Core Component Genie Libraries .....	2974
Common Components Genie Library .....	2975
Clock Timer Genie .....	2976
Control Mode Genie .....	2977
Label Genie .....	2978
MEO Genie .....	2979
Meter Value Genie .....	2980
Out Of Service (OOS) Genie .....	2981
Output Bar Genie .....	2982
Output Value Genie .....	2984
PV Value Genie .....	2985
Selection Adorner Genie .....	2986
Selection Adorner Auto Genie .....	2987
Status Indicator Genie .....	2988
Miscellaneous Components Genie Library .....	2989

Direction Arrow Genie .....	2990
Faceplate Components Genie Library .....	2991
Faceplate Button Genie .....	2991
Faceplate Command Button Genie .....	2993
Limit Label Genie .....	2995
Local/Remote Indicator Genie .....	2997
MEO Selector Genie .....	2998
Output Tracker Genie .....	2999
Output Indicator Genie .....	3000
Faceplate Output Value Genie .....	3001
Faceplate Selection Adorner Genie .....	3002
<b>StruxtureWare (SxW) Templates Project .....</b>	<b>3003</b>
The SxW Interface .....	3004
Use SxW Pages and Templates .....	3007
Normal Page Template .....	3008
Blank Page Template .....	3008
Data Browse Page Template .....	3009
Interactive Data Browse List .....	3011
Alarm Page Templates .....	3012
Alarm Page-specific Menus .....	3013
Interactive Alarm List .....	3013
Filtering Alarms using the Equipment Panel .....	3015
Process Analyst Page Templates .....	3016
SPC Trend Page Templates .....	3016
File Page Templates .....	3017
Creating a New Project .....	3017
Create a Privileged User .....	3019
Run the Setup Wizard .....	3019
Use Instant Trending .....	3019
Implement Audible Alarms .....	3020
Create Pages .....	3021
Creating New Pages .....	3021
Creating Custom Menus for SxW Templates .....	3022
Using Environment Variables in SxW Templates .....	3024
<b>Tab Style Templates Project .....</b>	<b>3024</b>
Use Pages and Templates .....	3025
Common Navigation Functionality .....	3030
Common Alarm Page Template Functionality .....	3035
Common Data Browse Template Functionality .....	3037
Alarm Equipment Tree-view Page Templates .....	3038
Common Functionality .....	3039
Action Tab .....	3040
Interactive Alarm List .....	3041
Column Sorting and Filter Status Indicator .....	3043
Filtering Alarms Using the Equipment Tree-view .....	3043
Create a New Project .....	3044
Create a Privileged User .....	3045
Run the Setup Wizard .....	3045
Use Instant Trending .....	3046

Display a Project on Multiple Monitors .....	3046
Implement Audible Alarms .....	3047
Create Pages .....	3048
Create Custom Tab Style Menus .....	3049
Configure Tab Style Popup Menus .....	3049
Using Environment Variables in Tab Style Templates .....	3050
Tab Style Template Reference .....	3050
Tab Style Template Functions .....	3051
TabAlarm_GetAn .....	3051
TabAlarm_GetAckPriv .....	3052
TabAlarm_GetDisablePriv .....	3052
TabAlarmSnd_GetPriv .....	3053
TabAlarmSnd_SoundState .....	3053
TabAlarmSnd_Silence .....	3054
TabAlarmSnd_SilenceOnAck .....	3054
TabAlarmSnd_ShowContextMenu .....	3055
DspFileGetAn .....	3055
DspRtfFileGetAn .....	3055
DspRtfFileGetName .....	3056
DspRtfFileSetName .....	3057
PageDspConfigMenu .....	3058
PageDspLoginMenu .....	3059
PageGetConfigMenuHnd .....	3059
PageHomeGetName .....	3060
PageParent .....	3060
PageParentGetName .....	3061
PagePrint .....	3061
PagePrintGetName .....	3062
<b>Library Controls Include Project .....</b>	<b>3062</b>
Use the Library Controls .....	3065
Alarm Table Library Control .....	3065
Use the Alarm Table Genie .....	3068
Calendar Library Control .....	3070
Data Browse Table Library Control .....	3072
Equipment Tree Library Control .....	3075
PageTabs Library Control .....	3079
Scroll Bar Horizontal Library Control .....	3082
Scroll Bar Vertical Library Control .....	3083
Slider Library Control .....	3084
SQL Table Library Control .....	3085
Tabs Library Control .....	3088
Use the Tabs Genie .....	3090
Table Library Control .....	3092
Use the Table Genie .....	3095
Table Row Library Control .....	3098
Tag Table Library Control .....	3098
Using the Tag Table Genie .....	3101
Tree Library Control .....	3104
Use the Tree Genie .....	3108

<b>Include Project Specifications .....</b>	<b>3110</b>
Predefined Commands .....	3111
System Keyboard Commands Database .....	3111
Predefined Keyboard Keys .....	3111
Predefined Character Sets .....	3113
Predefined Fonts .....	3114
Predefined Devices .....	3116
Predefined Cicode Files .....	3116
Predefined Color Names and Codes .....	3117
Predefined Keyboard Key Codes .....	3118
Predefined Labels .....	3126
 <b>Extensibility .....</b>	 <b>3132</b>
<b>CtAPI .....</b>	<b>3133</b>
I/O Point Count .....	3134
CtAPI Synchronous Operation .....	3135
Reading Data Using the CtAPI Functions .....	3136
Secure the Connection from a Remote CtAPI Application .....	3137
Error Codes .....	3138
Debug Tracing .....	3139
CtAPI Functions Reference .....	3139
ctCancelIO .....	3142
ctCiCode .....	3143
ctClientCreate .....	3145
ctClientDestroy .....	3146
ctClose .....	3146
ctCloseEx .....	3147
ctEngToRaw .....	3148
ctFindClose .....	3149
ctFindFirst .....	3149
ctFindFirstEx .....	3154
ctFindNext .....	3158
ctFindNumRecords .....	3159
ctFindPrev .....	3159
ctFindScroll .....	3160
ctGetOverlappedResult .....	3161
ctGetProperty .....	3163
ctHasOverlappedIoCompleted .....	3165
ctListAdd .....	3166
ctListAddEx .....	3167
ctListData .....	3168
ctListDelete .....	3169
ctListEvent .....	3170
ctListFree .....	3171
ctListNew .....	3172
ctListItem .....	3173
ctListRead .....	3175
ctListWrite .....	3177

ctOpen .....	3177
ctOpenEx .....	3180
ctRawToEng .....	3181
ctSetManagedBinDirectory .....	3182
ctTagGetProperty .....	3183
ctTagRead .....	3185
ctTagReadEx .....	3186
ctTagWrite .....	3188
ctTagWriteEx .....	3189
AlmQuery .....	3190
TrnQuery .....	3192
Display Mode .....	3193
Returned Data .....	3195
CtAPIAlarm .....	3196
CtAPITrend .....	3197
<b>Graphics Builder Automation Interface .....</b>	<b>3198</b>
Error Handling .....	3199
Automation Events .....	3200
Function Categories .....	3201
Alarm Indicator Functions .....	3202
AlarmIndicatorGetBorderIsInside .....	3204
AlarmIndicatorGetBorderPadding .....	3204
AlarmIndicatorGetBorderWidth .....	3205
AlarmIndicatorGetEnabled .....	3205
AlarmIndicatorGetEquipHierarchy .....	3206
AlarmIndicatorGetExpr .....	3207
AlarmIndicatorGetFlagEnabled .....	3207
AlarmIndicatorGetFlagLocation .....	3208
AlarmIndicatorGetFrameColor .....	3209
AlarmIndicatorGetInclude .....	3209
AlarmIndicatorSetBorderIsInside .....	3210
AlarmIndicatorSetBorderPadding .....	3211
AlarmIndicatorSetBorderWidth .....	3211
AlarmIndicatorSetEnabled .....	3212
AlarmIndicatorSetEquipHierarchy .....	3213
AlarmIndicatorSetExpr .....	3214
AlarmIndicatorSetFlagEnabled .....	3215
AlarmIndicatorSetFlagLocation .....	3216
AlarmIndicatorSetFrameColor .....	3216
AlarmIndicatorSetInclude .....	3217
Arrange and Position Functions .....	3218
PositionAt .....	3219
PositionBringForwards .....	3220
PositionBringToFront .....	3220
PositionMirrorHorizontal .....	3221
PositionMirrorVertical .....	3221
PositionRotate .....	3222
PositionSendBackwards .....	3222
PositionSendToBack .....	3223

Composite Genie Functions .....	3224
CompositeGeniePlace .....	3224
PageSelectFirstCompositeGenie .....	3225
PageSelectNextCompositeGenie .....	3225
Dynamic Properties Functions .....	3226
PropertiesAccessDisableGet .....	3229
PropertiesAccessDisablePut .....	3230
PropertiesAccessGeneralGet .....	3230
PropertiesAccessGeneralPut .....	3231
PropertiesAttributeNameGetEx .....	3232
PropertiesAttributeNamePutEx .....	3232
PropertiesButtonGet .....	3233
PropertiesButtonGetEx .....	3234
PropertiesButtonPut .....	3235
PropertiesCicodeObjectGet .....	3235
PropertiesCicodeObjectPut .....	3236
PropertiesDisplayValueGet .....	3237
PropertiesDisplayValuePut .....	3238
PropertiesDisplayValueTextGet .....	3239
PropertiesDisplayValueTextPut .....	3239
PropertiesFillColourColourGet .....	3240
PropertiesFillColourColourGetEx .....	3241
PropertiesFillColourColourPut .....	3242
PropertiesFillColourColourPutEx .....	3243
PropertiesFillColourGet .....	3244
PropertiesFillColourPut .....	3245
PropertiesFillLevelGet .....	3246
PropertiesFillLevelGetEx .....	3247
PropertiesFillLevelPut .....	3248
PropertiesFillLevelPutEx .....	3249
PropertiesInputKeyboardGet .....	3250
PropertiesInputKeyboardPut .....	3251
PropertiesInputTouchGet .....	3251
PropertiesInputTouchPut .....	3252
PropertiesShowDialog .....	3253
PropertiesSymbolSetGet .....	3253
PropertiesSymbolSetPut .....	3254
PropertiesSymbolSetSymbolGet .....	3255
PropertiesSymbolSetSymbolPut .....	3256
PropertiesTransCentreOffsetExpressGet .....	3257
PropertiesTransCentreOffsetExpressPut .....	3258
PropertiesTransformationGet .....	3259
PropertiesTransformationPut .....	3260
PropertiesTrendGet .....	3262
PropertiesTrendGetEx .....	3263
PropertiesTrendPut .....	3265
PropertiesTrendPutEx .....	3266
PropertyVisibility .....	3268
Event Functions .....	3268

BrokenLink .....	3269
PasteGenie .....	3270
PasteSymbol .....	3270
ProjectChange .....	3270
Selection .....	3271
SwapObject .....	3271
Library Object Functions .....	3271
LibraryFirstObject .....	3273
LibraryNextObject .....	3274
LibraryObjectFirstProperty .....	3274
LibraryObjectFirstPropertyEx .....	3275
LibraryObjectHotspotGet .....	3276
LibraryObjectHotspotPut .....	3276
LibraryObjectName .....	3277
LibraryObjectNextProperty .....	3278
LibraryObjectNextPropertyEx .....	3278
LibraryObjectPlace .....	3279
LibraryObjectPlaceEx .....	3280
LibraryObjectPutProperty .....	3281
LibraryShowPasteDialog .....	3282
LibSelectionHooksEnabled .....	3282
Metadata Functions .....	3283
PropertiesAddMetadata .....	3283
PropertiesDeleteMetadata .....	3284
PropertiesMetadataName .....	3285
PropertiesMetadataValue .....	3285
PropertiesSelectFirstMetadata .....	3285
PropertiesSelectMetadataByName .....	3286
PropertiesSelectNextMetadata .....	3287
Miscellaneous Functions .....	3287
BrokenLinkCancelEnabled .....	3288
ClipboardCopy .....	3289
ClipboardCut .....	3289
ClipboardPaste .....	3289
ConvertToBitmap .....	3289
GetOffBitmap .....	3290
GetOnBitmap .....	3290
Quit .....	3291
SelectionEventEnabled .....	3291
UnLockObject .....	3291
Object Drawing and Property Functions .....	3292
Attribute3dEffects .....	3295
Attribute3dEffectDepth .....	3296
AttributeAN .....	3297
AttributeBaseCoordinates .....	3298
AttributeClass .....	3299
AttributeCornerRadius .....	3299
AttributeEllipseStyle .....	3300
AttributeEndAngle .....	3300

AttributeExtentX .....	3301
AttributeExtentY .....	3302
AttributeFillColour .....	3302
AttributeFillOffColourEx .....	3303
AttributeFillOnColourEx .....	3304
AttributeGradientMode .....	3305
AttributeGradientOffColour .....	3305
AttributeGradientOnColour .....	3306
AttributeHiLightColour .....	3306
AttributeHiLightOffColourEx .....	3307
AttributeHiLightOnColourEx .....	3308
AttributeLineColour .....	3309
AttributeLineOffColourEx .....	3310
AttributeLineOnColourEx .....	3311
AttributeLineStyle .....	3311
AttributeLineWidth .....	3312
AttributeLoLightColour .....	3313
AttributeLoLightOffColourEx .....	3314
AttributeLoLightOnColourEx .....	3315
AttributeNodeCoordinatesFirst .....	3316
AttributeNodeCoordinatesNext .....	3316
AttributePolygonOpen .....	3317
AttributeRectangleStyle .....	3318
AttributeSetFill .....	3318
AttributeShadowColour .....	3319
AttributeShadowOffColourEx .....	3320
AttributeShadowOnColourEx .....	3321
AttributeStartAngle .....	3322
AttributeTransformationMatrixGet .....	3322
AttributeTransformationMatrixPut .....	3323
AttributeX .....	3324
AttributeY .....	3325
DrawButton .....	3325
DrawCicodeObject .....	3326
DrawEllipse .....	3327
DrawLine .....	3328
DrawNumber .....	3328
DrawPipeEnd .....	3329
DrawPipeSection .....	3329
DrawPipeStart .....	3330
DrawPolygonEnd .....	3331
DrawPolygonLine .....	3331
DrawPolygonStart .....	3332
DrawRectangle .....	3333
DrawSymbolSet .....	3333
DrawText .....	3334
DrawTrend .....	3335
Options Functions .....	3335
OptionDisplayPropertiesOnNew .....	3336

OptionSnapToGrid .....	3337
OptionSnapToGuidelines .....	3337
Page Functions .....	3338
PageActiveWindowHandle .....	3340
PageClose .....	3340
PageConvertWindowCoordinates.....	3341
PageDelete .....	3341
PageDeleteEx .....	3342
PageDeleteObject .....	3343
PageDeleteTemplate .....	3344
PageGroupSelectedObjects.....	3344
PageImport .....	3345
PageKeyboardCommandsPut .....	3345
PageNew .....	3346
PageNewEx .....	3347
PageNewLibrary .....	3348
PageNewTemplate .....	3349
PageOpen .....	3349
PageOpenEx .....	3350
PageOpenTemplate .....	3351
PagePrint.....	3352
PageSave .....	3352
PageSaveAs .....	3353
PageSaveAsEx .....	3353
PageSelect .....	3354
PageSelectFirst .....	3355
PageSelectFirstObject .....	3355
PageSelectFirstObjectEx .....	3356
PageSelectFirstObjectInGenie .....	3356
PageSelectFirstObjectInGroup .....	3357
PageSelectNext.....	3357
PageSelectNextObject .....	3358
PageSelectNextObjectEx .....	3358
PageSelectNextObjectInGenie .....	3359
PageSelectNextObjectInGroup .....	3359
PageSelectObject .....	3360
PageSelectObjectAdd .....	3360
PageTemplateSelectFirstObject .....	3361
PageTemplateSelectNextObject .....	3361
PageThumbnailToClipboard .....	3362
PageUngroupSelectedObject .....	3362
PageUpdated .....	3363
Page Properties Functions .....	3363
PageAddAssociation.....	3366
PageAppearanceGet .....	3366
PageAppearanceGetEx .....	3367
PageArea .....	3368
PageAssociationDefault .....	3369
PageAssociationDescription .....	3370

PageAssociationName .....	3370
PageAssociationValueOnError .....	3370
PageClusterInherit .....	3371
PageClusterName .....	3372
PageDeleteAssociation .....	3372
PageDescription .....	3373
PageEnvironmentAdd .....	3374
PageEnvironmentFirst .....	3374
PageEnvironmentNext .....	3375
PageEnvironmentRemove .....	3376
PageEventGet .....	3376
PageEventGetEx .....	3378
PageEventPut .....	3379
PageEventPutEx .....	3380
PageLogDevice .....	3382
PageName .....	3382
PageNext .....	3383
PagePrevious .....	3384
PageScanTime .....	3385
PageSelectAssociationByName .....	3385
PageSelectFirstAssociation .....	3386
PageSelectNextAssociation .....	3387
PageTitle .....	3387
<b>Project Functions .....</b>	<b>3388</b>
ProjectCompile .....	3389
ProjectFirst .....	3390
ProjectFirstInclude .....	3390
ProjectNext .....	3391
ProjectNextInclude .....	3392
ProjectPackDatabase .....	3392
ProjectPackLibraries .....	3393
ProjectSelect .....	3393
ProjectSelected .....	3394
ProjectUpdatePages .....	3395
ProjectUpgrade .....	3395
ProjectUpgradeAll .....	3396
<b>Specific Functions .....</b>	<b>3397</b>
Visible .....	3397
<b>Text Property Functions .....</b>	<b>3398</b>
AttributeText .....	3399
AttributeTextColour .....	3399
AttributeTextOffColourEx .....	3400
AttributeTextOnColourEx .....	3401
AttributeTextFont .....	3402
AttributeTextFontSize .....	3402
AttributeTextJustification .....	3403
AttributeTextStyle .....	3404
<b>Using External Databases .....</b>	<b>3405</b>
dBASE Databases .....	3405

SQL Databases .....	3406
Using Structured Query Language .....	3406
Limitations When Using Legacy SQL Functions .....	3408
Database Connection Objects .....	3408
Basic SQL Scenarios .....	3409
Connecting to an SQL Database .....	3410
Working with Recordsets .....	3410
Parameterized Queries .....	3411
Executing Legacy SQL Commands .....	3411
Using a Transaction .....	3412
Expressing Dates and Times in SQL .....	3413
Database Independent Date-Time Syntax .....	3413
Conversions between Database and Cicode Types .....	3413
<b>Exchanging Data with Other Applications.....</b>	<b>3415</b>
Using DDE (Dynamic Data Exchange) .....	3416
DDE Conversations and Client Syntax .....	3416
Setting up DDE Conversations .....	3417
DDE Function Types .....	3418
Exchanging Plant SCADA Data via DDE .....	3419
Connecting to the Tag Database using DDE .....	3419
Posting Select Data using DDE .....	3420
Writing Values to a DDE Application .....	3421
Reading Values from a DDE Application .....	3421
Using DDE with Microsoft Office Applications .....	3422
Long File Names in DDE .....	3422
Microsoft Office Security .....	3422
Network DDE .....	3423
Starting Network DDE Services .....	3423
Setting up Network DDE Shares .....	3424
DDE Shares .....	3425
Using DDE Trusted Shares .....	3425
Using Network DDE .....	3426
Connecting to a Network DDE Shared Application .....	3426
Using ODBC Drivers .....	3427
ODBC Compatibility .....	3430
Using Plant SCADA as an ODBC Server .....	3431
Reading Data from an Access Table with ODBC .....	3433
Appending Data with ODBC .....	3434
Editing Data with ODBC .....	3434
Deleting Rows from an Access Table .....	3435
Calling Action Queries with ODBC .....	3435
Parameter Queries .....	3435
Access and Cicode Date/Time Conversions .....	3437
<b>Integration with Historian .....</b>	<b>3437</b>
Preparing Variables for Historian Integration .....	3438
<b>Using Microsoft Excel to Edit DBF Tables .....</b>	<b>3438</b>
Project DBF AddIn Functionality .....	3440
<b>Using an OPC AE Server .....</b>	<b>3441</b>
Using a Third-party OPC Client to Access Alarm Server Data .....	3442

<b>Using an OPC DA Server .....</b>	<b>3442</b>
Configure an OPC DA Server .....	3443
OPC DA Server Properties .....	3443
OPC DA Server DCOM Settings .....	3445
Operating the OPC DA Server at Runtime .....	3447
Starting the OPC DA Server on a Client Connection .....	3448
OPC Item Browsing .....	3448
OPC DA Server Diagnostics .....	3449
OPC DA Server Reference Information .....	3450
Supported OPC DA Server Interfaces .....	3450
Client Request Data Type Conversions .....	3451
OPC Client Read Request Conversions .....	3451
OPC Client Write Request Conversions .....	3453
OPC DA Server Mandatory Properties .....	3454
 <b>Plant SCADA Access Anywhere .....</b>	 <b>3457</b>
<b>Access Anywhere Quick Start Guide .....</b>	<b>3457</b>
Licensing .....	3458
Installing and Configuring the Access Anywhere Server .....	3459
Configuring Windows Firewall .....	3460
Configuring Remote Desktop Service .....	3461
Configuring User Access to Plant SCADA Clients .....	3462
Installing and Configuring the Access Anywhere Secure Gateway .....	3462
Connecting to a Plant SCADA Access Anywhere Server .....	3464
Connecting Access Anywhere to a Plant SCADA Client .....	3466
<b>Access Anywhere Installation and Configuration Guide .....</b>	<b>3467</b>
Overview .....	3468
RDP Compression and Acceleration .....	3470
Before You Install .....	3471
Prerequisites .....	3471
Configuring Firewalls .....	3471
Binding Service to Every Network Interface .....	3474
Installation .....	3474
Installing Plant SCADA Access Anywhere Server .....	3475
Unattended Silent Installation .....	3479
Repairing an Installation .....	3481
Uninstalling Plant SCADA Access Anywhere .....	3483
Upgrading to a New Version .....	3485
Migrate an Existing Configuration from an Earlier Version .....	3488
Plant SCADA Access Anywhere Web Component .....	3489
WebSocket Connections .....	3489
Configuring the Plant SCADA Access Anywhere Server .....	3493
Configuring Remote Desktop Services .....	3493
Configuring User Access .....	3494
The Configuration Console .....	3496
General Tab .....	3496
Performance Tab .....	3497
Communication Tab .....	3497

Acceleration Tab .....	3498
Security .....	3498
Logging Tab .....	3499
Advanced .....	3500
Configuring Mobile Devices .....	3500
Supported Browsers .....	3500
Logging On to Plant SCADA Access Anywhere .....	3501
Automatic Display Resize .....	3504
Using Gestures on Client Portable Devices .....	3505
Advanced Configuration .....	3507
Static Configuration of Config.js .....	3507
Settings Table .....	3508
Defining Configuration Groups .....	3511
Settings Precedence .....	3511
Passing Credentials using Form POST .....	3512
Embedding Plant SCADA Access Anywhere in an iFrame .....	3512
SSL VPN Configuration .....	3513
Web Proxy with Juniper 7.4 (or later) .....	3513
Web Proxy with Older Juniper Versions .....	3513
Network Connect .....	3514
<b>Access Anywhere Secure Gateway Installation Guide .....</b>	<b>3514</b>
Introduction .....	3515
Architecture .....	3515
Installation .....	3516
Installation Prerequisites .....	3516
Installing the Secure Gateway .....	3517
Completing the Installation .....	3523
Repairing an Installation .....	3524
Uninstalling the Secure Gateway .....	3526
Migrate an Existing Configuration from an Earlier Version .....	3529
Post-Installation .....	3532
Configuring the Secure Gateway .....	3533
Dashboard .....	3535
Security .....	3536
Web Server .....	3537
Log Settings .....	3537
Built-In Authentication Server .....	3538
Download - Obtaining Log Files .....	3539
Admin .....	3539
SSL Certificates .....	3540
Configuring the SSL Certificates .....	3540
Built-In Web Server .....	3544
Connecting to the Web Server .....	3544
HTTP Redirect .....	3544
Disabling HTTP Filtering .....	3545
Advanced Configuration .....	3545
<b>Access Anywhere Web Client User Guide .....</b>	<b>3546</b>
Supported Browsers .....	3546
Viewing Plant SCADA with Plant SCADA Access Anywhere .....	3547

Logging on to Plant SCADA Access Anywhere .....	3547
Securely Connecting to Plant SCADA Access Anywhere .....	3551
Obtaining a Plant SCADA Runtime License .....	3551
Checking Connectivity .....	3552
Viewing a Plant SCADA Access Anywhere Session .....	3552
Plant SCADA Access Anywhere Toolbar .....	3553
Ending a Plant SCADA Access Anywhere Session .....	3554
Using Plant SCADA Access Anywhere on Portable Devices .....	3555
Tablet and Smartphones .....	3555
Gestures Support .....	3556
Automatic Display Resize .....	3557
Running Access Anywhere on Different Web Browsers .....	3557
Android Operating System .....	3557
Google Chromebooks .....	3558
iOS .....	3559
<b>AVEVA Cloud Services .....</b>	<b>3561</b>
<b>Plant SCADA and AVEVA Integration Studio .....</b>	<b>3561</b>
Access AVEVA Integration Studio .....	3562
Use Plant SCADA with AVEVA Integration Studio .....	3563
Add Plant SCADA Nodes to a Project Template .....	3565
Security Implications for Plant SCADA Simulations .....	3567
Configure Server Redundancy .....	3568
Known Issues .....	3569
<b>Reference .....</b>	<b>3570</b>
<b>Cicode Reference .....</b>	<b>3570</b>
About Cicode .....	3570
Using Cicode Files .....	3571
Restricted Cicode Keywords .....	3571
Using Cicode Commands .....	3572
Setting Variables .....	3572
Performing Calculations .....	3573
Using Multiple Command Statements .....	3574
Using Include (Text) Files .....	3574
Getting Runtime Operator Input .....	3575
Using Cicode Expressions .....	3576
Using Cicode Functions .....	3578
Calling Functions from Commands and Expressions .....	3578
Triggering Functions via Runtime Operator Input .....	3579
Evaluating Functions .....	3579
Combining Functions with Other Statements .....	3579
Passing Data to Functions (Arguments) .....	3580
Using String Arguments .....	3580
String Assignment .....	3581
Using the Caret Escape Sequence Character .....	3581
Using Multiple Arguments .....	3581

Using Numeric Arguments .....	3582
Using Variable Arguments .....	3582
Using Operator Input in Functions .....	3582
Returning Data from Functions .....	3583
Referencing an Object Using a Name at Runtime .....	3583
Working with Commonly Used Functions .....	3587
Writing Functions .....	3589
Cicode Function Structure .....	3590
Writing Groups of Functions .....	3591
Cicode Function Libraries .....	3591
Creating a Function Outline .....	3591
Pseudocode .....	3592
Considering Tag Value Quality in Cicode .....	3593
Using Comments in Cicode .....	3595
Using Comments for Debugging Functions .....	3596
Tag Reference/TagReadEx Behavior in Cicode Expressions .....	3596
Following Cicode Syntax .....	3597
Cicode Function Syntax .....	3598
End of Line Markers .....	3599
Function Scope .....	3600
Declaring the Return Data Type .....	3600
Declaring Functions .....	3601
Naming Functions .....	3602
Function Argument Structure .....	3603
Declaring Argument Data Type .....	3604
Naming Arguments .....	3605
Setting Default Values for Arguments .....	3606
Returning Values from Functions .....	3607
Using Variables .....	3608
Declaring Variable Properties .....	3609
Declaring the Variable Data Type .....	3609
Naming Variables .....	3611
Setting Default Variable Values .....	3611
Using Variable Scope .....	3612
Using Database Variables .....	3613
Using Arrays .....	3614
Using Cicode Macros .....	3617
IFDEF .....	3617
IFDEFAdvAlm .....	3619
IFDEFAnaAlm .....	3620
IFDEFDigAlm .....	3621
IFDEFHresAlm .....	3621
Macro Arguments .....	3622
Converting and Formatting Cicode Variables .....	3623
Working with Operators .....	3627
Using Mathematical Operators .....	3627
Using Bit Operators .....	3628
Using Relational Operators .....	3629
Using Logical Operators .....	3630

Order of Precedence of Operators .....	3630
Working with Conditional Executors .....	3631
Setting IF ... THEN Conditions .....	3631
Using FOR ... DO Loops .....	3632
Using WHILE ... DO Conditional Loops .....	3633
Nested Loops .....	3633
Using the SELECT CASE statement .....	3634
Performing Advanced Tasks .....	3635
Cicode Editor .....	3637
Creating Cicode Files .....	3638
Creating Functions .....	3639
Saving Files .....	3639
Opening Cicode Files .....	3639
Deleting Cicode Files .....	3640
Finding Text in Cicode Files .....	3640
Compiling Cicode Files .....	3641
Viewing Errors Detected by the Cicode Compiler .....	3641
Find User Function .....	3641
Cicode Editor Windows .....	3642
Breakpoint Window .....	3642
Output Window .....	3643
Global Variable Window .....	3643
Stack Window .....	3644
Thread Window .....	3645
Compile Errors Window .....	3645
Citect VBA Watch Window .....	3646
Files Window .....	3646
Cicode Editor Options .....	3647
Windows and Bars Tab .....	3647
Options Tab .....	3648
Docking Windows and Toolbars .....	3650
Debugging Cicode .....	3651
Cicode Errors .....	3653
Hardware/Cicode Errors .....	3653
Cicode and General Errors .....	3654
MAPI Errors .....	3682
Using Cicode Programming Standards .....	3684
Variable Declaration Standards .....	3685
Variable Scope Standards .....	3686
Variable Naming Standards .....	3686
Standards for Constants, Variable Tags, and Labels .....	3687
Constants .....	3687
Variable Tags .....	3688
Labels .....	3688
Formatting Simple Declarations .....	3688
Formatting Executable Statements .....	3689
Formatting Expressions .....	3690
Cicode Comments .....	3691
Formatting Functions .....	3691

Format Templates .....	3692
Function Naming Standards .....	3693
Source File Headers .....	3694
Function Headers .....	3694
Modular Programming .....	3695
Cohesion .....	3696
Coupling .....	3697
Defensive Programming .....	3697
Function Error Handling .....	3698
Debug Error Trapping .....	3700
DebugMsg Function .....	3701
Assert Function .....	3701
Cicode Function Categories .....	3702
Accumulator Functions .....	3703
AccControl .....	3703
AccumBrowseClose .....	3704
AccumBrowseFirst .....	3705
AccumBrowseGetField .....	3705
AccumBrowseNext .....	3706
AccumBrowseNumRecords .....	3707
AccumBrowseOpen .....	3708
AccumBrowsePrev .....	3709
ActiveX Functions .....	3709
_ObjectCallMethod .....	3710
_ObjectGetProperty .....	3711
_ObjectSetProperty .....	3712
AnByName .....	3713
CreateControlObject .....	3713
CreateObject .....	3715
ObjectAssociateEvents .....	3717
ObjectAssociatePropertyWithTag .....	3718
ObjectByName .....	3719
ObjectHasInterface .....	3720
ObjectIsValid .....	3721
ObjectToStr .....	3721
Alarm Functions .....	3722
AlarmAck .....	3728
AlarmAckRec .....	3730
AlarmAckTag .....	3731
AlarmActive .....	3732
AlarmCatGetFormat .....	3733
AlarmClear .....	3734
AlarmClearRec .....	3736
AlarmClearTag .....	3736
AlarmComment .....	3737
AlarmCommentRecID .....	3738
AlarmCount .....	3739
AlarmCountEquipment .....	3742
AlarmCountList .....	3745

AlarmDelete .....	3745
AlarmDisable .....	3747
AlarmDisableRec .....	3749
AlarmDisableTag .....	3751
AlarmDsp .....	3752
AlarmDspClusterAdd .....	3753
AlarmDspClusterInUse .....	3754
AlarmDspClusterRemove .....	3755
AlarmDspLast .....	3755
AlarmDspNext .....	3757
AlarmDspPrev .....	3758
AlarmEnable .....	3759
AlarmEnableRec .....	3760
AlarmEnableTag .....	3762
AlarmEventQue .....	3763
AlarmFirstCatRec .....	3764
AlarmFirstPriRec .....	3765
AlarmFirstTagRec .....	3767
AlarmGetDelay .....	3768
AlarmGetDelayRec .....	3769
AlarmGetDsp .....	3770
AlarmGetInfo .....	3771
AlarmGetFieldRec .....	3773
AlarmGetOrderByKey .....	3775
AlarmGetThreshold .....	3775
AlarmGetThresholdRec .....	3776
AlarmHelp .....	3777
AlarmHighestPriority .....	3778
AlarmListCreate .....	3780
AlarmListDestroy .....	3782
AlarmListDisplay .....	3782
AlarmListFill .....	3783
AlarmNextCatRec .....	3783
AlarmNextPriRec .....	3785
AlarmNextTagRec .....	3786
AlarmNotifyVarChange .....	3788
AlarmQueryFirstRec .....	3789
AlarmQueryNextRec .....	3790
AlarmSetDelay .....	3792
AlarmSetDelayRec .....	3793
AlarmSetInfo .....	3794
AlarmSetThreshold .....	3798
AlarmSetThresholdRec .....	3799
AlarmSplit .....	3801
AlarmSumAppend .....	3801
AlarmSumCommit .....	3803
AlarmSumDelete .....	3804
AlarmSumFind .....	3806
AlarmSumFirst .....	3808

AlarmSumGet .....	3809
AlarmSumLast .....	3810
AlarmSumNext .....	3811
AlarmSumPrev .....	3813
AlarmSumSet .....	3814
AlarmSumSplit .....	3815
AlarmSumType .....	3817
AlarmTagFromEquipment .....	3818
AlmBrowseAck .....	3818
AlmBrowseClose .....	3819
AlmBrowseDisable .....	3820
AlmBrowseEnable .....	3820
AlmBrowseFirst .....	3821
AlmBrowseGetField .....	3822
AlmBrowseNext .....	3823
AlmBrowseNumRecords .....	3824
AlmBrowseOpen .....	3824
AlmBrowsePrev .....	3826
AlmSummaryAck .....	3827
AlmSummaryCommit .....	3827
AlmSummaryClear .....	3828
AlmSummaryClose .....	3829
AlmSummaryDelete .....	3829
AlmSummaryDeleteAll .....	3830
AlmSummaryDisable .....	3831
AlmSummaryEnable .....	3832
AlmSummaryFirst .....	3833
AlmSummaryGetField .....	3833
AlmSummaryLast .....	3835
AlmSummaryNext .....	3835
AlmSummaryNumRecords .....	3836
AlmSummaryOpen .....	3837
AlmSummaryPrev .....	3840
AlmSummarySetFieldValue .....	3841
AlmTagsAck .....	3842
AlmTagsClear .....	3843
AlmTagsClose .....	3843
AlmTagsDisable .....	3844
AlmTagsEnable .....	3845
AlmTagsFirst .....	3845
AlmTagsGetField .....	3846
AlmTagsNext .....	3847
AlmTagsNumRecords .....	3848
AlmTagsOpen .....	3849
AlmTagsPrev .....	3850
HwAlarmQue .....	3851
Alarm Filter Functions .....	3852
AlarmFilterClose .....	3853
AlarmFilterEditAppend .....	3854

AlarmFilterEditAppendEquipment .....	3855
AlarmFilterEditClose .....	3856
AlarmFilterEditCommit .....	3857
AlarmFilterEditFirst .....	3857
AlarmFilterEditLast .....	3858
AlarmFilterEditNext .....	3859
AlarmFilterEditOpen .....	3860
AlarmFilterEditPrev .....	3861
AlarmFilterEditSet .....	3862
AlarmFilterEditHasField .....	3863
AlarmFilterForm .....	3864
AlarmFilterOpen .....	3865
AlarmGetFilterName .....	3866
AlarmResetQuery .....	3867
LibAlarmFilterForm .....	3867
Array Functions .....	3869
ArrayCopy .....	3870
ArrayCreate .....	3871
ArrayCreateByAn .....	3872
ArrayDestroy .....	3874
ArrayDestroyByAn .....	3875
ArrayExists .....	3875
ArrayExistsByAn .....	3876
ArrayFillFromAlarmDataByAn .....	3877
ArrayGetArrayByAn .....	3877
ArrayGetInfo .....	3878
ArrayGetInt .....	3879
ArrayGetIntByAn .....	3880
ArrayGetMapName .....	3881
ArrayGetMapNameByAn .....	3881
ArrayGetString .....	3882
ArrayGetStringByAn .....	3883
ArrayIsDirty .....	3884
ArraySetInt .....	3885
ArraySetIntByAn .....	3886
ArraySetIsDirty .....	3887
ArraySetString .....	3887
ArraySetStringByAn .....	3888
ArraySwap .....	3889
DspArrayByAn .....	3890
Clipboard Functions .....	3891
ClipCopy .....	3891
ClipPaste .....	3892
ClipReadLn .....	3892
ClipSetMode .....	3893
ClipWriteLn .....	3894
Cluster Functions .....	3894
ClusterActivate .....	3895
ClusterDeactivate .....	3896

ClusterFirst .....	3896
ClusterGetName .....	3897
ClusterIsActive .....	3897
ClusterNext .....	3898
ClusterServerTypes .....	3899
ClusterSetName .....	3899
ClusterStatus .....	3900
ClusterSwapActive .....	3901
Color Functions .....	3902
CitectColourToPackedRGB .....	3902
GetBlueValue .....	3903
GetGreenValue .....	3903
GetRedValue .....	3904
MakeColour .....	3904
PackedRGB .....	3905
PackedRGBToCitectColour .....	3906
Communication Functions .....	3906
ComClose .....	3907
ComOpen .....	3907
ComRead .....	3909
ComReset .....	3911
ComWrite .....	3911
SerialKey .....	3913
Dynamic Data Exchange Functions .....	3913
DDEExec .....	3914
DDEhExecute .....	3915
DDEhGetLastError .....	3916
DDEhInitiate .....	3917
DDEhPoke .....	3918
DDEhReadLn .....	3919
DDEhRequest .....	3920
DDEhSetMode .....	3921
DDEhTerminate .....	3921
DDEhWriteLn .....	3922
DDEPost .....	3923
DDERead .....	3924
DDEWrite .....	3925
Device Functions .....	3926
DevAppend .....	3928
DevClose .....	3928
DevControl .....	3929
DevCurr .....	3930
DevDelete .....	3931
DevDisable .....	3932
DevEOF .....	3933
DevFind .....	3933
DevFirst .....	3934
DevFlush .....	3935
DevGetField .....	3936

DevHistory .....	3937
DevInfo .....	3937
DevModify .....	3939
DevNext .....	3940
DevOpen .....	3941
DevPrev .....	3943
DevPrint .....	3943
DevRead .....	3944
DevReadLn .....	3945
DevRecNo .....	3946
DevSeek .....	3946
DevSetField .....	3947
DevSize .....	3948
DevWrite .....	3949
DevWriteLn .....	3950
DevZap .....	3950
Print .....	3951
PrintFont .....	3952
PrintLn .....	3953
Display Functions .....	3954
DspAnCreateControlObject .....	3959
DspAnFree .....	3960
DspAnGetArea .....	3960
DspAnGetMetadata .....	3961
DspAnGetMetadataAt .....	3962
DspAnGetPos .....	3963
DspAnGetPrivilege .....	3963
DspAnInfo .....	3964
DspAnInRgn .....	3966
DspAnMove .....	3966
DspAnMoveRel .....	3967
DspAnNew .....	3968
DspAnNewRel .....	3969
DspAnSetName .....	3969
DspAnSetMetadata .....	3970
DspAnSetMetadataAt .....	3971
DspBar .....	3972
DspBmp .....	3973
DspButton .....	3974
DspButtonFn .....	3975
DspChart .....	3976
DspClearClip .....	3977
DspCol .....	3978
DspDel .....	3979
DspDelayRenderBegin .....	3979
DspDelayRenderEnd .....	3980
DspDirty .....	3981
DspError .....	3982
DspFile .....	3983

DspFileGetInfo .....	3984
DspFileGetName .....	3985
DspFileScroll .....	3985
DspFileSetName .....	3986
DspFont .....	3987
DspFontHnd .....	3989
DspFullScreen .....	3989
DspGetAnBottom .....	3991
DspGetAnCur .....	3991
DspGetAnExtent .....	3992
DspGetAnRawExtent .....	3993
DspGetAnFirst .....	3994
DspGetAnFromName .....	3994
DspGetAnFromNameRelative .....	3995
DspGetAnFromPoint .....	3996
DspGetAnHeight .....	3997
DspGetAnLeft .....	3998
DspGetAnNext .....	3998
DspGetAnRight .....	3999
DspGetAnTop .....	4000
DspGetAnWidth .....	4000
DspGetEnv .....	4001
DspGetMetadataFromName .....	4001
DspGetMetadataFromNameRelative .....	4002
DspGetMouse .....	4003
DspGetMouseOver .....	4004
DspGetNameFromAn .....	4004
DspGetNearestAn .....	4005
DspGetParentAn .....	4006
DspGetSlider .....	4006
DspGetTip .....	4007
DspGrayButton .....	4008
DspInfo .....	4009
DspInfoDestroy .....	4011
DspInfoField .....	4011
DspInfoNew .....	4013
DspInfoValid .....	4014
DspIsButtonGray .....	4015
DspKernel .....	4015
DspMarkerMove .....	4017
DspMarkerNew .....	4017
DspMCI .....	4019
DspPlaySound .....	4019
DspPopupConfigMenu .....	4021
DspPopupMenu .....	4022
DspRichText .....	4024
DspRichTextEdit .....	4025
DspRichTextEnable .....	4025
DspRichTextGetInfo .....	4026

DspRichTextLoad .....	4027
DspRichTextPgScroll .....	4028
DspRichTextPrint .....	4029
DspRichTextSave .....	4030
DspRichTextScroll .....	4030
DspRubEnd .....	4031
DspRubMove .....	4032
DspRubSetClip .....	4033
DspRubSetColor .....	4034
DspRubStart .....	4035
DspSetClip .....	4036
DspSetCurColor .....	4037
DspSetMetadataFromName .....	4038
DspSetMetadataFromNameRelative .....	4039
DspSetPopupMenuFont .....	4040
DspSetSlider .....	4041
DspSetTip .....	4041
DspSetTooltipFont .....	4042
DspStatus .....	4043
DspStr .....	4044
DspSym .....	4045
DspSymAnm .....	4047
DspSymAnmEx .....	4048
DspSymAtSize .....	4049
DspText .....	4051
DspTipMode .....	4052
DspTrend .....	4053
DspTrendInfo .....	4055
DspWebContentGetURL .....	4056
DspWebContentSetURL .....	4057
<b>DLL Functions .....</b>	<b>4058</b>
DLLCall .....	4058
DLLCallEx .....	4059
DLLClose .....	4060
DLLOpen .....	4061
<b>Equipment Database Functions .....</b>	<b>4063</b>
EquipBrowseClose .....	4064
EquipBrowseFirst .....	4065
EquipBrowseGetField .....	4065
EquipBrowseNext .....	4066
EquipBrowseNumRecords .....	4067
EquipBrowseOpen .....	4068
EquipBrowsePrev .....	4069
EquipCheckUpdate .....	4070
EquipGetParameter .....	4070
EquipGetProperty .....	4071
EquipRefBrowseClose .....	4073
EquipRefBrowseFirst .....	4073
EquipRefBrowseGetField .....	4074

EquipRefBrowseNext .....	4075
EquipRefBrowseNumRecords .....	4076
EquipRefBrowseOpen .....	4076
EquipRefBrowsePrev .....	4078
EquipSetProperty .....	4078
EquipStateBrowseClose .....	4079
EquipStateBrowseFirst .....	4080
EquipStateBrowseGetField .....	4080
EquipStateBrowseNext .....	4082
EquipStateBrowseNumRecords .....	4082
EquipStateBrowseOpen .....	4083
EquipStateBrowsePrev .....	4083
Error Functions .....	4084
ErrCom .....	4085
ErrDrv .....	4085
ErrGetHw .....	4087
ErrInfo .....	4088
ErrLog .....	4089
ErrMsg .....	4089
ErrSet .....	4090
ErrSetHw .....	4092
ErrSetLevel .....	4094
ErrTrap .....	4095
IsError .....	4095
Event Functions .....	4096
CallEvent .....	4097
ChainEvent .....	4099
GetEvent .....	4100
OnEvent .....	4103
SetEvent .....	4107
File Functions .....	4109
FileClose .....	4110
FileCopy .....	4111
FileDelete .....	4112
FileEOF .....	4113
FileExist .....	4113
FileFind .....	4114
FileFindClose .....	4115
FileGetTime .....	4116
FileMakePath .....	4116
FileOpen .....	4117
FilePrint .....	4118
FileRead .....	4119
FileReadBlock .....	4120
FileReadLn .....	4121
FileRename .....	4121
FileRichTextPrint .....	4122
FileSeek .....	4123
FileSetTime .....	4124

FileSize .....	4124
FileSplitPath .....	4125
FileWrite .....	4126
FileWriteBlock .....	4127
FileWriteLn .....	4128
Form Functions .....	4128
FormActive .....	4130
FormAddList .....	4130
FormButton .....	4131
FormCheckBox .....	4132
FormComboBox .....	4134
FormCurr .....	4135
FormDestroy .....	4136
FormEdit .....	4137
FormField .....	4138
FormGetCurInst .....	4140
FormGetData .....	4141
FormGetInst .....	4141
FormGetText .....	4142
FormGoto .....	4143
FormGroupBox .....	4144
FormInput .....	4145
FormListAddText .....	4146
FormListBox .....	4147
FormListDeleteText .....	4148
FormListSelectText .....	4149
FormNew .....	4150
FormNumPad .....	4152
FormOpenFile .....	4153
FormPassword .....	4154
FormPosition .....	4155
FormPrompt .....	4155
FormRadioButton .....	4156
FormRead .....	4158
FormSaveAsFile .....	4159
FormSecurePassword .....	4160
FormSelectPrinter .....	4161
FormSetData .....	4161
FormSetInst .....	4162
FormSetText .....	4163
FormWndHnd .....	4164
Format Functions .....	4165
FmtClose .....	4165
FmtFieldHnd .....	4166
FmtGetField .....	4167
FmtGetFieldCount .....	4168
FmtGetFieldHnd .....	4168
FmtGetFieldName .....	4169
FmtGetWidth .....	4169

FmtOpen .....	4170
FmtSetField .....	4171
FmtSetFieldHnd .....	4172
FmtToStr .....	4173
Fuzzy Logic Functions .....	4173
FuzzyClose .....	4174
FuzzyGetCodeValue .....	4174
FuzzyGetShellValue .....	4175
FuzzyOpen .....	4176
FuzzySetCodeValue .....	4177
FuzzySetShellValue .....	4178
FuzzyTrace .....	4179
Group Functions .....	4179
GrpClose .....	4180
GrpDelete .....	4181
GrpFirst .....	4181
GrpIn .....	4182
GrpInsert .....	4183
GrpMath .....	4184
GrpName .....	4185
GrpNext .....	4185
GrpOpen .....	4186
GrpToStr .....	4187
I/O Device Functions .....	4188
DriverInfo .....	4188
IODeviceControl .....	4189
IODeviceInfo .....	4192
IODeviceStats .....	4196
Keyboard Functions .....	4196
KeyAllowCursor .....	4197
KeyBs .....	4198
KeyDown .....	4199
KeyGet .....	4200
KeyGetCursor .....	4200
KeyLeft .....	4201
KeyMove .....	4202
KeyPeek .....	4202
KeyPut .....	4203
KeyPutStr .....	4204
KeyReplay .....	4205
KeyReplayAll .....	4205
KeyRight .....	4206
KeySetCursor .....	4206
KeySetSeq .....	4207
KeyUp .....	4208
SendKeys .....	4209
Mail Functions .....	4211
MailError .....	4211
MailLogoff .....	4212

MailLogon .....	4213
MailRead .....	4214
MailSend .....	4215
Map Functions .....	4216
MapClear .....	4217
MapClose .....	4218
MapExists .....	4219
MapKeyCount .....	4219
MapKeyDelete .....	4220
MapKeyExists .....	4221
MapKeyFirst .....	4222
MapKeyNext .....	4223
MapOpen .....	4224
MapValueGet .....	4225
MapValueSet .....	4226
MapValueSetQuality .....	4227
Math and Trigonometry Functions .....	4228
Abs .....	4230
ArcCos .....	4230
ArcSin .....	4231
ArcTan .....	4231
Cos .....	4232
DegToRad .....	4233
Exp .....	4233
Fact .....	4234
HighByte .....	4234
HighWord .....	4235
Ln .....	4236
Log .....	4236
LowByte .....	4237
LowWord .....	4238
Max .....	4238
Min .....	4239
Pi .....	4240
Pow .....	4240
RadToDeg .....	4241
Rand .....	4241
Round .....	4242
Sign .....	4242
Sin .....	4243
Sqrt .....	4244
Tan .....	4244
Menu Functions .....	4245
MenuGetChild .....	4246
MenuGetFirstChild .....	4247
MenuGetGenericNode .....	4248
MenuGetNextChild .....	4248
MenuGetNodeByPath .....	4249
MenuGetPageNode .....	4250

MenuGetParent .....	4250
MenuGetPrevChild .....	4251
MenuGetWindowNode .....	4252
MenuNodeAddChild .....	4252
MenuNodeGetCurr .....	4253
MenuNodeGetDepth .....	4254
MenuNodeGetProperty .....	4255
MenuNodeGetTargetPage .....	4256
MenuNodeHasCommand .....	4256
MenuNodeIsDisabled .....	4257
MenuNodeGetExpanded .....	4258
MenuNodeIsHidden .....	4258
MenuNodeRemove .....	4259
MenuNodeRunCommand .....	4260
MenuNodeSetDisabledWhen .....	4260
MenuNodeSetExpanded .....	4261
MenuNodeSetHiddenWhen .....	4262
MenuNodeSetProperty .....	4263
MenuReload .....	4264
Miscellaneous Functions .....	4265
AreaCheck .....	4267
Assert .....	4268
Beep .....	4269
CitectInfo .....	4270
CodeTrace .....	4275
DebugBreak .....	4276
DebugMsg .....	4277
DebugMsgSet .....	4278
DelayShutdown .....	4279
DisplayRuntimeManager .....	4279
DumpKernel .....	4280
EngToGeneric .....	4281
Exec .....	4282
GetArea .....	4283
GetEnv .....	4283
GetLogging .....	4284
InfoForm .....	4285
InfoFormAn .....	4286
Input .....	4287
IntToReal .....	4287
KerCmd .....	4289
KernelQueueLength .....	4289
KernelTableInfo .....	4290
KernelTableItemCount .....	4291
LanguageFileTranslate .....	4292
Message .....	4293
ParameterGet .....	4294
ParameterPut .....	4295
ProcessIsClient .....	4295

ProcessIsServer .....	4296
ProcessRestart .....	4297
ProductInfo .....	4297
ProjectInfo .....	4298
ProjectRestartGet .....	4299
ProjectRestartSet .....	4300
ProjectSet .....	4300
Prompt .....	4301
Pulse .....	4301
ServerDumpKernel .....	4302
ServiceGetList .....	4303
SetArea .....	4304
SetLogging .....	4305
Shutdown .....	4306
ShutdownForm .....	4308
ShutdownMode .....	4309
SwitchConfig .....	4309
TestRandomWave .....	4310
TestSawWave .....	4311
TestSinWave .....	4312
TestSquareWave .....	4313
TestTriangWave .....	4314
Toggle .....	4315
TraceMsg .....	4315
Version .....	4316
.Net Functions .....	4317
DIIClassDispose .....	4317
DIIClassCreate .....	4318
DIIClassIsValid .....	4319
DIIClass SetProperty .....	4320
DIIClass GetProperty .....	4321
DIIClass CallMethod .....	4321
Conversion Table .....	4322
Page Functions .....	4323
PageAlarm .....	4326
PageBack .....	4327
PageDisabled .....	4327
PageDisplay .....	4328
PageExists .....	4330
PageFile .....	4330
PageFileInfo .....	4331
PageFileInfoEx .....	4332
PageForward .....	4332
PageGetInt .....	4333
PageGetStr .....	4334
PageGoto .....	4334
PageHardware .....	4336
PageHistoryDspMenu .....	4337
PageHistoryEmpty .....	4337

PageHome .....	4338
PageInfo .....	4338
PageLast .....	4340
PageListCount .....	4341
PageListCurrent .....	4342
PageListInfo .....	4343
PageListDisplay .....	4345
PageListDelete .....	4346
PageMenu .....	4347
PageNext .....	4348
PagePeekCurrent .....	4349
PagePeekLast .....	4349
PagePopLast .....	4350
PagePopUp .....	4351
PagePrev .....	4352
PageProcessAnalyst .....	4353
PageProcessAnalystPens .....	4354
PagePushLast .....	4355
PageRecall .....	4356
PageRichTextFile .....	4357
PageSelect .....	4358
PageSetInt .....	4359
PageSetStr .....	4360
PageSOE .....	4360
PageSummary .....	4361
PageTask .....	4362
PageTransformCoords .....	4363
PageTrend .....	4364
PageTrendEx .....	4365
Plot Functions .....	4366
PlotClose .....	4367
PlotDraw .....	4368
PlotGetMarker .....	4369
PlotGrid .....	4370
PlotInfo .....	4372
PlotLine .....	4373
PlotMarker .....	4376
PlotOpen .....	4377
PlotScaleMarker .....	4379
PlotSetMarker .....	4381
PlotText .....	4381
PlotXYLine .....	4383
Process Analyst Functions .....	4385
ProcessAnalystLoadFile .....	4385
ProcessAnalystPopup .....	4387
ProcessAnalystSelect .....	4388
ProcessAnalystSetPen .....	4389
ProcessAnalystWin .....	4390
Quality Functions .....	4391

QualityCreate .....	4392
QualityGetPart .....	4393
QualityIsControlInhibit .....	4400
QualityIsBad .....	4400
QualityIsGood .....	4401
QualityIsOverride .....	4402
QualityIsUncertain .....	4402
QualitySetPart .....	4403
QualityToStr .....	4404
VariableQuality .....	4406
Report Functions .....	4407
RepGetCluster .....	4407
RepGetControl .....	4408
Report .....	4409
RepSetControl .....	4410
Scheduler Functions .....	4412
SchdClose .....	4415
SchdConfigClose .....	4415
SchdConfigFirst .....	4416
SchdConfigGetField .....	4416
SchdConfigNext .....	4417
SchdConfigNumRecords .....	4418
SchdConfigOpen .....	4418
SchdConfigPrev .....	4420
SchdFirst .....	4421
SchdGetField .....	4422
SchdNext .....	4422
SchdNumRecords .....	4423
SchdOpen .....	4423
SchdPrev .....	4425
SchdSpecialAdd .....	4425
SchdSpecialClose .....	4426
SchdSpecialDelete .....	4427
SchdSpecialFirst .....	4427
SchdSpecialGetField .....	4428
SchdSpecialItemAdd .....	4428
SchdSpecialItemAddRange .....	4429
SchdSpecialItemClose .....	4430
SchdSpecialItemDelete .....	4431
SchdSpecialItemDeleteRange .....	4432
SchdSpecialItemFirst .....	4432
SchdSpecialItemGetField .....	4433
SchdSpecialItemModify .....	4434
SchdSpecialItemModifyRange .....	4435
SchdSpecialItemNext .....	4436
SchdSpecialItemNumRecords .....	4436
SchdSpecialItemOpen .....	4437
SchdSpecialItemPrev .....	4438
SchdSpecialModify .....	4438

SchdSpecialNext .....	4439
SchdSpecialNumRecords .....	4440
SchdSpecialOpen .....	4440
SchdSpecialPrev .....	4441
ScheduleItemAdd .....	4442
ScheduleItemDelete .....	4443
ScheduleItemModify .....	4443
ScheduleItemSetRepeat .....	4444
Screen Profile Functions .....	4446
ResetScreenProfile .....	4446
Security Functions .....	4447
FullName .....	4449
GetLanguage .....	4449
GetPriv .....	4450
Login .....	4451
LoginForm .....	4453
Logout .....	4454
LogoutIdle .....	4455
MultiSignatureForm .....	4456
MultiSignatureTagWrite .....	4456
Name .....	4458
UserCreate .....	4458
UserCreateByRole .....	4460
UsercreateForm .....	4461
UserDelete .....	4461
UserEditForm .....	4462
UserInfo .....	4463
UserLogin .....	4464
UserPassword .....	4466
UserPasswordExpiryDays .....	4467
UserPasswordField .....	4468
UserSetStr .....	4469
UserUpdateRecord .....	4470
UserVerify .....	4471
VerifyPrivilegeForm .....	4472
VerifyPrivilegeTagWrite .....	4473
WhoAmI .....	4474
Sequence of Events (SOE) Functions .....	4475
SOEArchive .....	4475
SOEDismount .....	4476
SOEEventAdd .....	4477
SOEMount .....	4478
Server Functions .....	4478
ServerBrowseClose .....	4479
ServerBrowseOpen .....	4480
ServerBrowseFirst .....	4481
ServerBrowseNext .....	4482
ServerBrowsePrev .....	4482
ServerBrowseGetField .....	4483

ServerBrowseNumRecords .....	4484
ServerGetProperty .....	4484
ServerInfo .....	4485
ServerInfoEx .....	4488
ServerIsOnline .....	4491
ServerReload .....	4491
ServerRestart .....	4493
SQL Functions .....	4494
ExecuteDTSPkg .....	4496
SQLAppend .....	4498
SQLBeginTran .....	4499
SQLCall .....	4501
SQLCancel .....	4502
SQLClose .....	4502
SQLCommit .....	4503
SQLConnect .....	4504
SQLCreate .....	4507
SQLDisconnect .....	4509
SQLDispose .....	4510
SQLEnd .....	4510
SQLErrMsg .....	4511
SQLExec .....	4512
SQLFieldInfo .....	4516
SQLGetField .....	4518
SQLGetRecordset .....	4519
SQLGetScalar .....	4520
SQLInfo .....	4522
SQLIsNullField .....	4523
SQLNext .....	4524
SQLNoFields .....	4524
SQLNumChange .....	4525
SQLNumFields .....	4526
SQLOpen .....	4527
SQLParamsClearAll .....	4528
SQLParamsSetAsInt .....	4529
SQLParamsSetAsReal .....	4532
SQLParamsSetAsString .....	4533
SQLPrev .....	4535
SQLQueryCreate .....	4536
SQLQueryDispose .....	4536
SQLRollBack .....	4537
SQLRowCount .....	4538
SQLSet .....	4539
SQLTraceOff .....	4540
SQLTraceOn .....	4541
SPC Functions .....	4542
SPCAlarms .....	4543
SPCCClientInfo .....	4544
SPCGetHistogramTable .....	4545

SPCGetSubgroupTable .....	4547
SPCPlot .....	4548
SPCProcessXRSGet .....	4549
SPCProcessXRSSet .....	4550
SPCSelLimit .....	4551
SPCSpecLimitGet .....	4552
SPCSpecLimitSet .....	4553
SPCSubgroupSizeGet .....	4554
SPCSubgroupSizeSet .....	4555
<b>String Functions .....</b>	<b>4556</b>
CharToStr .....	4558
HexToStr .....	4559
IntToStr .....	4559
PathToStr .....	4560
RealToStr .....	4561
StrCalcWidth .....	4562
StrClean .....	4562
StrEndsWith .....	4563
StrFill .....	4563
StrFormat .....	4564
StrGetChar .....	4565
StrLeft .....	4566
StrLength .....	4566
StrListContainsItem .....	4567
StrLower .....	4568
StrMid .....	4568
StrPad .....	4569
StrReplace .....	4570
StrRight .....	4571
StrSearch .....	4571
StrSetChar .....	4572
StrSplit .....	4573
StrToBool .....	4574
StrToChar .....	4575
StrToDate .....	4575
StrToFmt .....	4576
StrToGrp .....	4577
StrToHex .....	4578
StrToInt .....	4579
StrToLines .....	4579
StrToLocalText .....	4580
StrToPeriod .....	4581
StrToReal .....	4582
StrToTime .....	4583
StrToValue .....	4584
StrTrim .....	4585
StrTruncFont .....	4585
StrTruncFontHnd .....	4586
StrTruncFontTooltip .....	4587

StrUpper .....	4588
StrWord .....	4588
Super Genie Functions .....	4589
Ass .....	4591
AssChain .....	4592
AssChainPage .....	4593
AssChainPopUp .....	4594
AssChainWin .....	4595
AssChainWinFree .....	4596
AssEquipParameters .....	4598
AssEquipReferences .....	4599
AssGetProperty .....	4600
AssGetScale .....	4602
AssInfo .....	4604
AssInfoEx .....	4607
AssMetadata .....	4610
AssMetadataPage .....	4612
AssMetadataPopUp .....	4613
AssMetadataWin .....	4614
AssPage .....	4616
AssPopUp .....	4617
AssScaleStr .....	4618
AssTag .....	4620
AssTitle .....	4621
AssVarTags .....	4622
AssWin .....	4623
AssWinReplace .....	4625
Table (Array) Functions .....	4626
TableLookup .....	4627
TableMath .....	4627
TableShift .....	4629
Tag Functions .....	4630
SubscriptionAddCallback .....	4632
SubscriptionGetAttribute .....	4633
SubscriptionGetInfo .....	4634
SubscriptionGetQuality .....	4635
SubscriptionGetTag .....	4635
SubscriptionGetTimestamp .....	4636
SubscriptionGetValue .....	4637
SubscriptionRemoveCallback .....	4638
TagBrowseClose .....	4638
TagBrowseFirst .....	4639
TagBrowseGetField .....	4640
TagBrowseNext .....	4641
TagBrowseNumRecords .....	4641
TagBrowseOpen .....	4642
TagBrowsePrev .....	4645
TagDebug .....	4646
TagDebugForm .....	4647

TagEventFormat .....	4649
TagEventQueue .....	4649
TagGetProperty .....	4651
TagGetScale .....	4653
TagGetValue .....	4654
TagInfo .....	4656
TagInfoEx .....	4659
TagRamp .....	4662
TagRDBReload .....	4663
TagRead .....	4664
TagReadEx .....	4666
TagResolve .....	4668
TagScaleStr .....	4669
TagSetOverrideBad .....	4670
TagSetOverrideGood .....	4671
TagSetOverrideQuality .....	4672
TagSetOverrideUncertain .....	4673
TagSubscribe .....	4675
TagUnresolve .....	4678
TagUnsubscribe .....	4678
TagWrite .....	4679
TagWriteEventQue .....	4680
TagWriteIntArray .....	4682
TagWriteRealArray .....	4683
Task Functions .....	4685
CodeSetMode .....	4687
EnterCriticalSection .....	4688
Halt .....	4689
LeaveCriticalSection .....	4690
MsgBrdcst .....	4691
MsgClose .....	4691
MsgGetCurr .....	4692
MsgOpen .....	4693
MsgRead .....	4696
MsgRPC .....	4697
MsgState .....	4698
MsgWrite .....	4700
QueClose .....	4701
QueLength .....	4701
QueOpen .....	4702
QuePeek .....	4703
QueRead .....	4704
QueWrite .....	4705
ReRead .....	4706
SemClose .....	4707
SemOpen .....	4707
SemSignal .....	4708
SemWait .....	4709
ServerRPC .....	4710

Sleep .....	4711
SleepMS .....	4712
TaskCall .....	4713
TaskCluster .....	4714
TaskGetSignal .....	4715
TaskHnd .....	4715
TaskKill .....	4716
TaskNew .....	4717
TaskNewEx .....	4719
TaskResume .....	4721
TaskSetSignal .....	4722
TaskSuspend .....	4722
Time and Date Functions .....	4723
Date .....	4724
DateAdd .....	4726
DateDay .....	4726
DateInfo .....	4727
DateMonth .....	4728
DateSub .....	4729
DateWeekDay .....	4730
DateYear .....	4731
OLEDateToTime .....	4732
SysTime .....	4733
SysTimeDelta .....	4734
Time .....	4735
TimeCurrent .....	4736
TimeHour .....	4736
TimeInfo .....	4737
TimeMidNight .....	4738
TimeMin .....	4739
TimeSec .....	4740
TimeSet .....	4740
TimeToOLEDate .....	4741
TimeToStr .....	4742
TimeUTCOffset .....	4743
Timestamp Functions .....	4744
StrToTimestamp .....	4745
TimeIntToTimestamp .....	4745
TimestampAdd .....	4746
TimestampCurrent .....	4747
TimestampCreate .....	4748
TimestampDifference .....	4749
TimestampFormat .....	4750
TimestampGetPart .....	4752
TimestampSub .....	4753
TimestampToStr .....	4754
TimestampToInt .....	4756
VariableTimestamp .....	4756
Trend Functions .....	4758

TrendDspCursorComment .....	4762
TrendDspCursorScale .....	4762
TrendDspCursorTag .....	4763
TrendDspCursorTime .....	4764
TrendDspCursorValue .....	4765
TrendGetAn .....	4765
TrendPopUp .....	4766
TrendRun .....	4767
TrendSetDate .....	4767
TrendSetScale .....	4768
TrendSetSpan .....	4769
TrendSetTime .....	4770
TrendSetTimebase .....	4770
TrendWin .....	4771
TrendZoom .....	4773
TrnAddHistory .....	4774
TrnBrowseClose .....	4774
TrnBrowseFirst .....	4775
TrnBrowseGetField .....	4775
TrnBrowseNext .....	4776
TrnBrowseNumRecords .....	4777
TrnBrowseOpen .....	4778
TrnBrowsePrev .....	4779
TrnClientInfo .....	4780
TrnComparePlot .....	4781
TrnDelete .....	4783
TrnDelHistory .....	4783
TrnEcho .....	4784
TrnEventGetTable .....	4785
TrnEventGetTableMS .....	4787
TrnEventSetTable .....	4789
TrnEventSetTableMS .....	4790
TrnExportClip .....	4792
TrnExportCSV .....	4794
TrnExportDBF .....	4796
TrnExportDDE .....	4798
TrnFlush .....	4800
TrnGetBufEvent .....	4801
TrnGetBufTime .....	4802
TrnGetBufValue .....	4803
TrnGetCluster .....	4804
TrnGetCursorEvent .....	4804
TrnGetCursorMSTime .....	4805
TrnGetCursorPos .....	4806
TrnGetCursorTime .....	4806
TrnGetCursorValue .....	4807
TrnGetCursorValueStr .....	4808
TrnGetDefScale .....	4809
TrnGetDisplayMode .....	4810

TrnGetEvent .....	4811
TrnGetFormat .....	4812
TrnGetGatedValue .....	4813
TrnGetInvalidValue .....	4813
TrnGetMode .....	4814
TrnGetMSTime .....	4815
TrnGetPen .....	4817
TrnGetPenComment .....	4818
TrnGetPenFocus .....	4818
TrnGetPenNo .....	4819
TrnGetPeriod .....	4820
TrnGetScale .....	4821
TrnGetScaleStr .....	4821
TrnGetSpan .....	4823
TrnGetTable .....	4823
TrnGetTime .....	4826
TrnGetUnits .....	4827
TrnInfo .....	4828
TrnIsValidValue .....	4829
TrnNew .....	4830
TrnPlot .....	4831
TrnPrint .....	4832
TrnSamplesConfigured .....	4833
TrnScroll .....	4834
TrnSelect .....	4835
TrnSetCursor .....	4836
TrnSetCursorPosition .....	4837
TrnSetDisplayMode .....	4838
TrnSetEvent .....	4839
TrnSetPen .....	4840
TrnSetPenFocus .....	4841
TrnSetPeriod .....	4842
TrnSetScale .....	4843
TrnSetSpan .....	4844
TrnSetTable .....	4845
TrnSetTime .....	4847
Window Functions .....	4848
GetWinTitle .....	4850
HtmlHelp .....	4850
MultiMonitorStart .....	4851
WinCopy .....	4851
WinFile .....	4853
WinGetFirstChild .....	4855
WinFree .....	4855
WinFreeEx .....	4856
WinGetClicked .....	4857
WinGetFocus .....	4857
WinGetName .....	4858
WinGetNextChild .....	4859

WinGetParent .....	4859
WinGetWndHnd .....	4860
WinGoto .....	4861
WinMode .....	4862
WinMove .....	4862
WinNew .....	4863
WinNewAt .....	4864
WinNewPinAt .....	4867
WinNext .....	4870
WinNumber .....	4871
WinPos .....	4871
WinPrev .....	4872
WinPrint .....	4873
WinPrintFile .....	4874
WinSelect .....	4876
WinSetName .....	4877
WinSize .....	4877
WinStyle .....	4878
WinTitle .....	4879
WndFind .....	4880
WndGetFileProfile .....	4881
WndInfo .....	4881
WndMonitorInfo .....	4883
WndMonitorInfoEx .....	4884
WndPutFileProfile .....	4884
WndShow .....	4885
<b>XML Functions .....</b>	<b>4886</b>
XMLClose .....	4888
XMLCreate .....	4888
XMLGetAttribute .....	4889
XMLGetAttributeCount .....	4890
XMLGetAttributeName .....	4891
XMLGetAttributeValue .....	4892
XMLGetChild .....	4893
XMLGetChildCount .....	4894
XMLGetParent .....	4894
XMLGetRoot .....	4895
XMLNodeAddChild .....	4896
XMLNodeFind .....	4897
XMLNodeGetName .....	4898
XMLNodeGetValue .....	4899
XMLNodeRemove .....	4900
XMLNodeSetValue .....	4901
XMLOpen .....	4902
XMLSave .....	4902
XMLSetAttribute .....	4903
<b>Browse Function Field Reference .....</b>	<b>4904</b>
AccumBrowseOpen Fields .....	4905
AlmBrowseGetField Fields .....	4906

AlarmGetDsp Fields .....	4925
AlarmGetFieldRec Fields .....	4945
AlmBrowseOpen Fields .....	4964
AlmSummaryOpen Fields .....	4984
AlmTagsOpen Fields .....	5003
EquipBrowseOpen Fields .....	5023
EquipRefBrowseGetField Fields .....	5025
EquipRefBrowseOpen Fields .....	5027
EquipStateBrowseOpen Fields .....	5028
TagBrowseOpen Fields .....	5029
TrnBrowseOpen Fields .....	5033
Server Browse Function Fields .....	5038
<b>Parameters .....</b>	<b>5038</b>
About Parameters .....	5039
Parameter Precedence .....	5042
Hierarchical Parameters .....	5043
Computer Setup Editor .....	5045
Configure Parameters .....	5048
Configure Project Database Parameters .....	5048
Configure Citect.ini Parameters .....	5049
Open the Citect.ini File .....	5049
Edit Parameter Sections .....	5050
Edit Parameters .....	5051
Configure Alarm Parameters for a Specific Cluster or Process .....	5053
View Undocumented Parameters .....	5059
Add Comments to the Citect.ini File .....	5059
Use the Comparison Wizard .....	5060
Run the Comparison Wizard .....	5061
Interpret the Comparison Wizard .....	5061
Copy Values Between Compared Files .....	5061
Undo and Redo Changes to the Compared Files .....	5062
Save Changes to the Compared File .....	5062
Close Without Saving Changes .....	5063
Saving the Citect.ini File .....	5063
Parameters Reference .....	5064
Parameter Categories .....	5064
Computer-specific Parameters .....	5065
Accumulator Parameters .....	5065
[Accumulator]Debug .....	5066
[Accumulator]UpdateTime .....	5066
[Accumulator]WatchTime .....	5066
Address Forwarding Parameters .....	5067
Alarm Heading Parameters .....	5067
[AlarmHeading]AlarmField .....	5068
Alarm Log Parameters .....	5068
[AlarmLog]DefaultSearchDays .....	5069
[AlarmLog]Format .....	5069
[AlarmLog]NumFiles .....	5070
Alarm Parameters .....	5070

[Alarm]AckHold .....	5074
[Alarm]Active .....	5075
[Alarm]AlarmDisable .....	5075
[Alarm]AlarmListRequestTimeout .....	5076
[Alarm]ArchiveAfter .....	5076
[Alarm]BackgroundColorMode .....	5077
[Alarm]BrowseRowLimit .....	5077
[Alarm]CacheSize .....	5078
[Alarm]DBLogDBServer .....	5078
[Alarm]DBLogHistoric .....	5079
[Alarm]DBLogServerCore .....	5080
[Alarm]DefDspFmt .....	5082
[Alarm]DefSOEFmt .....	5082
[Alarm]DefSumFmt .....	5083
[Alarm]DeltaTimeUpdate .....	5083
[Alarm]DisableFilterOptimization .....	5084
[Alarm]DisableSummary .....	5084
[Alarm]DisableSOE .....	5085
[Alarm]DisplayDisable .....	5085
[Alarm]EnableErrorLogging .....	5086
[Alarm]EnableSmartCustomFilters .....	5086
[Alarm]EnableStateLogging .....	5087
[Alarm]EventFmt .....	5087
[Alarm]EventQue .....	5088
[Alarm]ExtendedDate .....	5088
[Alarm]FlushTimeLimit .....	5089
[Alarm]FutureMessages .....	5089
[Alarm]HardHoldTime .....	5090
[Alarm]HardwareDisable .....	5090
[Alarm]HeadingFont .....	5090
[Alarm]HeartbeatTimeout .....	5091
[Alarm]HighResOff .....	5091
[Alarm]Hres24HrDeadBand .....	5092
[Alarm]HresTimerExprDelay .....	5092
[Alarm]HresType .....	5093
[Alarm]HwAlarmFmt .....	5094
[Alarm]HwAlarmQueMax .....	5094
[Alarm]HwExclude .....	5094
[Alarm]IsolationDetectInterval .....	5095
[Alarm]IsolationDetectIP1 .....	5095
[Alarm]IsolationDetectIP2 .....	5096
[Alarm]IsolationDetectRetryCount .....	5097
[Alarm]KeepOnlineFor .....	5098
[Alarm]LastAlarmCategories .....	5098
[Alarm]LastAlarmDisplayMode .....	5099
[Alarm]LastAlarmFmt .....	5099
[Alarm]LastAlarmPriorities .....	5100
[Alarm]LastAlarmType .....	5100
[Alarm]MaxOptimisedQueries .....	5101

[Alarm]MaxQueryExecuteTime .....	5102
[Alarm]MemoryWarningLimit .....	5102
[Alarm]MonitorConnectTimeout .....	5102
[Alarm]MonitorRequestTimeout .....	5103
[Alarm]Period .....	5104
[Alarm]QueryCPUUsage .....	5104
[Alarm]QueryRowLimit .....	5105
[Alarm]QueryTimeout .....	5105
[Alarm]ReloadBackOffTime .....	5106
[Alarm]SavePrimary .....	5106
[Alarm]SaveSecondary .....	5107
[Alarm]ScanTime .....	5107
[Alarm]SetTimeOnAck .....	5108
[Alarm]SetTimeOnOff .....	5108
[Alarm]ShowAllConfigured .....	5109
[Alarm]Sort .....	5109
[Alarm]SortMode .....	5110
[Alarm]Sound<n> .....	5110
[Alarm]Sound<n>Interval .....	5111
[Alarm]SoundSilenceTimeoutOnAck .....	5112
[Alarm]StandbyCommandDelay .....	5112
[Alarm]StartTimeout .....	5113
[Alarm]StreamSize .....	5113
[Alarm]SummaryAutoRefreshMode .....	5114
[Alarm]SummaryBufferSize .....	5114
[Alarm]SummaryMode .....	5115
[Alarm]SummarySort .....	5116
[Alarm]SummarySortMode .....	5116
[Alarm]SummaryTimeout .....	5117
[Alarm]SummaryTimeoutTolerance .....	5117
[Alarm]SummaryShutdownMode .....	5118
[Alarm]SumStartupCopy .....	5118
[Alarm]SyncAllHistoricData .....	5119
[Alarm]TimeDate .....	5119
[Alarm]TransferConnectTimeout .....	5121
[Alarm]TransferInterleave .....	5121
[Alarm]TransferInterval .....	5122
[Alarm]TransferRequestTimeout .....	5122
[Alarm]UseConfigLimits .....	5123
Alarm Process Parameters .....	5124
[Alarm.<ClusterName>.<ServerName>]ClientConnectTimeout .....	5125
[Alarm.<ClusterName>.<ServerName>]ClientDisconnectTimeout .....	5125
[Alarm.<ClusterName>.<ServerName>]ClientRequestTimeout .....	5126
[Alarm.<ClusterName>.<ServerName>]Clusters .....	5126
[Alarm.<ClusterName>.<ServerName>]CPU .....	5126
[Alarm.<ClusterName>.<ServerName>]DisableConnection .....	5127
[Alarm.<ClusterName>.<ServerName>]Events .....	5128
[Alarm.<ClusterName>.<ServerName>]Priority .....	5128
[Alarm.<ClusterName>.<ServerName>]ShutdownCode .....	5129

[Alarm.<ClusterName>.<ServerName>]StartupCode .....	5130
AlarmFilterRule Parameters .....	5130
[AlarmFilterRuleList.Active]Rule<n> .....	5131
[AlarmFilterRuleList.Disabled]Rule<n> .....	5131
[AlarmFilterRuleList]Rule<n> .....	5132
[AlarmFilterRuleList.SOE]Rule<n> .....	5132
[AlarmFilterRuleList.Summary]Rule<n> .....	5133
[AlarmFilterRules]<RuleName> .....	5133
Animator Parameters .....	5134
[Animator]BoldWeight .....	5135
[Animator]ButtonCancelMode .....	5135
[Animator]CacheSize .....	5136
[Animator]ConfigureMouseCommand .....	5136
[Animator]EatMouseClick .....	5136
[Animator]EatMouseFocus .....	5137
[Animator]EnableWebContent .....	5137
[Animator]FastDisplay .....	5138
[Animator]FlashTime .....	5138
[Animator]FormFontWidth .....	5139
[Animator]FullScreen .....	5139
[Animator]InvalidRegionExpansion .....	5140
[Animator]InvalidRegionQueueDepth .....	5140
[Animator]LibraryError .....	5141
[Animator]LibraryLength .....	5141
[Animator]LibraryTime .....	5141
[Animator]MaxAn .....	5142
[Animator]MaximizedWindow .....	5142
[Animator]PrintXScale .....	5143
[Animator]PrintYScale .....	5143
[Animator]TabFactor .....	5143
[Animator]TemplateUpdate .....	5144
[Animator]TooltipFont .....	5144
[Animator]UseCTGIfNewer .....	5145
AnmCursor Parameters .....	5145
[AnmCursor]Color .....	5146
[AnmCursor]Height .....	5147
[AnmCursor]InvertText .....	5147
[AnmCursor]MouseSnapToCursor .....	5148
[AnmCursor]RubberBandColor .....	5148
[AnmCursor]Thickness .....	5149
[AnmCursor]Width .....	5149
Backup Parameters .....	5150
[Backup]SaveiniFiles .....	5150
BrowseTableView Parameters .....	5150
[BrowseTableView]<BrowseType>.<ViewName>.ColWidths .....	5150
[BrowseTableView]<BrowseType>.<ViewName>.Fields .....	5151
Client Parameters .....	5152
[Client]AdditionalClientsArePartOfTrustedNetwork .....	5153
[Client]AllowRPC .....	5154

[Client]AutoLoginClearPassword .....	5155
[Client]AutoLoginPage .....	5155
[Client]Clusters .....	5156
[Client]ComputerRole .....	5156
[Client]CPU .....	5157
[Client]DisableDisplay .....	5158
[Client]Events .....	5158
[Client]EvictTimeout .....	5158
[Client]ForceClient .....	5159
[Client]FullLicense .....	5159
[Client]PartOfTrustedNetwork .....	5160
[Client]PointCountRequired .....	5161
[Client]ReadOnly .....	5161
[Client]ResolveTimeout .....	5161
[Client]ShutdownCode .....	5162
[Client]StalenessPeriod .....	5162
[Client]StalenessPeriodTolerance .....	5163
[Client]StartupCode .....	5163
[Client]TagReadCachedMode .....	5164
[Client]TagReadRoundToFormat .....	5164
[Client]UseLocalLicense .....	5165
[Client]WaitForConnectAtStartup .....	5165
[Client]WaitForValidDataTimeout .....	5166
Code Parameters .....	5166
[Code]BackwardCompatibleErrHw .....	5168
[Code]CallbackThreads .....	5168
[Code]DebugMessage .....	5168
[Code]DIIICallErrorPopup .....	5169
[Code]DIIICallProtect .....	5169
[Code]DynamicCallDebug .....	5170
[Code]EchoError .....	5170
[Code]EnableErrorLogging .....	5171
[Code]Export .....	5171
[Code]HaltOnError .....	5172
[Code]HaltOnInvalidTagData .....	5172
[Code]IgnoreCase .....	5173
[Code]ProfilerEnabled .....	5173
[Code]Queue .....	5174
[Code]ScaleCheck .....	5174
[Code]ShutdownTime .....	5175
[Code]Stack .....	5175
[Code]StrictArgumentCheck .....	5175
[Code]Threads .....	5176
[Code]TimeData .....	5177
[Code]TimeSlice .....	5177
[Code]TimeSlicePage .....	5177
[Code]Unsigned .....	5178
[Code]VBASupport .....	5178
[Code]WriteLocal .....	5179

COM Parameters .....	5179
CrashHandler Parameters .....	5179
[CrashHandler]Enable .....	5180
[CrashHandler]KeepLogFiles .....	5181
CtAPI Parameters .....	5181
[CtAPI]AcceptThousandsSeparator .....	5182
[CtAPI]AllowLegacyServices .....	5182
[CtAPI]CpuLoadCount .....	5182
[CtAPI]CpuLoadSleepMS .....	5183
[CtAPI]Debug .....	5184
[CtAPI]EventLogging .....	5184
[CtAPI]MaxConnections .....	5185
[CtAPI]Remote .....	5185
[CtAPI]RoundToFormat .....	5186
CtCicode Parameters .....	5186
[CtCicode]FastFormat .....	5186
[CtCicode]MLCommentThreshold .....	5187
CtDraw.RSC Parameters .....	5187
[CtDraw.RSC]AllowEditSuperGeniePage .....	5188
[CtDraw.RSC]BitmapCompression .....	5188
[CtDraw.RSC]LibraryCacheSize .....	5189
[CtDraw.RSC]PurgeGenieLists .....	5189
CtEdit Parameters .....	5190
[CtEdit]ANSIToOEM .....	5191
[CtEdit]Backup .....	5191
[CtEdit]Bin .....	5192
[CtEdit]CompileSuccessfulCommand .....	5192
[CtEdit]CompileUnsuccessfulCommand .....	5194
[CtEdit]Config .....	5195
[CtEdit]Copy .....	5196
[CtEdit]Data .....	5197
[CtEdit]DbFiles .....	5198
[CtEdit]DbfNdxMode .....	5199
[CtEdit]Deploy .....	5199
[CtEdit]DisplayEquipmentItem .....	5199
[CtEdit]IncludeProjectEquipmentUpdate .....	5200
[CtEdit]IncrementalCompile .....	5200
[CtEdit]IncrementalEquipmentUpdate .....	5201
[CtEdit]IncrementalEquipmentUpdateAutoClose .....	5201
[CtEdit]Logs .....	5202
[CtEdit]LogLevel .....	5202
[CtEdit]MaxCicodeFunctions .....	5203
[CtEdit]MaxHelpRec .....	5203
[CtEdit]Run .....	5204
[CtEdit]SaveRetries .....	5205
[CtEdit]Starter .....	5205
[CtEdit]SubtAnValue .....	5206
[CtEdit]SubtPageValue .....	5206
[CtEdit]SuppressCompilerWarning .....	5207

[CtEdit]UpdateAnPage .....	5207
[CtEdit]UpdateAnPageVer5 .....	5208
[CtEdit]Upgrade .....	5208
[CtEdit]User .....	5209
CtExplore Parameters .....	5209
[CtExplore]DefaultTemplateStyle .....	5209
[CtExplore]DefaultTemplateResolution .....	5210
[CtExplore]DefaultTemplateTitleBar .....	5210
DBCClient Parameters .....	5211
[DBCClient]Enabled .....	5211
[DBCClient]FileBase .....	5212
[DBCClient]MaxFiles .....	5212
[DBCClient]MaxSize .....	5213
[DBCClient]OldFiles .....	5213
DDE Parameters .....	5213
[DDE]AllowCicode .....	5214
[DDE]AllowWrites .....	5214
[DDE]Enable .....	5214
[DDE]Timeout .....	5215
Debug Parameters .....	5215
[Debug]ArchiveFiles .....	5217
[Debug]CategoryFilter .....	5218
[Debug]CategoryFilterMode .....	5220
[Debug]ChromiumDevToolsPort .....	5220
[Debug]CodeDebug .....	5221
[Debug]CrashOnError .....	5221
[Debug]CrashReserveManagedMB .....	5222
[Debug]CrashReserveNativeMB .....	5222
[Debug]DBLogEnable .....	5223
[Debug]DBLogMaxFiles .....	5223
[Debug]DBLogMaxSize .....	5224
[Debug]DBLogOldFiles .....	5224
[Debug]DebugAllTrans .....	5225
[Debug]DebugLogHistoryFiles .....	5226
[Debug]DebugLogSize .....	5226
[Debug]DisableWinTop .....	5226
[Debug]DriverCheck .....	5227
[Debug]DriverTrace .....	5228
[Debug]DriverTracePort .....	5228
[Debug]DriverTraceMask .....	5229
[Debug]DriverTraceElements .....	5232
[Debug]DrWatson .....	5232
[Debug]EnableHardwareAlarmLogging .....	5233
[Debug]EnableLogging .....	5234
[Debug]EnablePSICounters .....	5234
[Debug]EnableReloadLogging .....	5235
[Debug]EnableTaskFrameworkCounters .....	5235
[Debug]EnableTransportCounters .....	5236
[Debug]FlushPeriod .....	5236

[Debug]LogDirect .....	5237
[Debug]LogsDriveMinimumFreeSpace .....	5237
[Debug]LogShutDown .....	5238
[Debug]MaximumFileSize .....	5238
[Debug]MaxMiniDumps .....	5239
[Debug]MaxTableOutputLength .....	5239
[Debug]Menu .....	5240
[Debug]MiniDumpType .....	5240
[Debug]Priority .....	5241
[Debug]SeverityFilter .....	5242
[Debug]SeverityFilterMode .....	5243
[Debug]Shutdown .....	5243
[Debug]ShutDownProtect .....	5244
[Debug]SnapshotEnable .....	5244
[Debug]SnapshotInterval .....	5245
[Debug]SnapshotMaxFiles .....	5245
[Debug]SnapshotMaxSize .....	5246
[Debug]SnapshotOldFiles .....	5246
[Debug]SysErrDsp .....	5247
[Debug]SysLogArchive .....	5247
[Debug]SysLogSize .....	5248
[Debug]UserTraceMode .....	5248
Deployment Parameters .....	5249
[Deployment]AskRestartArgs .....	5249
[Deployment]AskRestartFunc .....	5250
[Deployment]Enabled .....	5250
Device Parameters .....	5251
[Device]AlwaysCreateHistory .....	5251
[Device]CreateHistoryFiles .....	5252
[Device]DiscardSpoolOnError .....	5252
[Device]ForceHistoryOpenMode .....	5253
[Device]FormLength .....	5253
[Device]LptDosANSIToOEM .....	5253
[Device]MaxSpool .....	5254
[Device]SQLSelect .....	5254
[Device]WatchTime .....	5255
Dial Parameters .....	5255
[Dial]CacheRefresh .....	5256
[Dial]CallerIDTimeout .....	5256
[Dial]ConnectionRetries .....	5256
[Dial]Debug .....	5257
[Dial]DebugLevel .....	5257
[Dial]DefaultCallerID .....	5258
[Dial]MaxConnectionTime .....	5258
[Dial]MaxQueueTime .....	5259
[Dial]MaxTimeAfterDisconnect .....	5259
[Dial]MissedScheduleTolerance .....	5260
[Dial]ModemRetryDelay .....	5260
[Dial]PersistCache .....	5260

[Dial]ReadThroughCache .....	5261
[Dial]RingCount .....	5261
[Dial]WatchTime .....	5261
DisableIO Parameters .....	5262
[DisableIO]<DeviceName> .....	5262
[DisableIO]<ServerName> .....	5263
DiskDrv Parameters .....	5264
[DiskDrv]UpDateTime .....	5264
Driver Parameters .....	5264
[<DriverName>]OverrideOSProtection .....	5264
Event Parameters .....	5265
[Event]InhibitEvent .....	5265
[Event]Server .....	5266
[Event]WatchTime .....	5266
Font Parameters .....	5267
[Font]Echo .....	5267
[Font]Error .....	5267
[Font]General .....	5268
[Font]Print .....	5268
[Font]Prompt .....	5268
Format Parameters .....	5269
[Format]FormatName .....	5269
General Parameters .....	5271
[General]AdvancedDDEPoll .....	5273
[General]AELManagerUrl .....	5274
[General]BufferIO .....	5274
[General]CheckAddressBoundary .....	5274
[General]ClusterReplication .....	5275
[General]Ctl3D .....	5276
[General]DeadlockTimeout .....	5276
[General]DeadlockWarning .....	5276
[General]DebugInfo .....	5277
[General]DisablePlotSetupForm .....	5277
[General]DisableRemoteBlocking .....	5278
[General]EnforceTagGlobalUniqueness .....	5278
[General]FormatCheck .....	5279
[General]InfoDestroy .....	5279
[General]LicenseReservationTimeout .....	5280
[General]LicensingFullLogging .....	5280
[General]LockDelay .....	5280
[General]LockRetry .....	5281
[General]LongFilename .....	5281
[General]Multiprocess .....	5282
[General]OSErrors .....	5282
[General]PasswordExpiry .....	5283
[General]PointCountHigh .....	5283
[General]PointCountHighHigh .....	5283
[General]PrinterColourMode .....	5284
[General]RegionalNumbersFormat .....	5284

[General]ServerMonitoringPeriod .....	5285
[General]ShareFiles .....	5285
[General]ShowDriverError .....	5286
[General]Stack .....	5286
[General]StartDelay .....	5287
[General]SymbolicInfo .....	5287
[General]TagDB .....	5287
[General]TagDBReloadOnChange .....	5288
[General]TagStartDigit .....	5289
[General]TagWriteEventFmt .....	5290
[General]TagWriteEventQue .....	5290
[General]TrnPrinter .....	5291
[General]TypeRemapAllowed .....	5291
[General]Verbose .....	5291
[General]VerboseToSysLog .....	5292
[General]WatermarkedPointCount .....	5292
Interlock Parameters .....	5293
[Interlock]StateItem .....	5293
IOServer Parameters .....	5294
[IOServer]AlwaysCallStopChannel .....	5295
[IOServer]AlwaysCallStopUnit .....	5296
[IOServer]AsyncConnect .....	5296
[IOServer]BlockWrites .....	5297
[IOServer]CacheDebug .....	5297
[IOServer]CacheIgnoreDriver .....	5297
[IOServer]CacheOptMode .....	5298
[IOServer]CacheReadAhead .....	5299
[IOServer]CancelTimeout .....	5299
[IOServer]DisableConnectivityPropertiesAndExtensions .....	5300
[IOServer]DebugLogTagHandshake .....	5300
[IOServer]EnableEventQueue .....	5301
[IOServer]HeartTime .....	5301
[IOServer]HWAlarmOnDeviceDisable .....	5302
[IOServer]InitMsg .....	5302
[IOServer]LogTagHandshake .....	5303
[IOServer]MaxEventsDrop .....	5303
[IOServer]MaxEventsQueued .....	5303
[IOServer]MaxTagHandshakeSize .....	5304
[IOServer]MaxTimeInQueueMs .....	5304
[IOServer]PeerServerConnectTimeOut .....	5304
[IOServer]RedundancyDebug .....	5305
[IOServer]SaveFile .....	5305
[IOServer]SaveNetwork .....	5306
[IOServer]StrictTagHandshake .....	5306
[IOServer]TagAddressNoCase .....	5306
[IOServer]WatchDog .....	5307
[IOServer]WatchDogPrimary .....	5307
IOServer Process Parameters .....	5308
[IOServer.<ClusterName>.<ServerName>]Clusters .....	5308

[IOServer.<ClusterName>.<ServerName>]CPU .....	5309
[IOServer.<ClusterName>.<ServerName>]Events .....	5309
[IOServer.<ClusterName>.<ServerName>]ShutdownCode .....	5310
[IOServer.<ClusterName>.<ServerName>]StartupCode .....	5311
IPC Parameters .....	5311
[IPC]bPipeToSocket .....	5312
[IPC]EventLogging .....	5312
[IPC]HeartBeat .....	5313
[IPC]SocketSendTimeout .....	5313
[IPC]SocketShutdownTimeout .....	5313
Kernel Parameters .....	5314
[Kernel]AlwaysOnTop .....	5315
[Kernel]BufPoolProtect .....	5315
[Kernel]DeltaLateCount .....	5316
[Kernel]DeltaLateTime .....	5316
[Kernel]DeltaSleepTime .....	5317
[Kernel]ErrorBuffers .....	5318
[Kernel]ExceptionWatchdog .....	5318
[Kernel]Idle .....	5319
[Kernel]MaxLateTime .....	5320
[Kernel]MemoryMinimum .....	5320
[Kernel]Queue .....	5321
[Kernel]Retries .....	5322
[Kernel]Task .....	5322
[Kernel]Watchdog .....	5323
[Kernel]WatchdogTime .....	5324
[Kernel]WinShutdown .....	5324
Keyboard Parameters .....	5325
[Keyboard]ButtonOnlyLeftClick .....	5325
[Keyboard]ButtonRaw .....	5326
[Keyboard]EchoPopUp .....	5326
[Keyboard]LogExtendedDate .....	5327
[Keyboard]NonPrint .....	5327
[Keyboard]NonPrintChar .....	5327
[Keyboard]Type .....	5328
LAN Parameters .....	5328
[LAN]AddressScope .....	5330
[LAN]AddressType .....	5330
[LAN]AllowRemoteReload .....	5331
[LAN]BackOffTime .....	5331
[LAN]ClientRetryTime .....	5331
[LAN]ConnectionLogging .....	5332
[LAN]Delay .....	5332
[LAN]EarliestLegacyVersion .....	5333
[LAN]HeartbeatPeriod .....	5333
[LAN]HeartbeatTimeout .....	5334
[LAN]HighWaterMark .....	5335
[LAN]LocalNet .....	5335
[LAN]LowWaterMark .....	5336

[LAN]ListenerRetryTime .....	5336
[LAN]Node .....	5337
[LAN]ReadOnlyLegacyConnections .....	5338
[LAN]ReadPool .....	5338
[LAN]Sessions .....	5338
[LAN]SocketNoDelay .....	5339
[LAN]TCPIP .....	5339
[LAN]WaitBufTime .....	5340
[LAN]WritePool .....	5340
Language Parameters .....	5341
[Language]CaseSensitive .....	5341
[Language]CharSet .....	5342
[Language]ClientTranslateFile .....	5342
[Language]CodePage .....	5343
[Language]DisplayError .....	5344
[Language]LocalLanguage .....	5345
[Language]LocalLanguageOnly .....	5346
[Language]SuppressWarningUnsupported .....	5346
Memory Parameters .....	5346
[Memory]Free .....	5347
[Memory]MinPhyK .....	5347
[Memory]PageLock .....	5348
[Memory]Segment .....	5348
MultiMonitor Parameters .....	5349
[MultiMonitors]Context<n> .....	5349
[MultiMonitors]DisableAutoStart .....	5350
[MultiMonitors]Monitors .....	5351
[MultiMonitors]ScreenProfile .....	5351
[MultiMonitors]StartupPage<Name> .....	5351
[MultiMonitors]StartupPage<n> .....	5352
[MultiMonitors]Workspaces .....	5352
ODBC Parameters .....	5353
[ODBC]Server .....	5353
OID Parameters .....	5354
OpcDaServer Parameters .....	5354
[OpcDaServer]AutoStartByClient .....	5354
[OpcDaServer]RoundToFormat .....	5355
Page Parameters .....	5355
[Page]AddDefaultMenu .....	5359
[Page]Alarm .....	5359
[Page]AlarmPage .....	5360
[Page]AllowHScroll .....	5360
[Page]AllowHScrollBar .....	5361
[Page]AllowVScroll .....	5361
[Page]AllowVScrollBar .....	5361
[Page]AnmDelay .....	5362
[Page]BackgroundColor .....	5362
[Page]BadDitheringColor .....	5363
[Page]BadDitheringDensity .....	5364

[Page]BadText .....	5364
[Page]BadTextBackgroundColor .....	5365
[Page]CenterStartupPage .....	5365
[Page]ControlInhibitDitheringColor .....	5365
[Page]ControlInhibitDitheringDensity .....	5366
[Page]ControlInhibitTextBackgroundColor .....	5366
[Page]Date .....	5367
[Page]DefaultQualityFormat .....	5367
[Page]DefaultTimestampFormat .....	5368
[Page]Delay .....	5369
[Page]DelayRenderAdvancedAnimation .....	5369
[Page]DelayRenderAll .....	5369
[Page]DisabledAlarm .....	5370
[Page]DisabledGrayTextColor .....	5370
[Page]DisabledPage .....	5371
[Page]DynamicSizing .....	5371
[Page]EnableQualityToolTip .....	5372
[Page]ErrorDitheringColor .....	5373
[Page]ErrorDitheringDensity .....	5373
[Page]ErrorTextBackgroundColor .....	5374
[Page]HomePage .....	5374
[Page]HardwarePage .....	5375
[Page]HwAlarm .....	5375
[Page]IgnoreValueQuality .....	5375
[Page]InheritParentScale .....	5377
[Page]KeyEcho .....	5378
[Page]LastAlarm .....	5378
[Page]Logo .....	5379
[Page]MaintainAspectRatio .....	5379
[Page]MaxInt .....	5380
[Page]MaxLast .....	5380
[Page]MaxList .....	5380
[Page]MaxRecursion .....	5381
[Page]MaxStr .....	5381
[Page]MenuDisable .....	5382
[Page]MenuReloadOnChange .....	5382
[Page]MenuShutdown .....	5382
[Page]Name .....	5383
[Page]OriginAdjust .....	5383
[Page]OverrideDitheringColor .....	5384
[Page]OverrideDitheringDensity .....	5384
[Page]OverrideTextBackgroundColor .....	5384
[Page]PagesReloadOnChange .....	5385
[Page]PrintPage .....	5385
[Page]ProcessAnalystPage .....	5386
[Page]ProcessAnalystPopupPage .....	5386
[Page]Prompt .....	5387
[Page]RangeCheck .....	5387
[Page]RelocateNonResizedOnCreation .....	5387

[Page]ScaleTextToMax .....	5388
[Page]ScanTime .....	5388
[Page]ShowBadText .....	5389
[Page]ShowErrorText .....	5389
[Page]ShowUncertainText .....	5390
[Page]SOEPage .....	5390
[Page]Splash .....	5391
[Page]SplashTimeout .....	5391
[Page]SplashWinName .....	5392
[Page]Startup .....	5392
[Page]StartUpCancel .....	5393
[Page]StartupDelay .....	5393
[Page]StartupHeight .....	5394
[Page]StartupMode .....	5394
[Page]StartupWidth .....	5395
[Page]StartupWinName .....	5396
[Page]StartupX .....	5396
[Page]StartupY .....	5396
[Page]SummaryPage .....	5397
[Page]Time .....	5397
[Page]Tip .....	5398
[Page]TipHelp .....	5398
[Page]TipTimeOut .....	5398
[Page]Title .....	5399
[Page]UncertainDitheringColor .....	5399
[Page]UncertainDitheringDensity .....	5400
[Page]UncertainText .....	5400
[Page]UncertainTextBackgroundColor .....	5401
[Page]ValueAbnormalPattern .....	5401
[Page]WaitForValidData .....	5402
[Page]Windows .....	5402
[Page]WinTitle .....	5403
Path Parameters .....	5403
[Path]<PathName> .....	5403
Privilege Parameters .....	5404
[Privilege]AckAlarms .....	5404
[Privilege]DisableAlarms .....	5405
[Privilege>EditUser .....	5405
[Privilege]Exclusive .....	5405
[Privilege]Shutdown .....	5406
[Privilege]SilenceAlarms .....	5406
ProcessAnalyst Parameters .....	5407
Protocol Parameters .....	5407
Proxi Parameters .....	5408
PubSub Parameters .....	5408
[PubSub]CicodeDeviceDelaySwitchInterval .....	5409
[PubSub]CicodeDeviceDelaySwitchTime .....	5409
[PubSub]LogDevice .....	5410
[PubSub]LogLevel .....	5410

RemoteDB Parameters .....	5411
[RemoteDB]DefaultComment .....	5411
[RemoteDB]DefaultEngFull .....	5411
[RemoteDB]DefaultEngZero .....	5412
[RemoteDB]DefaultEU .....	5412
[RemoteDB]DefaultRawFull .....	5412
[RemoteDB]DefaultRawZero .....	5413
Report Parameters .....	5413
[Report]ComBreak .....	5414
[Report]Debug .....	5414
[Report]Disable .....	5415
[Report]HeartBeat .....	5415
[Report]HeartBeatTime .....	5416
[Report]InhibitEvent .....	5416
[Report]RtfRollOver .....	5417
[Report]RunStandby .....	5417
[Report]Startup .....	5418
[Report]TxtRollOver .....	5418
[Report]WatchTime .....	5419
Report Process Parameters .....	5419
[Report.<ClusterName>.<ServerName>]Clusters .....	5420
[Report.<ClusterName>.<ServerName>]CPU .....	5420
[Report.<ClusterName>.<ServerName>]DisableConnection .....	5421
[Report.<ClusterName>.<ServerName>]Events .....	5422
[Report.<ClusterName>.<ServerName>]Priority .....	5422
[Report.<ClusterName>.<ServerName>]ShutdownCode .....	5423
[Report.<ClusterName>.<ServerName>]StartupCode .....	5423
Runtime Manager Parameters .....	5424
[RuntimeManager]AllowDumpKernel .....	5424
[RuntimeManager]AllowReload .....	5425
[RuntimeManager]AutoRestartTrigger .....	5425
[RuntimeManager]ExitAfterShutdown .....	5425
[RuntimeManager]NotRespondingRetries .....	5426
[RuntimeManager]Profile .....	5426
Scheduling Parameters .....	5427
[Scheduling]AlwaysExecuteEntryAction .....	5427
[Scheduling]BACnetSchedulePollPeriod .....	5427
[Scheduling]PersistPath .....	5428
[Scheduling]StartDelay .....	5428
Security Parameters .....	5429
[Security]AuthenticationLogging .....	5429
[Security]BlockExec .....	5430
[Security]CreateUserByRole .....	5430
[Security]DisableDEP .....	5430
[Security]DeleteWebContentProfile .....	5431
Server Parameters .....	5432
[Server]AutoLoginMode .....	5432
Shutdown Parameters .....	5432
[Shutdown]NetworkIgnore .....	5432

[Shutdown]NetworkStart .....	5433
[Shutdown]Phase .....	5433
SPC Parameters .....	5434
[SPC]AlarmBufferSize .....	5434
[SPC]PartialSubgroup .....	5435
[SPC]RAboveUCL .....	5435
[SPC]RBelowLCL .....	5436
[SPC]ROutsideCL .....	5436
[SPC]XAboveUCL .....	5436
[SPC]XBelowLCL .....	5437
[SPC]XDownTrend .....	5437
[SPC]XErratic .....	5438
[SPC]XFreak .....	5438
[SPC]XGradualDown .....	5438
[SPC]XGradualUp .....	5439
[SPC]XMixture .....	5439
[SPC]XOutsideCL .....	5439
[SPC]XOutsideWL .....	5440
[SPC]XStratification .....	5440
[SPC]XUptrend .....	5441
SQL Parameters .....	5441
[SQL]DefaultTimestampFormat .....	5441
[SQL]MaxConnections .....	5442
[SQL]MaxRecordsets .....	5443
[SQL]QueryTimeout .....	5443
[SQL]TextColWidth .....	5443
Time Parameters .....	5444
Trend Parameters .....	5444
[Trend]AbortStartupOnError .....	5446
[Trend]AcquisitionTimeout .....	5447
[Trend]AllFiles .....	5447
[Trend]BlockByIODevice .....	5448
[Trend]BytesReadBeforeSleep .....	5448
[Trend]BytesWrittenBeforeSleep .....	5448
[Trend]CacheSize0 .....	5449
[Trend]CacheSize1 .....	5450
[Trend]CacheSize2 .....	5451
[Trend]CacheSize3 .....	5451
[Trend]CacheSize4 .....	5452
[Trend]CacheSize5 .....	5453
[Trend]CacheSize6 .....	5454
[Trend]CacheSize7 .....	5455
[Trend]CacheSize8 .....	5456
[Trend]ClientRequestTime .....	5456
[Trend]CopyFilesMessage .....	5457
[Trend]CursorPosition .....	5457
[Trend]DeleteIfIncompatible .....	5458
[Trend]Disable .....	5458
[Trend]EnableBackfill .....	5459

[Trend]FileCreationWriteSize .....	5459
[Trend]FileNameType .....	5460
[Trend]FilterViewByPrivilege .....	5460
[Trend]GapFillMode .....	5461
[Trend]GapFillSamples .....	5461
[Trend]GapFillTime .....	5462
[Trend]InhibitEvent .....	5462
[Trend]InstantTrendIdleTime .....	5463
[Trend]InstantTrendSamples .....	5463
[Trend]MaxBackfillsAtOnce .....	5464
[Trend]MaxRdnSamples .....	5464
[Trend]MaxRequestLength .....	5465
[Trend]MaxSetTableQue .....	5465
[Trend]MaxSetTableStnbyPending .....	5466
[Trend]MissedSamplesAlarmTime .....	5466
[Trend]RangeCheck .....	5467
[Trend]ReadWatchTime .....	5467
[Trend]ReloadBackOffTime .....	5468
[Trend]ShowCacheSizes .....	5468
[Trend]TrendDebug .....	5469
[Trend]WatchTime .....	5469
[Trend]WriteWatchTime .....	5470
Trend Process Parameters .....	5470
[Trend.<ClusterName>.<ServerName>]Clusters .....	5471
[Trend.<ClusterName>.<ServerName>]CPU .....	5471
[Trend.<ClusterName>.<ServerName>]DisableConnection .....	5472
[Trend.<ClusterName>.<ServerName>]Events .....	5473
[Trend.<ClusterName>.<ServerName>]Priority .....	5473
[Trend.<ClusterName>.<ServerName>]ShutdownCode .....	5474
[Trend.<ClusterName>.<ServerName>]StartupCode .....	5474
Win Parameters .....	5475
[Win]AltEsc .....	5475
[Win]AltSpace .....	5476
[Win]AltTab .....	5476
[Win]Configure .....	5477
[Win]ScreenSaver .....	5477
[Win>ShowAbout .....	5478
Workspace Parameters .....	5478
[Workspace]AlarmSortOrder .....	5479
[Workspace]EquipmentNavigationModelFilter .....	5480
[Workspace]InfoZoneAlarms.IncludeReferences .....	5481
[Workspace]InfoZoneTrends .....	5481
[Workspace]InfoZoneTrends.Type .....	5482
[Workspace]NumberOfTopPriorities .....	5482
Logging Parameters .....	5483
Advanced Logging Parameters .....	5485
Template Parameters .....	5486
Tab_Style Template Parameters .....	5486
Alarm Parameters .....	5486

Alarm Heading Parameters .....	5487
Format Parameters .....	5487
Page Parameters .....	5487
Privilege Parameters .....	5487
[TabAlarm.Custom] Parameters .....	5488
[TabAlarm.Custom]Function.AlarmGetDsp .....	5488
[TabAlarm.Custom]Function.Row.ShowContextMenu .....	5489
[TabAlarm.Custom]Function.Row.ShowHWContextMenu .....	5490
[Tabmenu.Custom] Parameters .....	5491
[Tabmenu.Custom]Font.Disabled .....	5491
[Tabmenu.Custom]Font.Normal .....	5492
[Tabmenu.Custom]Function.CreateDefaultMenu .....	5492
Situational Awareness Parameters .....	5494
Format Parameters .....	5494
[MultiMonitors] Parameters .....	5494
[SA_Library.Meter] Parameters .....	5495
[SA_Library.Meter]UseDefaultPLCLimits .....	5495
[SA_Library.Meter]PLCLimitNames .....	5495
[SA_Library.Meter]PLCLimitLabels .....	5496
[SA_Library.Meter]DefaultPLCLimits .....	5496
[Workspace] Parameters .....	5498
<b>VBA Programming Reference .....</b>	<b>5498</b>
Integrating VBA into a Project .....	5499
Using VBA in Command or Expression Fields .....	5501
Multithread Considerations with VBA .....	5505
Understanding VBA Language Basics .....	5506
VBA Files .....	5507
Cicode Editor .....	5507
Scope of VBA .....	5507
VBA Statements .....	5509
Comments .....	5510
Header Information .....	5510
Labels .....	5511
VBA Line Continuation Character .....	5511
Naming .....	5511
Option Statements .....	5512
VBA Data Types .....	5513
Constants .....	5514
Declaration of Constants .....	5515
Intrinsic Constants .....	5516
Variables .....	5517
Variable Declaration .....	5517
Variable Initialization Values .....	5518
Arrays of Variables .....	5519
Variable Array Declaration .....	5519
Array Subscripts .....	5520
Fixed Size Arrays .....	5521
Multi-Dimensional Arrays .....	5522
Dynamic Size Arrays .....	5522

Variant Declaration .....	5523
Variant Data Types and Coercion .....	5523
Numbers in Variants .....	5524
Numbers .....	5524
Numeric Data Types .....	5525
Exponential Notation .....	5525
Floating Point Calculation Rules .....	5526
Rounding Numbers .....	5526
Rounding Down .....	5527
Rounding Up .....	5527
Arithmetic Rounding .....	5527
Banker's Rounding .....	5527
Random Rounding .....	5527
Alternate Rounding .....	5528
Date and Time Handling .....	5528
Date Constants .....	5529
Formatting Date Values .....	5529
Text .....	5530
Day .....	5530
Month .....	5530
Year .....	5531
Period/Era .....	5531
Time .....	5531
Date and Time Data Constraints .....	5532
Date Data Type Structure .....	5533
Date-values .....	5533
Time-values .....	5534
Dates in Databases Using Different Calendars .....	5534
Operators .....	5535
Assignment Operator .....	5535
Arithmetical (Math) Operators .....	5536
Relational Operators .....	5537
Logical Operators .....	5537
Operator Precedence .....	5537
Strings .....	5538
String Comparisons .....	5539
String Concatenation .....	5539
Control Structures .....	5540
GoTo Statement .....	5541
Do Statement .....	5541
While Statement .....	5542
For Statement .....	5542
If Statement .....	5542
Select Case Statement .....	5544
End Statement .....	5545
Exit Statement .....	5546
OnError Statement .....	5546
Stop Statement .....	5547
With Statement .....	5547

Subroutines and Functions .....	5547
Subroutines .....	5548
Functions .....	5549
Arguments .....	5550
DLLs and APIs .....	5552
Accessing Functions in DLLs .....	5553
Declare Statement Structure .....	5553
Declare - Function Statement .....	5553
Declare - Lib Statement .....	5553
Declare - Alias Statement .....	5554
Passing Variables Byref and ByVal .....	5554
Passing Arguments to DLL Functions from VBA .....	5555
OLE Services .....	5556
OLE Terminology .....	5556
OLE Automation Objects .....	5557
Declaration of OLE Automation Objects .....	5558
Assigning References to OLE Automation Objects .....	5558
Using OLE Automation Objects .....	5559
Accessing the Object Model of OLE Automation Server Applications .....	5560
Understanding Object Models in OLE Automation .....	5561
Using the Microsoft Word Object Model .....	5563
OLE Automation Example Using the Microsoft Word Object .....	5563
Using the Microsoft Excel Object Model .....	5563
Deleting OLE Automation Objects .....	5564
File Input/Output with VBA .....	5565
VBA Function Reference .....	5565
Array Functions .....	5565
Dim .....	5566
Erase .....	5567
Lbound .....	5567
Option Base .....	5568
ReDim .....	5569
Ubound .....	5569
Conditional Statements .....	5570
Do Loop .....	5571
End Function .....	5571
Exit .....	5572
For .....	5572
GoTo .....	5573
If .....	5573
OnError .....	5575
Select .....	5575
Stop .....	5576
While...Wend .....	5577
With .....	5577
Conversion Functions .....	5577
ASCII Character Code Conversion .....	5577
Asc .....	5577
Chr .....	5578

Date Conversion .....	5579
CDate .....	5579
CDbl .....	5580
CInt .....	5580
CLng .....	5581
CSng .....	5582
CStr .....	5582
CVar .....	5583
Date and Time Formatting/Conversion .....	5583
DateSerial .....	5583
TimeSerial .....	5584
Number and String Conversion .....	5584
Format .....	5585
Hex .....	5592
Oct .....	5592
Str .....	5593
Val .....	5594
Declarations .....	5594
CreateObject .....	5595
Const .....	5596
Declare .....	5597
Dim .....	5598
IsDate .....	5599
IsEmpty .....	5599
IsNull .....	5600
IsNumeric .....	5601
Nothing .....	5601
Option Base .....	5602
Option Compare .....	5603
ReDim .....	5604
Set .....	5604
Static .....	5605
VarType .....	5606
Date and Time Functions .....	5607
Date .....	5607
Date Statement .....	5608
DateValue .....	5609
Day .....	5610
Hour .....	5610
Minute .....	5611
Month .....	5611
Now .....	5612
Second .....	5612
Time .....	5613
Time (statement) .....	5614
Timer .....	5614
TimeValue .....	5615
WeekDay .....	5616
Year .....	5617

File I/O Functions .....	5617
ChDir .....	5619
ChDrive .....	5620
Close .....	5620
CurDir, CurDir\$ .....	5621
Dir .....	5622
EOF .....	5624
FileCopy .....	5624
FileLen .....	5625
FreeFile .....	5626
Get # .....	5626
GetAttr .....	5628
Input .....	5629
Kill .....	5631
Line Input # .....	5631
Loc .....	5632
LOF .....	5633
MkDir .....	5634
Name .....	5635
Open .....	5636
Print (function) .....	5638
Print # .....	5638
Put # .....	5640
RmDir .....	5642
Seek .....	5643
Write # .....	5644
Math/Trigonometry Functions .....	5645
Numeric Functions .....	5645
Abs .....	5645
Exp .....	5646
Fix .....	5647
Int .....	5647
Log .....	5648
Rnd .....	5649
Sgn .....	5650
Sqrt .....	5650
Trigonometric Functions .....	5651
Atn .....	5651
Cos .....	5652
Sin .....	5652
Tan .....	5653
Miscellaneous Functions .....	5653
Beep .....	5654
Randomize .....	5654
Rem .....	5655
SendKeys .....	5655
Procedural Statements .....	5656
Call .....	5656
CicodeCallOpen .....	5657

CicodeCallReturn .....	5659
End Function .....	5661
End Sub .....	5661
Function .....	5662
Sub .....	5663
VbCallOpen Function .....	5664
VbCallReturn Function .....	5665
VbCallRun Function .....	5666
String Functions .....	5667
Asc .....	5668
Chr .....	5668
InStr .....	5669
LCase .....	5670
Left, Left\$ .....	5671
Len .....	5672
Line Input # .....	5672
LTrim .....	5673
Mid .....	5674
Option Compare .....	5675
Option Explicit .....	5676
Right .....	5676
RTrim .....	5677
Space .....	5678
StrComp .....	5678
String .....	5679
Trim .....	5680
UCase .....	5680
ASCII/ANSI Character Code Listings .....	5681
<b>Driver Reference Help .....</b>	<b>5691</b>
Safety Information .....	5692
Industry-Standard Drivers .....	5693
ABCLX Driver .....	5694
ControlLogix PLCs via Ethernet .....	5694
ControlLogix PLCs - Device Address .....	5695
ControlLogix PLCs - Software Requirements .....	5695
ControlLogix PLCs - Communication Settings .....	5695
Data Types .....	5698
Data Types - Data Coercion .....	5699
Adding Tags to Your Project .....	5699
Individual Tag Addressing .....	5700
Array Tag Addressing .....	5701
Mapping to Individual Elements of an Array .....	5702
Individual Tag Address Type Mapping .....	5702
Mapping to More Than an Individual Element of an Array .....	5703
Mapping Data Types for Array Tag Addressing .....	5704
Addressing Program Tags .....	5706
Configuring Quality and Timestamp Tags .....	5707
Addressing Statistics Tags .....	5709
Advanced Configuration and Maintenance .....	5710

Using a Route Path To Locate a Remote CPU .....	5710
Configuring Redundancy .....	5712
Stop_Unit Handling .....	5712
Performance Tuning .....	5712
Optimizing Driver to PLC Communications .....	5712
Optimizing Client To Server Communications .....	5713
Customizing a Project using Citect.ini Parameters .....	5713
ABCLX Specific Parameters .....	5714
Logging Parameters .....	5719
Global and Device-Specific Parameters .....	5720
Session Control .....	5721
Status Tags .....	5722
Mode Detection .....	5723
Identifying Bad Tags Using 'FailOnBadData' .....	5724
Scan Rates .....	5724
ABCLX Specific Errors .....	5725
Logging .....	5726
Driver Kernel Additional Fields .....	5728
Frequently Asked Questions .....	5729
ABMLXEIP Driver .....	5731
Safety Information .....	5731
Allen-Bradley MicroLogix via Ethernet .....	5733
Device Address .....	5734
Software Requirements .....	5734
Communication Settings .....	5734
Tag Addressing .....	5736
Output Data File .....	5737
Input Data File .....	5737
Status Data File .....	5738
Bit Data File .....	5739
Timer Data File .....	5740
Counter Data File .....	5740
Control Data File .....	5741
Integer Data File .....	5742
Float Data File .....	5742
Long Word Data File .....	5743
String Data File .....	5743
Advanced Configuration and Maintenance .....	5744
Configuring Redundancy .....	5744
Optimizing Communications .....	5745
Customizing a Project using Citect.ini Parameters .....	5745
Common Parameters .....	5746
Specific Parameters .....	5748
Logging Parameters .....	5748
Troubleshooting .....	5749
Logging .....	5750
Specific Errors .....	5751
Kernel Diagnostics .....	5752
ABTCP Driver .....	5753

Safety Information .....	5753
Supported Devices via TCP/IP .....	5755
Allen-Bradley PLC-2 via TCP/IP .....	5755
Allen-Bradley PLC-2 via TCP/IP - Device Address .....	5756
Allen-Bradley PLC-2 via TCP/IP - Hardware Setup .....	5756
Allen-Bradley PLC-2 via TCP/IP - Communication Settings .....	5757
Allen-Bradley PLC-2 via TCP/IP - Data Types .....	5758
Allen-Bradley PLC-3 via TCP/IP .....	5759
Allen-Bradley PLC-3 via TCP/IP - Device Address .....	5759
Allen-Bradley PLC-3 via TCP/IP - Hardware Setup .....	5760
Allen-Bradley PLC-3 via TCP/IP - Communication Settings .....	5760
Allen-Bradley PLC-3 via TCP/IP - Data Types .....	5761
Allen-Bradley PLC-5 via TCP/IP .....	5765
Allen-Bradley PLC-5 via TCP/IP - Device Address .....	5766
Allen-Bradley PLC-5 via TCP/IP - Hardware Setup .....	5766
Allen-Bradley PLC-5 via TCP/IP - Communication Settings .....	5767
Allen-Bradley PLC-5 via TCP/IP - Data Types .....	5769
Allen-Bradley PLC-5/250 via TCP/IP .....	5773
Allen-Bradley PLC-5/250 via TCP/IP - Device Address .....	5773
Allen-Bradley PLC-5/250 via TCP/IP - Hardware Setup .....	5774
Allen-Bradley PLC-5/250 via TCP/IP - Communication Settings .....	5774
Allen-Bradley PLC-5/250 via TCP/IP - Data Types .....	5776
Allen-Bradley SLC-500 via TCP/IP .....	5779
Allen-Bradley SLC-500 via TCP/IP - Device Address .....	5780
Allen-Bradley SLC-500 via TCP/IP - Hardware Setup .....	5780
Allen-Bradley SLC-500 via TCP/IP - Communication Settings .....	5781
Allen-Bradley SLC-500 via TCP/IP - Data Types .....	5782
Supported Devices via Data Highway .....	5784
Allen-Bradley PLC-2 via Data Highway .....	5785
Allen-Bradley PLC-2 via Data Highway - Device Address .....	5785
Allen-Bradley PLC-2 via Data Highway - KT Board Setup .....	5786
Allen-Bradley PLC-2 via Data Highway - 5136-SD-PCI Setup .....	5788
Allen-Bradley PLC-2 via Data Highway - 5136-SD-PCI Testing .....	5788
Allen-Bradley PLC-2 via Data Highway - Communication Settings .....	5789
Allen-Bradley PLC-2 via Data Highway - Data Types .....	5792
Allen-Bradley PLC-3 via Data Highway .....	5792
Allen-Bradley PLC-3 via Data Highway - Device Address .....	5793
Allen-Bradley PLC-3 via Data Highway - KT Board Setup .....	5793
Allen-Bradley PLC-3 via Data Highway - 5136-SD-PCI Setup .....	5796
Allen-Bradley PLC-3 via Data Highway - 5136-SD-PCI Test .....	5796
Allen-Bradley PLC-3 via Data Highway - Communication Settings .....	5796
Allen-Bradley PLC-3 via Data Highway - Data Types .....	5799
Allen-Bradley PLC-5 via Data Highway .....	5803
Allen-Bradley PLC-5 via Data Highway - Device Address .....	5804
Allen-Bradley PLC-5 via Data Highway - KT Board Setup .....	5804
Allen-Bradley PLC-5 via Data Highway - 5136-SD-PCI Setup .....	5807
Allen-Bradley PLC-5 via Data Highway - 5136-SD-PCI Test .....	5807
Allen-Bradley PLC-5 via Data Highway - Communication Settings .....	5807
Allen-Bradley PLC-5 via Data Highway - Data Types .....	5810

Allen-Bradley PLC-5/250 via Data Highway .....	5814
Allen-Bradley PLC-5/250 via Data Highway - Device Address .....	5815
Allen-Bradley PLC-5/250 via Data Highway - KT Board Setup .....	5815
Bradley PLC-5/250 via Data Highway - 5136-SD-PCI Setup .....	5818
Allen-Bradley PLC-5/250 via Data Highway - 5136-SD-PCI Test .....	5818
Allen-Bradley PLC-5/250 via Data Highway - Communication Settings .....	5818
Allen-Bradley PLC-5/250 via Data Highway - Data Types .....	5821
Allen-Bradley SLC-500 via Data Highway .....	5825
Allen-Bradley SLC-500 via Data Highway - Device Address .....	5826
Allen-Bradley SLC-500 via Data Highway - KT Board Setup .....	5826
Allen-Bradley SLC-500 via Data Highway - 5136-SD-PCI Setup .....	5828
Allen-Bradley SLC-500 via Data Highway - 5136-SD-PCI Test .....	5829
Allen-Bradley SLC-500 via Data Highway - Communication Settings .....	5829
Allen-Bradley SLC-500 via Data Highway - Data Types .....	5832
Offlink Addressing .....	5834
ABTCP Driver Parameters .....	5835
ABTCP Driver Specific Errors .....	5838
BACNET Driver .....	5841
Safety Information .....	5841
About the BACnet Protocol .....	5843
Communicating with BACnet .....	5844
Mapping Tags to BACnet Objects .....	5845
BACnet Data Type Representation .....	5846
Reading BACnet Properties .....	5847
Writing to BACnet Properties .....	5847
Writing to Specific Calendar and Schedule Properties .....	5848
Configuring Your Project .....	5853
BACnet Device Address .....	5853
Communication Settings .....	5853
BACstac Configuration .....	5856
BACNET Supported Object Types .....	5856
Analog Input Objects .....	5857
Analog Output Objects .....	5859
Analog Value Objects .....	5861
Binary Input Objects .....	5862
Binary Output Objects .....	5864
Binary Value Objects .....	5866
Calendar Objects .....	5868
DateList Sub-Properties .....	5869
Device Objects .....	5870
Life Safety Point Objects .....	5873
Sub-Properties of Life Safety Point .....	5876
Life Safety Zone Objects .....	5877
Sub-Properties of Life Safety Zone .....	5880
Loop Objects .....	5881
Multi-state Input Objects .....	5884
Multi-state Output Objects .....	5886
Multi-state Value Objects .....	5887
Notification Class Objects .....	5889

Positive Integer Value Objects .....	5891
Schedule Objects .....	5895
Sub-Properties of EffectivePeriod .....	5897
Sub-Properties of WeeklySchedule .....	5897
Sub-Properties of ExceptionSchedule .....	5898
Sub-Properties of Schedule .....	5899
Structured View Objects .....	5900
Timer Objects .....	5900
BACNET Parameters .....	5903
Blocking Parameters .....	5904
Discovery Parameters .....	5904
Read Parameters .....	5905
Write Parameters .....	5906
APDU Parameters .....	5907
Logging Parameters .....	5908
Logging .....	5909
BACnet Tag Import .....	5910
CTICMP Driver .....	5912
Safety Information .....	5912
TCP/IP Hosts .....	5914
TCP/IP Hosts - Device Address .....	5915
TCP/IP Hosts - Hardware Setup .....	5915
TCP/IP Hosts - Communication Settings .....	5915
TCP/IP Host Data Types .....	5916
Customizing a Project using Citect.ini Parameters .....	5918
CTICMP Driver Parameters .....	5918
Logging Parameters .....	5919
Logging .....	5920
CTICMP Driver Specific Errors .....	5922
DNPR Driver .....	5922
Safety Information .....	5923
The DNP 3.0 Protocol .....	5925
Device Setup .....	5926
Configure Your Project .....	5926
DNPR - Device Address .....	5927
Communication Settings .....	5927
Configure Variables .....	5930
Monitoring Data Types - Binary Inputs .....	5931
Monitoring Data Types - Binary Outputs .....	5933
Monitoring Data Types - Counters .....	5936
Monitoring Data Types - Analog Inputs .....	5940
Monitoring Data Types - Analog Outputs .....	5942
Monitoring Data Types - Time and Date .....	5945
Monitoring Data Types - IIN Flags .....	5945
Control Data Types - Binary Outputs .....	5948
Control Data Types - Counters .....	5951
Control Data Types - Analog Outputs .....	5951
Control Data Types - Time and Date .....	5953
Control Data Types - Utilities .....	5954

Global Data Types.....	5954
Polling Regime Data Types.....	5955
Miscellaneous Data Types .....	5963
Projects Using Super Genies .....	5963
Hints and Tips .....	5964
Advanced Configuration and Maintenance.....	5965
Customize a Project using Citect.ini Parameters .....	5965
Basic Parameters .....	5966
Logging/Diagnostic Parameters.....	5969
Driver-specific Parameters.....	5970
Peer Redundancy Parameters.....	5972
Port-based Parameters.....	5975
Event Processing Parameters.....	5975
Communication Parameters .....	5977
Polling Regime Parameters .....	5983
Data Point and Cache Parameters .....	5989
Miscellaneous Parameters .....	5997
Device Group Parameters .....	5999
DNPR Redundancy .....	6001
Redundancy Configuration .....	6002
Manual Redundancy Switchover (Usurping) .....	6003
Automatic Redundancy Switchover (Usurping) .....	6003
Port-level Redundancy .....	6004
Driver Cache Replication .....	6005
Trend and Alarm Backfilling .....	6005
Value Writes .....	6006
Action on Becoming Active .....	6006
Data Acquisition within DNP3 .....	6006
Blocking .....	6008
Device Polling Regime .....	6009
Device Recovery .....	6010
Device Recovery - Normal .....	6011
Device Recovery - Fast .....	6011
Device Recovery - Startup Considerations .....	6012
SELECT and OPERATE .....	6013
Virtual Units .....	6013
Virtual Units Example .....	6014
Virtual Units Configuration .....	6019
Configuring Variables for Virtual Units .....	6021
Virtual Units - Polling Regime Data Types .....	6022
Virtual Units - Usrp Data Types .....	6028
Virtual Units - Control Data Types .....	6031
Virtual Units - Monitoring Data Types .....	6032
Troubleshooting .....	6033
DNPR Driver-specific Errors .....	6033
Driver Statistics .....	6037
Logging .....	6038
Debugging Messages .....	6039
GENERIC Driver .....	6040

<b>GENERIC Driver Communications Settings</b>	6040
<b>GENERIC Driver Data Types</b>	6042
<b>GETCP Driver</b>	6042
<b>Safety Information</b>	6043
<b>Supported Devices</b>	6045
<b>GE Fanuc GE9030 via TCP/IP</b>	6045
<b>GE Fanuc GE9030 via TCP/IP - Device Address</b>	6046
<b>GE Fanuc GE9030 via TCP/IP - Hardware Setup</b>	6046
<b>GE Fanuc GE9030 via TCP/IP - Communication Settings</b>	6046
<b>GE Fanuc GE9030 via TCP/IP - Data Types</b>	6048
<b>GE Fanuc GE9070 via TCP/IP</b>	6049
<b>GE Fanuc GE9070 via TCP/IP - Device Address</b>	6050
<b>GE Fanuc GE9070 via TCP/IP - Hardware Setup</b>	6050
<b>GE Fanuc GE9070 via TCP/IP - Communication Settings</b>	6050
<b>GE Fanuc GE9070 via TCP/IP - Data Types</b>	6052
<b>GE Fanuc GERx3i via TCP/IP</b>	6053
<b>GE Fanuc GERx3i via TCP/IP - Device Address</b>	6053
<b>GE Fanuc GERx3i via TCP/IP - Hardware Setup</b>	6053
<b>GE Fanuc GERx3i via TCP/IP - Communication Settings</b>	6054
<b>GE Fanuc GERx3i via TCP/IP - Data Types</b>	6055
<b>GETCP Driver Parameters</b>	6057
<b>GETCP Driver Specific Errors</b>	6058
<b>IEC870IP Driver</b>	6059
<b>Safety Information</b>	6060
<b>Supported Devices</b>	6062
<b>Setting up Device Communications</b>	6062
<b>Using the Express Communications Wizard</b>	6062
<b>Communications Settings</b>	6063
<b>Configuring Variable Tags</b>	6065
<b>Read Tags</b>	6065
<b>Output Tags</b>	6066
<b>Control Tags</b>	6068
<b>Array Tags</b>	6069
<b>Virtual Tags</b>	6070
<b>Structured Tag Addressing</b>	6070
<b>Value Quality Support</b>	6070
<b>Quality Descriptors</b>	6071
<b>Quality Mapping</b>	6072
<b>Value Timestamping</b>	6073
<b>Timestamp Handling</b>	6073
<b>Advanced Configuration and Maintenance</b>	6074
<b>Using Read Commands to Retrieve Data</b>	6074
<b>Events Handling</b>	6075
<b>Configuring Redundancy</b>	6075
<b>Customizing a Project using Citect.ini Parameters</b>	6076
<b>General Parameters</b>	6077
<b>Communication Parameters</b>	6079
<b>Data Interrogation Parameters</b>	6082
<b>Write Parameters</b>	6085

Counter Parameters .....	6087
Time Synchronization Parameters .....	6088
Polling Parameters .....	6089
Logging Parameters .....	6089
Input Tags .....	6090
Port-specific Parameters .....	6091
Configuring Commands for Individual IOAs .....	6093
Troubleshooting .....	6094
Driver Statistics .....	6095
Logging .....	6096
Logging Messages .....	6097
Appendix .....	6100
IEC 870-5-104 Interoperability List .....	6100
KNX Driver .....	6110
Safety Information .....	6111
Communicating with KNX Networks .....	6113
Communication Settings .....	6116
Tag Addressing .....	6117
KNX Data Type Mappings .....	6119
KNX Tag Import .....	6125
Advanced Configuration and Maintenance .....	6127
Configuring Redundancy .....	6127
Quality and Timestamps .....	6127
Background Polling .....	6128
Write Requests .....	6128
Data Type Mismatches .....	6128
Maximum Transmission Rate Control .....	6128
Customize a Project using Citect.ini Parameters .....	6129
KNX Parameters .....	6129
Logging Parameters .....	6134
Troubleshooting .....	6135
Logging .....	6135
Specific Errors .....	6137
Frequently Asked Questions .....	6138
MELSECQ Driver .....	6139
Safety Information .....	6139
Supported Devices .....	6141
MelsecQ via TCP/IP .....	6142
MelsecQ - Device Address .....	6142
MelsecQ - Hardware Setup .....	6143
MelsecQ - Hints and Tips .....	6145
MelsecQ - Communication Settings .....	6145
MelsecQ - Data Types .....	6147
MelsecQnA via TCP/IP .....	6149
MelsecQnA - Device Address .....	6149
MelsecQnA - Hardware Setup .....	6150
MelsecQnA - Wiring Diagram 1 .....	6154
MelsecQnA - Wiring Diagram 2 .....	6154
MelsecQnA - Hints and Tips .....	6155

MelsecQnA - Communication Settings .....	6156
MelsecQnA - Data Types .....	6157
Melsec iQ-R via TCP/IP .....	6159
Melsec iQ-R - Device Address .....	6160
Melsec iQ-R - Communications Settings .....	6160
Melsec iQ-R - Data Types .....	6162
Customize a Project using Citect.ini Parameters .....	6164
MELSECQ Driver Parameters .....	6165
Logging Parameters .....	6166
MELSECQ Driver Specific Errors .....	6167
Logging .....	6168
MODBUS Driver .....	6169
Safety Information .....	6169
Supported Devices .....	6171
MODBUS Generic Devices .....	6172
MODBUS Generic Devices - Device Address .....	6172
MODBUS Generic Devices - Hardware Setup .....	6172
MODBUS Generic Devices - Wiring Diagram 1 .....	6173
MODBUS Generic Devices - Wiring Diagram 2 .....	6174
MODBUS Generic Devices - Wiring Diagram 3 .....	6174
MODBUS Generic Devices - Wiring Diagram 4 .....	6175
MODBUS Generic Devices - Communication Settings .....	6175
MODBUS Generic Devices - Data Types .....	6176
Modicon 484 PLCs .....	6177
Modicon 484 PLCs - Device Address .....	6178
Modicon 484 PLCs - Hardware Setup .....	6178
Modicon 484 PLCs - Wiring Diagram 1 .....	6179
Modicon 484 PLCs - Wiring Diagram 2 .....	6179
Modicon 484 PLCs - Wiring Diagram 3 .....	6180
Modicon 484 PLCs - Wiring Diagram 4 .....	6180
Modicon 484 PLCs - Communication Settings .....	6181
Modicon 484 PLCs - Data Types .....	6182
Modicon 584 PLCs .....	6183
Modicon 584 PLCs - Device Address .....	6183
Modicon 584 PLCs - Hardware Setup .....	6183
Modicon 584 PLCs - Wiring Diagram 1 .....	6184
Modicon 584 PLCs - Wiring Diagram 2 .....	6185
Modicon 584 PLCs - Wiring Diagram 3 .....	6185
Modicon 584 PLCs - Wiring Diagram 4 .....	6186
Modicon 584 PLCs - Communication Settings .....	6186
Modicon 584 PLCs - Data Types .....	6187
Modicon 884 PLCs .....	6188
Modicon 884 PLCs - Device Address .....	6189
Modicon 884 PLCs - Hardware Setup .....	6189
Modicon 884 PLCs - Wiring Diagram 1 .....	6190
Modicon 884 PLCs - Wiring Diagram 2 .....	6190
Modicon 884 PLCs - Wiring Diagram 3 .....	6191
Modicon 884 PLCs - Wiring Diagram 4 .....	6191
Modicon 884 PLCs - Communication Settings .....	6192

Modicon 884 PLCs - Data Types .....	6193
Modicon 984 PLCs .....	6194
Modicon 984 PLCs - Device Address .....	6194
Modicon 984 PLCs - Hardware Setup .....	6194
Modicon 984 PLCs - Wiring Diagram 1 .....	6195
Modicon 984 PLCs - Wiring Diagram 2 .....	6196
Modicon 984 PLCs - Wiring Diagram 3 .....	6196
Modicon 984 PLCs - Wiring Diagram 4 .....	6197
Modicon 984 PLCs = Communication Settings .....	6197
Modicon 984 PLCs - Data Types .....	6198
Moore Products IOExpress .....	6199
Moore Products IOExpress - Device Address .....	6200
Moore Products IOExpress - Hardware Setup .....	6200
Moore Products IOExpress - Wiring Diagram 1 .....	6201
Moore Products IOExpress - Wiring Diagram 2 .....	6201
Moore Products IOExpress - Wiring Diagram 3 .....	6202
Moore Products IOExpress - Wiring Diagram 4 .....	6202
Moore Products IOExpress - Communication Settings .....	6203
Moore Products IOExpress - Data Types .....	6204
Working With Non-standard Devices .....	6205
Protocol Variants .....	6205
Customizing a Project using Citect.ini Parameters .....	6206
MODBUS Driver Parameters .....	6207
Device/Group-specific Parameters .....	6214
MODBUS Driver Errors .....	6216
MODBUSA Driver .....	6217
Safety Information .....	6217
Supported Devices .....	6219
Generic Devices .....	6219
Generic Devices - Device Address .....	6220
Generic Devices - Hardware Setup .....	6220
Generic Devices - Wiring Diagram 1 .....	6221
Generic Devices - Wiring Diagram 2 .....	6221
Generic Devices - Wiring Diagram 3 .....	6222
Generic Devices - Wiring Diagram 4 .....	6222
Generic Devices - Communications Settings .....	6223
Generic Devices - Data Types .....	6224
Working With Non-standard Devices .....	6225
Protocol Variants .....	6226
Customizing a Project using Citect.ini Parameters .....	6227
MODBUSA Driver Parameters .....	6227
Device/group-specific parameters .....	6234
MODBUSA Driver Errors .....	6236
MODNET Driver .....	6237
Safety Information .....	6237
Supported Devices .....	6239
Generic MODBUS TCP/IP Devices .....	6239
Generic MODBUS TCP/IP Devices - Device Address .....	6240
Generic MODBUS TCP/IP Devices - Hardware Setup .....	6240

Generic MODBUS TCP/IP Devices - Communication Settings .....	6240
Generic MODBUS TCP/IP Devices - Data Types .....	6242
Generic MODBUS TCP/IP Devices - Array Support .....	6244
M340, TSX Quantum™ and Premium™ PLCs .....	6245
M340, TSX Quantum and Premium PLCs - Device Address .....	6245
M340, TSX Quantum and Premium PLCs - Hardware Setup .....	6246
M340, TSX Quantum and Premium PLCs - Communication Settings .....	6246
M340, TSX Quantum and Premium PLCs - Data Types .....	6247
MODNET Protocol Variants .....	6249
MODNET Driver Parameters .....	6250
Device/Group-specific Parameters .....	6262
MODNET Driver Specific Errors .....	6264
OFSCOPC Driver .....	6265
Safety Information .....	6266
System Requirements .....	6269
Setting up Device Communication .....	6269
Device Address .....	6269
Communication Settings .....	6270
Data Dictionary Support .....	6271
Configuring Variable Tags .....	6272
Array Support .....	6273
Timestamp and Quality Support .....	6274
Tag Addressing for Derived Data Types (DDT) .....	6274
Poll Engine for #Specific Tags .....	6275
Advanced Configuration and Maintenance .....	6275
Configuring Redundancy .....	6275
Using the Batch-write Mechanism .....	6276
Push Data Support .....	6277
OFS Group Management .....	6277
Source Timestamped Events Support .....	6277
TS Quality Support .....	6278
Support for Version 7.20 .....	6279
Support for Version 7.30 .....	6280
Timestamp Event Replication .....	6281
Diagnostic Buffer Support .....	6281
Record Browsing Interface .....	6281
GetNextEvent .....	6282
GetEventField .....	6282
Device Resynchronization Interface .....	6283
GetResendState .....	6283
ResendAll .....	6284
Customizing a Project using Citect.ini Parameters .....	6284
Driver-specific Parameters .....	6285
Group Management Parameters .....	6286
Logging Parameters .....	6287
BatchWrite Parameters .....	6288
Diagnostic Buffer Parameters .....	6289
Dynamic Status Item Parameters .....	6289
Performance Tuning .....	6291

Tuning Parameters .....	6291
OFS Server Configuration .....	6291
Device Settings .....	6293
Communications Settings .....	6293
OFSOPC Driver Parameters .....	6294
Troubleshooting .....	6296
Driver Errors .....	6296
OFSOPC Driver Specific Errors .....	6296
Driver Statistics .....	6298
Logging .....	6299
OMFINS Driver .....	6301
Safety Information .....	6301
Communicating with Omron PLCs .....	6303
Omron PLCs via Ethernet .....	6304
Device Address .....	6305
Unit Address .....	6305
Hardware Setup .....	6306
Communication Settings .....	6306
Data Types .....	6308
Omron 'Series 1' Devices .....	6309
Omron 'Series 2' Devices .....	6310
Omron 'Series 3' Devices .....	6313
OMFINS Parameters .....	6314
Port/IO Device-specific Parameters .....	6316
Protocol-specific Parameters .....	6318
OMFINS Driver Errors .....	6318
OMFINS Specific Errors .....	6319
OPC Driver .....	6324
Safety Information .....	6324
Preparing the OPC System .....	6326
Hardware Requirements .....	6326
Software Requirements .....	6327
OPC Devices and Clients .....	6327
Device Address Information .....	6327
Communication Settings .....	6328
Data Types .....	6330
Include Project Support .....	6331
Arrays Support .....	6331
Quality Values .....	6333
Timestamp and Quality Support .....	6334
Advanced Configuration and Maintenance .....	6336
Status Tags .....	6336
Customizing a Project using Citect.ini Parameters .....	6337
Common Driver Parameters .....	6338
Driver-specific Parameters .....	6339
Logging Parameters .....	6347
OPC Device-specific Parameters .....	6348
OPC Access Path Parameters .....	6350
OPC Status Tag Parameters .....	6351

Overriding [OPC]CacheRead .....	6354
Configuring Redundancy .....	6354
Scan Rates .....	6354
Troubleshooting .....	6355
Driver Errors .....	6355
Common Protocol Errors .....	6355
OPC Protocol Errors .....	6356
Driver Statistics .....	6357
Logging .....	6358
Maintaining the Project Database .....	6360
OPCLX Driver .....	6360
Safety Information .....	6361
Prepare the OPC System .....	6363
OPC Devices and Clients .....	6364
Device Address .....	6364
Communication Settings .....	6364
Data Types .....	6367
Include Project Support .....	6368
Configure Variable Tags .....	6368
Tag Address Configuration .....	6368
Arrays Support .....	6370
Quality Values .....	6373
Time Stamp and Quality Tags Configuration .....	6374
Advanced Configuration and Maintenance .....	6377
Status Tags .....	6377
Customize a Project Using Citect.ini Parameters .....	6378
Common Driver Parameters .....	6378
Driver-specific Parameters .....	6380
OPCLX Device-specific Parameters .....	6389
OPCLX Access Path Parameters .....	6391
Status Tag Parameters .....	6391
Logging Parameters .....	6393
Overriding [OPCLX]CacheRead .....	6394
Configure Redundancy .....	6394
Scan Rates .....	6395
Diagnostics .....	6395
Migrate from ABCLX to OPCLX .....	6396
Migrate a Project from ABCLX to OPCLX .....	6396
Migration of I/O Devices .....	6398
Tag Address Formats .....	6399
Troubleshooting .....	6401
Detected Driver Errors .....	6401
Detected Common Protocol Errors .....	6401
Detected OPCLX Protocol Errors .....	6401
Driver Statistics .....	6403
Logging .....	6404
Maintaining the Project Database .....	6405
OPCUA Driver .....	6406
Safety Information .....	6406

Third Party Software .....	6408
Manually Install PCS Framework .....	6418
Setting up Device Communications .....	6419
Communication Settings .....	6419
Configuring Variable Tags .....	6421
Array Support .....	6423
Extension Object Support .....	6424
Advanced Configuration and Maintenance .....	6424
Configuring Redundancy .....	6425
Subscription Management .....	6425
Security Configuration .....	6425
Customizing a Project using Citect.ini Parameters .....	6428
Connection Parameters .....	6429
Subscription Parameters .....	6431
Security Parameters .....	6433
Logging Parameters .....	6434
Device-specific Parameters .....	6435
Tag Import .....	6435
Troubleshooting .....	6436
OPC UA Server Connection .....	6437
Driver Errors .....	6437
Driver Statistics .....	6438
Logging .....	6438
S7TCP Driver .....	6439
Safety Information .....	6440
Supported Devices .....	6442
Setting up Device Communications .....	6443
Device Address .....	6443
Communication Settings .....	6443
Configuring Variable Tags .....	6445
Advanced Configuration and Maintenance .....	6447
Configuring Redundancy .....	6447
Customizing a Project using Citect.ini Parameters .....	6448
S7TCP Driver Parameters .....	6449
Logging Parameters .....	6451
Using Device or Port-specific Parameters .....	6452
Tag Import .....	6453
Troubleshooting .....	6454
Driver Errors .....	6454
Driver Statistics .....	6454
Logging .....	6455
SBUS Driver .....	6456
Safety Information .....	6457
SAIA PCD PLCs .....	6458
Device Address .....	6459
Hardware Setup .....	6459
Software Setup .....	6460
Wiring Diagram .....	6461
Communication Settings .....	6461

Data Types .....	6462
SBUS Driver Parameters .....	6463
Port-specific Parameters .....	6466
SBUS Driver Specific Errors .....	6466
SNMPII Driver .....	6467
Safety Information .....	6467
Preparing the SNMPII System .....	6469
Hardware Requirements .....	6469
Installing the SNMP Service .....	6470
Setup Tips and Techniques .....	6470
Configuration Overview .....	6471
SNMP Redundancy .....	6473
SNMP Enterprise Numbers .....	6474
SNMPII Driver Recommended Settings .....	6474
Device Address .....	6474
Communication Settings .....	6474
Data Types .....	6476
Numeric Data Grouping Considerations .....	6478
SNMPII Protocol-specific Parameters .....	6478
Driver-Specific Parameters .....	6479
I/O Device-specific Parameters .....	6482
SNMPII Blocking .....	6483
Debug Messages .....	6483
Driver-generated Statistics .....	6483
SNMP Traps .....	6484
Structure of a Trap .....	6484
Special Tags for Reading and Creating Traps .....	6485
Reading Queued Traps .....	6486
Trap Forwarding .....	6487
Generating Traps .....	6487
Queue Management .....	6487
Notes .....	6488
Troubleshooting .....	6488
Driver Errors .....	6488
Kernel Diagnostics .....	6489
Kernel I/O Devices Window .....	6490
Kernel Main Window .....	6493
Tag-based Driver Considerations .....	6494
Project IDs .....	6494
Variable Databases .....	6494

# Welcome to AVEVA Plant SCADA

AVEVA Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution that is used to manage and monitor processes in manufacturing, primary production, utilities delivery and facilities management.

The easy-to-use configuration tools and powerful features help customers to develop and deploy solutions for small and large applications, with robust visualization and operational capabilities driving operational efficiency, helping to mitigate risk and deliver actionable insights faster.

Known for its reliability, flexibility and scalability, and used in a wide range of industries, Plant SCADA enables you to increase your return on assets by delivering highly scalable control and monitoring systems to:

- Reduce operating costs
- Improve productivity
- Improve product quality.

Plant SCADA is a leading industrial automation software platform, with over 30 years of industrial automation expertise and an extensive customer-base spanning several market segments. Plant SCADA's library of device and protocol drivers to PLC and RTU hardware available in the market as well as interoperability with AVEVA's portfolio of Engineering and Industrial Automation software make Plant SCADA an extremely versatile solution.

To learn more about Plant SCADA, see [About Plant SCADA](#).

## Legal information

2015-2024 AVEVA Group Limited or its subsidiaries. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, photocopying, recording, or otherwise, without the prior written permission of AVEVA Group Limited. No liability is assumed with respect to the use of the information contained herein.

Although precaution has been taken in the preparation of this documentation, AVEVA assumes no responsibility for errors or omissions. The information in this documentation is subject to change without notice and does not represent a commitment on the part of AVEVA. The software described in this documentation is furnished under a license agreement. This software may be used or copied only in accordance with the terms of such license agreement. AVEVA, the AVEVA logo and logotype, OSIsoft, the OSIsoft logo and logotype, ArchestrA, Avantis, Citect, DYN SIM, eDNA, EYESIM, InBatch, InduSoft, InStep, Intelatrac, InTouch, Managed PI, OASyS, OSIsoft Advanced Services, OSIsoft Cloud Services, OSIsoft Connected Services, OSIsoft EDS, PIPEPHASE, PI ACE, PI Advanced Computing Engine, PI AF SDK, PI API, PI Asset Framework, PI Audit Viewer, PI Builder, PI Cloud Connect, PI Connectors, PI Data Archive, PI DataLink, PI DataLink Server, PI Developers Club, PI Integrator for Business Analytics, PI Interfaces, PI JDBC Driver, PI Manual Logger, PI Notifications, PI ODBC Driver, PI OLEDB Enterprise, PI OLEDB Provider, PI OPC DA Server, PI OPC HDA Server, PI ProcessBook, PI SDK, PI Server, PI Square, PI System, PI System Access, PI Vision, PI Visualization Suite, PI Web API, PI WebParts, PI Web Services, PRISM, PRO/II, PROVISION, ROMeo, RLINK, RtReports, SIM4ME, SimCentral, SimSci, Skelta, SmartGlance, Spiral Software, WindowMaker, WindowViewer, and Wonderware are trademarks of AVEVA and/or its subsidiaries. All other brands may be trademarks of their respective owners.

### U.S. GOVERNMENT RIGHTS

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the license agreement with AVEVA Group Limited or its subsidiaries and as provided in DFARS 227.7202, DFARS 252.227-7013, FAR 12-212, FAR 52.227-19, or their successors, as applicable.

AVEVA Legal Resources: <https://www.aveva.com/en/legal/>

AVEVA Third Party Software Notices and Licenses: <https://www.aveva.com/en/legal/third-party-software-license/>

Publication date: Monday, December 16, 2024

Publication ID: 1356401

## Contact information

AVEVA Group Limited

High Cross  
Madingley Road  
Cambridge  
CB3 0HB. UK

<https://sw.aveva.com/>

For information on how to contact sales and customer training, see <https://sw.aveva.com/contact>.

For information on how to contact technical support, see <https://sw.aveva.com/support>.

To access the AVEVA Knowledge and Support center, visit <https://softwaresupport.aveva.com>.

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

### DANGER

**DANGER** indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.

**⚠ WARNING**

**WARNING** indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.

**⚠ CAUTION**

**CAUTION** indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.

**NOTICE**

**NOTICE** used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.

**Please Note**

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

**Before You Begin**

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ WARNING****LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

## Security Guidelines for Industrial Control Systems

The security guidelines provided here will assist you to securely deploy the Plant SCADA product as an Industrial Control Systems (ICS) application.

**Note:** This section is not meant to be comprehensive, and it does not provide any detailed instructions. It is only a collection of basic concepts and recommendations that you can use as a checklist to secure your own systems.

### See Also

[General considerations for security](#)

## General considerations for security

Before you review the information in this section, it is recommended that you go through the following checklist to ensure you plan to cover the security areas that apply to your ICS and organization.

Security Area	Reference Section
Physical and virtual access to the host	<a href="#">General guidelines for securing the host</a>
Latest Windows patches applied	<a href="#">Windows updates</a>
Protecting the host from viruses and malware	<a href="#">Scanning the Host</a>

Security Area	Reference Section
Access to content on the host	Protect the applications and content on the host
Securing your network	Secure the network
Configuring services and ports	Manage network services and ports
Securing client/server communication	Secure communication between the client and server
User and group management	Secure systems through authentication and authorization
Planning for emergencies	Plan contingency

For a list of security feature help topics refer to the table at the end of this section.

## Introduction

This appendix provides a general overview on how to securely deploy your AVEVA software product as an Industrial Control Systems (ICS) application.

**This appendix is not meant to be comprehensive, and it does not provide any detailed instructions.** It is only a collection of basic concepts and recommendations that you can use as a checklist to secure your own systems. If you need help with a specific item in this guide, see the official documentation for that item -- for example, if you need help with your anti-virus software, see the documentation for that software.

AVEVA's approach to securing site networks and ICS software is driven by the following principles:

- View security from both Management and Technical perspectives
- Ensure that security is addressed from both IT and ICS perspectives.
- Design and develop multiple network, system and software security layers.
- Ensure industry, regulatory and international standards are taken into account.
- Aim to prevent security breaches, supported by detection and mitigation.

These principles are realized by implementing the following security recommendations:

- Prevent security breaches using the following components:
  - Firewalls
  - Network-based intrusion prevention/detection
  - Host-based intrusion prevention/detection
- Segregate IT and Plant networks
- Include a clearly defined and clearly communicated change management policy. For example, firewall configuration changes.

**Note:** AVEVA strongly recommends following the guidelines prescribed by the U.S. Department of Commerce for securing ICS software. The document "Guide to Industrial Control Systems (ICS) Security" [NIST Special Publication 800-82 Revision 2] (<https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r2.pdf>) provides detailed information about ICS, typical system topologies, security threats and vulnerabilities, and

---

recommendations for implementing security measures.

---

## Secure the host

Given the sensitive nature of industrial control, it is important to secure not only the ICS software, but also:

- the host on which it runs
- the network to which it is connected
- the hardware used for the ICS software.

---

**Note:** The "host" is the Windows computer or Windows Embedded device on which your ICS software is installed and running.

---

There are several factors to consider for securing the host including:

- Access to the host
- Keeping track of and applying the latest Windows updates
- Keeping the host computer free of viruses and malware
- Protecting the applications and content on the host

Each of these factors is covered in the sections below.

## General guidelines for securing the host

Here are a few guidelines to secure the host:

- Use an account with administrative privileges to install the ICS software, and one without administrative privileges to run the ICS software.
- Restrict configuration of ICS to a limited set of users.
- Consider running the ICS software as a Windows service, if that option is available. If the ICS software is run as a service, run it as a low privileged virtual service account.
- Once the host is fully configured and placed in its permanent location, restrict physical access and remote access to it so that only authorized personnel (for example, system administrators, application engineers, run-time operators) can use it.
- Consider disabling or removing physical ports (for example, USB, memory card) that might be used to connect external storage devices and then transfer data.

## Windows updates

Check that the Windows operating system on the host is a version that is under what Microsoft calls "mainstream support", which means Microsoft actively maintains and releases updates for it. Older versions of Windows are under Microsoft "extended support", which means they are not actively maintained and therefore might become vulnerable without notice. For more information about the different versions of Windows and the different levels of support, see [Windows lifecycle fact sheet](<https://support.microsoft.com/en-us/help/13853/windows-lifecycle-fact-sheet>).

Automate Microsoft product updates using Microsoft Windows Server Update Services (WSUS), which enables you to manage and distribute updates to computers on your network. For more information about WSUS, see [Windows Server Update Services](<https://docs.microsoft.com/en-us/windows-server/administration/windows-server-update-services/get-started/windows-server-update-services-wsus>). If the host does not or will not have a reliable connection to the WSUS server, perhaps because it is located on a private network, you can either develop a procedure to manually apply updates or consider changing the operating system to a Long-Term Servicing Channel (LTSC) version of Windows, which is updated less frequently.

In addition, AVEVA ICS software is tested for compatibility with Microsoft updates the results of which are published on the Security Central site (<https://softwaresupportsp.aveva.com/#/securitycentral>). Security advisories and bulletins are also published on this site.

## ICS software updates

Check that the ICS software on the host has all the recommended patches and hot fixes installed.

Some AVEVA applications release regular updates, which should be applied as soon as possible as they may contain security-related fixes.

---

**Note:** AVEVA's Global Customer Support (GCS) group publishes a Technology Matrix (<https://gcsresource.aveva.com/TechnologyMatrix/Home/Index>) for AVEVA software products. This matrix lists the Windows operating system versions against which a software product has been tested for compatibility. In addition, it lists compatible runtime, browser, and virtualization environments for the software. It also includes a list of other products that can be installed on the same computer and lists other products with which this software can communicate.

---

## Scanning the Host

Use both anti-virus and anti-malware software and file integrity checking software to regularly scan the host.

Windows includes Windows Defender by default, but you may choose to install and use additional software that scans for more types of malware or performs other functions. If you do that, make sure the software is provided by a reputable company. And, as with the operating system, if the host does not or will not have reliable access to the software's update service, develop a procedure to manually apply updates. If you develop a manual update procedure, it should account for all devices on a network or at a site, because a single outdated device can leave the entire network or site vulnerable.

## Protect the applications and content on the host

To protect the applications and content on the host:

- Enable Windows Firewall, and configure it to close all ports that are not used by the ICS software. For more information about port usage, see [Manage network services and ports](#).
- Disable Windows features like remote desktop and file sharing, and remove unnecessary programs like games and social media.
- Restrict access to the files, databases, registry and other resources on the host.
- Use Windows BitLocker to encrypt the hard drive of computers that are either mobile or not located in a secure facility. However, BitLocker may impact the performance of computers.
- Consider using server-class storage (SANs) infrastructure to avoid storing sensitive data on mobile devices.

- If your application stores data in SQL Server, Windows authentication can provide better application security than SQL Authentication. If you switch from Windows Authentication to SQL Authentication, a pop up dialog will appear recommending that you use Windows Authentication for this reason. If you choose to ignore this warning and proceed with SQL Authentication, select **OK**. A similar message will be logged in the OCMC (SMC) Log Viewer.

AVEVA leverages the security built into the Windows operating system to store and manage encryption keys. The encryption keys are stored in a local storage location called the encryption store. For more information about the Windows encryption store, refer to the Microsoft documentation, located at Certificate Stores (<https://docs.microsoft.com/en-us/windows-hardware/drivers/install/certificate-stores>).

## Phases of Data Protection

Data exists in three different phases, and protection must be provided for each phase:

- At rest
- In transit
- In use

### Data at rest

Data at rest is data that is not currently being used or accessed, such as data stored on a hard drive, laptop, flash drive, RAID array, network attached storage (NAS), storage area network (SAN), or archived/stored in some other way. Data protection at rest aims to secure inactive data stored on any device or network. For protecting data at rest, you can simply encrypt sensitive files prior to storing them and/or choose to encrypt the storage drive itself. BitLocker Drive Encryption, which you can invoke via the Windows Control Panel, can be used to invoke whole-drive encryption.

In the context of SCADA and ICS systems, data at rest includes stored configuration data, historical data, backups, and other static data. The duration of storage, that is, long term or short term, does not impact this classification of data at rest. Protection for data at rest is applicable for as long as the data exists in this condition; it is not a fixed condition.

Proper authorization rights need to be set in place to ensure the data is not being viewed by unauthorised users. Other steps can also help, such as storing individual data elements in separate locations, such as a corporate-approved offline backup to decrease the likelihood of attackers gaining enough information to commit fraud or other crimes. Offline backups are the best mitigation against the threat of ransomware.

### Data in transit

Data in transit, or data in motion, is data that is actively moving from one location to another.

In the context of SCADA and ICS systems, this encompasses deploying a project to a run-time node, transmitting process variables, VTQ data, and other data that is sent between nodes in a running, production system. This includes alerts and alarms.

Data protection in transit is the protection of this data while the data traveling, including the following examples:

- From node to node within a network
- From network to network

- Accessed via internet
- Transferred from a local storage device to a cloud storage device

Wherever data is moving, effective data protection measures for in-transit data are critical as data is often considered less secure while in motion. Best security practice is to ensure TLS 1.2 encryption is used for all communications using the HTTPS protocol.

## Data in use

Data in use refers to data that is being processed or accessed either locally or remotely. This generally involves placing data into memory (RAM) for access and processing by applications and users, potentially multiple users across different computers, mobile devices, remote terminals or other device. Data in use is particularly vulnerable to attack. To protect data in use, encryption, user authentication, and identity management is highly recommended.

In the context of SCADA and ICS systems, data in use can apply to databases, such as those used actively by a historian or deployed to a run-time node. This needs to be safeguarded by a secure transfer channel.

## Configure encryption in SQL server

We recommend you enable encrypted connections for SQL Server. You enable encrypted connections for an instance of the SQL Server Database Engine and use SQL Server Configuration Manager to specify a certificate. The server computer must have a certificate provisioned. To provision the certificate on the server computer, you import it into Windows. The client machine must be set up to trust the certificate's root authority.

SQL Server can use Transport Layer Security (TLS) to encrypt data that is transmitted across a network between an instance of SQL Server and a client application. The TLS encryption is performed within the protocol layer and is available to all supported SQL Server clients.

Enabling TLS encryption increases the security of data transmitted across networks between instances of SQL Server and applications. However, when all traffic between SQL Server and a client application is encrypted using TLS, the following additional processing is required:

- An extra network round trip is required at connect time.
- Packets sent from the application to the instance of SQL Server must be encrypted by the client TLS stack and decrypted by the server TLS stack.
- Packets sent from the instance of SQL Server to the application must be encrypted by the server TLS stack and decrypted by the client TLS stack.

## Certificate Requirements

For SQL Server to load a TLS certificate, the certificate must meet the following conditions:

- The certificate must be in either the local computer certificate store or the current user certificate store.
- The SQL Server Service Account must have the necessary permission to access the TLS certificate.
- The current system time must be after the **Valid from** property of the certificate and before the **Valid to** property of the certificate.

## Install on a Server

With SQL Server 2019 (15.x), certificate management is integrated into the SQL Server Configuration Manager. SQL Server Configuration Manager for SQL Server 2019 (15.x) can be used with earlier versions of SQL Server.

If using SQL Server 2012 (11.x) through SQL Server 2017 (14.x), and SQL Server Configuration Manager for SQL Server 2019 (15.x) is not available, follow these steps:

1. On the **Start** menu, select **Run**, and in the **Open** box, type **MMC** and select **OK**.
2. In the MMC console, on the **File** menu, select **Add/Remove Snap-in**.
3. In the **Add/Remove Snap-in** dialog box, select **Add**.
4. In the **Add Standalone Snap-in** dialog box, select **Certificates**, select **Add**.
5. In the **Certificates snap-in** dialog box, select **Computer account**, and then select **Finish**.
6. In the **Add Standalone Snap-in** dialog box, select **Close**.
7. In the **Add/Remove Snap-in** dialog box, select **OK**.
8. In the **Certificates snap-in**, expand **Certificates**, and then expand **Personal**.
9. Right-click **Certificates**, point to **All Tasks**, and then click **Import**.
10. Right-click the imported certificate, point to **All Tasks** and then select **Manage Private Keys**. In the **Security** dialog box, add read permission for the user account used by the SQL Server service account.
11. Complete the **Certificate Import Wizard**, to add a certificate to the computer, and close the MMC console. For more information about adding a certificate to a computer, see your Windows documentation.

## Export Server Certificate

To export the server certificate:

1. From the **Certificates** snap-in, locate the certificate in the **Certificates / Personal** folder.
2. Right-click the **Certificate**, point to **All Tasks**, and then select **Export**.
3. Complete the **Certificate Export Wizard**, storing the certificate file in a convenient location.

## Configure Server

Configure the server to force encrypted connections. The SQL Server service account must have read permissions on the certificate used to force encryption on the SQL Server. For a non-privileged service account, read permissions will need to be added to the certificate. Failure to do so can cause the SQL Server service restart to fail.

To configure the server:

1. In **SQL Server Configuration Manager**, expand **SQL Server Network Configuration**, right-click **Protocols for <server instance>**, and then select **Properties**.
2. In the **Protocols for <instance name> Properties** dialog box, on the **Certificate** tab, select the desired certificate from the drop-down for the **Certificate** box, and then select **OK**.
3. On the **Flags** tab, in the **ForceEncryption** box, select **Yes**, and then select **OK** to close the dialog box.
4. Restart the SQL Server service.

## Configure Client

Configure the client to request encrypted connections.

1. Copy either the original certificate or the exported certificate file to the client computer.
2. On the client computer, use the **Certificates** snap-in to install either the root certificate or the exported certificate file.
3. Using SQL Server Configuration Manager, right-click **SQL Server Native Client Configuration**, and then select **Properties**.
4. On the Flags page, in the **Force protocol encryption** box, select **Yes**.

---

**Note:** The sections in this topic have been derived from the Microsoft documentation. For more information, refer to the topic 'Enable encrypted connections to the Database Engine' in Microsoft Documentation.

---

## Secure the network

Usually, the host computer will have some sort of network access; it is increasingly rare for an ICS device to run as an entirely standalone device. The host may use the network to communicate with other ICS components such as controllers, sensors, databases, remote clients, and even other hosts in peer-to-peer relationships. You may also use the network to manage several ICS devices from a development or supervisory computer.

Once you determine that the host will have network access, decide how it will connect to the network. In recent years there has been a shift from wired networks (that is, "Ethernet") to wireless networks ("Wi-Fi"), even for business and industrial uses. We recommend against using Wi-Fi for your ICS network because you do not have physical control over who or what might access the network. Any computer or device within range of the Wireless Access Point (WAP) can try to access the network, and even if the network is ostensibly secure, an intruder can intercept and analyze network traffic and potentially discover a vulnerability.

Nevertheless, if you decide to use Wi-Fi for your ICS network, enable all access control features on the WAP including encryption (for example, WPA/WPA2), a strong password and a list of authorized MAC addresses. Do not try to "hide" the Wi-Fi network by disabling broadcast of the Service Set Identifier (SSID), because doing so actually generates more network traffic that can be intercepted and analysed.

## Segment the ICS network

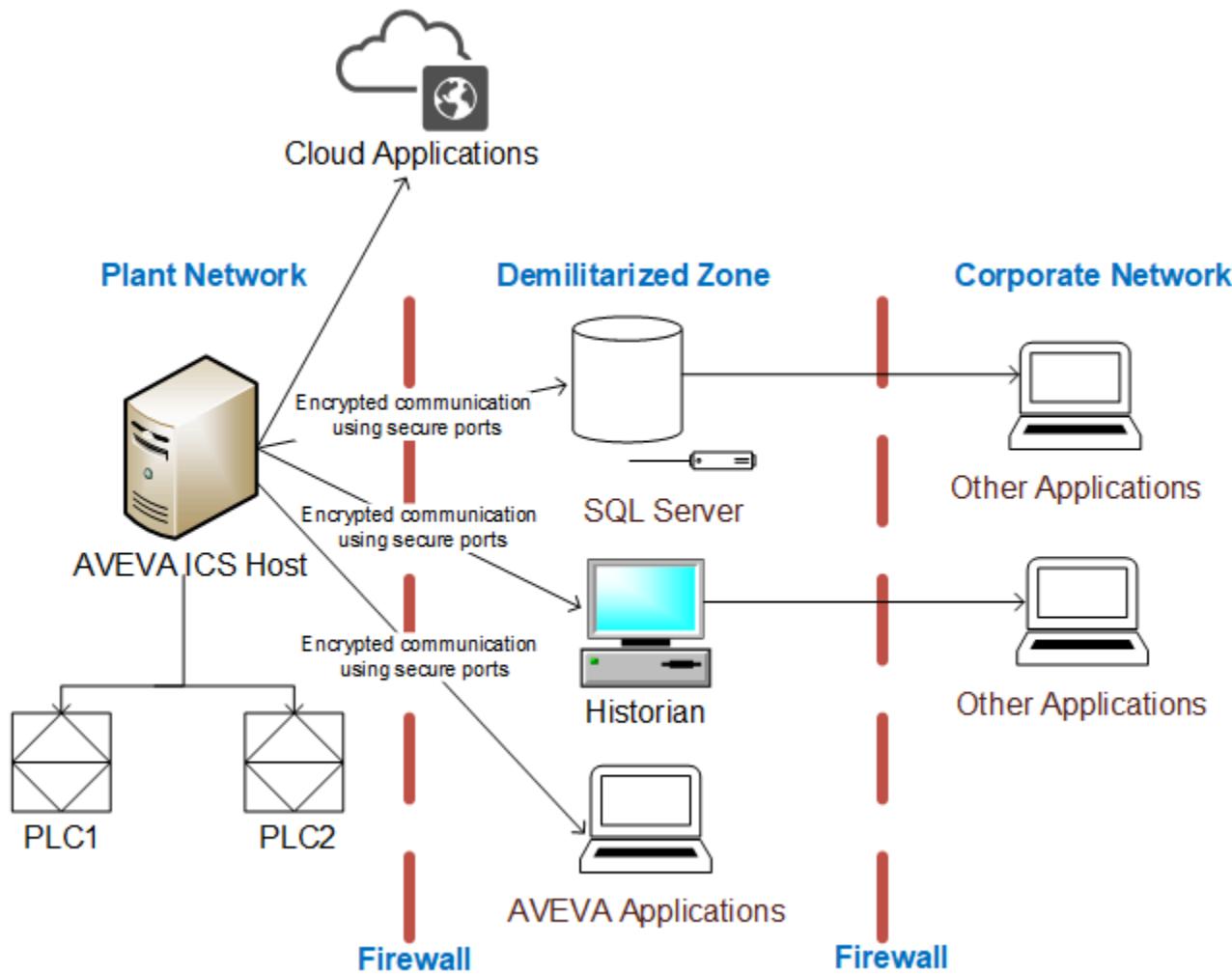
The ICS network itself can be either physically or logically segmented from your other corporate networks. A physically segmented network is by definition the most secure. The network hardware and all computers and devices connected to it form a single closed network with no physical connection to any other network, so an intruder cannot access the network unless they also have access to the physical location.

In contrast, a logically segmented network is physically connected to your other corporate networks and/or the public internet, but it uses various methods to segregate ICS network traffic from other network traffic. This may include:

- Using a unidirectional gateway
- Implementing a Demilitarized Zone (DMZ) network architecture with firewalls to prevent network traffic from passing directly between the corporate and ICS networks
- Having different authentication mechanisms and credentials for users of the corporate and ICS networks.

- The ICS should also use a network topology that has multiple layers, with the most critical communications occurring in the most secure and reliable layer.

Given below is a sample deployment topology.



In no case should your ICS network and devices be directly accessible from the public internet. If there is some part of your ICS that you want to be accessible, (for example, if you want be able to view web-enabled HMI screens from a browser or smart phone), your ICS software should include features that securely pass the necessary traffic between your ICS network and a public-facing server.

## Manage network services and ports

A network port is an endpoint of communication in an operating system. While the term is also used for hardware devices, in software it is a logical construct that identifies a specific process or a type of service. In other words, a network port is conceptually different from hardware ports like USB, memory card, and even the wired network connection.

Computers and devices can access many different network services at the same time by communicating on different network ports. Each network service or communication protocol has an associated port number. Some port numbers are specified by international standards, and therefore they are universally recognized. Other port numbers are claimed by proprietary software, and in most cases they can be changed in the software settings if

there is a conflict with other software or services.

Firewalls control network traffic by either accepting or refusing communication on these network ports. If a port is open, it accepts communication, and if a port is closed, it refuses communication. Almost every layer of a network -- from the operating system on an individual computer or device, to the router that manages traffic within a network, to the gateway that manages traffic between networks -- has its own firewall.

The documentation for your ICS software should include a list of network ports that are commonly used by the software. Given the nature of ICS, the list typically includes services like web, email, file transfer, external databases, device drivers, and the ICS software itself for server-client communications. Configure the firewalls to open only those network ports that are actually used by your ICS. Disable all unused services and close all unused ports.

## Secure communication between the client and server

Like most server-client applications, your ICS software should support secure communication between the server and client in order to prevent the messages sent between those two stations from being read by any other stations on the same network. Note that this is different from securing the network itself in order to prevent unauthorized access to the network.

This sort of communication is also sometimes known as "Encrypted Channel" because it uses the Transport Layer Security (TLS) standard to encrypt the server-client messages. The latest version of the standard is TLS 1.3 (released August 2018), but it is not yet in common use. The latest version of the standard in common use is TLS 1.2 (released August 2008). TLS supersedes the earlier Secure Sockets Layer (SSL) standard, although SSL is still used in older applications.

### Certificates

TLS and SSL use a system of certificates and keys to digitally "sign" the messages sent between the server and client. When the server establishes communication with the client (and vice versa), it presents its certificate which identifies its name, network address, organization, physical location, and so on. The client can then choose to either accept or refuse the certificate as presented. If it accepts the certificate, it agrees to accept messages encrypted with the same certificate, and it uses the associated key to decrypt those messages.

When you configure this sort of communication, you need to choose one of the following:

- Using self-signed certificate
- Using certificates signed by a Public Certificate Authority (CA)
- Using Domain-issued certificates or certificates signed by a Private Certificate Authority using systems like Microsoft Active Directory Certificate Service (AD CS)

A self-signed certificate is issued and signed by the same application that presents it. Self-signed certificates are easy to create and manage, but they are secure only if you control both the server and the client and therefore control which certificates are installed on each.

In contrast, CA-signed certificates are slightly difficult and expensive to acquire, but they are more flexible than self-signed certificates because you do not need to control both the server and the client. If you configure the server to present a CA-signed certificate, the client will accept the certificate because it recognizes the Certificate Authority.

Domain-issued certificates are internal certificates typically managed by your IT department. They are issued and validated by an Active Directory Certificate Authority. Domain-issued certificates are free and can be issued instantly.

You need to renew CA-signed and Domain-issued certificates at regular intervals.

For more information about how to enable Encrypted Channel features and manage self-signed certificates in your ICS software, see the documentation for that software. However, acquiring a CA-signed certificate and then using it to sign other certificates is typically beyond the scope of ICS software documentation.

---

**Note:** Encrypted and unencrypted communications typically use different network ports.

---

## Cloud-based systems

It is possible that your ICS software might access cloud-based solutions, or might itself be hosted on the Cloud. It is important to mitigate the risks associated with cloud-based access and hosting.

### Access cloud-based solutions

Several AVEVA applications are now being made available through the Cloud, and ICS software may need to connect to these applications. One of the main risks associated with accessing cloud-based applications is unauthorized access. Connecting ICS software to Cloud solutions must be done in a secure manner, and needs to use secure protocols such as Transport Layer Security (TLS).

It is important that data integrity is maintained at all times. Use data classification to identify data that is sensitive and data that can be made public. Secure computers, storage and networking in order to secure the data that is stored and transmitted. Work with your Cloud Service Provider (CSP) to configure users, assign access levels and monitor and control access. Ensure that the CSP's buildings are physically secure and protected from unauthorized access.

### Cloud-based ICS software

While hosting ICS software on the Cloud provides several benefits such as flexibility, scalability and availability, it is also fraught with security risks such as susceptibility to hacking resulting in damage to the organization's reputation. Therefore, it is important to implement a security strategy before you make your ICS software accessible on the Cloud. For securing ICS software on the Cloud, you need to consider the following:

- Securing access points by putting in place authentication, monitoring and support mechanisms.
- Implementing cloud-based, centralized security measures including encrypting communications using TLS.

---

**Note:** It is recommended that you review the NIST Cybersecurity Framework (<https://www.nist.gov/cyberframework>) for additional information.

---

## Secure systems through authentication and authorization

Typically, ICS software is comprised of a large number of systems, each accessed by a variety of users including engineers, operators and managers. The level of access that each type of user requires is different. So, it is necessary to manage user authentication and authorization to secure the system.

### Authentication

Authentication is the process of verifying a user's/system's identity. Authentication can be managed in the following ways:

- Within the ICS software through application accounts
- Through Windows accounts, which can be local to a single computer
- Through Authentication systems (see the next section for details)

While ICS software allows for user and role management, it can become cumbersome and complicated to manage a large number of user accounts as employees and roles change. Because of this, use of Windows accounts is generally preferred.

### **Authentication systems**

Authentication systems such as Active Directory and Lightweight Directory Access Protocol (LDAP), referred to as authentication servers, are a repository of and provide centralized management for all system accounts and individual user accounts. An authentication protocol is used for all communication between authentication servers and the user or server requesting authentication.

Even though use of authentication systems provides improved scalability, the following factors must be considered depending upon the size and complexity of your operations:

- It is important that the authentication servers are highly secured.
- The authentication server system creates a single system for managing all system accounts. Therefore, it requires to be available at all times. To ensure minimal disruption during an emergency, redundancy must be considered.
- Permit caching of user credentials only for users who have authenticated their identity recently.
- Networks that support the authentication protocol must be reliable and secure to assist in trouble-free authentication.

It may also be worthwhile implementing two-factor authentication using additional applications such as PingID.

### **Authorization**

Authorization is the process of providing the correct level of privileges to users by applying access rules to authenticated users, systems (HMI, field devices and SCADA servers) and networks (remote sites' LANs).

## Manage users and groups through windows

When you configure security, you may choose to do one of the following:

- Keep the configuration local to a single application.
- Share the configuration between multiple applications.
- Manage the configuration as part of the network domain (for example, using Active Directory). This option typically allows users to have the same user account for the network, the host, and the ICS software. Using Active Directory gives you the following advantages:
  - A centralized repository for user and group data, enabling effective implementation of security policies and procedures.
  - Provides a single point of access to all network resources after the user is identified and authenticated.

To manage users and groups:

- First define a specific role for each group, and then configure the group privileges to fit that role.
- Groups may overlap, but it is often better to have clearly separate groups and then assign individual users to

multiple groups, if necessary.

- Set or change the password for the ICS software's default user (e.g., "guest").
- Define stringent password policies to force users to create strong passwords. Enforce mandatory password updates on a regular basis.

## Manage users and groups through ICS software

Your ICS software should have a built-in security system that controls who may use the software and what privileges they have.

Users should be assigned permissions that determine what each user is authorized to do within the ICS system. Permissions can be managed either on a per-account basis or on a group basis by making use of roles. Group or role-based access control is preferred as it greatly simplifies management. Users can be moved from one role to another as the organization's needs change, and can also be members of multiple roles if required.

Each user should have their own user account with a unique user name and a strong password. The user account can then be assigned to one or more groups.

Accounts should always be assigned the least privileges necessary to perform their functions. Accounts with Windows Administrator permissions should be reduced to the minimum, and typically only used to install and configure the software. Likewise, accounts with SQL Server SysAdmin privileges should be reduced to the minimum, and typically only used to install and configure the software.

In most cases, the ICS software will allow associating Windows Groups with roles within the product. While defining and assigning roles, consider the following:

- Roles should be defined to have the least privileges necessary for their functionality.
- Roles should be limited to a single purpose in order to simplify the permissions assigned to them.
- Users can be members of multiple roles if necessary.

## Plan contingency

Incidents are inevitable. It is, therefore, important to develop a strategy to detect an incident quickly and respond to it in a timely manner in order to minimize loss and protect your system. An organization must consider contingencies arising from incidents such as fire, flood and so on, and those arising from failure of hardware or software components. Cyber attacks such as ransomware are becoming more common and must also be considered.

An organization should have contingency plans in place to cover the entire range of failures and eventualities. Employees should be trained and be familiar with the contents of the contingency plans.

As part of planning for contingencies, it is important to establish a site, physically separated from the central one, that has replication capability. Doing so will ensure the integrity of an operational system where the central site is at risk from fire, floods or other disasters. The replication capability includes having duplicated hardware, and requires software configuration and key state information to be periodically propagated from the central site to the recovery site. Each recovery scenario is unique, so it is important to consult with system integration experts regarding the design of communications equipment, hardware and the configuration of the software.

Protecting the data stored in your system is also of paramount importance. Full and incremental backups must be scheduled on a regular basis. Backups should be verified by running tests to restore from backed up data. Backups should be stored offline so that they are safe from cyber attacks such as ransomware.

Organizations should also have business continuity and disaster recovery plans that are similar to contingency plans. These plans are covered briefly in the sections to follow.

## Audit and log

As part of implementing security for ICS software, it is important to incorporate auditing and logging activities on various systems and networks.

Auditing and logging provide information on the current state of your ICS, and help to ensure that the system is functioning as expected. If an incident occurs, you can use the activity logs to trace the origin of the incident to a computer, user or network. Auditing and logging can also help with troubleshooting issues.

If you are connecting to cloud-based solutions, audit all virtual machines (VMs) to ensure data integrity.

## Plan business continuity

Business continuity planning addresses strategies to maintain or re-establish production in the event of any disruption. These disruptions may be caused by a natural disaster (flood, earthquake, etc), by an intentional or unintentional human-made event (arson, operator error, power outage, etc), or by system failure.

Depending upon the duration of the potential ICS application outage caused by a disruption, operational recovery plans for short-term outages and disaster recovery plans for long-term outages must be formulated. It is also important to employ physical security for areas of a production site that house data acquisition and control systems that might have higher-level risks. Your business continuity plan should specify system and data recovery procedures for your systems. Once the recovery procedures are documented, a schedule should be developed to test the recovery procedures. Particular attention must be paid to the verification of backups of system configuration data and product or production data. The procedures should be reviewed periodically.

If you are accessing cloud-based solutions, ensure that systems are available at all times. In case of a disaster, services should switch to a new physical location to provide continued service.

## Plan disaster recovery

A disaster recovery plan (DRP) is a set of procedures to protect and recover an IT infrastructure in case of a disaster. It contains the procedures to follow before, during and after a disaster. Disasters can be natural, environmental or human-made (intentional or unintentional).

A DRP is essential for continued availability of the ICS, and should cover the following:

- When the DRP should be activated depending upon an event, its duration and its severity.
- Detailed course of action for operating the ICS manually until external connections are secured.
- Personnel responsible for each procedure.
- Processes for securely backing up data and restoring it. This should cover:
  - Requirements for building redundancy.
  - File backup procedures.
  - Frequency of backups.
  - Storage mechanism for full and incremental backups.
  - Safe storage of installation media, license keys and configuration information.

- List of individuals responsible for performing, testing, maintaining and restoring backups.
- List of personnel with physical and virtual access to the ICS.
- Detailed configuration information about the components of the ICS.
- Schedule for testing the DRP.

## Conclusion

Security lapses present a serious threat to ICS software and infrastructure. Therefore, it is important for every organization to:

- Be proactive about preventing security lapses
- Identify potential lapses
- Detect them in a timely manner when they occur
- Address lapses to ensure minimum disruption and maximum availability

To this end:

- Computers and networks must be secured
- Users and groups must be authenticated and authorized
- Contingency plans must be in place to recover from untoward or intentional events

Refer to the document "Guide to Industrial Control Systems (ICS) Security" [NIST Special Publication 800-82 Revision 2](<https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r2.pdf>) for additional details and recommendations.

## Security Configuration for Plant SCADA

The table below lists the security areas that you need to configure for Plant SCADA and the details of the section(s) in this guide that provide the corresponding instructions.

Security Area(s)	Topic(s) in this guide	Summary
Manage access through users and groups	<a href="#">Runtime System Security</a>	You can restrict which functions a user is allowed to perform by linking those functions to a defined set of areas and privileges.
Secure a runtime system through authentication and authorization	<a href="#">Security Roles</a>	Plant SCADA uses a set of security roles to manage access to sensitive components within the runtime system. Each role provides a different level of access to features, applications and project resources.

Security Area(s)	Topic(s) in this guide	Summary
Use Windows users accounts	<a href="#">Integrate Windows™ User Groups</a>	Plant SCADA provides the ability to integrate Windows user accounts into the security model for a runtime system. You can achieve this by linking domain Windows user accounts to a role defined within Plant SCADA.
Protect the application on the host runtime computer	<a href="#">Securing Runtime Computers</a>	There are several ways you can limit runtime access to the host operating system or software other than Plant SCADA.
Enable encrypted communications	<a href="#">Encrypted Communications</a>	Communications between Plant SCADA processes, computers and CtAPI can be encrypted. A System Management Server is required to manage the certificates that enable trusted communications.

## See Also

[Security](#)

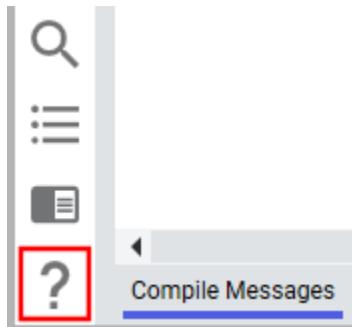
## Information and Support

There are several options available to you when seeking information and assistance with Plant SCADA.

- **Product documentation**

If you have questions about using Plant SCADA, you should initially consult the product documentation.

You can open the documentation in a browser via the **Help** button at the bottom of Plant SCADA Studio's Activity Bar.



You can browse the **Contents** to find the topics you are interested in, or you can enter a term in the **Search**

field.

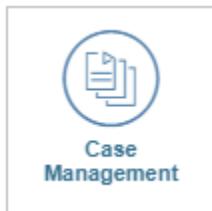
You can also access context-sensitive documentation via the **F1** key. This will open a topic in the documentation that relates to the task you are currently performing in Plant SCADA Studio.

See [AVEVA™ Help](#).

- **AVEVA Knowledge and Support Center**

If you seek more technical information than is provided in the documentation, check the [AVEVA™ Knowledge & Support Center](#).

If you cannot find the information you need, you can raise a case with Technical Support via the **Case Management** page.



- **Consulting Services**

Consulting services are also available upon request. Contact your local authorized AVEVA distributor.

## AVEVA™ Help

The documentation for Plant SCADA is presented using AVEVA Help, a browser-based tool that allows any installed AVEVA products to share a common platform for the delivery of product information.

AVEVA Help is accessible from the **AVEVA Documentation** branch of the Windows **Start** menu. When launched, it will display in your default browser.

The panel to the left includes a list of any locally installed AVEVA products that utilize the AVEVA Help platform. If installed, this will include:

- AVEVA Enterprise Licensing
- Operations Control Logger.

You can open the documentation for a product on its own browser tab by selecting the following icon:



Plant SCADA's context-sensitive help will also open content directly in AVEVA Help.

The **Search** functionality uses straightforward text-based matching, and you can save a search for repeated use.

The Plant SCADA help is also now available on AVEVA's Documentation cloud portal ([docs.aveva.com](https://docs.aveva.com)).

## See Also

[AVEVA™ Knowledge & Support Center](#)

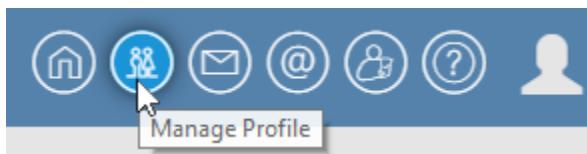
## AVEVA™ Knowledge & Support Center

The AVEVA Knowledge & Support Center is an online resource you can use to access technical information, updates and tools that support the company's software products. It is located at <https://softwaresupport.aveva.com/>.

**Note:** The Citect Global Support website is no longer available. If you try to visit this site, you will automatically be redirected to the AVEVA Knowledge & Support Center.

If you are new to the Knowledge & Support Center website, you will need to register for access. Use an organizational email address as generic email addresses are not supported.

If you have an existing CustomerFIRST account, it will provide access to restricted content. Be aware that it can take up to 24 hours to link your website registration to a CustomerFIRST account as it is a manual process. You can check if your registration is complete by viewing the **Manage Profile** page.



If a number appears in the **Tech Support ID** field, it indicates that your registration has been processed. The **Account Information** appears at the bottom of the page.

The home page provide access to the pages described in the table below. If you are familiar with the old Global Support Page, you will now find the existing content on the following pages.

- **Product Downloads** — Product Hub
- **Updates & Patches** — Product Hub
- **Web Help** — Product Hub
- **Forums** — Community
- **Compatibility Matrix** — Technology Matrix
- **DriverWeb** — Communication Drivers
- **License Generator** — License Activation.

Page	Description
	<b>Case Management</b> Allows you view and manage your <b>Cases</b> and <b>Service Requests</b> lodged with the support team.
	<b>Product Hub</b> Lists all products and provides the following: <ul style="list-style-type: none"><li>• Product information</li><li>• Product downloads</li><li>• Updates and patches</li><li>• Online documentation</li></ul>

Page	Description
	<ul style="list-style-type: none"><li>• Instructional videos</li><li>• Product support lifecycle.</li></ul>
	<b>Solutions Hub</b> Lists all product bundles and industry applications. It includes product documentation and videos.
	<b>Knowledge Base</b> Includes a list of Knowledge Base articles, Tech Notes, Tech Alerts and FAQs for all software products.
	<b>Videos</b> Includes instructional videos for all software products.
	<b>Reports</b> Includes external and internal reports. Report access is determined by your site access permissions.
	<b>Community</b> Links to the AVEVA Software Community ( <a href="https://softwareforums.aveva.com/">https://softwareforums.aveva.com/</a> ). Select <b>Citect Community</b> to see the forums related to Plant SCADA.
	<b>Training Center</b> Links to the <b>AVEVA Training Services</b> website, which provides details about the training courses that are available for Plant SCADA and other AVEVA products.
	<b>Subscriptions</b> Use this page to set, edit or delete your subscriptions. When content for your subscription changes, you will receive an email notification. To subscribe to Tech Notes, FAQs, Tech Alerts and Notifications, navigate to the <b>Knowledge Base</b> page, open an article, and subscribe to the specific article, the document type, or the product covered by the article.
	<b>Technology Matrix</b> Links to the <b>AVEVA Technology Matrix</b> website, which provides a form that allows you to confirm if AVEVA software products align with your compatibility

Page	Description
	requirements.
	<b>Communication Drivers</b> This page lists all OI, DA and other connectivity applications, including product documentation and videos. In the case of Plant SCADA, this is where you find all information related to drivers and the old Driver Web. Click the <b>Related Tech Notes, FAQs</b> button to view the Tech Notes and FAQs related to the selected products.
	<b>License Activation</b> Select <b>Citect</b> to activate licenses for Plant SCADA. See <a href="#">Licensing</a> for further instructions.
	<b>Security Central</b> Provides Microsoft™ Security Update Reports, Product Cyber Security Updates, and Policies & Guidelines. Use the <b>Select Product Line</b> menu to display information related to Plant SCADA.
	<b>Digital Exchange</b> Links to the AVEVA Digital Exchange, which allows you to browse available solutions for specific industries.
	<b>SAM Software</b> Links to the Software Asset Manager (SAM) download site, which provides access to the files and instructions necessary to run Software Asset Manager (SAM) at your plant. SAM is a software application from AVEVA that helps you manage Plant SCADA software installations and licenses.

Most pages will allow you to filter a list according to **Product** and/or content type. For example, to see the available **Tech Notes** for **Plant SCADA**, you would use the following filter.

**Filters**

Document # or Legacy #

Title...

**Document Type**

Select All  
 Tech Notes  
 Tech Alerts  
 FAQS  
 Notifications

**Products**

Select All  
 Ampla  
 Avantis  
 Citect  
     Citect Historian  
     Citect SCADA  
 OASyS

You can also use the **Manage Profile** page to set **Preferences** based on the products you use. Every time you view a page with a filter, your preferences will be preselected.

# What's New in Plant SCADA

Plant SCADA 2023 R2 incorporates the following new or modified features:

- Support for Plant SCADA on AVEVA Integration Studio
- Double Point Status Alarms
- Kernel Access Improvements
- New Installation Framework
- AVEVA™ Help
- Setup Wizard supports HMI Computer Role
- Unique IDs for Equipment
- Equipment Update Improvements
- Project DBF AddIn supports 64-bit Excel.

---

**Note:**

- With the release of Plant SCADA 2023 R2, the **CSV\_Include** project is no longer supported as a system project. If you have any projects based on the CSV\_Include templates, you will need to create a separate backup of the CSV\_Include project and restore manually it in version 2023 R2 (or later). You can then include it as a user project.
  - Process Analyst no longer supports CitectHistorian integration.
- 

For updates to Citect.ini parameters, see [Citect.ini Parameters in Plant SCADA 2023 R2](#).

For updated to Cicode functions, see [Cicode Functions in Plant SCADA 2023 R2](#).

For details on how to upgrade an existing project to 2023 R2, refer to [Upgrade Plant SCADA](#).

## Support for Plant SCADA on AVEVA Integration Studio

AVEVA Integration Studio is an infrastructure-as-a-service virtual development environment that is delivered on AVEVA's cloud services platform CONNECT.

Integration Studio lets you run and manage software applications without the complexity of building and maintaining the physical infrastructure typically required when developing and launching software. You can create templates for virtual machines and deploy multiple instances on demand.

AVEVA Integration Studio now allows you to include Plant SCADA nodes in a project template. These nodes can be used to create multiple instances of virtual machines with a fully licensed version of Plant SCADA installed.

For more information, see [Plant SCADA and AVEVA Integration Studio](#).

## Double Point Status Alarms

Plant SCADA 2023 R2 now supports double point status alarms.

A double point status alarm responds to eight states in a field device that are represented in Plant SCADA as a single integer tag.

For each of the eight states you can specify the following:

- **State Name <n>** — a name that identifies the condition that the state represents.
- **State Type <n>** — the action that occurs when the device transitions to a defined state. The options are:
  - "Alarm" - an alarm is added to the Active Alarms page and an event is logged to the SOE page. The State Name will appear in the State field.
  - "Event" - indicates an OFF state has occurred. An event is logged on the SOE page, even if an alarm state transition did not occur. The State Name will appear in the State field.
  - "None" - indicates an OFF state has occurred. The State Name will appear in the State field on both the SOE and Active Alarm pages when an alarm transition occurs.

---

**Note:** If an alarm is currently in an "Event" or "None" state, a further transition to a "None" state will not be reported.

---

This means when the value of the integer tag is 0, the alarm will go into "State Name 0". When the value of the tag is 1, it will go into "State Name 1", and so on. When this occurs, the associated State Type action will be triggered.

Just like time stamped analog and time stamped digital alarms, double point status alarms are timestamp based and designed to work with the Driver Runtime Interface (DRI). They need to use tags where values are pushed to the alarm server by a driver.

See [Add a Double Point Status Alarm](#).

## Kernel Access Improvements

Access to the Kernel has been changed to provide improved security while monitoring a runtime system. These changes also allow you to display the Kernel from Runtime Manager while running Plant SCADA as a service.

You can still launch the Kernel on a display client in the following ways:

- From the **Control** menu (see [Display the Kernel from the Control Menu](#))
- Via the Cicode function DspKernel (see [Define a Runtime Command](#))
- From the Runtime Manager (see [Use the Kernel with a Selected Process](#)).

To display the Kernel window for a display client, the user that is currently logged in now needs to be assigned to a **Roles** that has the **Kernel Access** property set to "Full Access" or "Read Only". A Read Only user can access the Kernel, but cannot perform some privileged commands like running Cicode.

---

**Note:** The special Kernel user that enabled Kernel access in previous versions no longer has default access to the Kernel window.

---

To display the Kernel window for a server process (including when running Plant SCADA as a service), you will need to log in using the dialog that is displayed when the Kernel window is launched.

You will need to enter a user that is assigned to a Role that has the **Kernel Access** property set to "Full Access" or "Read Only".

---

**Note:** Before commissioning a system, it is recommended that you review the **Kernel Access** setting for each Role. This will avoid accidental or unauthorized use of the Kernel.

---

## See Also

[Display the Kernel Window](#)

## New Installation Framework

Plant SCADA 2023 R2 includes a new installation framework that simplifies the selection and installation of required components.

Plant SCADA's core components appear as a single entry within the **Programs and Features** list in Windows Control Panel, making it easier to uninstall. The new framework also supports **Modify** functionality, which means you can easily add and remove components (see [Uninstall or Change Components](#)).

The installation of Plant SCADA can also be executed via a command line, enabling support for an unattended silent installation. The required components are specified via a configurable response file that represents the forms normally seen when interacting with the user interface of the Plant SCADA installer.

Four preconfigured response files are provided. They install a typical set of required components, or they can be used as reference for your own customized response file.

- **All.txt** – installs all components
- **DevelopmentWorkstation.txt** – installs components for a development workstation
- **RuntimeClient.txt** – installs components for a runtime client
- **Server.txt** – installs components for a runtime server.

## See Also

[Unattended Silent Installation](#)

## AVEVA™ Help

The documentation for Plant SCADA is now presented using AVEVA Help, a browser-based tool that allows any installed AVEVA products to share a common platform for the delivery of product information.

AVEVA Help is accessible from the **AVEVA Documentation** branch of the Windows **Start** menu. When launched, it will display in your default browser.

The panel to the left includes a list of any locally installed AVEVA products that utilize the AVEVA Help platform. You can open the documentation for a product on its own browser tab by selecting the following icon:



Plant SCADA's context-sensitive help will also open content directly in AVEVA Help.

The **Search** functionality uses straightforward text-based matching, and you can save a search for repeated use.

The Plant SCADA help is also now available on AVEVA's Documentation cloud portal at <https://docs.aveva.com/bundle/plant-scada/>.

## Setup Wizard supports HMI Computer Role

The Setup Wizard and Profile Wizard now allow you to assign the role **HMI (Standalone, no networking)** to a computer within a Plant SCADA system. This change aligns the available computer roles in a Plant SCADA system with the license types supported by AVEVA Enterprise Licensing.

If your Enterprise License Server hosts a combination of client, server and HMI licenses, you can now specify the HMI role for any computers that require a dedicated HMI license. This means the computer will not pick up a server license that was not intended for it.

It also means you can host both server and HMI licenses on the same Enterprise License Server. Previously, you had to setup two different systems to host these licenses

The **HMI (Standalone, no networking)** role can be configured on the [Computer Role Setup](#) page of the Setup Wizard and Profile Wizard.

### See Also

[Allocating AVEVA™ Enterprise Licenses](#)

## Unique IDs for Equipment

A unique ID is now applied to any equipment and equipment types in a Plant SCADA project. This is done to provide consistent identification for integration with future software releases.

The unique IDs are Globally Unique Identifiers that are generated by Plant SCADA whenever equipment is added to a project. The IDs are not available to access at runtime.

The Migration Tool also includes an option to **Create unique IDs for Variables and Equipment**.

For more information, see [Unique IDs for Variable Tags and Equipment](#).

## Equipment Update Improvements

The performance of an Equipment Update has been improved. In most cases, the time taken to generate tags and equipment will be significantly reduced compared to earlier versions.

## Project DBF AddIn supports 64-bit Excel

The Plant SCADA Project DBF AddIn now supports 64-bit versions of Microsoft Excel.

### See Also

[Using Microsoft Excel to Edit DBF Tables](#)

# Citect.ini Parameters in Plant SCADA 2023 R2

This topic lists the parameters that have changed in Plant SCADA 2023 R2.

## Removed Parameters

The following parameters have been removed from this release.

### Debug Parameters

[Debug]Kernel	You can no longer launch the Kernel at startup.
---------------	---

### CSV\_Include Parameters

[Alarm]ActiveHeading	The CSV_Include project is no longer supported by Plant SCADA as a system project.
[Alarm]DisabledHeading	
[Alarm]SummaryHeading	
[Alarm]HardwareHeading	
[MultiMonitors]LastPageStackSize	
[MultiMonitors]ScreenWidth	
[Page]NewPageInBase	
[Page]DelayRepaint	
[Navigation]AlarmPage	
[Navigation]DisabledPage	
[Navigation]FilePage	
[Navigation]HardwarePage	
[Navigation]HelpPage	
[Navigation]HomePage	
[Navigation]MenuBackColour	
[Navigation]MenuForeColour	
[Navigation]MenuXPos	
[Navigation]NetworkPage	
[Navigation]MenuYPos	

[Navigation]ProcessAnalystPage
[Navigation]SummaryPage
[Navigation]TrendPage
[Navigation]ToolsPage
[Printer]Port
[Privilege]EngTools
[TrendX]Duration
[TrendX]KeySeq
[TrendX]TagListEnable

## See Also

[What's New in Plant SCADA](#)

# Cicode Functions in Plant SCADA 2023 R2

Some Cicode functions may have been introduced or modified. The following sections detail these changes:

## Removed Functions

### Error Functions

ErrHelp	This function is now obsolete.
---------	--------------------------------

### Windows Functions

WndHelp	This function is now obsolete.
WndViewer	This function is now obsolete.

## Reinstated Functions

### Error Functions

HtmlHelp	Invokes the Microsoft HTML Help application (hh.exe)
----------	--

[ ] to display an HTML help file (.chm).

## See Also

[Citect.ini Parameters in Plant SCADA 2023 R2](#)

## What's New - Previous Releases

This section lists the features that were added in previous releases of Plant SCADA, and provides links to the relevant sections in the online help.

- [Introduced in Plant SCADA 2023](#)
- [Introduced in Plant SCADA 2020 R2](#)
- [Introduced in Citect SCADA 2018 R2](#)
- [Introduced in Citect SCADA 2018](#)
- [Introduced in Citect SCADA 2016](#)
- [Introduced in Citect SCADA 2015 SP1](#)
- [Introduced in Citect SCADA 2015](#)
- [Introduced in Citect SCADA 7.40 SP1](#)
- [Introduced in Citect SCADA 7.40](#)
- [Introduced in Citect SCADA 7.30](#)
- [Introduced in Citect SCADA 7.20](#)
- [Introduced in Citect SCADA 7.10](#)
- [Introduced in Citect SCADA 7.0](#)

## See Also

[What's New in Plant SCADA](#)

## What's New in Plant SCADA 2023

Plant SCADA 2023 incorporates the following new or modified features:

- [AVEVA™ Enterprise Licensing](#)
- [Configuration Parameters Database for Equipment Types](#)
- [Add Web Content to a Graphics Page](#)
- [Tag Writes Enabled for Industrial Graphics Applications](#)
- [Tag Writes Enabled for OPC UA Client Applications](#)
- [Alarm Filtering Performance Improvements](#)
- [Unique IDs for Variables](#)

- Unique IDs for Projects
- Debug Cicode without Compiling.

The colors, buttons and other elements used for the Plant SCADA Studio interface have been changed to provide more efficient workflows and a common look-and-feel across AVEVA's products.

**Note:**

The following features are no longer supported with this release of Plant SCADA.

- Schneider Electric's **Floating License Manager** is no longer supported by Plant SCADA. Existing users of the Floating License Manager will need to reconfigure their system to use AVEVA Enterprise Licensing.
- Plant SCADA's legacy Active X Web Client is no longer supported.

For updates to Citect.ini parameters, see [Citect.ini Parameters in Plant SCADA 2023](#).

For updates to Cicode functions, see [Cicode Functions in Plant SCADA 2023](#).

## AVEVA™ Enterprise Licensing

In Plant SCADA 2023, AVEVA Enterprise Licensing is now fully integrated for use with Plant SCADA.

AVEVA Enterprise Licensing is a common platform solution that allows you to manage AVEVA Enterprise Software product licenses.

The system is comprised of a browser-based License Manager and a License Server that allow you to share and deliver licenses to installed AVEVA applications. This distributed architecture enables centralized management of activated licenses and flexible topologies that support systems of any size.

AVEVA Enterprise Licensing is required if you intend to use AVEVA Industrial Graphics.

**Note:** With this release, Schneider Electric's Floating License Manager is no longer supported by Plant SCADA. Existing users of the Floating License Manager will need to reconfigure their system to use AVEVA Enterprise Licensing.

## See Also

[AVEVA™ Enterprise Licensing](#)

## Configuration Parameters Database for Equipment Types

In Plant SCADA 2023, "configuration parameters" replace "custom parameters" for Equipment Types.

[Configuration Parameters](#) provide the same functionality as custom parameters. They can be used to represent specific information about a physical piece of equipment, such as a set point or delay setting. However, they are stored in a dedicated configuration parameters database. This means you will no longer be restricted by a 256 character limitation, providing greater flexibility if you need to add multiple parameters.

Only Equipment Types created in Plant SCADA 2023 (or later) will make use of the new database. You can identify one of these new Equipment Types via the following version "2.0" label in the source XML:

```
<template version="2.0" desc="Equipment template">
```

**Note:** This change requires any existing equipment types to be migrated to a new format. If Plant SCADA detects any version 1.0 equipment types, a dialog will appear asking if you want to migrate them. If this occurs, see [Migrate Custom Parameters to the Configuration Parameters Database](#).

## See Also

[Add Configuration Parameters to an Equipment Type](#)

## Add Web Content to a Graphics Page

Plant SCADA now allows you to natively incorporate modern web-based content in your runtime system. This is achieved by adding a **Web Content** object to a page in Graphics Builder (see [Add a Web Content Object](#)).

Each instance of a Web Content object has a **URL** property that allows you to specify the content that is displayed when the host graphics page is launched.

---

**Note:** If one of your projects uses the Microsoft Web Browser ActiveX object on a graphics page, it is recommended that you replace it with the new Web Content object.

---

## See Also

[Web Content](#)

## Tag Writes Enabled for Industrial Graphics Applications

You can now write to Plant SCADA tags and equipment.items directly from Industrial Graphics applications. This involves the following changes to Plant SCADA:

- The "Industrial Graphics R/W Users" security role is now enabled on any computer hosting an Industrial Graphics Server. It is used to manage access for client application users that require read/write access to a Plant SCADA runtime system.
- Variable tags now include a **Write Roles** property that is used to enable writes for users from specific Plant SCADA roles. The current user of an Industrial Graphics application needs to be associated with the role specified in this property to write to a variable tag.

For more information, see [Enable Tag Writes for Industrial Graphics Applications](#).

## Tag Writes Enabled for OPC UA Client Applications

You can now write to Plant SCADA tags directly from an OPC UA client application connected to a Plant SCADA OPC UA Server.

To do this, you need to set the **Write Roles** property for each variable tag that will have writes enabled. The Property Grid allows you to select one of the roles configured in your Plant SCADA project, or you can manually enter a comma-separated list to include multiple roles. See [Add a Variable Tag](#).

To write to a variable tag, the user that is currently logged in to the OPC UA client application needs to be part of a Windows™ domain group that is associated with the role specified in the **Write Roles** property.

---

**Note:** Tag writes will only work if encryption is enabled for the OPC UA Server.

---

For more information, see [Confirm the Settings for an OPC UA Server](#).

## Alarm Filtering Performance Improvements

The performance of filtering operations on the **Sequence of Events** and **Alarm Summary** pages have been improved.

Queries that return a small number of alarm records from a large alarm database will see the greatest performance improvement. For example, when you filter alarms that have occurred on a single piece of equipment.

In addition, a new parameter has been added that allows you to switch off alarm filter optimization. For a description of the circumstances where this may be beneficial, see [\[Alarm\]DisableFilterOptimization](#).

## Unique IDs for Variables

A unique ID is applied to each of the variable tags in a Plant SCADA project to provide consistent identification for integration with future software releases. The unique IDs are Globally Unique Identifiers that are generated by Plant SCADA whenever tags are added to a project.

The Migration Tool has a new option to automatically add IDs to any existing project.

For more information, see [Unique IDs for Variable Tags and Equipment](#).

## Unique IDs for Projects

With the release of Plant SCADA 2023, unique IDs are now applied to each Plant SCADA project. This provides consistent identification across the lifetime of a project.

Each unique ID is a Globally Unique Identifier (GUID) generated by Plant SCADA using the following format:

XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

If you [Copy a Project](#) or [Restore a Project](#) a project, you will need to consider how its Unique ID will be impacted.

## See Also

[Plant SCADA Projects](#)

[Edit a Project's Properties](#)

## Debug Cicode without Compiling

To start a debugging session without the project being recompiled, you can use **Start Debugging Without Compile** to quickly start debugging your Cicode.

The **Start Debugging Without Compile** option can be selected from Cicode Editor's **Debug** menu.

---

**Note:** You can only use this option if you have already compiled a project at least once.

---



### UNINTENDED EQUIPMENT OPERATION

Do not use the following command to start Plant SCADA if you have uncompiled project changes that are

necessary for the safe operation of your plant and process.

- Start Debugging Without Compile

This option is only intended to assist debugging an already compiled project.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## See Also

[Debugging Cicode](#)

## Citect.ini Parameters in Plant SCADA 2023

This topic lists the parameters that have changed in Plant SCADA 2023.

### New Parameters

The following parameters are new in this release. For an entire list of the system parameters, refer to the Parameters documentation.

#### Alarm Parameters

[Alarm]DisableFilterOptimization	Allows you to turn off filter optimization for alarm queries.
[Alarm]StandbyCommandDelay	Sets a delay (in milliseconds) that a standby alarm server will apply when executing alarm commands received from the main alarm server.

#### AnmCursor Parameters

[AnmCursor]RubberBandColor	Sets the color of the trend selection rubber band.
----------------------------	--

#### CTAPI Parameters

[CtAPI]MaxConnections	Specifies the maximum number of connections supported by a CTAPI client.
-----------------------	--

#### Debug Parameters

[Debug]ChromiumDevToolsPort	Enables debugging for a Web Content object located on a Plant SCADA graphics page.
-----------------------------	--

#### General Parameters

[General]AELManagerUrl	This parameter stores the current URL for the AVEVA™ Enterprise License Manager.
[General]EnforceTagGlobalUniqueness	Instructs the compiler to generate a warning message for each tag name that is not globally unique.

**OpcDaServer Parameters**

[OpcDaServer]RoundToFormat	Specifies if tag values published to an OPC DA Server are rounded.
----------------------------	--

**Security Parameters**

[Security]DeleteWebContentProfile	Use this parameter to automatically delete the application data associated with users of a Web Content control.
-----------------------------------	---

**Removed Parameters**

The following parameters have been removed from this release.

**Address Forwarding Parameters**

[AddressForwarding]LicenseStore	Floating License Manager is no longer supported by Plant SCADA.
---------------------------------	---

**CtEdit Parameters**

[CtEdit]WebDeploy	The Plant SCADA IIS Web Client is no longer supported.
-------------------	--

**DNS Parameters**

[DNS]Primary	The Plant SCADA Internet Display Client is no longer supported.
[DNS]Standby	The Plant SCADA Internet Display Client is no longer supported.

**Internet Parameters**

[Internet]Client	The Plant SCADA Internet Display Client is no longer supported.
[Internet]Display	The Plant SCADA Internet Display Client is no longer supported.
[Internet]FTPTimeout	The Plant SCADA Internet Display Client is no longer supported.
[Internet]FTPWatchTime	The Plant SCADA Internet Display Client is no longer supported.
[Internet]IPAddress	The Plant SCADA Internet Display Client is no longer supported.
[Internet]LogFile	The Plant SCADA Internet Display Client is no longer supported.

[Internet]Manager	The Plant SCADA Internet Display Client is no longer supported.
[Internet]Password	The Plant SCADA Internet Display Client is no longer supported.
[Internet]Port	The Plant SCADA Internet Display Client is no longer supported.
[Internet]Redundancy	The Plant SCADA Internet Display Client is no longer supported.
[Internet]RunFTP	The Plant SCADA Internet Display Client is no longer supported.
[Internet]SearchCurrentPath	The Plant SCADA Internet Display Client is no longer supported.
[Internet]Server	The Plant SCADA Internet Display Client is no longer supported.
[Internet]ShowFileFTPDlg	The Plant SCADA Internet Display Client is no longer supported.
[Internet]ShowSetupDlg	The Plant SCADA Internet Display Client is no longer supported.
[Internet]UpdateTime	The Plant SCADA Internet Display Client is no longer supported.
[Internet]ZipFiles	The Plant SCADA Internet Display Client is no longer supported.

**Web Server Parameters**

[WebServer]DeployRoot	The Plant SCADA IIS Web Client is no longer supported.
[WebServer]WebClientCab	The Plant SCADA IIS Web Client is no longer supported.

**See Also**

[What's New in Plant SCADA 2023](#)

**Cicode Functions in Plant SCADA 2023**

Some Cicode functions have been introduced or modified. The following sections detail these changes:

## New Functions

### Color Functions

MakeColour	Creates a color from red, green and blue component parts. For backwards compatibility, you can refer to this function using the alias "MakeCitectColour".
------------	---

### Display Functions

DspRubSetColor	Sets the color of the rubber band tool that is used to select a section within a trend object.
DspWebContentGetURL	Gets the current URL for a Web Content control on a graphics page.
DspWebContentSetURL	Sets the URL for a Web Content control on a graphics page.

### Security Functions

UserCreateByRole	Creates a record for a new Plant SCADA user. The user can be assigned to a specified role. Not available when logged in as Windows user.
------------------	--

## Deprecated Functions

### Security Functions

UserCreate	Creates a record for a new Plant SCADA user. This function is now deprecated and will be removed in the next release. Use the function instead.
------------	---

## Removed Functions

### FTP Functions

FTPClose	The Plant SCADA Internet Display Client is no longer supported.
FTPFileCopy	The Plant SCADA Internet Display Client is no longer supported.
FTPFileFind	The Plant SCADA Internet Display Client is no longer supported.
FTPFileFindClose	The Plant SCADA Internet Display Client is no longer supported.
FTPOpen	The Plant SCADA Internet Display Client is no longer

	supported.
--	------------

## See Also

[Citect.ini Parameters in Plant SCADA 2023](#)

## What's New in Plant SCADA 2020 R2

Plant SCADA 2020 R2 incorporates the following new or modified features:

- [Rebranding to Plant SCADA](#)
- [Integration of AVEVA™ Industrial Graphics](#)
- [Integrated OPC UA Server Functionality](#)
- [IPv6 Support](#)
- [Graphics Usage](#)
- [Graphics Editing Enhancements](#)
- [Driver Classifications](#).

For new Citect.ini Parameters, see [Citect.ini Parameters in Plant SCADA 2020 R2](#).

---

**Note:**

The following features are no longer supported with this release of Plant SCADA.

- **EcoStruxure Web Services (EWS)** are no longer supported. You will not be able to create or use an EWS server or client.
  - The **Vijeo Web Gate Control** has been deprecated and is no longer part of the Plant SCADA installation. The control will still work with Plant SCADA. However, to continue to use it you will need to manually reinstall it from another source as it will have been uninstalled when the previous version of Citect SCADA was uninstalled.
  - The **Pelco ActiveX Control** has been deprecated and is no longer part of the Plant SCADA installation. If you have upgraded your computer from a previous version with this control installed, it will continue to function. However, it is recommended that you communicate with Pelco cameras using a different method.
- 

## Rebranding to Plant SCADA

The 2020 R2 release signifies the transition away from the Citect SCADA brand to AVEVA™ Plant SCADA. The new name reflects the positioning of Plant SCADA within the broader family of AVEVA products.

The rebranding incorporates several changes, including updates to splash screens and graphics, icons, installer and installation directories, start menu paths, and more.

To determine the new directory locations, see [View a Project's Folders](#).

## Integration of AVEVA™ Industrial Graphics

AVEVA Industrial Graphics is the common graphics technology used across AVEVA's SCADA, HMI and cloud solutions. You can create HTML5 screens and dashboards in Plant SCADA and access them in desktop browsers and mobile devices.

Industrial Graphics are accessible in Plant SCADA Studio for integrated browsing and filtering. From here, they can be launched within the new Industrial Graphics Editor for all editing activities. AVEVA Industrial Graphics are accessed by clients in an HTML browser via an Industrial Graphics Server. It authenticates clients details and provides access to the graphical and functional content.

"IndustrialGraphics\_Include" is a system project designed for the Industrial Graphics platform. This project is included in the "ExampleSA" project for demonstration of Industrial Graphics. It contains a default set of Industrial Graphics which includes charting and dashboard symbols, as well as ISA symbols.

For more information, see [AVEVA Industrial Graphics](#).

## Integrated OPC UA Server Functionality

You can now make your I/O data accessible to third-party systems with the new OPC UA Server. This enables IT and OT convergence through secure connectivity.

For more information, see [Install an OPC UA Server](#).

## Graphics Usage

Supporting both Plant SCADA native graphics and AVEVA Industrial Graphics, the new Graphics Usage feature allows you to identify the graphical references that exist in your system. For example, you can confirm where a Genie or symbol is used, or if a project contains unused graphics.

To take advantage of this feature, see the new **Used By** section of the **Libraries** view in the **Visualization** activity (see [Browse Libraries in Plant SCADA Studio](#) and/or [Browse Industrial Graphics Libraries in Plant SCADA Studio](#)).

## Graphics Editing Enhancements

Graphics Builder now persists its window positions and the size of the "Go To Object" dialog.

### See also

[Object Groups](#)

## IPv6 Support

Plant SCADA now supports Internet Protocol version 6 (IPv6) and will accept 128 bit IP addresses (including any acceptable abbreviations). For example:

- 2001:0db8:0000:0000:8a2e:0123:1234
- or

- 2001:db8::8a2e:123:1234 (abbreviated).

You can now operate a Plant SCADA system on an IPv6 network. IPv6 protocol can be used for communication between servers and clients, and for any drivers that support IPv6.

IPv6 support can be enabled via the [Network Model](#) page of the Setup Wizard.

## See Also

[Add Network Addresses](#)

[Server Redirection Using Address Forwarding](#)

## Driver Classifications

As part of an ongoing effort to set clear expectations and provide better support to our customers, we have introduced a classification system for our driver portfolio to communicate the level of ongoing maintenance and version updates that customers can reasonably expect. These classifications and status of each driver can be found online at <https://gcsresource.aveva.com/PlantSCADA/Drivers/>.

In line with these classifications, we have limited the number of drivers included by default on the Plant SCADA installation media to a reduced set of 'core' drivers that have undergone recent updates, however all other drivers continue to be available for download from the **Communication Drivers** page at the [AVEVA Knowledge and Support Center](#).

It is recommended that you visit on a regular basis and subscribe to the forum so that you are alerted to any updates that occur.

For more information, see [Install Communication Drivers](#).

## Citect.ini Parameters in Plant SCADA 2020 R2

This topic lists the parameters that have changed in Plant SCADA 2020 R2.

## New Parameters

The following parameters are new in this release. For an entire list of the system parameters, refer to the [Parameters documentation](#).

### Client Parameters

[Client]TagReadRoundToFormat	Determines if the variable tag values returned by the TagRead and TagReadEx will be rounded to the format of the variable tag.
[Client]AdditionalClientsArePartOfTrustedNetwork	Tells an /X Client process to attempt to authenticate using the stored server password, which results in the client node becoming part of a "trusted network".

### LAN Parameters

[LAN]AddressScope	Specifies the address scope types for IPv6 addresses.
[LAN]AddressType	Enables support for IPv4 or IPv6 network types, or both.

### General Parameters

[General]EnforceTagGlobalUniqueness	Instructs the compiler to generate a warning message for each tag name that is not globally unique.
-------------------------------------	---

### Scheduling Parameters

[Scheduling]AlwaysExecuteEntryAction	This parameter determines how entry actions are executed by Scheduler.
--------------------------------------	--

### Removed Parameters

[Server]EWSAllowAnonymousAccess	Removed as the EcoStruxure Web Services (EWS) are no longer supported by Plant SCADA.
---------------------------------	---

### See Also

[What's New in Plant SCADA](#)

## What's New in Citect SCADA 2018 R2

Citect SCADA 2018 R2 incorporates the following new or modified features:

- [Graphics Browsing](#)
- [Size and Position Objects](#)
- [Encrypted Connections](#) between Citect SCADA server, clients and other processes
- [Publish Plant SCADA Data to the Cloud](#)
- [Situational Awareness Improvements.](#)

For changes to:

- Cicode functions, see [Cicode Functions in Citect SCADA 2018 R2](#).
- Citect.ini Parameters, see [Citect.ini Parameters in Citect SCADA 2018 R2](#).

**Note:** The Plant SCADA drivers that are installed with version 2018 R2 have been recreated to allow Aveva Group Plc branding.

## Graphics Browsing

Enhancements have been made to the **Visualization** activity that allow you to browse the pages and library items

in the active project and its included projects.

Browsing provides you with a thumbnails view or list view of pages and library items, allowing you to easily see their attributes. You can perform the following operations in these views:

- Browse, search and filter pages (see [Browse Pages in Plant SCADA Studio](#)).
- Browse, search and filter library items by project and/or library (see [Browse Libraries in Plant SCADA Studio](#)).
- Perform bulk operations directly from Plant SCADA Studio including delete, update pages and pack libraries.
- Open existing pages or library items in Graphics Builder directly from Plant SCADA Studio.
- Create new pages or library items in Graphics Builder directly from Plant SCADA Studio.

## Size and Position Objects

Editing graphics has been simplified, as you can manually enter the size and position of a selected object.

It is also possible to move or nudge an object using the keyboard.

See [Adjust Position and Size Dialog](#).

## Encrypted Connections

Citect SCADA 2018 R2 allows for [Encrypted Communications](#) between Citect SCADA servers, clients and other processes. Security certificates are centrally configured for all nodes (when on a domain), including secure deployment of configuration to remote clients.

This provides enhanced protection against external security threats by providing secure connectivity for remote CtAPI clients.

## Publish Plant SCADA Data to the Cloud

A new interface is now available to AVEVA Insight, which allows you to publish your Plant SCADA data to the cloud easily and securely. You can:

- Access your data from browsers and mobile devices.
- Create and share customized content for your process/plan.
- Automatically detect anomalous patterns in your historical data and display these via the AVEVA Insight "newsfeed".

For more information, refer to the **AVEVA Insight** documentation.

## Situational Awareness Improvements

The following enhancements have been made to the workspace for Situational Awareness projects.

1. The Navigation Zone can be configured to support icons for four alarm priorities.
2. The equipment tree view on alarm pages can also support icons for four alarm priorities.

See [Configure Display Properties for a Fourth Alarm Priority](#).

You can also add a fourth alarm priority filter button to your alarm pages (see [Add a Fourth Alarm Priority Filter to Alarm Pages](#)).

Additional topics have been added to the documentation to provide an overview of key functionality and concepts. These topics include:

- [Alarms](#)
- [Libraries](#)
- [Interlocks](#)
- [Page Levels](#).

The workspace will also support a new blue theme (see [Apply a Color Theme to a Workspace](#)).

## Citect.ini Parameters in Citect SCADA 2018 R2

This topic lists the parameters that have changed in Citect SCADA 2018 R2.

### New Parameters

The following parameters are new or have been altered in this release. For an entire list of the system parameters, refer to the Parameters documentation.

#### Accumulator Parameters

[Accumulator]Debug	Enables trace logging messages for accumulators.
--------------------	--

#### DDE Parameters

[DDE]Enable	Allows Citect SCADA Runtime to function as a DDE Server.
-------------	--

#### Page Parameters

[Page]ValueAbnormalPattern	Displays a diagonal pattern across a graphics object when the corresponding device goes offline, that is, it loses communication.
----------------------------	---

#### Scheduler Parameters

[Scheduling]BACnetSchedulePollPeriod	Sets the subscription poll period (rate) for the BACnet tags.
--------------------------------------	---

#### Report Parameters

[Report]Debug	Enables trace logging messages for the report server process.
---------------	---

#### Workspace Parameters

[Workspace]NumberOfTopPriorities	Determines how many alarm priorities are included in
----------------------------------	--

	the alarm counts presented in the Tree View on alarm and trend pages, and the tabs and buttons in the Navigation Zone.
--	--

## Modified Parameters

### Page Parameters

[Page]KeyEcho	Zero (0) has been added as an allowable value. It can be used to disable the supported functionality.
[Page]Prompt	Zero (0) has been added as an allowable value. It can be used to disable the supported functionality.

## Removed Parameters

[CtAPI]AllowLegacyConnections	Allowed connections from client applications compiled against version 7.00 (and earlier).
-------------------------------	---

## See Also

[Cicode Functions in Citect SCADA 2018 R2](#)

## Cicode Functions in Citect SCADA 2018 R2

Some Cicode functions have been introduced or modified. The following sections detail these changes:

## New Functions

### Alarm Functions

StrReplaceAlarmFilterEditHasField	Checks whether an alarm field name or any field in a set of field names is used in the filter specified in an alarm filter edit session.
-----------------------------------	--

### String Functions

StrReplace	Replaces a substring in a string with a replacement substring.
------------	--

### Scheduler Functions

SchdSpecialItemAddRange	Adds a range of special days in a specified special day group.
SchdSpecialItemDeleteRange	Deletes a range of special days in a specified special

	day group.
SchdSpecialItemModifyRange	Modifies a range of special days in a specified special day group.

## Modified Functions

### Alarm Functions

AlarmSetInfo	The function was enhanced to allow alarms to be sorted by type.
--------------	---

### Trend Functions

TrnGetTable	New display mode options have been introduced to allow users to synchronize samples to the specified time instead of the nearest display period.
-------------	--

## Deprecated Functions

No functions have been deprecated for Citect SCADA 2018 R2.

## Removed Functions

No functions have been removed for Citect SCADA 2018 R2.

## See Also

[Citect.ini Parameters in Citect SCADA 2018 R2](#)

## What's New in Citect SCADA 2018

Citect SCADA 2018 incorporates the following new or modified features:

- Situational Awareness Starter Project
- Additional Example Project
- Alarm Management
- Graphics Improvements
- Screen Profiles
- Equipment References
- Composite Genies
- Refresh Super Genie Associations
- Grid Column Resizing in Plant SCADA Studio
- Display Name for Equipment

- Menu Configuration Enhancements
- Array Functions
- Indirect Tag Value Lookups in Foreground Code.

For changes to:

- Cicode functions, see [Cicode Functions in Citect SCADA 2018](#).
- Citect.ini Parameters, see [Citect.ini Parameters in Citect SCADA 2018](#).

## Situational Awareness Starter Project

The Situational Awareness Starter Project can be used to create a Plant SCADA project that is designed to support abnormal situation management for operators. This provides accurate information to an operator in a way that can be perceived and acted upon quickly, without overwhelming their cognitive abilities. A consistent look and feel is applied to all graphical content, and standardized colors settings are used to emphasize important process data and alarms.

Situational Awareness projects differ to other pure template-based projects as they use "Workspace" technology to manage the content of a Master Page that appears on each configured client screen. Each pane in a Master Page displays a page that can be updated independently at runtime based on navigational or contextual changes. The Workspace is responsible for managing contextual updates to those panes by using the association between the hierarchical location of a detected change, and the type of content each pane is configured to display.

A set of library objects are also available via an included library project (named "SA\_Library"). The included objects support the Workspace autofill functionality at runtime and maintain a consistent look and feel across a project's display pages.

You can also use the workspace without equipment to build a project that does not utilize the new context features. This is achieved using a Pane Genie on a customized master page.

For more information, see [Situational Awareness Projects](#).

---

**Note:** This release of Plant SCADA also includes a [Additional Example Project](#) to demonstrate a Situational Awareness Project.

---

## See Also

[What's New in Citect SCADA 2018](#)

## Additional Example Project

Citect SCADA now includes an additional example project, ExampleSA.

ExampleSA is a project created using the [Situational Awareness Starter Project](#). It features the default layout used, including elements such as a header bar, content area, and a dashboard that displays contextual information and navigational tools.

It also demonstrates the autofill functionality of the workspace, which enables contextual updates managed via a project's equipment hierarchy. This is driven by an association between the hierarchical location of the current context, and the type of content each pane is configured to display.

The project pages are built using objects that are available in an included library project (named "SA\_Library").

These objects (and associated faceplates) support the autofill functionality of the workspace and maintain a consistent look and feel across a project's display pages.

For more information, see [Situational Awareness Projects](#).

## See Also

[What's New in Citect SCADA 2018](#)

## Alarm Management

Citect SCADA 2018 includes the following enhancements to the way operators can manage alarms at runtime.

### Alarm Cause and Response

Citect SCADA 2018 allows you to associate cause and response information with an alarm tag. This means you can describe the circumstances that are likely to cause the alarm, and the appropriate course of action required to address the alarm. This information can then be presented to an operator at runtime. See [Provide Cause and Response Information to Operators](#).

To specify cause and response information for an alarm tag, you need to use the new **Alarming > Cause and Response** view in the Setup activity. See [Add Cause and Response Information to Alarms](#).

### Alarm Shelving

When you shelf an alarm, it is disabled for a specified period of time, or until a specified time is reached. This allows an operator to temporarily disable alarms that are causing an unnecessary distraction.

Shelving is available via the Workspace's new Alarm tab in the Information zone.

If your project uses the SxW or Tab Style templates (accessible via a starter project), you can shelf alarms during runtime from the Active Alarms and Alarm Summary page.

You can also shelf alarms by setting the *EndTime* argument for the following Cicode functions:

- [AlarmDisable](#)
- [AlarmDisableRec](#)
- [AlarmDisableTag](#)
- [AlmBrowseDisable](#)
- [AlmSummaryDisable](#)
- [AlmTagsDisable](#).

For more information, see [Shelve Alarms](#).

### Alarm Indicators

Citect SCADA 2018 allows you to use alarm indicators. Consisting of an alarm border and flag that appear around the extent of an object group or Genie, an alarm indicator provides a prominent visual indication of alarm occurrences at runtime.



See [Use Alarm Indicators](#).

To configure an alarm indicator for an object group or Genie, there are two tasks you need to consider:

1. You need to define display properties for the alarm priorities that an indicator will represent. See [Configure Display Properties for an Alarm Priority](#).
2. For each object group or Genie, you need to define the Alarm Indicator properties in Graphics Builder. See [Alarm Indicator](#).

## Alarm Priority Display Properties

You can configure optional display properties for an alarm priority that allow you to:

- Specify a name for a priority value to provide a meaningful representation of its purpose.
- Define background and foreground colors that support a visual representation of priority on a graphics page.
- Define a Large and Small genie to visually represent the priority on the Alarm Indicator and Alarm Lists.

These additional properties are configured in the **Setup** activity (see [Configure Display Properties for an Alarm Priority](#)).

For more information, see [Prioritize Alarms](#).

## See Also

[What's New in Citect SCADA 2018](#)

## Graphics Improvements

Citect SCADA 2018 includes the following enhancements to Graphics and the Graphics builder.

## Name a Graphics Object

In Citect SCADA 2018 you can now name graphics objects ( for example, a Polyline, Rectangle, Text and Genie). Use this name to reference a graphics object via expressions and Cicode at runtime.

Configure the animation name when defining the general access properties of a graphics object.

See the following topics for more information.

- [General Access to Objects](#)
- [Animation Points](#)
- [Naming Graphic Objects](#)
- [PropertiesAttributeNameGetEx](#)
- [PropertiesAttributeNamePutEx](#)

- [PropertiesAttributeNamePutEx](#)
- [Referencing an Object Using a Name at Runtime](#)
- [DspAnSetName.](#)

## Animated Polygon Vertices

Plant SCADA now allows you to associate a tag or Cicode expression with the vertices in a polygon. This means the shape of a polygon can be manipulated at runtime in response to values generated by a production system.

This is achieved by applying offset values to each vertex that define a path along which the vertex can move. As the return value for an expression changes, a vertex will move along the path defined for it, altering the shape of the polygon.

For more information see [Animate a Polygon at Runtime](#).

## Pinned Windows

Windows can now be "pinned" at specified locations within a main window or within other pinned windows. A new Cicode function WinNewPinAt() has been added that when configured opens a new display window at a specified location, relative to the current active window, with a selected page displayed. The window can later be removed with the WinFree() function.

For more information see [WinNewPinAt](#).

## Activate/Deactivate

You can now configure events to run when a window is activated or deactivated. This is useful when using pinned windows on a page. For example, the activate event can be used to run an event if the user clicks on the pinned window.

## Dynamically Display Symbols

In Plant SCADA 2018 you can use the [DspSym](#) Cicode function with a Genie to dynamically display content that is only generated at runtime.

You can also use bitmap and non-bitmap images as part of a symbol set in Graphics Builder.

See [Dynamically Instantiate a Genie at Runtime](#) for more information.

## Runtime Cursor

The runtime input focus cursors' color can now be set to transparent. Also, a new function DspSetCurColor() has been added to allow simpler dynamic setting of the cursor color.

## See Also

[What's New in Citect SCADA 2018](#)

## Screen Profiles

A [screen profile](#) specifies the physical characteristics of one or more workstation screens and how these screens are arranged with respect to each other. Each screen icon that appears in a screen profile represents a physical screen.

You can create, manage and configure screen profiles with the Screen Profile Editor, which is displayed below the Screen Profile table in the **Setup** activity. You can create as many screen profiles as you want.

A screen profile can also be set up for your project through the [Screen Setup](#).

## See Also

[What's New in Citect SCADA 2018](#)

## Equipment References

Citect SCADA 2018 allows you to link a piece of equipment to items, or tags belonging to another piece of equipment in your Citect SCADA project. This means you can establish a relationship with equipment or equipment.items outside the equipment hierarchy. With equipment reference browse functions you can then track, group and display all information for the equipment and referenced equipment in one interface at runtime.

Add equipment references within the **System Model > Equipment References** view. Refer to [Equipment References](#) for more information.

## See also

[What's New in Citect SCADA 2018](#)

## Composite Genies

Citect SCADA provides several Composite Genies for use in graphics pages. A Composite Genie is a collection of individual Genies assembled to form a single object based on parameters/options chosen by the Engineer. The individual objects and their parameters/options are defined in an XML template file along with layouts for the collection, that is, the Composite Genie. You can insert multiple instances of the composite object on to a graphics page, and specify different parameter settings including values, alignment and display options for each instance to tailor the Composite Genie to suit your requirements.

For example, you want to build a mirrorable compressor object that is capable of optionally displaying a label, a mode indicator and a meter alongside the symbol. Prior to Citect SCADA 2018, you could either create separate genies for each of the permutations, or you could build a single genie that used visibility to show/hide elements at runtime. With Composite Genies, you can simply select the XML template for a compressor, which presents a set of presentation options for you to configure label, orientation, etc. All you need to do is set values for these options and insert the object on to a page. You can change parameters/options at any time. For more information, see [Composite Genies](#).

## See Also

[What's New in Citect SCADA 2018](#)

## Refresh Super Genie Associations

A new Cicode function called AssWinReplace allows you to update the associations for a Super Genie without the need to reload the entire page. The function removes the associations on a specified Super Genie window, and applies any pending associations.

For more information, see [AssWinReplace](#).

## See Also

[What's New in Citect SCADA 2018](#)

## Grid Column Resizing in Plant SCADA Studio

You can now re-size the columns in the Plant SCADA Studio Grid Editor according to their content.

To do this, select **Size Columns to Content** from the menu on the left side of the Grid Editor header. Alternatively, you can double-click the divider of a specific column to auto resize it.



For more information, see [Customize Grid Editor Columns](#).

## See Also

[What's New in Citect SCADA 2018](#)

## Display Name for Equipment

Equipment definitions now support a **Display Name** property. This allows you to specify a meaningful name that can identify a piece of equipment at runtime.

The Display Name can also be used to reference equipment in a Genie.

For more information, see [Define Equipment in Plant SCADA Studio](#).

## See Also

[What's New in Citect SCADA 2018](#)

## Menu Configuration Enhancements

Plant SCADA's menu system has been enhanced. You can now associate equipment with a menu item, allowing you to align the equipment model with page navigation. In the case of a Situational Awareness project, this

drives the functionality that enables a runtime system to respond to the current context of a display client (see [Prepare the Navigation Menu](#)).

Further enhancements include:

1. Support for two more menu levels
2. Support for up to eight **Custom** properties for each menu item
3. All properties can be queried at runtime.

For more information, see [Menu Configuration](#).

## See Also

[What's New in Citect SCADA 2018](#)

## Array Functions

Citect SCADA now includes a set of functions that allow you to create arrays that use up to three dimensions (x,y,z). This can offer performance benefits when handling large amounts of data, for example, when generating an alarms list.

For more information, see [Array Functions](#).

## See Also

[What's New in Citect SCADA 2018](#)

## Indirect Tag Value Lookups in Foreground Code

A new function [TagGetValue](#) has been added that allows values of tags to be automatically looked up in foreground cicode expressions. This function automatically sets up and maintains the subscriptions for the referenced tag / equipment.item.

## See also

[What's New in Citect SCADA 2018](#)

## Citect.ini Parameters in Citect SCADA 2018

This topic lists the parameters that have changed in Citect SCADA 2018.

## New Parameters

The following parameters are new or have been altered in this release. For an entire list of the system parameters, refer to the Parameters documentation.

### CtEdit Parameters

[CtEdit]IncrementalEquipmentUpdateAutoClose	Determines if the Update Equipment confirmation dialog appears when an incremental equipment update has completed.
---	--

**Debug Parameters**

[Debug]DebugLogHistoryFiles	Determines how many "debug.<process name>.log" files are kept when rolled over.
[Debug]DebugLogFileSize	Sets the maximum size of the "debug.<process name>.log" file.

**MultiMonitors Parameters**

[MultiMonitors]Context<n>	Allows you to share a single context with one or more screens.
[MultiMonitors]ScreenProfile	Sets the arrangement of monitors on a client workstation.
[MultiMonitors]StartupPage<Name>	Sets the pages that appear when using a particular screen profile.
[MultiMonitors]Workspaces	Indicates the number of workspace pages to be shown at startup.

**SA\_Library.Meter Parameters**

[SA_Library.Meter]UseDefaultPLCLimits	Allows you to use PLC alarm limits for faceplates.
[SA_Library.Meter]PLCLimitNames	Names for PLC alarm limits. This is a comma-separated list of values.
[SA_Library.Meter]PLCLimitLabels	Display labels that correspond to each PLC alarm limit defined. This is a comma-separated list of values.
[SA_Library.Meter]DefaultPLCLimits	A number that determines the order in which to use the defined PLC alarm limits. 0 corresponds to the lowest PLC alarm limit defined, while the largest number corresponds to the highest limit defined.

**Workspace Parameters**

[Workspace]AlarmSortOrder	Defines a customized sort order for the default alarms pages in a Situational Awareness project.
[Workspace] EquipmentNavigationModelFilter	Filters the equipment model loaded by the workspace.
[Workspace]InfoZoneAlarms.IncludeReferences	Determines if alarms from referenced equipment are included in the Alarms page on the Information Zone.
[Workspace]InfoZoneTrends	Determines the pens that will be displayed in the

	Information Zone of the workspace based on the equipment items configured.
[Workspace]InfoZoneTrends.Type	Determines the pens that will be displayed in the Information Zone of the workspace based on the equipment types configured in the Equipment Editor.

## Modified Parameters

[Format]FormatName	Defines a display format for tabular data and identifies it with a specified name. This allows you to name a customized display format that you can then apply to an alarms list.
[Code]Thread	Defines the number of Cicode threads (tasks) that can run concurrently. The default value has changed from 64 to 128.

## Cicode Functions in Citect SCADA 2018

Some Cicode functions have been introduced or modified. The following sections detail these changes:

### New Functions

#### Alarm Filter Functions

AlarmFilterEditAppendEquipment	Appends the provided expression that can include equipment names or category to the current filter session content without validation.
--------------------------------	--

#### Alarm Functions

AlarmCommentRecID	Allows an operator to add a comment to a selected alarm summary or SOE entry during runtime.
AlarmHighestPriority	Returns the alarm priority and/or state of the current highest priority alarm for the given equipment in conjunction with the selected filter criteria.
AlarmListCreate	Creates an alarms list at a specified AN.
AlarmListDisplay	Displays an alarms list at a specified AN.
AlarmListDestroy	Destroys an alarms list at a specified AN.
AlarmListFill	Fills an alarms list at a specified AN to be used by another routine (e.g. the interlocks processing routine).

AlarmTagFromEquipment	Returns the first alarm tag associated with a piece of equipment.
-----------------------	---

**Array Functions**

ArrayCopy	Makes a copy of an array.
ArrayCreate	Creates an array.
ArrayCreateByAn	Creates an array at a specified AN.
ArrayDestroy	Destroys an array.
ArrayDestroyByAn	Destroys an array associated with the specified AN.
ArrayExists	Determines if an array with a specified handle exists.
ArrayExistsByAn	Determines if an array is associated with a specified AN.
ArrayFillFromAlarmDataByAn	This function is used to fill an array with information from internal alarm record caches.
ArrayGetArrayByAn	Retrieves an array associated with a specified AN.
ArrayGetInfo	Retrieves the size of the x-, y-, or z-dimension for an array.
ArrayGetInt	Retrieves an integer value from an array.
ArrayGetIntByAn	Retrieves an integer value from an array associated with a specified AN.
ArrayGetMapName	Retrieves the name of the map associated with an array.
ArrayGetMapNameByAn	Retrieves the name of the map associated with an array at a specified AN.
ArrayGetString	Retrieves a string from an array.
ArrayGetStringByAn	Retrieves a string from an array associated with a specified AN.
ArrayIsDirty	Determines if an array is 'dirty' (information in the array has changed).
ArraySetInt	Sets an integer value within an array.
ArraySetIntByAn	Sets an integer value in an array associated with a specified AN.

ArraySetIsDirty	Allows you to indicate that an array is dirty (meaning the information in the array has changed).
ArraySetString	Sets a string value in an array.
ArraySetStringByAn	Sets a string value in an array associated with a specified AN.
ArraySwap	Swaps the contents of two arrays.
DspArrayByAn	Displays an alarms list. The array that contains the alarm data is associated with the specified AN.

**Display Functions**

DspAnSetName	Using a valid AN set the name of an animation object.
DspClearClip	Clears the clipping rectangle surrounding the object or group of objects.
DspGetAnFromName	Name used to retrieve the AN of an object on the page.
DspGetAnFromNameRelative	Retrieves the animation number (AN) of an object on the page relative to a given Animation Number(AN).
DspGetAnRawExtent	Gets the extent of the object from the graphics page at the specified AN.
DspGetNameFromAN	Using a valid An the animation name of the object is returned.
DspGetMetadataFromName	Name used to retrieve the metadata of an object on the page.
DspGetMetadataFromNameRelative	Retrieves the metadata of an object on the page relative to a given Animation Number (AN).
DspSetCurColor	Sets the color of the focus rectangle.
DspSetMetadataFromName	Name used to set the metadata of an object on the page.
DspSetPopupMenuFont	Sets the font for popup menu text at runtime.

**Equipment Database Functions**

EquipGetParameter	Reads a runtime parameter of an equipment database record from the EQPARAM.RDB database file.
EquipRefBrowseClose	Terminates an active data browse session and cleans up resources associated with the session.

EquipRefBrowseFirst	Places the reference browse cursor at the first record.
EquipRefBrowseGetField	Retrieves the value of the specified field from the record the data browse cursor is currently referencing.
EquipRefBrowseNext	Moves the data browse cursor forward one record.
EquipRefBrowseNumRecords	Returns the number of records that match the filter criteria.
EquipRefBrowseOpen	Initiates a new browse session and returns a handle to the new session that can be used in subsequent equipment reference browse function calls.
EquipRefBrowsePrev	Moves the data browse cursor back one record.

**Map Functions**

MapClear	Clears all entries in a map and returns the error status.
MapClose	Deletes a previously created map.
MapExists	Check if the map exists using a returned error code.
MapKeyCount	Retrieves the number of keys in a map.
MapKeyDelete	Delete a key and its value from a map.
MapKeyExists	Check if a key exists in a map.
MapKeyFirst	Retrieves the first key in a maps so that all the keys in a map can be discovered.
MapKeyNext	Retrieves the next key after the supplied key in a map so that all the keys in a map can be discovered.
MapOpen	Creates a new map or to open an existing map.
MapValueGet	Retrieves the value from a key in a map.
MapValueSet	Sets a value of a map key.
MapValueSetQuality	Sets the quality of a property in a map.

**Menu Functions**

MenugetNodeByPath	Returns a menu handle corresponding to a menu item expressed as a string path in the format <level>.<level>.<level>.<level>.
MenuNodeGetCurr	Allows you to get the handle of the menu node when this associated command is called.

MenuNodeGetDepth	Returns the depth of a specified menu node within a menu hierarchy.
MenuNodeGetExpanded	Returns the expansion state value of the specified menu node.
MenuNodeGetTargetPage	Returns the target page for a specified menu node.
MenuNodeSetExpanded	Sets the expansion state value of the specified menu node.

**Page Functions**

PageFileInfoEx	Returns the width, height, content type, parent or title of an unopened page.
----------------	---

**Screen Profile Functions**

ResetScreenProfile	Returns all top level windows back to their original screen starting position as defined by the screen profile in a multi-monitor setup.
--------------------	--

**String Functions**

StrEndsWith	Verifies whether the given string ends with a specific string.
StrListContainsItem	Checks whether the string passed is an item contained in a delimited list of strings.
StrReplace	Replaces a substring in a string with replacement substring.
StrSplit	Splits a string into sub-strings based on the specified delimiter. Sub-strings are stored in a new array and a handle to this array is returned.
StrTruncFontToolkit	Returns a truncated string with ellipsis, and sets an AN with a tooltip containing the complete string if a truncation occurs.

**Super Genie Functions**

AssEquipParameters	Associates a set of equipment parameters defined in <b>System Model   Equipment   Runtime Parameters</b> with a page.
AssWinReplace	Removes the associations on a specified Super Genie window and applies any pending associations.

**Tag Functions**

TagGetValue	Reads the value, quality and timestamp of a tag based
-------------	---

	on the tag subscription.
--	--------------------------

**Windows Functions**

WinGetClicked	Gets the number of the Plant SCADA window that has most recently been clicked on using the left mouse button.
WinGetName	Gets the name previously associated with a particular window number using WinSetName.
WinNewPinAt	Opens a new display window at a specified location, relative to the current active window, with a selected page displayed.
WinFreeEx	Removes the active display window and allows you to pass a Windows handle to the function for closing the window.
WinGetFirstChild	Gets the window number of the first child of a parent window.
WinGetNextChild	Gets the window number of the next child in a child link.
WinGetParent	Retrieves the window number of the specified window's parent or root window.
WndMonitorInfoEx	Returns information about a particular monitor using the screen name of the screen profile.

**Modified Functions****Alarm Functions**

AlarmDisable	<i>EndTime</i> and <i>Comment</i> arguments have been added to support alarm shelving.
AlarmDisableRec	<i>EndTime</i> and <i>Comment</i> arguments have been added to support alarm shelving.
AlarmDisableTag	<i>EndTime</i> and <i>Comment</i> arguments have been added to support alarm shelving.
AlarmEnable	<i>bAcknowledge</i> argument has been added to force acknowledgment of an alarm after it is enabled.
AlarmEnableRec	<i>bAcknowledge</i> argument has been added to force acknowledgment of an alarm after it is enabled.
AlarmEnableTag	<i>bAcknowledge</i> argument has been added to force

	acknowledgment of an alarm after it is enabled.
AlarmGetDsp	Alarm response properties are now accessible via the <i>Field</i> argument.
AlarmGetFieldRec	Alarm response properties are now accessible via the <i>Field</i> argument.
AlmBrowseDisable	<i>EndTime</i> and <i>Comment</i> arguments have been added to support alarm shelving.
AlmBrowseEnable	<i>bAcknowledge</i> argument has been added to force acknowledgment of an alarm after it is enabled.
AlmBrowseGetField	Alarm response properties are now accessible via the <i>Field</i> argument.
AlmSummaryDisable	<i>EndTime</i> and <i>Comment</i> arguments have been added to support alarm shelving.
AlmSummaryEnable	<i>bAcknowledge</i> argument has been added to force acknowledgment of an alarm after it is enabled.
AlmTagDisable	<i>EndTime</i> and <i>Comment</i> arguments have been added to support alarm shelving.
AlmTagEnable	<i>bAcknowledge</i> argument has been added to force acknowledgment of an alarm after it is enabled.

**Display Functions**

DspAnInfo	Mode 3 added as an option for the <i>nType</i> argument.
DspText	<i>bTooltip</i> argument added.
DspStr	<i>bTooltip</i> argument added.

**Form Functions**

FormButton	"Disabled" has been added as an option for the <i>Mode</i> argument.
FormPrompt	<i>Width</i> and <i>Height</i> arguments have been added to create an area that the prompt string will wrap within.

**Reinstated Functions**

No functions have been re-instated in Citect SCADA 2018.

## Deprecated Functions

No functions have been deprecated for Citect SCADA 2018.

## Removed Functions

No functions have been removed for Citect SCADA 2018.

## What's New in Citect SCADA 2016

Citect SCADA 2016 incorporates the following new or modified features.

Introduced in Citect SCADA 2016:

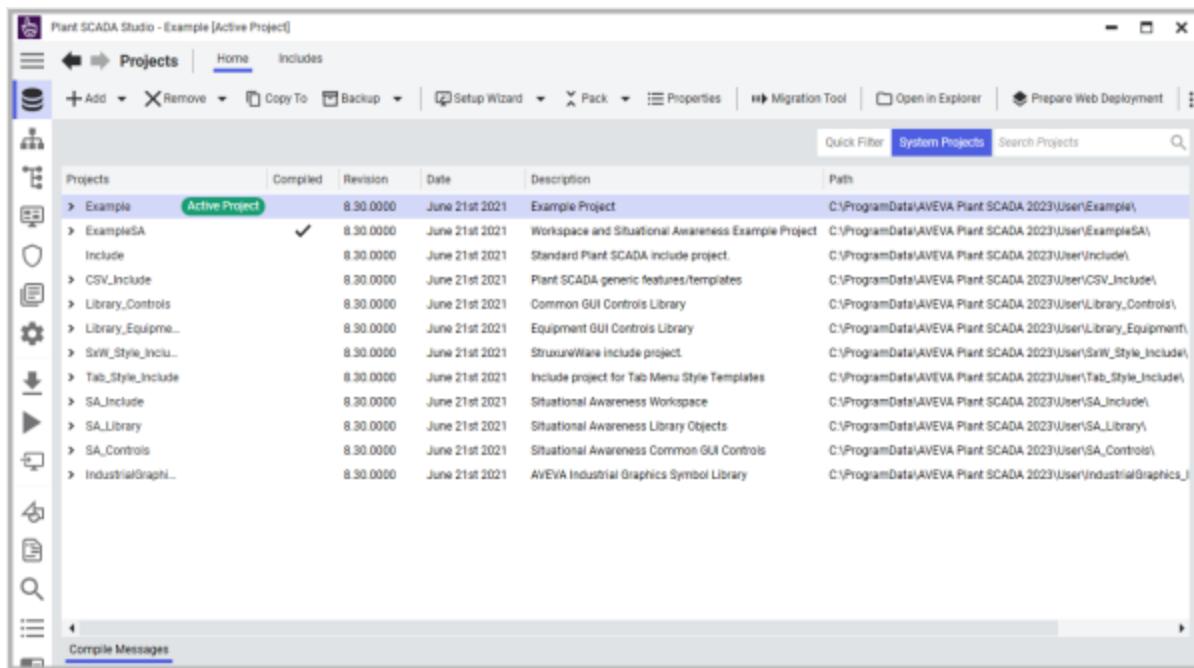
- [Plant SCADA Studio](#)
- [Topology](#)
- [Deployment](#)
- [Calculated Variables](#)
- [Schedule Integration](#)

For changes to:

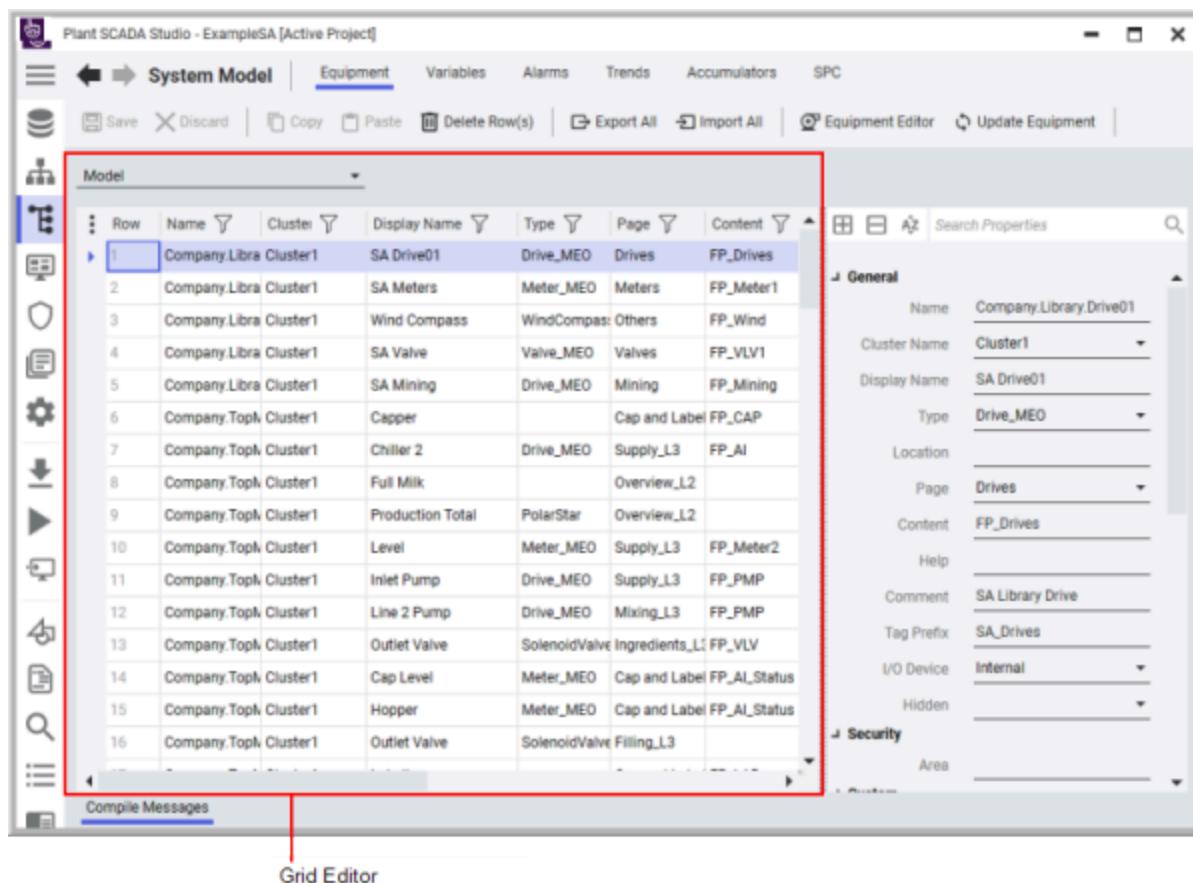
- Citect.ini parameters, see [Citect.ini Parameters in Citect SCADA 2016](#).
- Cicode functions, see [Cicode Functions in Citect SCADA 2016](#).

## Plant SCADA Studio

Plant SCADA 2016 comes with a brand new user interface that is intuitive and easy to use. The new interface replaces the Citect Explorer and the Project Editor and allows you access to projects, topology, system model information such as equipment, variable tags and alarms, and other configuration areas.



The highlight of [Plant SCADA Studio](#) is the Grid Editor, which allows tabular editing of alarms, variable tags, equipment across your entire project hierarchy. The [The Grid Editor](#) makes editing values very easy as a change made to a single record can be propagated throughout the system. The Grid Editor allows for bulk editing of multiple records, which makes it easy to maintain projects. Properties of a selected record are displayed next to the Grid Editor in a [Property Grid](#) where you can update values. Grid Editor views also allow multi-column sorting and filtering.



## See Also

[Plant SCADA Studio](#)

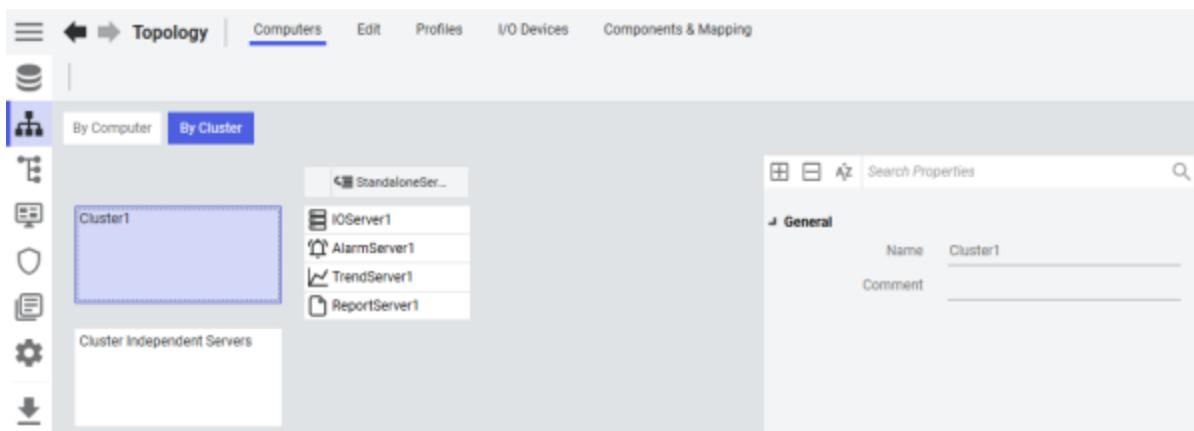
[Plant SCADA Activities](#)

[The Grid Editor](#)

[Property Grid](#)

## Topology

Plant SCADA 2016 allows you to quickly visualize the computers, network addresses, clusters and services being used in your system through a new interface that displays all related information in a single view. This view is available from the [Topology](#) activity of Plant SCADA Studio.



## See Also

[Computers](#)

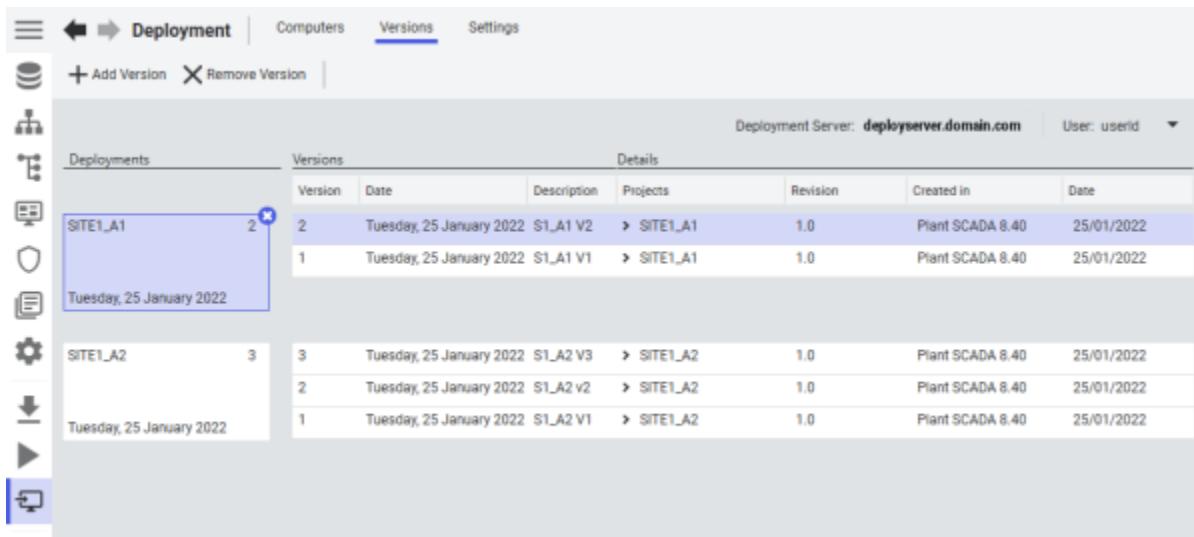
[Clusters](#)

[Server Processes](#)

## Deployment

Plant SCADA 2016 allows you to roll out small and large project changes to computers in your SCADA system located at a single site or across multiple sites quickly and efficiently through the Deployment feature. This feature allows for centralized management of project versions to be deployed to servers and clients including Citect Anywhere clients. Changes can be deployed automatically or through a customizable notification to the operator. Deployed changes can be rolled back if required.

When new project versions are deployed, only the project changes are sent to the computers instead of the entire project. The amount of bandwidth available when deploying changes on the network can also be throttled.



**Note:** It is recommended that you use Deployment instead of the Run/Copy functionality. The Run/Copy functionality is also available for use if required.

## See Also

- [Deployment](#)
- [Prepare Your System for Deployment](#)
- [Using the Deployment Activity](#)

## Calculated Variables

Calculated Variables in Plant SCADA 2016 allow users to generate tags at runtime from a Cicode expression and to call internal Cicode functions. A tag value is calculated for I/O devices that have their protocol set to "CICODE" and a tag associated with the I/O device that has a valid Cicode expression as its address.

Calculated variables significantly reduce the coding required in Cicode functions to manipulate tag values, thus improving code maintainability.

## See Also

- [Calculated Variables](#)
- [Add a Calculated Variable](#)

## Schedule Integration

Scheduler now supports the integration of schedules that are configured on a BACNet device. Locally configured schedule items on a BACNet device can be imported into Plant SCADA. You can also view and modify schedule-object and calendar-object properties on a BACNet device at runtime.

Schedule integration is enabled via the following process:

- A BACnet I/O device is configured in Plant SCADA Studio and connected to Plant SCADA via the BACNET driver.
- Variable tags and equipment definitions that represent the device's schedule and calendar objects are configured on the I/O server through tag import.
- The reports server subscribes to the tags on the I/O server and generates schedule entries in Scheduler.
- Any changes to the schedule entries are written back to the BACnet device via the I/O server.

## See Also

- [Integrate BACnet Schedules into Scheduler](#)

## Citect.ini Parameters in Citect SCADA 2016

This topic lists the parameters that have changed in Citect SCADA 2016.

## New Parameters

The following parameters are new or have been altered in this release. For an entire list of the system parameters, refer to the Parameters documentation.

### Deployment Parameters

[CtEdit]Deploy	The location where a project will be stored when a deployment package is received from the deployment server.
[Deployment]AskRestartArgs	Passes arguments to the Cicode function called by [Deployment]AskRestartFunc.
[Deployment]AskRestartFunc	Calls a Cicode function instead of displaying a restart notification dialog when a prompted deployment occurs.
[Deployment]Enabled	Determines if Runtime Manager runs a project that has been deployed from the deployment server, or the Active Project.

## Modified Parameters

[Win]Configure	Determines whether Plant SCADA Studio and Graphics Builder options are displayed on the control many of the runtime system.
----------------	---

## Removed Parameters

[Lan]SecureLogin	[LAN]SecureLogin is no longer supported.
------------------	--

## Obsolete Parameters

### OID Parameters

[OID]Reset	Resets all OIDs (Object IDs) at compile. This parameter has been removed.
[CtEdit]MaxFields	The maximum number of fields that can display on a Citect Project Editor form.
[CtEdit]ShowToolbar	Shows / hides the toolbar in the Citect Project Editor.

## Cicode Functions in Citect SCADA 2016

Some Cicode functions have been introduced. The following sections detail the changes made to these functions:

### New Functions

#### Alarm Functions

AlarmCountEquipment	Counts the available alarms for specified equipment in conjunction with the selected filter criteria.
---------------------	---

#### Modified Functions

No functions have been modified for Citect SCADA 2016.

#### Reinstated Functions

No functions have been re-instated in Citect SCADA 2016.

#### Deprecated Functions

No functions have been deprecated for Citect SCADA 2016.

#### Removed Functions

No functions have been removed for Citect SCADA 2016.

## What's New in Citect SCADA 2015 SP1

Citect SCADA 2015 SP1 incorporates the following new or modified features.

### Scheduler integration for Bacnet

New properties have been added to the Scheduler ActiveX control to allow you to control:

- The selected cluster
- The selected equipment
- The selected tab
- The selected special day category
- To remove the configuration view.

## Tag Import improvements for Bacnet

Variable tags for Bacnet schedules can now be imported directly from Bacnet compatible devices into CitectSCADA using the Tag Import feature. Equipment can automatically be created Bacnet schedules and Bacnet calendar objects which allows integration into the Scheduler.

## Generate Project Summary

The project summary is an xml based file that contains information about the current project and usercreated include projects. The project summary contains statistics on:

- Variable Tags
- Alarms
- Trends
- I/O Devices
- Internal Cicode functions used

A project summary can be generated by selecting “Generate Project Summary” from the tools menu in Project Editor.

All information in the project summary is anonymous.

## Cicode Profiler

The performance of user created Cicode functions can now be measured using the Cicode profiler. This feature will create a report showing the amount of time each Cicode function has taken to execute. This is useful for debugging large projects with complex Cicode.

## Exclusive Redundancy

I/O Devices now support exclusive redundancy. This can be enabled by setting the “Exclusive” setting to true for an I/O Device and all of its matching redundant devices.

When Exclusive redundancy is enabled, only one redundant device will send requests to the field device at any one time. All I/O Devices will still create a channel to the field device.

## See Also

[Cicode Functions in Citect SCADA 2015](#)

## Cicode Functions in Citect SCADA 2015 SP1

Some Cicode functions have been introduced. The following sections detail the changes made to these functions:

## New Functions

### Page Functions

PageExists	Determines if a page exists or not
------------	------------------------------------

### Tag Functions

TagWriteArrayInt	Writes an array of integer values to a tag.
TagWriteArrayReal	Writes an array of real values to a tag.

## Modified Functions

No functions have been modified for 2015 SP1.

## Reinstated Functions

No functions have been reinstated for 2015 SP1.

## Deprecated Functions

No functions have been deprecated for 2015 SP1.

## Removed Functions

No functions have been removed for 2015 SP1.

## What's New in Citect SCADA 2015

Citect SCADA 2015 incorporates the following new or modified features.

- Alarm Server Upgrade
- Backup and Restore Included Projects
- Incremental Equipment Update
- Link a Genie to an Equipment Type
- Partial Associations can be used with Dynamic Associations
- Running Plant SCADA as a Windows Service
- Extended Memory Mode
- OPC Factory Server Version Update
- Time Scheduler support for Swedish and Norwegian.

For changes to:

- Cicode functions in 2015 and service pack/patches updates, see [Cicode Functions in Citect SCADA 2015](#).

- Citect.ini Parameters 2015 and service pack/patches updates, see [Citect.ini Parameters in Citect SCADA 2015](#).

## Citect.ini Parameters in Citect SCADA 2015

This topic lists the parameters that have changed in Citect SCADA 2015.

### New Parameters

The following parameters are new or have been altered in this release.

#### Alarm Parameters

[Alarm]AlarmListRequestTimeout	Specifies the length of time (in seconds) that an alarm display will wait to receive data from all clusters.
[Alarm]DBLogDBServer	Use to turn on logging for the ClearSCADA Database Server.
[Alarm]DBLogHistoric	When set to 119 this parameter provides logging of historic ClearSCADA data.
[Alarm]DBLogServerCore	Used to find redundancy and synchronization issues.
[Alarm]DeltaTimeUpdate	Determines if DeltaTime (duration) field is set on non-OFF alarms by calculating volatile duration between current time and the time when the alarm was activated.
[Alarm]DisableSOE	Used to turn off the processing for the event journal.
[Alarm]DisableSummary	Allows a user to turn off processing for the summary events in the alarm server.
[Alarm]IsolationDetectInterval	Sets the interval between ICMP packets to detect network isolation on alarm servers.
[Alarm]IsolationDetectIP1	Determines status of the disconnected alarm server when network communication has been interrupted.
[Alarm]IsolationDetectIP2	Determines status of the disconnected alarm server when network communication has been interrupted.
[Alarm]IsolationDetectRetryCount	Sets the ICMP retry count to detect network isolation on alarm servers.
[Alarm]MaxQueryExecuteTime	Creates a log entry if an internal SQL query takes longer than a specified amount of time.
[Alarm]MemoryWarningLimit	Value in Mb, of the threshold of the alarm server memory.

[Alarm]SummaryAutoRefreshMode	Represents the default value for AlarmSetInfo type 15.
[Alarm]SummaryTimeoutTolerance	The length of time from timeout after which an alarm summary entry is committed to Summary Device regardless the fact that Off Time is not set.

**Alarm Process Parameters**

[Alarm<ClusterName><ServerName>]IsolationDetectInterval	Sets the interval between ICMP packets to detect network isolation on alarm servers.
[Alarm<ClusterName><ServerName>]IsolationDetectIP1	Determines status of the disconnected alarm server when network communication has been interrupted.
[Alarm<ClusterName><ServerName>]IsolationDetectIP2	Determines status of the disconnected alarm server when network communication has been interrupted.
[Alarm<ClusterName><ServerName>]IsolationDetectRetryCount	Sets the ICMP retry count to detect network isolation on alarm servers.

**CtEdit Parameters**

[CtEdit]IncrementalEquipmentUpdate	Determines whether an incremental equipment update will occur.
------------------------------------	--

**DBClient Parameters**

[DBClient]Enabled	Enables ODBC logging.
[DBClient]FileBase	Specifies a location for the ODBC log files.
[DBClient]MaxFiles	Specifies the maximum number of ODBC log files that are retained.
[DBClient]MaxSize	Specifies the maximum size for an ODBC log file (in kilobytes).
[DBClient]OldFiles	Specifies the maximum number of log file sets that can be retained.

**LAN Parameters**

[LAN]HeartbeatPeriod	Controls how frequently a tran channel sends a heartbeat packet to the other peer.
[LAN]HeartbeatTimeout	Controls how much idle time on network is accepted prior to dropping the tran connection.

**Modified Parameters**

[Alarm]DisplayDisable	In Plant SCADA 2015, when you set this parameter to
-----------------------	---

	1 (disabled alarms are suppressed), disabled alarm will now be listed on the Alarm Summary page.
[Debug]CategoryFilter	New alarm filters are now supported.
[LAN]EarliestLegacyVersion	The allowable values were updated and the default value is now "7500".

## Reinstated Parameters

[Alarm]StartTimeout	Sets the timeout period for loading each packet from the primary Alarms Server. This parameter has been reinstated for v2015 only.
---------------------	--

## Obsolete Parameters

### Alarm Parameters

[Alarm]ArgyleTagValueTimeout	Defines the length of time that the alarm server will wait for argyle tag values to become available (without error) before starting to scan for argyle alarms.
[Alarm]DefaultSOETimeRange	Applies a time range filter to all SOE queries.
[Alarm]SOERowLimit	Defined the maximum number of SOE rows per cluster that can be displayed on an SOE page.
[Alarm]SummaryLength	The maximum number of alarm summary entries that can be held in memory.
[Alarm]SumStateFix	Determined whether an alarm summary entry maintained its state information when the alarm changed to an OFF state.

### LAN Parameters

[LAN]KeepAliveInterval	Sets the length of time between two keep alive transmissions by the client.
[LAN]KeepAliveTime	Sets the length of time between two keep alive transmissions in idle conditions.

## Cicode Functions in Citect SCADA 2015

Some Cicode functions have been introduced. The following sections detail the changes made to these functions:

## New Functions

### .Net Functions

DllClassDispose	Use this function to clean up resources used by the .Net object and any other .Net objects created via the use of the object.
DllClassCreate	Use this function to instantiate a new .Net object by specifying the path, class and arguments required for the matching constructor of the class.
DllClassGetProperty	Use this function to get a property of the .Net object.
DllClassIsValid	Use this function to validate class. Uses the handle of the class returned from DllClassCreate.
DllClassCallMethod	Use this function to call a method of a .Net object, passing in the method name and any arguments required for the matching prototype of the method.
DllClassSetProperty	Use this function to set a property of the .Net object. The property may be of any type or an object itself.

## Modified Functions

No functions have been modified for Citect SCADA 2015.

## Reinstated Functions

### Alarm Functions

AlarmDelete	Deletes alarm summary entries that are currently displayed.
AlarmSplit	Splits an alarm summary entry which has no Off time.
AlarmSumAppend	Appends a new blank record to the alarm summary.
AlarmSumCommit	Commits the alarm summary record to the alarm summary device.
AlarmSumDelete	Deletes alarm summary entries.
AlarmSumFind	Finds an alarm summary index for an alarm record and alarm on time.
AlarmSumFirst	Gets the oldest alarm summary entry.
AlarmSumGet	Gets field information from an alarm summary entry.

AlarmSumLast	Gets the latest alarm summary entry.
AlarmSumNext	Gets the next alarm summary entry.
AlarmSumPrev	Gets the previous alarm summary entry.
AlarmSumSet	Sets field information in an alarm summary entry.
AlarmSumSplit	Duplicates an alarm summary entry.
AlarmSumType	Retrieves a value that indicates a specified alarm's type.

## Deprecated Functions

No functions have been deprecated for Citect SCADA 2015.

## Removed Functions

No functions have been removed for Citect SCADA 2015.

## What's New in Citect SCADA 7.40 SP1

Citect SCADA 7.40 SP1 incorporates the following new or modified features.

Introduced in 7.40 SP1:

- Alarm Performance Improvement
- Enhancements to the Equipment Editor
- Library\_equipment include project
- EcoStruxure Web Services Client

---

**Note:** With the release of Plant SCADA version 2020 R2, EcoStruxure Web Services (EWS) are no longer supported.

---

For changes to:

- Cicode functions in 7.40 SP1 and service pack/patches updates, see [Cicode Functions in 7.40 SP1](#)
- Citect.ini Parameters 7.40 SP1 and service pack/patches updates, see [Citect.ini Parameters in 7.40 SP1](#)

## Citect.ini Parameters in 7.40 SP1

This topic lists the parameters that have been added in version 7.40 SP1 of Citect SCADA.

## New Parameters

The following parameters are new in version 7.20. For an entire list of the system parameters, refer to the

Parameters documentation.

### Alarm Parameters

[Alarm]WebClientUpdatePollPeriod	Sets the polling period in milliseconds for web client to get data updates.
[Alarm]ClientUpdatePollPeriod	Sets the polling period in milliseconds for the display client to get data updates.

### Modified Parameters

No parameters were modified in 7.40 SP1

### Obsolete Parameters

No parameters were made obsolete in 7.40 SP1

### Cicode Functions in 7.40 SP1

Some Cicode functions have been introduced. The following sections detail the changes made to these functions:

### New Functions

#### Security Functions

GetLanguage	Gets the language currently used on the display client.
-------------	---

### Modified Functions

#### Alarm Functions

AlarmGetInfo	Gets data on the alarm list displayed at a specified AN. A new type of 13 was added to return the ready state of the data on an alarm display view.
--------------	---

### Reinstated Functions

No functions have been reinstated for 7.40 SP1.

### Deprecated Functions

No functions have been deprecated for 7.40 SP1.

## Removed Functions

No functions have been removed for 7.40 SP1.

## What's New in Citect SCADA 7.40

Citect SCADA 7.40 incorporates the following new or modified features.

- Referencing a variable tag using Equipment.Item
- New Cicode functions to maintain a list of unique Pages "opened"
- New Equipment Editor
- New SxW Templates
- OPC Factory Server (OFS) Security Updates

For changes to:

- Cicode functions in 7.40 and service pack/patches updates, see [Cicode Functions in 7.40](#)
- Citect.ini Parameters 7.40 and service pack/patches updates, see [Citect.ini Parameters in 7.40](#)

## Citect.ini Parameters in 7.40

This topic lists the parameters that have been added or changed in version 7.40 of Citect SCADA.

### New Parameters

The following parameters are new in version 7.20. For an entire list of the system parameters, refer to the Parameters documentation.

#### CTEdit Parameters

[CTEDIT]DisplayEquipmentItem	Used to control the population of the variable tag list, or equipment item list in graphics builder.
------------------------------	--

#### General Parameters

[General]TagDBReloadOnChange	Determines whether the Variable Tags database is checked for changes and reloaded when a new page is displayed.
------------------------------	---

#### Page Parameters

[Page]MaxList	The maximum number of pages that can be placed on the page list stack.
---------------	--

#### Server Parameters

[Server]AllowAnonymousAccess	Determines whether the EWS Server will allow the
------------------------------	--

	EWS Client anonymous data access.
--	-----------------------------------

## Modified Parameters

### Code Parameters

[Code]Stack	The size of the Cicode stack. The default has been changed from 127 to 256.
-------------	---

### General Parameters

[General]TagDB	Determines whether the Variable Tags database is loaded at runtime. The Variable Tags database needs to be loaded to allow tags to be referenced with the Equipment.Item syntax.
----------------	--

## Cicode Functions in 7.40

Some Cicode functions have been introduced, and modified. The following sections detail the changes made to these functions:

## New Functions

### Page Functions

PageListCount	Gets number of pages in the page list of the current window.
PageListCurrent	Gets index of the current page in the page list of the current window.
PageListInfo	Gets information of a page at the specific index in the page list of current window.
PageListDisplay	Displays a page at the specific index in the page list of the current window, and moves the current index to the page. When a page is recalled, the original parameters (such as cluster context, super genie associations, PageTask arguments if applicable) used to display the page will be restored.
PageListDelete	Deletes a page at the specific index from the page list of the current window.

### XML Functions

XMLClose	Deletes an XML document in memory
XMLCreate	Creates a new XML document in memory
XMLGetAttribute	Retrieves the attribute value of the node from an XML document in memory
XMLGetAttributeCount	Retrieves the number of attributes (properties of a node. Each attribute has a name and a value) within an XML document in memory
XMLGetAttributeName	Retrieves the name of an attribute (property of a node. Each attribute has a name and a value) within an XML document in memory
XMLGetAttributeValue	Retrieves the value of an attribute (property of a node. Each attribute has a name and a value) within an XML document in memory
XMLGetChild	Retrieves the child node for the specified parent node in XML document in memory
XMLGetChildCount	Retrieves the total number of child nodes for the specified parent node in an XML document in memory
XMLGetParent	Retrieves the parent node within the contents of an XML document in memory
XMLGetRoot	Retrieves the root node of an XML document in memory
XMLNodeAddChild	Creates an element node with the specified Name and Namespace and appends the node to the end of the list of child nodes of specified parent node in the XML document.
XMLNodeFind	Selects a single node from the contents of an XML document in memory
XMLNodeGetName	Retrieves the name of the specified node
XMLNodeGetValue	Retrieves the value of a node from the contents of an XML document in memory
XMLNodeRemove	Removes specified XML node from its parent and XML document
XMLNodeSetValue	Sets the value of the specified node.
XMLOpen	Loads an XML file from disk

XMLSave	Saves an XML file to disk
XMLSetAttribute	Sets the value of specified attribute of the node in the XML document. If the attribute does not exist, it will be created.

## Modified Functions

### Super Genie Functions

Ass	Associates a variable tag with a Super Genie. Now supports Equipment.Item.
-----	--

### Tag Functions

TagGetProperty	Gets a property for a variable tag from the datasource. Now supports Equipment.Item.
TagGetScale	Gets the value of a tag at a specified scale from the datasource. Now supports Equipment.Item.
TagInfo	Gets information about a variable tag. Now supports Equipment.Item.
TagInfoEx	Gets information about a variable tag. Now supports Equipment.Item.
TagRead	Reads a variable from the I/O device. Now supports Equipment.Item.
TagReadEx	Reads the value, quality or timestamp of a particular tag from the I/O device. Now supports Equipment.Item.
TagResolve	Used to increment a reference count on a tag to keep it resolved, making it readily available to a client. Now supports Equipment.Item.
TagScaleStr	Gets the value of a tag at a specified scale. Now supports Equipment.Item.
TagSetOverrideBad	Sets a quality Override element for a specified tag to Bad Non Specific. Now supports Equipment.Item.
TagSetOverrideGood	Sets a quality Override element for a specified tag to Good Non Specific. Now supports Equipment.Item.
TagSetOverrideUncertain	Sets a quality Override element for a specified tag to Uncertain Non Specific. Now supports Equipment.Item.

TagSetOverrideQuality	Sets a quality Override element for a specified tag. Now supports Equipment.Item.
TagSubscribe	Subscribes a tag for periodic monitoring and event handling. Now supports Equipment.Item.
TagWrite	Writes to an I/O Device variable. Now supports Equipment.Item.

## Reinstated Functions

No functions have been reinstated for 7.40.

## Deprecated Functions

No functions have been deprecated for 7.40.

## Removed Functions

No functions have been removed for 7.40.

## CtAPI Functions in 7.40

The following section details the changes made to CtAPI functions in Plant SCADA 7.40.

## New Functions

No new functions have been added for 7.40.

## Modified Functions

ctListAdd	Adds a tag to the list. Now accepts "Equipment.item".
ctListAddEx	Adds a tag to the list with a specified poll period. Now accepts "Equipment.item".
ctTagGetProperty	Gets the given property of the tag. Now accepts "Equipment.item".
ctTagRead	Reads the current value from the given I/O Device variable tag. Now accepts "Equipment.item".
ctTagReadEx	Performs the same as ctTagRead, but with an additional new argument. Now accepts "Equipment.item".

ctTagWrite	Writes the given value to the I/O Device variable tag. Now accepts "Equipment.item".
ctTagwriteEx	Performs the same as ctTagWrite, but with an additional new argument. Now accepts "Equipment.item".

## Obsolete Functions

No functions were made obsolete in 7.40.

## What's New in Citect SCADA 7.30

Citect SCADA 7.30 incorporates the following new or modified features.

- ActiveX Data Objects (ADO.NET)
- Alarm Enhancements
- GUI Enhancements
- Equipment Hierarchy
- Importing Equipment
- New Licensing Method
- OPC DA Server
- Software Update
- Supportability
- Tag Browsing
- Scheduler

For changes to:

- Cicode functions in 7.30 and service pack/patches updates, see [Cicode Functions in 7.30](#)
- Citect.ini Parameters 7.30 and service pack/patches updates, see [Citect.ini Parameters in 7.30](#)

## Citect.ini Parameters in 7.30

This topic lists the parameters that have been added or changed in version 7.30 of Citect SCADA.

### New Parameters

The following parameters are new in version 7.20 . For an entire list of the system parameters, refer to the Parameters documentation.

#### Alarm Parameters

[Alarm]DefaultSOETimeRange	Specifies the default time range, in days, for SOE views
----------------------------	--

	that have no other time-based filter.
[Alarm]DefSOEFormat	Specifies an SOE display format to use if the SOE Display Format field is blank (in Alarm Categories).
[AlarmFilterRuleList.Active]Rule<n>	Defines the name of rules to appear on the Simple Rule dropdown list of the active alarm filter form.
[AlarmFilterRuleList.Disabled]Rule<n>	Defines the name of rules to appear on the Simple Rule dropdown list of disabled alarm filter form.
[AlarmFilterRuleList.SOE]Rule<n>	Defines the name of rules to appear on the Simple Rule dropdown list of alarm summary filter form.
[AlarmFilterRuleList.Summary]Rule<n>	Defines the name of rules to appear on the Simple Rule dropdown list of alarm summary filter form.
[AlarmFilterRules]<RuleName>	Defines the filter expression represented by the rule name.
AlarmFilterRuleList].Rule<n>	Defines the name of the common rules to appear on the Simple Rule dropdown list of all alarm filter form.

**AlarmFilterRules Parameters**

[AlarmFilterRuleList.Active]Rule<n>	Defines the name of rules to appear on the Simple Rule dropdown list of the active alarm filter form.
[AlarmFilterRuleList.Disabled]Rule<n>	Defines the name of rules to appear on the Simple Rule dropdown list of disabled alarm filter form.
[AlarmFilterRuleList.SOE]Rule<n>	Defines the name of rules to appear on the Simple Rule dropdown list of alarm summary filter form.
[AlarmFilterRuleList.Summary]Rule<n>	Defines the name of rules to appear on the Simple Rule dropdown list of alarm summary filter form.
[AlarmFilterRules]<RuleName>	Defines the filter expression represented by the rule name.
AlarmFilterRuleList].Rule<n>	Defines the name of the common rules to appear on the Simple Rule dropdown list of all alarm filter form.

**Alarm Process Parameters**

Alarm.<ClusterName>.<ServerName> ArchiveAfter	The archive after time (Event Journal) is the amount of time between each archive of Event Journal data.
Alarm.<ClusterName>.<ServerName> CacheSize	Defines the amount of memory (in megabytes) dedicated to the storage of event data.
Alarm.<ClusterName>.<ServerName>	Defines the amount of time, in milliseconds, in which

ClientConnectTimeout	the client can attempt to make a connection.
Alarm.<ClusterName>.<ServerName> ClientDisconnectTimeout	Defines the amount of time, in milliseconds, in which the client can attempt to terminate a connection to the server.
Alarm.<ClusterName>.<ServerName>ClientRequestTimeout	Defines the amount of time, in milliseconds, in which the client can request data from a server.
Alarm.<ClusterName>.<ServerName> FutureMessages	Event Journal records that have a time stamp with a date and time in the future can be stored historically.
Alarm.<ClusterName>.<ServerName> HeartbeatTimeout	Defines how long a server will wait before terminating a link that has been used for receiving heartbeat poll requests from its pair server, but is currently idle.
Alarm.<ClusterName>.<ServerName> KeepOnlineFor	The Event Journal Life is the amount of time for which the Alarm Server stores event messages on-line.
Alarm.<ClusterName>.<ServerName> MonitorConnectTimeout	Defines the amount of time, in seconds, that the server will wait for a monitor connection to occur.
Alarm.<ClusterName>.<ServerName> MonitorRequestTimeout	Defines the amount of time, in seconds, that the server will wait for a response from the other server in the pair.
Alarm.<ClusterName>.<ServerName>QueryCPUUsage	Defines the percentage of processor use you want to allocate to query searches.
Alarm.<ClusterName>.<ServerName> QueryRowLimit	Defines the maximum number of rows that can be returned in the result set for a single query.
Alarm.<ClusterName>.<ServerName>QueryTimeout	Defines the amount of time (in seconds) that is permitted for query searches.
Alarm.<ClusterName>.<ServerName> StreamSize	Defines the amount of data that is included in each event data file.
Alarm.<ClusterName>.<ServerName> SyncAllHistoricData	On multi-server systems, the Primary server and Standby server synchronize their data so that the Standby server contains an accurate, up to date backup of the Primary server's data.
Alarm.<ClusterName>.<ServerName>TransferConnectTimeout	Defines the amount of time, in seconds, that the Primary server will wait for a connection to occur.
Alarm.<ClusterName>.<ServerName> TransferInterleave	Controls how often the data synchronization is triggered by the Primary to the Standby Server.
Alarm.<ClusterName>.<ServerName>TransferInterval	Defines the number of seconds between each attempt to update the data on the Standby server.

**BrowseTableView Parameters**

[BrowseTableView]<BrowseType>.<ViewName>.ColWidths	Sets the column widths in pixels of the current data browse table.
[BrowseTableView]<BrowseType>.<ViewName>.Fields	Sets the field names of the columns in the current data browse table under the View Name configured on the page.

**ClientParameters**

[Client]PointCountRequired	Specifies what license point count a client requires.
----------------------------	---

**General Parameters**

[General]ClusterReplication	Controls whether tag will be replicated in a multi-cluster system.
[General]LicenseReservationTimeout	Specifies the number of seconds to reserve a license for a given IP address in cases where a remote client connection is lost.

**Page Parameters**

[Page]SOEPage	The name of the graphics page to display when you call up an sequence of events (SOE) page via the Cicode function PageSOE().
---------------	---

**SQL Parameters**

[SQL]MaxConnections	Defines the maximum number of DB connection objects.
---------------------	--

**Scheduler Parameters**

[Scheduling]PersistPath	Directs where the configuration data for the scheduler is stored.
[Scheduling]StartDelay	Sets the delay from when the Scheduler's server components are initialized to the point when Scheduler begins processing active schedule entries.

**Modified Parameters****Alarm Parameters**

[Alarm]SavePrimary	This parameter is now used only to import alarm history from previous versions of Plant SCADA.
[Alarm]SaveSecondary	This parameter is now used only to import alarm history from previous versions of Plant SCADA.

[Alarm]SummaryLength	The maximum number of alarm summary entries that can be held in memory. The maximum number for this parameter has been modified from 4096000 to 100000.
----------------------	---

**Language Parameters**

Language]LocalLanguage	Used to set the default language during start-up.
------------------------	---

**SQL Parameters**

[SQL]QueryTimeout	Sets the timeout period for SQL queries globally.
-------------------	---

**Tab Style Template Parameters**

[Format]FormatName	Define the display format by name.
--------------------	------------------------------------

**Obsolete Parameters**

[Alarm]Ack	Determined whether Plant SCADA acknowledges current alarms on startup.
[Alarm]AckHold	Determined whether alarms that have become inactive (and have been acknowledged) remain in the OFF ACKNOWLEDGED alarm list.
[Alarm]CacheLength	The maximum number of alarms that can be held in the cache of a client
[Alarm]FilterViewByPrivilege	If privilege is not checked, a user with no privilege (0) can browse and view trends and alarms that require privilege 1. This behavior is the same as [Alarm]FilterViewByPrivilege = 0 in 7.20.  The set of records returned from browse is now filtered by area.
[Alarm]SavePeriod	Set the path to the primary save file.
[Alarm]SaveStyle	Determines whether alarms records are identified by their record number or alarm tag.
[Alarm]StartTimeout	Sets the timeout period for loading data from the primary Alarms Server.

**Cicode Functions in 7.30**

Some Cicode functions have been introduced, modified, deprecated or removed. The following sections detail the changes made to these functions:

## New Functions

### Alarm Functions

AlarmAckTag	Acknowledge a specified alarm.
AlarmCount	Counts the available alarms for the selected filter criteria.
AlarmCountList	Counts the available alarms for the selected alarm list (selected by its animation).
AlmBrowseAck	Acknowledges the alarm tag at the current cursor position in an active data browse session.
AlmBrowseClose	Closes an alarm tags browse session.
AlmBrowseDisable	Disables the alarm tag at the current cursor position in an active data browse session.
AlmBrowseEnable	Enables the alarm tag at the current cursor position in an active data browse session.
AlmBrowseFirst	Gets the oldest alarm tags entry.
AlmBrowseGetField	Gets the field indicated by the cursor position in the browse session.
AlmBrowseNext	Gets the next alarm tags entry in the browse session.
AlmBrowseNumRecords	Returns the number of records in the current browse session.
AlmBrowseOpen	Opens an alarm tags browse session.
AlmBrowsePrev	Gets the previous alarm tags entry in the browse session.
AlarmFilterClose	Removes named filter from memory.
AlarmFilterEditAppend	Appends the provided expression to the current filter session content without any validation.
AlarmFilterEditClose	Removes the session from the memory.
AlarmFilterEditCommit	Validates the filter built in this session and, if valid, applies the filter to the list associated with the session.
AlarmFilterEditFirst	Retrieves the first part of the filter.
AlarmFilterEditLast	Retrieves the last part of the filter.

AlarmFilterEditNext	Retrieves the next part of the filter.
AlarmFilterEditOpen	Creates a session for the historical list associated with the provided animation number (AN).
AlarmFilterForm	Displays a form for specifying filtering criteria for either an alarm list or a named filter.
AlarmFilterOpen	Creates a named filter.
AlmFilterEditPrev	Retrieves the previous part of the filter.
AlmFilterEditSet	Replaces the current filter session content by the provided expression without any validation.
AlarmGetFilterName	Retrieves the name of the linked filter for the supplied AN.
AlarmResetQuery	Clears the filter of the specified filter source. Used to reset the filter set up by the Cicode function AlarmFilterForm().
LibAlarmFilterForm	Displays a generic alarm filter pop-up for specifying filtering criteria for either an alarm list or a named filter.

**Equipment Functions**

EquipSetProperty	Sets the property of an item of equipment.
EquipStateBrowseClose	Terminates a browsing session and cleans up the resources used by the session.
EquipStateBrowseFirst	Places the data browse cursor at the first record.
EquipStateBrowseGetField	Returns the value of the particular field in a record to which the data browse cursor is currently referencing.
EquipStateBrowseNext	Places the data browse cursor at the next available record.
EquipStateBrowseNumRecords	Returns the number of records that match the current filter criteria.
EquipStateBrowseOpen	Initiates a new session for browsing the equipment states configured.
EquipStateBrowsePrev	Places the data browse cursor at the previous record.

**Page Functions**

PageSOE	Displays a category of sequence of events (SOE)
---------	---

	entries on the SOE page.
--	--------------------------

**Scheduler Functions**

SchdClose	Terminates a browsing session and cleans up the resources used by the session.
SchdConfigClose	Terminates a browsing session and cleans up the resources used by the session.
SchdConfigFirst	Places the data browse cursor at the first record.
SchdConfigGetField	Returns the value of the particular field in a record to which the data browse cursor is currently referencing.
SchdConfigNext	Places the data browse cursor at the next available record.
SchdConfigNumRecords	Returns the number of records that match the current filter criteria.
SchdConfigOpen	Initiates a new session for browsing the schedules configured.
SchdConfigPrev	Places the data browse cursor at the previous record.
SchdFirst	Places the data browse cursor at the first record.
SchdGetField	Returns the value of the particular field in a record to which the data browse cursor is currently referencing.
SchdNext	Places the data browse cursor at the next available record.
SchdNumRecords	Returns the number of records that match the current filter criteria.
SchdOpen	Initiates a new session for browsing the runtime schedules.
SchdPrev	Places the data browse cursor at the previous record.
SchdSpecialAdd	Adds a new special day group to the scheduler engine.
SchdSpecialClose	Terminates a browsing session and cleans up the resources used in the session.
SchdSpecialDelete	Deletes an existing special day group.
SchdSpecialFirst	Places the data browse cursor at the first record.
SchdSpecialGetField	Returns the value of the particular field in a record to

	which the data browse cursor is currently referencing.
SchdSpecialItemAdd	Adds a new special day to the scheduler engine.
SchdSpecialItemClose	Terminates a browsing session and cleans up the resources used in the session.
SchdSpecialItemDelete	Deletes an existing schedule.
SchdSpecialItemFirst	Places the data browse cursor at the first record.
SchdSpecialItemGetField	Returns the value of the particular field in a record to which the data browse cursor is currently referencing.
SchdSpecialItemModify	Modifies an existing special day.
SchdSpecialItemNext	Places the data browse cursor at the next available record.
SchdSpecialItemNumRecords	Returns the number of records that match the current filter criteria.
SchdSpecialItemOpen	Initiates a new session for browsing the special days.
SchdSpecialItemPrev	Places the data browse cursor at the previous record.
SchdSpecialModify	Modifies an existing special day group.
SchdSpecialNext	Places the data browse cursor at the next available record.
SchdSpecialNumRecords	Returns the number of records that match the current filter criteria.
SchdSpecialOpen	Initiates a new session for browsing the special day groups.
SchdSpecialPrev	Places the data browse cursor at the previous record.
ScheduleItemAdd	Adds a new schedule to the scheduler engine.
ScheduleItemDelete	Deletes an existing schedule.
ScheduleItemModify	Modifies an existing schedule.
ScheduleItemSetRepeat	Adds recurrence information for an existing schedule to the scheduler engine.

**Sequence of Event Functions**

SOEArchive	Archives event journal.
SOEDismount	Use to dismount archive volume.

SOEventAdd	Inserts a new event into the event journal.
SOEMount	Use to mount archive volume.

**SQL Functions**

SQLCall	Executes an SQL query on a database
SQLClose	Closes a SQL connection between the DB connection object specified by the function's parameter and a database
SQLCreate	Creates an internal DB connection object and returns a handle to the object for use by the other DB functions
SQLDispose	Disposes the DB connection object
SQLGetRecordset	Executes an SQL query on a database
SQLGetScalar	Executes an SQL query on a database
SQLIsNullField	Checks presence of null value in field from a recordset
SQLNumFields	Gets the number of fields or columns that were returned by the last SQL statement
SQLOpen	Opens an SQL connection between the DB connection object
SQLParamsClearAll	Turns on a debug trace
SQLParamsSetAsInt	Adds or replaces a parameterized query's parameter as integer and its value in the specified connection
SQLParamsSetAsReal	Adds or replaces a parameterized query's parameter as real and its value in the specified connection
SQLParamsSetAsString	Adds or replaces a parameterized query's parameter as string and its value in the specified connection
SQLPrev	Gets the previous database record from an SQL query.
SQLQueryCreate	The function creates a new query and returns its handle
SQLQueryDispose	The function disposes the query which handle is given as the argument.
SQLRowCount	Gets the number of rows in the recordset.

**Timestamp Functions**

StrToTimestamp	Converts timestamp in a STRING format into a TIMESTAMP format
----------------	---

**Tag Functions**

TagBrowseClose	Close an existing browsing session
TagBrowseFirst	Move to the first record
TagBrowseGetField	Get the specified field of a record.
TagBrowseNext	Move to the next record
TagBrowseNumRecords	Get the number of records for a given browsing session.
TagBrowseOpen	Opens a new browsing session.
TagBrowsePrev	Move to the previous record

**Modified Functions****Alarm Functions**

AlarmAck	Acknowledges an active alarm.
AlarmCatGetFormat	Returns the display format string of the specified alarm category. Type has been extended to include SOE format.
AlarmDisable	Disables an alarm.
AlarmDsp	Displays alarms.
AlarmDspNext	Displays the next page of alarms. Works with new SOE display type.
AlarmDspPrev	Displays the previous page of alarms. Works with new SOE display type.
AlarmEnable	Enables a disabled alarm.
AlarmFirstTagRec AlarmFirstCatRec AlarmFirstPriRec AlarmFirstQueryRec AlarmNextTagRec AlarmNextCatRec AlarmNextPriRec AlarmNextQueryRec AlarmAckRec	Alarm 'Rec' functions listed are now executed in the client process, with the function MsgRPC no longer required when called remotely to the Alarm Server.

AlarmEnableRec	
AlarmDisableRec	
AlarmGetDelayRec	
AlarmSetDelayRec	
AlarmGetThresholdRec	
AlarmSetThresholdRec	
AlarmGetFieldRec	
AlarmGetDsp	Retrieves field data from the alarm record that is displayed at the specified AN. Works with new SOE display type.
AlarmGetInfo	Retrieves data on the alarm list displayed at a specified AN. New type 12 added.
AlarmSetInfo	Controls different aspects of the alarm list displayed at a specified AN. Supports automatic refresh of the new SOE display type.
AlmSummaryGetField	Gets the field indicated by the cursor position in the browse session. Now supports Equipment field.
AlmSummaryOpen	Opens an alarm summary browse session. Now supports Equipment field. Will not return data for 'NODE' field name.
AlmTagsGetField	Gets the field indicated by the cursor position in the browse session. Now supports Equipment field.
AlmTagsOpen	Opens an alarm tags browse session. Now supports Equipment field. Will not return data for 'NODE' field name.

**Accumulator Functions**

AccumBrowseGetField	Gets the field indicated by the cursor position in the browse session. Now supports Equipment field.
AccumBrowseOpen	Opens an accumulator browse session. Now supports Equipment field.

**Equipment Functions**

EquipBrowseGetField	Gets the field indicated by the cursor position in the browse session. Now supports Parent and Composite fields.
EquipBrowseOpen	Opens an equipment database browse session. Now supports Parent and Composite fields.
EquipGetProperty	Reads a property of an equipment database record

	from the EQUIP.DBF file. Now supports Parent and Composite fields.
--	--

**Format Functions**

FmtOpen	Opens a format template. mode has been extended to include SOE format.
---------	--

**Security Functions**

Login	Logs a user into the Plant SCADA system, using Plant SCADA security and gives users access to the areas and privileges assigned to them in the Users database. New sLanguage parameter added.
UserLogin	Logs a user into the Plant SCADA system, using either Windows security or Plant SCADA security and gives users access to the areas and privileges assigned to them in the Users database. New sLanguage parameter added.

**Server Functions**

ServeGetProperty	Returns information about a specified server and can be called from any client.
ServerReload	Reloads the server specified by cluster and server name.

**Super Genie Functions**

AssGetProperty	Gets association information about the current Super Genie from the datasource.
AssInfo	Gets association information about the current Super Genie.
AssInfoEx	Gets association information about the current Super Genie.

**SQL Functions**

SQLGetField	Gets field or column data from a database record.
SQLInfo	Gets information about a database connection. No longer supports type 3 and 4.
SQLNoFields	Gets the number of fields or columns that were returned by the last SQL statement.

**Tag Functions**

TagGetProperty	Gets a property for a variable tag from the datasource. Now supports Equipment field.
TagInfo	Gets information about a variable tag. Now supports Equipment field.
TagInfoEx	Gets information about a variable tag. Now supports Equipment field.

**Trend Functions**

TrnBrowseGetField	Gets the field indicated by the cursor position in the browse session. Now supports Equipment field.
TrnBrowseOpen	Opens a trend browse session. Now supports Equipment field.

**Reinstated Functions**

No functions have been reinstated for 7.30.

**Deprecated Functions**

AlmTagsEnable	Enables the alarm tag at the current cursor position in an active data browse session.
AlmTagsDisable	Disables the alarm tag at the current cursor position in an active data browse session.
AlmTagsNext	Gets the next alarm tags entry in the browse session.
AlmTagsAck	Acknowledges the alarm tag at the current cursor position in an active data browse session.
AlmTagsClear	Clears the alarm tag at the current cursor position in an active data browse session.
AlmTagsClose	Closes an alarm tags browse session.
AlmTagsFirst	Gets the oldest alarm tags entry.
AlmTagsGetField	Gets the field indicated by the cursor position in the browse session.
AlmTagsNumRecords	Returns the number of records in the current browse session.

AlmTagsOpen	Creates a session for the historical list associated with the provided animation number (aN).
AlmTagsPrev	Gets the previous alarm tags entry in the browse session.

## Removed Functions

AlmBrowseClear	Clears the alarm tag at the current cursor position in an active data browse session. Now obsolete.
AlarmClear	Clears acknowledged, inactive alarms from the active alarm list.
AlarmClearRec	Clear an alarm by its record number. Now obsolete.
AlarmDelete	Deletes alarm summary entries that are currently displayed. Now obsolete.
AlarmSetQuery	Specifies a query to be used in selecting alarms for display. Now Obsolete. Use the new Alarm Filter Edit functions.
AlarmSumAppend	Appends a new blank record to the alarm summary. Now obsolete.
AlarmSumCommit	Commits the alarm summary record to the alarm summary device. Now obsolete.
AlmSummaryCommit	Commits the alarm summary record to the alarm summary device. Now obsolete.
AlarmSplit	Duplicates an alarm summary entry where the cursor is positioned. Now obsolete.
AlarmSumDelete	Deletes alarm summary entries. Now obsolete.
AlarmSumFind	Finds an alarm summary index for an alarm record and alarm on time. Now obsolete.
AlarmSumFindExact	Finds the alarm summary index for an alarm specified by the alarm record identifier and alarm activation time.
AlarmSumFirst	Gets the oldest alarm summary entry. Now

	obsolete.
AlarmSumGet	Gets field information from an alarm summary entry. Now obsolete.
AlarmSumLast	Gets the latest alarm summary entry. Now obsolete.
AlarmSumNext	Gets the next alarm summary entry. Now obsolete.
AlarmSumPrev	Gets the previous alarm summary entry. Now obsolete.
AlarmSumSet	Sets field information in an alarm summary entry. Now obsolete.
AlmSummarySetFieldValue	Sets the value of the field indicated by the cursor position in the browse session. Now obsolete.
AlarmSumSplit	Duplicates an alarm summary entry. Now obsolete.
AlarmSumType	Retrieves a value that indicates a specified alarm's type. Now obsolete.
QueryFunction	The user-defined query function set in AlarmsetQuery. Now obsolete.

**Miscellaneous Functions**

SetLanguage	Sets the language database from which the local translations of native strings in the project will be drawn, and specifies the character set to be used. Now obsolete. Use the Login(), UserLogin() and LoginForm() to set the preferred language.
-------------	--

**CtAPI Functions in 7.30**

The following section details the changes made to CtAPI functions in Plant SCADA 7.30.

**New Functions**

ctFindNumRecords	Get the number of records for a given browsing session
------------------	--

## Modified Functions

CtFindFirst	Searches for the first object in the specified table, device, trend, tag, or alarm data which satisfies the filter string. Now accepts "Equipment" as an argument.
CtFindFirstEX	Performs the same as ctFindFirst, but with an additional new argument. Now accepts "Equipment" as an argument.
CtGetProperty	Retrieves an object property or meta data for an object. Now accepts "Equipment" as an argument.

## Obsolete Functions

None

## What's New in Citect SCADA 7.20

Citect SCADA 7.20 incorporates the following new features.

Introduced in 7.20:

- Control SCADA Client Connections
- Dynamically Optimized Writes
- Environment Variable Changes
- Graphics Enhancements
- Improved Installation Process
- Improved Citect SCADA Security
- Microprocessor Support in Demo Mode
- New Example Project
- New Web-based Help
- OPC Factory Server (OFS) Driver
- Performance Improvements
- Persisted I/O Memory Mode
- Post Compile Commands
- Improved Client Side Online Changes
- Server Side Online Changes
- Microsoft® Windows 7 Support
- Supportability Enhancements

- Tab Menu Templates
- Tag Extensions

For changes to:

- Citect.ini parameters, see [Citect.ini Parameters in 7.20](#).
- Cicode functions, see [Cicode Functions in 7.20](#).

## Citect.ini Parameters in 7.20

This topic lists the parameters that have been added or changed in version 7.20 of Citect SCADA.

It includes the following:

### New Parameters

The following parameters are new in version 7.20 . For an entire list of the system parameters, refer to the Parameters documentation.

#### Alarm Parameters

[Alarm.ClusterName.ServerName]DisableConnection	Specifies if a client will not connect to a server.
[Alarm.ClusterName.ServerName]Priority	Specifies the client priority for the server connection.
[Alarm]ReloadBackOffTime	Back-off time configured to control the pace of the reload on an alarm server.

#### Client Parameters

[Client]AutoLoginClearPassword	When set to 1 the cache is cleared of any client login credentials.
[Client]DisableDisplay	Sets whether to allow the client process to run in the background without a visible window.
[Client]EvictTimeout	Sets the amount of time a tag reference is cached before it is evicted.
[Client]PartOfTrustedNetwork	Tells a Client process to attempt to authenticate using the stored server password. It is automatically set by the Setup Wizard.
[Client]StalenessPeriod	Number of seconds to use for tag staleness period.
[Client]StalenessPeriodTolerance	Staleness period tolerance

#### CtAPI Parameters

[CtAPI]RoundToFormat	Indicates to the user if values rounded to format.
----------------------	--

#### CtDraw.RSC Parameters

[CtDraw.RSC]AllowEditSuperGeniePage	When set enables the user to choose whether or not to open and edit a Super Genie page.
-------------------------------------	---

**CtEdit Parameters**

[CtEdit]CompileSuccessfulCommand	Indicates to the compiler an optional command, script or batch file to execute after a successful compile.
[CtEdit]CompileUnsuccessfulCommand	Indicates to the compiler an optional command, script or batch file to execute after an unsuccessful compile.
[CtEdit]Starter	Specifies the directory where the starter projects are located.

**Debug Parameters**

[Debug]ArchiveFiles	Archives log files once the size specified by [Debug]MaximumFileSize is reached.
[Debug]CategoryFilter	Allows you to filter logging messages by component category.
[Debug]CategoryFilterMode	Enables logging of categories declared by the [Debug]CategoryFilter value.
[Debug]EnableLogging	Enables or disables the logging mechanism.
[Debug]MaximumFileSize	Sets the maximum size for a log file.
[Debug]Priority	Allows you to filter logging messages according to their priority.
[Debug]SeverityFilter	Allows you to filter logging messages according to their severity.
[Debug]SeverityFilterMode	Enables logging of severities declared by the [Debug]SeverityFilter value.

**General Parameters**

[General]MiniumlUpdateRate	Specifies the time period (sec) at which a DataSource will send tag update value notifications to the subscription clients.
[General]StalenessPeriod	Specifies the time period (sec) after which a tag value is considered to be "stale" if it was not updated during this period.

**IOServer Parameters**

[IOServer]EnableEventQueue	Enables the event queue.
[IOServer]MaxEventsDrop	Sets the number of events that are dropped when too

	many are queued.
[IOServer]MaxEventsQueued	Sets the total number of events that can be queued.
[IOServer]MaxTimeInQueueMs	Sets the total time for which an event can be queued.

**LAN Parameters**

[LAN]AllowRemoteReload	Enables remote reloading of servers from a client.
[LAN]ClientRetryTime	Sets the length of time between connection attempts by a client.
[LAN]EarliestLegacyVersion	Specify the minimum legacy version from which the current version will accept connections.
[LAN]HighWaterMark	The number of messages waiting to be sent on a particular network connection at which the high water mark event will occur.
[LAN]KeepAliveInterval	Sets the length of time between two keep alive transmissions by the client.
[LAN]KeepAliveTime	Sets the length of time between two keep alive transmissions in idle conditions.
[LAN]ListenerRetryTime	Sets the length of time a server waits between attempts to listen for a client connection.
[LAN]LowWaterMark	After the high water mark has been reached on a particular network connection, the low water mark represents the number of messages waiting to be sent at which we will resume normal operations.
[LAN]NoSocketDelay	Switches off the delay on a socket caused by the use of the Nagle algorithm.
[LAN]ReadOnlyLegacyConnections	When set to 1 version 7.10 clients can only communicate in read-only mode. This parameter overrides 'EarliestLegacyVersion' .

**Multi-Monitor Parameters (CSV Include project)**

[MultiMonitor]DisableAutoStart	Disables the new multi-monitor functionality.
--------------------------------	---

**Page Parameters**

[Page]AddDefaultMenu	Determines whether to add the default menu items to the tabbed menu bar.
[Page]BadDitheringColor	Sets the dithering color for graphics elements which are dithered if the value quality is "bad".

[Page]BadDitheringDensity	Sets the dithering density for graphics elements which are dithered if the value quality is "bad".
[Page]BadText	Text Objects can be displayed as #COM type errors, or as the text overlaid with a dithered pattern if the 'display value' expression has "bad" quality.
[Page]BadTextBackgroundColor	Sets the background color for numeric / text graphics objects to indicate "bad" quality.
[Page]EnableQualityToolTip	Set by default it controls the quality tooltip
[Page]ErrorDitheringColor	Sets the dithering color for graphics elements which are dithered if an internal error occurs.
[Page]ErrorDitheringDensity	Sets the dithering density for graphics elements which are dithered if an internal error occurs.
[Page]ErrorTextBackgroundColor	Sets the background color for numeric / text graphics objects to indicate an internal error.
[Page]IgnoreValueQuality	Defines the value quality handling by graphics pages.
[Page]OverrideDitheringColor	Sets the dithering color for graphics elements which are dithered if their values are override ("forced").
[Page]OverrideDitheringDensity	Sets the dithering density for graphics elements which are dithered if an internal error occurs.
[Page]OverrideTextBackgroundColor	Sets the background color for numeric / text graphics objects to indicate that the value presented on the objects is override ("forced").
[Page]ShowBadText	Text Objects can be displayed as #BAD text, or as the text overlaid with a dithered pattern if the "display value" expression has "bad" quality.
[Page]ShowErrorText	Text Objects can be displayed as #COM type errors, or as the text overlaid with a dithered pattern if the 'display value' expression has "uncertain" quality.
[Page]ShowUncertainText	Text Objects can be displayed as #UNC text, or as the text overlaid with a dithered pattern if the "display value" expression has "uncertain" quality.
[Page]Splash	Specify the name of the Splash Screen page.
[Page]SplashTimeout	Time in milliseconds for the Splash Screen to display.
[Page]SplashWinName	Specify the label of the Splash Window for use with the Cicode function WinNumber().

[Page]StartupDelay	Milliseconds between when Splash Screen and Start Screen are displayed.
[Page]StartupHeight	Height of the Start Page on main display monitor.
[Page]StartupMode	Mode of Start Page on main display monitor.
[Page]StartupWidth	Width of the Start Page on main display monitor.
[Page]StartupWinName	Specify the label of the Start Window for use with the Cicode function WinNumber().
[Page]StartupX	X coordinate of Start Page on main display monitor.
[Page]StartupY	Y coordinate of Start Page on main display monitor.
[Page]UncertainDitheringColor	Sets the dithering color for graphics elements which are dithered if the value quality is "uncertain".
[Page]UncertainDitheringDensity	Sets the dithering density for graphics elements which are dithered if the value quality is "uncertain".
[Page]UncertainText	Text Objects can be displayed as #COM type errors, or as the text overlaid with a dithered pattern if the 'display value' expression has "uncertain" quality.
[Page]UncertainTextBackgroundColor	Sets the background color for numeric / text graphics objects to indicate "uncertain" quality.
[Page]WaitForValidData	Specifies whether the animation system will attempt to wait for valid data from subscriptions necessary to draw a graphics page before it is animated.

**Report Parameters**

[Alarm.ClusterName.ServerName]DisableConnection	Specifies if a client will not connect to a server.
[Alarm.ClusterName.ServerName]Priority	Specifies the client priority for the server connection.

**Runtime Manager Parameters**

[RuntimeManager]AllowReload	Enables or disables the reload option in the Runtime Manager menu.
-----------------------------	--

**Security Parameters**

[Security]DisableDEP	Set to turn off DEP protection for the Plant SCADA runtime.
----------------------	---

**Server Parameters**

[Server]AutoLoginMode	Determines the auto login method the server will use when establishing connections to other servers.
-----------------------	--

**Trend Parameters**

[Trend]AcquisitionTimeout	Sets a timeout to stop a trend server infinitely acquiring a valid data sample from an I/O device.
[Trend.ClusterName.ServerName]DisableConnection	Specifies if a client should not connect to a server.
[Trend.ClusterName.ServerName]Priority	Specifies the client priority for the server connection.
[Trend]ReloadBackOffTime	Back-off time configured to control the pace of the reload on an Trend server.

**Modified Parameters****CtEdit Parameters**

[CtEdit]Copy	Supports runtime changes, it enables you to switch the SCADA node to use a new runtime configuration by pointing to a new location.
--------------	---

**Re-instanted Parameters****IOServer Parameters**

[IOServer]BlockWrites	Determines whether Plant SCADA will try to block optimize writes to I/O devices.
-----------------------	--

**Obsolete Parameters****AnmCursor Parameters**

[AnmCursor]Colour	Replaced with [AnmCursor]Color. Sets the color of the cursor.
-------------------	---

**General Parameters**

[General]TagAssMode	Validates the tag name in the Association Function. Refer to [General]TagDB instead.
---------------------	--

**LAN Parameters**

[LAN]AllowLegacyConnections	Set to allow previous versions of client to connect to the server.  Replaced with [LAN]EarliestLegacyVersion and the new trusted network authentication between SCADA servers. The Setup Wizard now allows a server password to be set on each server on your network.
[LAN]ServerLoginEnabled	Set to disable default server login.

	Replaced with [LAN]EarliestLegacyVersion and the new trusted network authentication between SCADA servers. The Setup Wizard now allows a server password to be set on each server on your network.
--	--

**Page Parameters**

[Page]BackgroundColour	Replaced with [Page]BackgroundColor. Specifies the color used to fill in the background when a page is smaller than the minimum width of a window.
[Page]ComBreak	Determines whether an error status is displayed on the screen if a communication error occurs.  Replaced with new page quality settings such as [Page]IgnoreValueQuality, [Page]BadText,[Page]BadDitheringDenisty.
[Page]ComBreakText	Determines the display of text objects if a communication error occurs that affects the text.  Replaced with new page quality settings such as [Page]IgnoreValueQuality, [Page]BadText,[Page]BadDitheringDenisty.
[Page]DynamicComBreakColour	Replaced with [Page]DynamicComBreakColor. Sets the color of the ComBreak dithering.
[Page]DynamicComBreakDensity	Sets the density of the ComBreak.  Replaced with new page quality settings such as [Page]IgnoreValueQuality, [Page]BadText,[Page]BadDitheringDenisty.

**Time Parameters**

[Time]Deadband	The deadband time checked by the Time Server before it adjusts the time on the client(s).
[Time]Disable	Enables/disables the processing of time messages from the Time Server.
[Time]Name	Enables the time synchronization functionality.
[Time]PollTime	The period that the Time Server uses to synchronize other Plant SCADA computers on the network.
[Time]RTsync	Determines whether the Time Server will synchronize with the hardware clock.
[Time]Server	Determines whether this computer is a Time Server.

**Trend Parameters**

[Trend]CursorColour	Replaced with [Trend]CursorColor. Allows the cursor color to be specified.
---------------------	--

## Cicode Functions in 7.20

Some Cicode functions have been introduced, modified, deprecated or removed. The following sections detail the changes made to these functions:

### New Functions

#### Alarm Functions

AlarmCatGetFormat	Returns the display format string of the specified alarm category.
AlarmDspClusterAdd	Adds a cluster to a client's alarm list.
AlarmDspClusterInUse	Determines if a cluster is included in a client's alarm list.
AlarmDspClusterRemove	Removes a cluster from a client's alarm list.

#### Display Functions

DspAnGetMetadata	Retrieves the field value of the specified metadata entry.
DspAnGetMetadataAt	Retrieves metadata information at the specified index.
DspAnSetMetadata	Non-blocking function, that sets the value of the specified metadata entry.
DspAnSetMetadataAt	Sets the value of a metadata entry.
DspPopupConfigMenu	Displays the contents of a menu node as a pop-up (context) menu, and run the command associated with the selected menu item.

#### Format Functions

FmtGetFieldCount	Retrieves the number of fields in a format object.
FmtGetFieldName	Retrieves the name of a particular field in a format object.
FmtGetFieldWidth	Retrieves the width of a particular field in a format object.

#### Menu Functions

MenuGetChild	Returns the handle to the child node with the specified name.
MenuGetFirstChild	Returns the handle to the first child of a menu node.
MenuGetGenericNode	Returns the root node of the default menu tree.
MenuGetNextChild	Returns the next node that shares the same parent.
MenuGetPageNode	Returns the Base menu node of a specific page
MenuGetParent	Returns the parent node of the menu item.
MenuGetPrevChild	Returns the previous node that shares the same parent.
MenuGetWindowNode	Returns the handle of the root menu node for a given window.
MenuNodeAddChild	Dynamically adds a new item to the menu at runtime.
MenuNodeGetProperty	Return the item value of the specified menu node.
MenuNodeHasCommand	Checks whether the menu node has a valid Cicode command associated with it.
MenuNodeIsDisabled	Checks whether the menu node is disabled by evaluating its DisabledWhen Cicode expression.
MenuNodeIsHidden	Checks whether the menu node is hidden by evaluating its HiddenWhen Cicode expression.
MenuNodeRemove	Remove the menu node from the menu tree.
MenuNodeRunCommand	Run the associated command for a menu node.
MenuNodeSetDisabledWhen	Set the DisabledWhen expression for a newly added node.
MenuNodeSetHiddenWhen	Set the HiddenWhen expression for a newly added node.
MenuNodeSetProperty	Set the item value of the specified menu node.
MenuReload	Reload base Menu Configuration from the compiled database.

**Miscellaneous Functions**

GetLogging	Gets the current value for one or more logging parameters.
SetLogging	Adjusts logging parameters while online.
ProductInfo	Returns information about the Plant SCADA product.
ProjectInfo	Returns information about a particular project, which is identified by a project enumerated number.

**Page Functions**

PageBack	Displays the previously displayed page in the Window.
PageForward	PageForward() restores the previously displayed page in the window following a PageBack command.
PageHistoryDspMenu	Displays a pop-up menu which lists the page history of current window.
PageHistoryEmpty	Returns whether page history of the current window is empty.
PageHome	Displays the predefined home page in the window.
PagePeekCurrent	Return the index in the page stack for the current page.
PageProcessAnalyst	Displays a Process Analyst page (in the same window) preloaded with the pre-defined Process Analyst View (PAV) file.
PageProcessAnalystPens	Displays a Process Analyst page (in the same window) preloaded with the pre-defined Process Analyst View (PAV) file and specified trend or variable tags.
PageRecall	Displays the page at a specified depth in the stack of previously displayed pages.
PageTask	Used for running preliminary Cicode before displaying a page in a window.
PageTransformCoords	Converts Page coordinates to absolute screen coordinates.

**Process Analyst Functions**

ProcessAnalystLoadFile	Loads the specified PAV file to a Process Analyst object, which is identified by parameter ObjName.
ProcessAnalystPopup	Displays a Process Analyst page (in the same window) preloaded with the pre-defined Process Analyst View

	(PAV) file and specified trend or variable tags.
ProcessAnalystSelect	Allows a set of pens to be selected before displaying the PA page.
ProcessAnalystSetPen	Allows a new pen to be added to a PA display.
ProcessAnalystWin	Displays a Process Analyst page (in a new window) preloaded with the pre-defined Process Analyst View (PAV) file.

**Quality Functions**

QualityCreate	Creates a quality value based on the quality fields provided.
QualityGetPart	Extracts a requested part of the Quality value from the QUALITY variable.
QualityIsBad	Returns a value indicating whether the quality is bad.
QualityIsGood	Returns a value indicating whether the quality is good.
QualityIsUncertain	Returns a value indicating whether the quality is uncertain.
QualitySetPart	Sets a Quality part's value to the QUALITY variable.
QualityToStr	Returns a textual representation of the Plant SCADA quality.
QualityIsOverride	Returns a value indicating whether the tag is in Override Mode.
QualityIsControlInhibit	Returns a value indicating whether the tag is in Control inhibit mode.
VariableQuality	Extracts the quality from a given variable.

**Server Functions**

ServerBrowseClose	This function terminates an active data browse session and cleans up resources associated with the session.
ServerBrowseFirst	This function places the data browse cursor at the first record.
ServerBrowseGetField	This function retrieves the value of the specified field from the record the data browse cursor is currently referencing.
ServerBrowseNext	This function moves the data browse cursor forward

	one record.
ServerBrowseNumRecords	This function returns the number of records that match the filter criteria.
ServerBrowseOpen	This function initiates a new browse session and returns a handle to the new session that can be used in subsequent data browse function calls.
ServerBrowsePrev	This function moves the data browse cursor back one record.
ServerGetProperty	This function returns information about a specified server and can be called from any client.
ServerReload	This function reloads the server specified by cluster and server name.
ServerIsOnline	This function checks if the given server can be contacted by the client for giving the online/offline status of the server.

**String Functions**

StrCalcWidth	Retrieves the pixel width of a string using a particular font.
StrTruncFont	Returns the truncated string using a particular font (specified by name) or the specified number of characters.
StrTruncFontHnd	Returns the truncated string using a particular font (specified by font number) or the specified number of characters.

**Super Genie Functions**

AssMetadata	Performs Super Genie associations using the "Name" and "Value" fields.
AssMetadataPage	Uses the metadata information from the current animation point for the page associations for a new Super Genie page, and displays the new Super Genie in the current page.
AssMetadataPopup	Uses the metadata information from the current animation point for the associations for a new Super Genie page, and displays the new Super Genie in a new pop up window.
AssMetadataWin	Uses the metadata information from the current

	animation point for the associations for a new Super Genie page, and displays the new Super Genie in a new window.
--	--

**Tag Functions**

SubscriptionGetInfo	Reads the specified text information about a subscribed tag.
SubscriptionGetQuality	Reads quality of a subscribed tag.
SubscriptionGetTag	Reads a value, quality and timestamps of a subscribed tag.
SubscriptionGetTimestamp	Reads the specified timestamp of a subscribed tag.
SubscriptionGetValue	Reads a value of a subscribed tag.
TagSetOverrideBad	Sets a quality Override element for a specified tag to Bad Non Specific.
TagSetOverrideGood	Sets a quality Override element for a specified tag to Good Non Specific.
TagSetOverrideUncertain	Sets a quality Override element for a specified tag to Uncertain Non Specific.
TagSetOverrideQuality	Sets a quality of Override element for a specified tag.

**Task Functions**

TaskCall	Calls a Cicode function by specifying the function name and providing an arguments string.
----------	--

**Timestamp Functions**

TimestampToStr	Converts a TIMESTAMP variable into a string.
TimestampDifference	Returns a difference between two TIMESTAMP variables as a number of milliseconds.
TimestampCreate	Returns a timestamp variable created from the parts.
TimestampFormat	Format a TIMESTAMP variable into a string.
TimestampGetPart	Returns one part (year, month, day, etc) of the timestamp variable.
TimestampToInt	Converts a TIMETSTAMP variable into a time INTEGER which is represented as a number of seconds since 01/01/1970.
TimeIntTo Timestamp	Converts a time INTEGER which is represented as a

	number of seconds since 01/01/1970 to a TIMETSTAMP
TimestampCurrent	Returns the current system date and time as a TIMESTAMP variable.
TimestampAdd	Adds time (in milliseconds) to a TIMESTAMP variable.
TimestampSub	Subtracts time (in milliseconds) from a TIMESTAMP variable.
VariableTimestamp	Extract the TIMESTAMP from a given variable.

**Window Functions**

MultiMonitorStart	Displays a Plant SCADA window on each of the configured monitors when a display client starts up.
WinSetName	Associates a name with a particular window by its window number.
WndMonitorInfo	Returns information about a particular monitor.

**Modified Functions****Accumulator Functions**

AccumBrowseOpen	Opens an accumulator browse session.
-----------------	--------------------------------------

**Alarm Functions**

AlarmDsp	Displays alarms.
AlarmDspLast	Displays the latest, unacknowledged alarms.
AlmSummaryOpen	Opens an alarm summary browse session.
AlmTagsOpen	Opens an alarm tags browse session.

**Display Functions**

DspStr	Displays a string at an AN.
DspText	Displays text at an AN.

**Format Functions**

FmtOpen	Creates a format template.
---------	----------------------------

**Miscellaneous Functions**

Shutdown	Ends Plant SCADA operation.
----------	-----------------------------

**Page Functions**

PageGetInt	Gets a local page-based integer.
PageGetStr	Gets a local page-based string.
PageInfo	Gets information about the current page.
PagePeekLast	Gets any page on the PageLast stack.
PageSetInt	Stores a local page-based integer.
PagesetStr	Stores a local page-based string.

**Security Functions**

Login	Logs an operator into the Plant SCADA system. Not available when logged in as Windows user.
-------	---

**Super Genie Functions**

The following functions were updated to accept string identifiers for substitution parameters.

Ass	Associates a variable tag with a Super Genie.
AssGetProperty	Retrieves association information about the current Super Genie from the datasource.
AssGetScale	Gets scale information about the associations of the current Super Genie from the datasource (that is scale information about a variable tag that has been substituted into the Super Genie)
AssInfo	Gets association information about the current Super Genie (that is information about a variable tag that has been substituted into the Super Genie).
AssInfoEx	Retrieves association information about the current Super Genie (that is information about a variable tag that has been substituted into the Super Genie).
AssScaleStr	Gets scale information about the associations of the current Super Genie (that is scale information about a variable tag that has been substituted into the Super Genie).

**Tag Functions**

SubscriptionGetAttribute	Reads an attribute value of a tag subscription.
TagRead	Reads the value of a particular tag element.
TagWrite	Writes a tag element value for the tag elements which have read/write access.

TagSubscribe	Subscribes to a particular tag element.
--------------	---

### Window Functions

WinNumber	Gets the window number of the active Plant SCADA window.
WndInfo	Gets the Windows system metrics information.

### Reinstated Functions

Following functions have been reinstated for 7.20.

#### Time and Date Functions

TimeSet	Sets the new system time. Requires UAC to be disabled in order for the time to be set.
---------	--

## CtAPI Functions in 7.20

The following section details the changes made to CtAPI functions in Plant SCADA 7.20.

### New Functions

ctListItem	Gets the tag element item data.
ctTagReadEx	Performs the same as ctTagRead, but with an additional new argument

### Modified Functions

ctTagRead	Reads the current value from the given I/O device variable tag element value.
ctTagWrite	Writes the given value to the I/O device variable tag elements which have read/write access.
ctTagWriteEx	Asynchronously writes the given value to the I/O device variable tag element value for the tag elements which have read/write access.
ctListAdd	Adds a tag element value to the list.
ctListAddEx	Adds a tag element value to the list.

## Obsolete Functions

None

## What's New in Citect SCADA 7.10

Citect SCADA 7.10 incorporates the following new features.

Introduced in 7.10:

- Windows® Integrated Security
- Citect SCADA Security Enhancements
- Multi-Signature Support
- Edit DBF Files in Microsoft® Excel
- Enhanced Driver Installation
- New Font Selection for Graphics Button
- Microsoft® Windows Vista™ Support
- New Location for Configuration and Project Files
- New Alarm Field Enhancements
- New Time Synchronization Service
- New Equipment Database Functions and Forms

For changes to:

- Citect.ini parameters, see [Citect.ini Parameters in 7.10](#).
- Cicode functions, see [Cicode Functions in 7.10](#).

## Citect.ini Parameters in 7.10

This topic lists the parameters that have been added or changed in version 7.10 of Citect SCADA.

### New Parameters

The following parameters are new in version 7.10. For an entire list of the system parameters, refer to the Parameters documentation.

#### Alarm Parameters:

[Alarm]ArgyleTagValueTimeout	Defines the length of time that the alarm server will wait for argyle tag values to become available (without error) before starting to scan for argyle alarms.
------------------------------	---

#### Code Parameters:

[Code]HaltOnInvalidTagData	When enabled will cause the Cicode to halt when any tag read returns invalid data.
----------------------------	--

**Client Parameters:**

[Client]AutoLoginPage	Set to enable auto login. Users can select one of seven modes.
-----------------------	--

**CtApi Parameters:**

[CtAPI]AllowLegacyConnections	When enabled current version of CTAPI server can accept connections from previous versions of CTAPI client.
[CtAPI]AllowLegacyServices	When enabled the Citect Web Service and the Citect OLEDB Provider can connect to the CTAPI server.

**DDE Parameters:**

[DDE]AllowCicode	Allows Cicode to be run on the Citect server via the DDE Execute command.
[DDE]AllowWrites	Allows tag writes to the Citect server via the DDE Poke command.

**Kernel Parameters:**

[Kernel]ErrorBuffers	The total number of error buffers available for logging to the syslog.dat file.
----------------------	---

**LAN Parameters:**

[LAN]AllowLegacyConnections	Set to allow previous versions of client to connect to the server.
[LAN]AnonymousLoginName	The name of the default identifier to allow 'view-only' data access for a client process to the SCADA server(s).
[LAN]SecureLogin	When set to 0 security measures are disabled and the system acts as it did in versions prior to 7.10.
[LAN]ServerLoginEnabled	Set to disable default server login.
[LAN]ServerLoginName	The name of the default identifier to allow data access for a server process to another SCADA server process(es).

**ODBC Parameters:**

[ODBC]Server	Set to enable ODBC connections.
--------------	---------------------------------

**Page Parameters:**

[Page]AllowHScroll	Defines the default behavior for horizontal scrolling.
[Page]AllowHScrollBar	Defines the default behavior when displaying horizontal scroll bars.
[Page]AllowVScroll	Defines the default behavior for vertical scrolling.
[Page]AllowVScrollBar	Defines the default behavior when displaying vertical scroll bars.

## Obsolete Parameters

### Win Parameters:

[Win]CtrlEsc	Determines whether the Windows key command [Ctrl]+[Esc] can be used in the runtime system (to display the start menu).
--------------	--

### COM Parameters:

[Com]StartTimeout	Determines the period to wait for I/O Devices to come online before displaying any data.
-------------------	--

## Cicode Functions in 7.10

Some Cicode functions have been introduced, modified, deprecated or removed. The following sections detail the changes made to these functions:

## New Functions

### Security Functions

FormSecurePassword	Adds both a password prompt and edit field to the current form.
MultiSignatureForm	Displays a form that allows up to 4 users to have their credentials verified in order to approve an operation.
MultiSignatureTagWrite	Displays a form that allows up to 4 users to have their credentials verified in order to approve a write of a specific value to a specific tag.
UserLogin	Logs an operator into the Citect SCADA system using a secure password string.
UserVerify	Uses the authentication functionality in the user login system

VerifyPrivilegeForm	Displays a form that allows a single user to enter their credentials.
VerifyPrivilegeTagWrite	Displays a form that allows any single user to enter their credentials in order to approve a write of a specific value to a specific tag.

**Miscellaneous Functions**

KernelQueueLength	Obtains the number of rows in a queue.
KernelTableInfo	Provides a consistent method of accessing items within Kernel Table.
KernelTableItemCount	Obtains the number of rows in a Kernel Table
ProcessRestart	Restarts the current process in which Cicode is running.
ServerRestart	Restart any alarm, report, trend or I/O server from any Cicode node in system, without affecting other server processes running on same machine.

**Tag Functions**

TagRDBReload	Works in conjunction with the TagInfo function. Reloads the variable tag database so when TagInfo is called it picks up online changes to the tag database.
--------------	---

**Windows Functions**

WinStyle	Switches on and off scrolling and scroll bar features for existing windows.
----------	---

**Modified Functions**

None

**Obsolete Functions****Window Functions**

WndGetProfile	Gets the value of a WIN.INI parameter. If called it will return 0. No longer available due to Vista support.
WndPutProfile	Updates a parameter in WIN.INI. If called it will return 0. No longer available due to Vista support.

**Time and Date Functions**

TimeSet	Sets the new system time. No longer available due to
---------	--

	Vista support.
--	----------------

## What's New in Citect SCADA 7.0

Citect SCADA 7.0 incorporates the following new features.

Introduced in 7.0:

- The Migration Tool
- Clustering
- Local Variables
- Publish Alarm Property
- Memory Mode
- Client-side Online Changes
- Dual Network Support
- Project-Based Network Configuration

For changes to:

- Citect.ini parameters, see [Citect.ini Parameters in version 7.0](#).
- Cicode functions, see [Cicode Functions in 7.0](#).

### Citect.ini Parameters in version 7.0

The following sections detail the changes made to Citect.ini parameters in Citect SCADA version 7.0:

- New Parameters
- Obsolete Parameters

#### New Parameters

The following parameters are new in version 7.0. For a complete list of the system parameters, refer to the Parameters help.

##### Alarm Parameters:

[Alarm.ClusterName.ServerName]Clusters	Sets which clusters this Alarm Server process connects to at startup
[Alarm.ClusterName.ServerName]CPU	Sets the CPU that the Alarm Server process is assigned to
[Alarm.ClusterName.ServerName]Events	The list of events that this Alarm Server process enables
[Alarm.ClusterName.ServerName]ShutdownCode	Determines the Cicode function to run when Alarm

	Server process shuts down
[Alarm.ClusterName.ServerName]StartupCode	Determines the Cicode function to run when Alarm Server process starts up

**Note:** The default alarm property write behavior was to write the new value to DBF/RDB. This has changed in version 7.0 onwards. Refer to the parameter [Alarm]UseConfigLimits.

#### Backup Parameters:

[Backup]SaveiniFiles	Determines whether the "Save ini files" check box is checked by default during Backup.
----------------------	--

#### Client Parameters:

[Client]Clusters	Selects which clusters the client is connected to at startup.
[Client]ComputerRole	Specifies the role of the computer
[Client]Events	Sets the events to be enabled on the client.
[Client]ForceClient	Starts only the client process, and connects to a configured Citect servers using a network connection.
[Client]FullLicense	Specifies that a Control Client will use a full server license
[Client]ShutdownCode	Determines the Cicode function to run when DisplayClient component shuts down.
[Client]StartupCode	Determines the Cicode function to run when DisplayClient component starts up.
[Client]WaitForConnectAtStartup	Specifies that connection to a server will wait until it can establish a connection to the server processes before starting up.

#### CtCicode Parameters:

[CtCicode]FastFormat	Controls whether fast formatting is used in the Cicode Editor
----------------------	---

#### CtEdit Parameters:

[CtEdit]Config	The directory where the Citect SCADA configuration files such as citect.ini are located.
[CtEdit]Logs	The directory where the Citect SCADA log files are located.

#### Dial Parameters:

[Dial]MissedScheduleTolerance	Specifies how many consecutive scheduled dial
-------------------------------	---

	attempts can be missed before the cache becomes stale
--	---

**Driver Parameters:**

[<DriverName>]OverriderOSProtection	Determines whether to override the protection mechanism built-in to the I/O Server for drivers that may not be compatible with Windows Vista.
-------------------------------------	---

**General Parameters:**

[General]Multiplexor	Determines whether Citect SCADA runs as a multi-process or single-process application.
----------------------	--

**IOServer Parameters:**

[IOServer.ClusterName.ServerName]Clusters	Sets the clusters that the I/O Server process will connect to at startup
[IOServer.ClusterName.ServerName]CPU	Sets the CPU that the I/O Server process is assigned to
[IOServer.ClusterName.ServerName]Events	The list of events that this I/O Server process enables
[IOServer.ClusterName.ServerName]ShutdownCode	Determines the Cicode function to run when I/O Server process shuts down
[IOServer.ClusterName.ServerName]StartupCode	Determines the Cicode function to run when I/O Server process starts up

**Report Parameters:**

[Report.ClusterName.ServerName]Clusters	Sets the clusters this Reports Server process will connect to at startup
[Report.ClusterName.ServerName]CPU	Sets the CPU that this Reports Server process is assigned to
[Report.ClusterName.ServerName]Events	The list of events that this Reports Server process enables
[Report.ClusterName.ServerName]ShutdownCode	Determines the Cicode function to run when this Reports Server process shuts down
[Report.ClusterName.ServerName]StartupCode	Determines the Cicode function to run when this Reports Server process starts up

**Trend Parameters:**

[Trend.ClusterName.ServerName]Clusters	Sets the clusters this Trends Server process will connect to at startup
[Trend.ClusterName.ServerName]CPU	Sets the CPU that the Trends Server process is assigned to

[Trend.ClusterName.ServerName]Events	The list of events that this Trends Server process enables
[Trend.ClusterName.ServerName]ShutdownCode	Determines the Cicode function to run when Trends Server process shuts down
[Trend.ClusterName.ServerName]StartupCode	Determines the Cicode function to run when Trends Server process starts up

## Obsolete Parameters

The following parameters are no longer supported in version 7.0:

### Alarm Parameters:

[Alarm]CPU	Sets the CPU that the Alarm Server component is assigned to. Replaced with a cluster specific CPU parameter in the following format:[ServerType.ClusterName.ServerName]CPU.
[Alarm]Name	The name of the default Alarm Server. To retrieve the server name in your Cicode, refer to the ServiceGetList Cicode function.
[Alarm]Primary	Determines if this Alarm Server is the Primary Alarm Server. Information is now configured within the project via the server form.
[Alarm]Process	Sets the Citect SCADA process the Alarm Server component is assigned to. No longer required as each process is separate when running multi-process.
[Alarm]Server	Determines whether this computer is an Alarm Server. To retrieve the server enabled status in your Cicode, refer to the ProcessIsServer Cicode function.

### Client Parameters:

[Client]Display	Sets the Citect SCADA computer as a Control Client. To retrieve the client status in your Cicode, refer to the ProcessIsClient Cicode function or the parameter [Client]ComputerRole. The computer role parameter can be set to the following: 0 = Server & Control client, 1 = Control client, 2 =View only client.
[Client]Manager	Sets the Citect SCADA computer as a View-only Client.

	To retrieve the client status in your Cicode, refer to the ProcessIsClient Cicode function or the parameter [Client]ComputerRole. The computer role parameter can be set to the following: 0 = Server & Control client, 1 = Control client, 2 =View only client.
[Client]Primary	The name of the primary Citect SCADA server
[Client]Process	Sets the Citect SCADA process the Control Client component is assigned to.  No longer required as each process is separate when running multi-process.
[Client]Standby	The name of the standby Citect SCADA server
[Client]Shutdown	Determines the Cicode function to run when DisplayClient component shuts down.  Replaced with [Client]ShutdownCode.
[Client]Startup	Determines the Cicode function to run when DisplayClient component starts up.  Replaced with [Client]StartupCode.

**Code Parameters:**

[Code]AlarmShutdown	Determines the Cicode function to run when Alarm Server component shuts down.  Replaced with a cluster specific code startup and shutdown parameters in the following format:[ServerType.ClusterName.ServerName]StartupCode and [ServerType.ClusterName.ServerName]ShutdownCode .
[Code]AlarmStartup	Determines the Cicode function to run when Alarm Server component starts up.  Replaced with a cluster specific code startup and shutdown parameters in the following format:[ServerType.ClusterName.ServerName]StartupCode and [ServerType.ClusterName.ServerName]ShutdownCode .
[Code]AutoReRead	Controls whether the ReRead() function is automatically called
[Code]IOServerShutdown	Determines the Cicode function to run when I/O Server component shuts down.  Replaced with a cluster specific code startup and

	shutdown parameters in the following format:[ServerType.ClusterName.ServerName]StartupCode and [ServerType.ClusterName.ServerName]ShutdownCode .
[Code]IOServerStartup	Determines the Cicode function to run when I/O Server component starts up. Replaced with a cluster specific code startup and shutdown parameters in the following format:[ServerType.ClusterName.ServerName]StartupCode and [ServerType.ClusterName.ServerName]ShutdownCode .
[Code]Process	Sets the process a code component is assigned to
[Code]ReportShutdown	Determines the Cicode function to run when Reports Server component shuts down. Replaced with a cluster specific code startup and shutdown parameters in the following format:[ServerType.ClusterName.ServerName]StartupCode and [ServerType.ClusterName.ServerName]ShutdownCode .
[Code]ReportStartup	Determines the Cicode function to run when Reports Server component starts up. Replaced with a cluster specific code startup and shutdown parameters in the following format:[ServerType.ClusterName.ServerName]StartupCode and [ServerType.ClusterName.ServerName]ShutdownCode .
[Code]Shutdown	Determines the Cicode function to run when Control Client component shuts down
[Code]Startup	Determines the Cicode function to run when Control Client component starts up
[Code]TrendShutdown	Determines the Cicode function to run when Trends Server component shuts down. Replaced with a cluster specific code startup and shutdown parameters in the following format:[ServerType.ClusterName.ServerName]StartupCode and [ServerType.ClusterName.ServerName]ShutdownCode .

	.
[Code]TrendStartup	<p>Determines the Cicode function to run when Trends Server component starts up.</p> <p>Replaced with a cluster specific code startup and shutdown parameters in the following format:[ServerType.ClusterName.ServerName]StartupCode and [ServerType.ClusterName.ServerName]ShutdownCode</p> <p>.</p>

**DNS Parameters:**

[DNS]<Server name>	<p>Determines the IP address (or fully qualified host name) of the primary I/O Server.</p> <p>Functionality replaced by the server network address configuration form and the server redirection parameters with the following format:[AddressForwarding].=</p>
--------------------	---

**Event Parameters:**

[Event]Alarm	<p>The classes of events to be enabled by the Alarm Server.</p> <p>Replaced with a cluster specific event parameter in the following format:[ServerType.ClusterName.ServerName]Events.</p>
[Event]IOMonitor	<p>The classes of events to be enabled by the I/O Server.</p> <p>Replaced with a cluster specific event parameter in the following format:[ServerType.ClusterName.ServerName]Events.</p>
[Event]Name	<p>The classes of events assigned to the Name entry to be enabled.</p> <p>Has been replaced with the parameter [Client]Events.</p>
[Event]Report	<p>The classes of events to be enabled by the Reports Server.</p> <p>Replaced with a cluster specific event parameter in the following format:[ServerType.ClusterName.ServerName]Events.</p>
[Event]Trend	<p>The classes of events to be enabled by the Trends Server.</p> <p>Replaced with a cluster specific event parameter in the following format:[ServerType.ClusterName.ServerName]Events.</p>

**General Parameters:**

[General]BadOptimise	Determines whether certain strings are replaced with labels on compile
[General]CitectRunningCheck	Checks if a project is currently running on the local machine when a compile is triggered

**IOServer Parameters:**

[IOServer]BlockWrites	Determines whether Citect SCADA will try to block optimize writes to I/O Devices. The IOserver will not block writes
[IOServer]CPU	Sets the CPU that the I/O Server component is assigned to. Replaced with a cluster specific CPU parameter in the following format:[ServerType.ClusterName.ServerName]CPU.
[IOServer]Name	The name of the default I/O Server. To retrieve the server name in your Cicode, refer to the ServiceGetList Cicode function.
[IOServer]Process	Sets the Citect SCADA process the I/O Server component is assigned to. No longer required as each process is separate when running multi-process.
[IOServer]SaveBackup	This parameter has been superseded by SaveNetwork
[IOServer]Server	Determines whether this computer is an I/O Server . To retrieve the server enabled status in your Cicode, refer to the ProcessIsServer Cicode function.

**LAN Parameters:**

[LAN]Bridge	Determines the bridge level
[LAN]CancelOnClose	For users with Novell NetBIOS emulator issues
[LAN]Disable	Enables/disables Citect SCADA from the LAN
[LAN]GroupName	Determines whether Citect SCADA uses the group name 'CITECT STATION50' or the computer name specified by the [Lan]Node parameter.
[LAN]KillPiggyBackAck	Controls whether Citect SCADA will try to optimize network protocols which support piggyback ACK
[LAN]LanA	Defines the protocol stack that Citect SCADA uses for

	NetBIOS communication
[LAN]NetBIOS	Enables/disables NetBIOS
[LAN]NetTrace	Determines whether the NetBIOS window is enabled on startup
[LAN]NetTraceBuff	Sets the number of trace buffers.
[LAN]NetTraceErr	Enables the error mode of the NetBIOS window
[LAN]NetTraceLog	Enables the logging mode of the NetBIOS window
[LAN]Poll	Puts Citect SCADA LAN communications into polled mode
[LAN]RemoteTimeOut	The timeout period for remote I/O Device write requests from a Control Client to the I/O Server
[LAN]Retry	The number of times to retry establishing communications after a timeout - before an alert message is generated
[LAN]SendTimeOut	The timeout to send a network packet across the network
[LAN]SesRecBuf	The number of receive NetBIOS Control Blocks (NCBs) that Citect SCADA uses for every session
[LAN]SesSendBuf	The number of send NetBIOS control blocks that Citect SCADA uses for every session
[LAN]TimeOut	The timeout to send a network packet across the network

**Proxi Parameters:**

[Proxi]<I/O Server name>	Defines a list of proxy server associations
--------------------------	---

**Report Parameters:**

[Report]CPU	Sets the CPU that the Report Server component is assigned to.  Replaced with a cluster specific CPU parameter in the following format:[ServerType.ClusterName.ServerName]CPU.
[Report]Name	The name of the default Report Server.  To retrieve the server name in your Cicode, refer to the ServiceGetList Cicode function.

[Report]Primary	Determines if this Reports Server is the Primary Reports Server.  Information is now configured within the project via the server form.
[Report]Process	Sets the Citect SCADA process the Reports Server component is assigned to.  No longer required as each process is separate when running multi-process.
[Report]Server	Determines whether this computer is a Reports Server .  To retrieve the server enabled status in your Cicode, refer to the ProcessIsServer Cicode function.

**Server Parameters:**

[Server]Name	The name of the Citect SCADA server .  To retrieve the server enabled status in your Cicode, now refer to the ProcessIsServer Cicode function.
--------------	--

**Trend Parameters:**

[Trend]BlockByIODevice	Verifies that I/O problems causing gaps for a trend tag do not cause gaps for every trend tag.  Trend blocking is no longer required due to the new subscription architecture.
[Trend]CPU	Sets the CPU that the Trends Server component is assigned to.  Replaced with a cluster specific CPU parameter in the following format:[ServerType.ClusterName.ServerName]CPU.
[Trend]Name	The name of the default Trend Server.  To retrieve the server name in your Cicode, refer to the ServiceGetList Cicode function.
[Trend]Primary	Determines if this Trend Server is the Primary Trend Server.  Information is now configured within the project via the server form.
[Trend]Process	Sets the Citect SCADA process the Trends Server component is assigned to.  No longer required as each process is separate when running multi-process.

[Trend]Redundancy	Enables/disables trend redundancy action.  Trend blocking is no longer required due to the new subscription architecture.
[Trend]Server	Determines whether this computer is a Trends Server.  To retrieve the server enabled status in your Cicode, refer to the ProcessIsServer Cicode function.
[Trend]StaggerRequestSubgroups	Reduces network traffic by spacing out trend sample requests.  Trend blocking is no longer required due to the new subscription architecture.

## Cicode Functions in 7.0

Some Cicode functions have been introduced, modified, deprecated or removed. The following sections detail the changes made to these functions:

### New Functions

#### Miscellaneous Functions

AccControl	Controls accumulators for example motor run hours.
AccumBrowseClose	Closes an accumulator browse session.
AccumBrowseFirst	Gets the oldest accumulator entry.
AccumBrowseGetField	Gets the field indicated by the cursor position in the browse session.
AccumBrowseNext	Gets the next accumulator entry in the browse session.
AccumBrowseNumRecords	Returns the number of records in the current browse session.
AccumBrowseOpen	Opens an accumulator browse session.
AccumBrowsePrev	Gets the previous accumulator entry in the browse session.
ProcessIsClient	Determines if the currently executing process contains a Client component
ProcessIsServer	Determines if the currently executing process contains a particular server component.

ServiceGetList	Gets information about services running in the component calling this function.
----------------	---

**Alarm Functions:**

AlmDspLast	Displays the latest unacknowledged alarms.
AlmSummaryAck	Acknowledges the alarm at the current cursor position in an active data browse session.
AlmSummaryClear	Clears the alarm at the current cursor position in an active data browse session.
AlmSummaryClose	Closes an alarm summary browse session.
AlmSummaryCommit	Commits the alarm summary record to the alarm summary device.
AlmSummaryDelete	Deletes alarm summary entries from the browse session.
AlmSummaryDeleteAll	Deletes alarm summary entries from the browse session.
AlmSummaryDisable	Disables the alarm at the current cursor position in an active data browse session.
AlmSummaryEnable	Enables the alarm at the current cursor position in an active data browse session.
AlmSummaryFirst	Gets the oldest alarm summary entry.
AlmSummaryGetField	Gets the field indicated by the cursor position in the browse session.
AlmSummaryLast	Places the data browse cursor at the latest summary record from the last cluster of the available browsing cluster list.
AlmSummaryNext	Gets the next alarm summary entry in the browse session.
AlmSummaryOpen	Opens an alarm summary browse session.
AlmSummaryPrev	Gets the previous alarm summary entry in the browse session.
AlmSummarySetFieldValue	Sets the value of the field indicated by the cursor position in the browse session.
AlmTagsAck	Acknowledges the alarm tag at the current cursor position in an active data browse session.

AlmTagsClear	Clears the alarm tag at the current cursor position in an active data browse session.
AlmTagsDisable	Disables the alarm tag at the current cursor position in an active data browse session.
AlmTagsEnable	Enables the alarm tag at the current cursor position in an active data browse session.
AlmTagsFirst	Gets the oldest alarm tags entry.
AlmTagsGetField	Gets the field indicated by the cursor position in the browse session.
AlmTagsNext	Gets the next alarm tags entry in the browse session.
AlmTagsNumRecords	Returns the number of records in the current browse session.
AlmTagsOpen	Opens an alarm tags browse session.
AlmTagsPrev	Gets the previous alarm tags entry in the browse session.

**Super Genie Functions**

AssGetProperty	Gets association information about the current Super Genie from the datasource
AssGetScale	Gets scale information about the associations of the current Super Genie from the datasource
AssInfoEx	Replaces the AssInfo function and supports online changes.

**Cluster Functions**

ClusterActivate	Allows the user to activate an inactive cluster.
ClusterDeactivate	Allows the user to deactivate an active cluster.
ClusterFirst	Allows the user to retrieve the first configured cluster in the project.
ClusterIsActive	Allows the user to determine if a cluster is active.
ClusterNext	Allows the user to retrieve the next configured cluster in the project.
ClusterServerTypes	Allows the user to determine which servers are defined for a given cluster.

ClusterStatus	Allows the user to determine the connection status from the client to a server on a cluster.
ClusterSwapActive	Allows the user to deactivate an active cluster at the same time as activating an inactive cluster.

**I/O Device Functions**

SubscriptionAddCallback	Adds a callback function to a tag subscription.
SubscriptionGetAttribute	Reads an attribute value of a tag subscription.
SubscriptionRemoveCallback	Removes a callback function from a tag subscription
TagGetProperty	Gets a property for a variable tag from the datasource.
TagGetScale	Gets the value of a tag at a specified scale from the datasource
TagSubscribe	Subscribes a tag for periodic monitoring and event handling.
TagUnsubscribe	Unsubscribes a tag for periodic monitoring and event handling.

**Tag Functions**

TagInfoEx	Supports online changes.
TagWriteEventQue	Opens the tag write event queue.

**Task Functions**

TaskCluster	Gets the name of the cluster context in which the current task is executing
-------------	---

**Trend Functions**

TrnBrowseClose	Closes a trend browse session.
TrnBrowseFirst	Gets the oldest trend entry.
TrnBrowseGetField	Gets the field indicated by the cursor position in the browse session.
TrnBrowseNext	Gets the next trend entry in the browse session.
TrnBrowseNumRecords	Returns the number of records in the current browse session.
TrnBrowseOpen	Opens a trend browse session.
TrnBrowsePrev	Gets the previous trend entry in the browse session.

TrnGetCluster	Gets the name of the cluster the trend graph is associated with.
TrnGetPenComment	Gets the comment of a trend pen.

**Report Functions**

RepGetCluster	Retrieves the name of the cluster the report is running on.
---------------	---

**Modified Functions****Alarm Functions**

AlarmAck	Acknowledges alarms.
AlarmAckRec	Acknowledges alarms by record number
AlarmActive	Determines if any alarms are active in the user's area.
AlarmClear	Clears acknowledged, inactive alarms from the active alarm list.
AlarmClearRec	Clear an alarm by its record number
AlarmDelete	Deletes alarm summary entries.
AlarmDisable	Disables alarms.
AlarmDisableRec	Disables alarms by record number
AlarmDsp	Displays alarms.
AlarmDspLast	Displays the latest unacknowledged alarms.
AlarmEnable	Enables alarms.
AlarmEnableRec	Enables alarms by record number
AlarmFirstTagRec	Searches for the first occurrence of an alarm tag, name, and description
AlarmGetDelayRec	Gets the delay setting for an alarm via the alarm record number
AlarmGetFieldRec	Gets alarm field data from the alarm record number
AlarmGetThresholdRec	Gets the thresholds of analog alarms by the alarm record number
AlarmNextTagRec	Searches for the next occurrence of an alarm tag,

	name, and description.
AlarmNotifyVarChange	Activates a time-stamped digital or time-stamped analog alarm
AlarmSumAppend	Appends a new blank record to the alarm summary.
AlarmSumCommit	Commits the alarm summary record to the alarm summary device.
AlarmSumDelete	Deletes alarm summary entries.
AlarmSumFind	Finds an alarm summary index for an alarm record and alarm on time.
AlarmSumFirst	Gets the oldest alarm summary entry.
AlarmSumGet	Gets field information from an alarm summary entry.
AlarmSumLast	Gets the latest alarm summary entry.
AlarmSumNext	Gets the next alarm summary entry.
AlarmSumPrev	Gets the previous alarm summary entry.
AlarmSumSet	Sets field information in an alarm summary entry.
AlarmSumSplit	Duplicates an alarm summary entry.
AlarmSumType	Retrieves a value that indicates a specified alarm's type.

**I/O Device Functions**

DriverInfo	Provides information about the driver for a particular I/O Device.
IODeviceControl	Provides control of individual I/O Devices. The I/O device client operations used by the function (via the mode parameter) are no longer available.
IODeviceInfo	Gets information on an I/O Device.

**Miscellaneous Functions**

AccControl	Controls accumulators for example motor run hours.
ServerInfo	Gets client and server information.
ServerInfoEx	Gets client and server information from a specified process in a multiprocessor environment.
Shutdown	Ends Plant SCADA's operation.

**Report Functions**

RepGetControl	Gets report control information.
Report	Runs a report.
RepSetControl	Sets report control information.

**SPC Functions**

SPCALarms	Returns the status of the specified SPC alarm.
SPCProcessXRSGet	Gets the process mean, range and standard deviation overrides.
SPCProcessXRSSet	Sets the process mean, range and standard deviation overrides.
SPCSpecLimitGet	Gets the specification limits (USL and LSL) for the specified tag.
SPCSpecLimitSet	Sets the specification limits (USL and LSL) for the specified tag.
SPCSubgroupSizeGet	Gets the size of a subgroup for the specified SPC tag.
SPCSubgroupSizeSet	Sets the subgroup size for the specified SPC tag.

**Super Genie Functions**

Ass	Associates a variable tag with a Super Genie.
AssPage	Associates up to eight variable tags with a Super Genie and displays the Super Genie in the current window.
AssPopUp	Associates up to eight variable tags with a Super Genie and displays the Super Genie in a popup window.
AssTag	Associates a variable tag with the current Super Genie. The association will be created for the current Super Genie only, and will only come into effect after you re-display the Super Genie.
AssVarTags	Associates up to eight variable tags with a Super Genie. This association is only made for the next Super Genie you display (either in the current window or in a new window). You can use this function repeatedly to associate more than 8 variable tags to a Super Genie.
AssWin	Associates up to eight variable tags with a Super Genie, and displays the Super Genie in a new window.

**Tag Functions**

TagGetProperty	This function reads a property of a variable tag from the datasource
TagGetScale	Gets the value of a tag at a specified scale from the datasource
TagRamp	This function will increment a Tag by the amount defined by iPercentInc
TagRead	Reads a variable from the I/O Device
TagScaleStr	Gets the value of a tag at a specified scale
TagWrite	Writes to an I/O Device variable by specifying the variable tag.

**Task Functions**

MsgOpen	Opens a message session with a Plant SCADA server or client.
---------	--

**Trend Functions**

TrendDspCursorTag	Displays the tag name of the current pen.
TrnAddHistory	Restores an old history file to the trend system.
TrnDelHistory	Deletes an old history file from the trend system.
TrnEventSetTable	Sets trend data from a table, for a specified trend tag.
TrnEventSetTableMS	Sets event trend data and time data (including milliseconds) for a specified trend tag.
TrnFlush	Flushes the trend to disk.
TrnGetDefScale	Gets the default engineering zero and full scales of a trend tag.
TrnGetPen	Gets the trend tag of a pen.
TrnGetTable	Stores trend data in an array.
TrnInfo	Gets the configured values of a trend tag.
TrnNew	Creates a new trend at run time.
TrnSelect	Sets up a page for a trend.

**Window Functions**

WinCopy	Copies the active window to the Windows clipboard.
WinFile	Writes the active window to a file.
WinNewAt	Opens a new display window at a specified location, with a selected page displayed.
WinPrint	Prints the active window.

## Obsolete Functions

### Cluster Functions

ClusterGetName	Returns the names of the primary and standby cluster servers. Has been replaced with the cluster specific events parameter formatted as [ServerType.ClusterName.ServerName]Events
ClusterSetName	Connects to a specific cluster server. Has been replaced with the cluster specific events parameter formated as [ServerType.ClusterName.ServerName]Events

### Display Functions

DspCol	Displays a color at an AN.
--------	----------------------------

### Task Functions

ReRead	<p>Causes Plant SCADA to re-read the I/O Device data associated with the current Cicode task.</p> <p>Tags are now subscribed at the start of a function and updated tag values are sent to the subscribing function.</p> <p>Tag subscriptions are made at the update rate of:</p> <ul style="list-style-type: none"> <li>• the graphics page if called from a page</li> <li>• the default subscription rate as determined by [Code]TimeData in the Parameters online help if called from Cicode (default 250 ms)</li> <li>• the update rate requested of a task created using TaskNewEx</li> <li>• the update rate requested of a subscription created using TagSubscribe</li> </ul> <p>Verify that the subscription update rate matches the requirements of your function.</p> <p>After removing ReRead from looping code you may</p>
--------	--

	<p>need to extend the period of the Sleep function. This is to replace the pause ReRead created while it read the tag values.</p> <p><b>Note:</b> The parameter[Code]TimeData is used to set the Cicode refresh rate. when using the TaskNew Cicode function, use mode 4 to replace ReRead at the start of the function.</p>
--	--

## CtAPI Functions in 7.0

The following section details the changes made to CtAPI functions in Plant SCADA 7.0.

### Obsolete Functions

Previously available "Point" related functions are now no longer available, and if used will detect and return an error indicating that they are not supported. In order to obtain the same result as was previously invoked by those function, replace them with the Tag based equivalent using the appropriate Tag arguments and conditions. The "point" functions that are no longer available are listed below along with their replacement functions:

Function	Replacement
ctPointGetProperty	ctTagGetProperty
ctPointNew	ctListNew or ctTagWrite and ctTagRead
ctPointRead	ctTagRead or ctListRead with ctListData
ctPointWrite	ctTagWrite or ctListWrite

If you are using the point functions on single tags, use the ctTagRead, ctTagWrite functions instead. If you are building up multiple tags into one point, use ctListNew and add tags to the list through ctListAdd. Then use ctListWrite, ctListRead and ctListData to write and read from the tags.

The following functions are not relevant to tag based operations. They are obsolete and there is no replacement function.

- ctPointBitShift
- ctPointClose
- ctPointCopy
- ctPointSize
- ctPointToStr
- ctStrToPoint
- ctTagToPoint

## Kernel Commands in Version 7.0

The following Kernel commands are obsolete in Plant SCADA from Version 7.0:

- Kernel Alarm
- Kernel Trend
- Kernel Report
- Kernel IOServer
- Kernel Client
- Probe
- NetBIOS
- PageNetstat

# About Plant SCADA

AVEVA Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution that is used to manage and monitor processes in manufacturing, primary production, utilities delivery and facilities management.

Plant SCADA can be configured as a standalone or a distributed client-server system, including native redundancy and load sharing capabilities. Plant SCADA's client-server architecture allows system components to be distributed across a number of computers on a LAN, creating a system that offers geographical flexibility and performance benefits over standalone systems.

For detailed information on Plant SCADA, see:

- [Working with Plant SCADA](#)
- [Typical System Scenarios](#)
- [Licensing](#)
- [Upgrade Plant SCADA](#)
- [What's New in Plant SCADA](#)
- [What's New - Previous Releases](#)

## Working with Plant SCADA

This section of the help provides an overview of Plant SCADA by introducing the fundamental activities you need to consider when building and operating a system. They include the following:

- [Configure Plant SCADA Projects](#)
- [Define a Topology](#)
- [Build a System Model](#)
- [Create Graphics Pages](#)
- [Secure Your System](#)
- [Compile and Run a Project](#)
- [Deploy a Project.](#)

These steps outline a typical work flow that is reflected in Plant SCADA Studio and runtime tools.

## See Also

[Plant SCADA Studio](#)

## Configure Plant SCADA Projects

Everything you create and configure for a Plant SCADA system is stored in projects. This can include graphic pages, variable tags, alarms, communications settings, equipment definitions, programming content, and so on.

Plant SCADA allows you to work with projects in a way that logically reflects the architecture of the SCADA system. If the system you are building represents a large production facility, you could manage the required content using multiple projects that operate in tandem. Each project could represent a geographical area within the plant, or a particular production flow.

- **Included Projects**

To enable the runtime integration of multiple projects, Plant SCADA allows you to "include" a project within another project. This means you can share content across a number of projects while maintaining each as a separate entity.

For example, you could create separate projects to represent each of the production lines within a factory. You could then include these projects in a main project that represents the whole factory. By doing this, you are able to develop and test the included projects independently.

- **System Projects**

Plant SCADA also comes with a number of system projects that you can include in the projects you create. These projects contain pre-configured content (such as page templates and control objects) that you can add to your own content.

This framework is fully realized with Plant SCADA's "starter projects". A starter project allows you to create a new project that includes the basic components required to start building your own system. A set of Plant SCADA's system projects are automatically included in any new projects you create (based on a selected template style), which means a complete set of page templates and library controls are ready for you to use.

To prepare a project for use in a runtime environment, you need to compile it and distribute the required files to the computers that will be engaged in the runtime system. This initially requires the creation of a system topology (see [Define a Topology](#)).

For more information, see [Plant SCADA Projects](#).

## Define a Topology

Plant SCADA supports a client-server network architecture. This provides the flexibility to adapt a Plant SCADA system to any production scenario, with support for scalability, server clustering, and system redundancy. When you define a topology, you are identifying the **computers** and **I/O devices** that are built into this architecture, and the role each will perform at runtime.

## Computers

In a Plant SCADA system, a computer can act as a server, or a client, or both. Servers are used to manage communication with plant equipment and collate production data, while clients provide the interface to assess and interact with the system. A topology allows you to view how your servers and clients are arranged.

There are several different types of server processes used in a Plant SCADA system. These include:

- Alarms server process
- I/O server process
- Trends server process
- Reports servers process.

How you arrange these sever process types across the computers included in your system topology will be determined by the following:

- **Operational requirements** — a standalone system may contain a single computer that hosts every component of a system, whereas a distributed system may have a separate computer dedicated to each server type.
- **Redundancy** — the servers associated with a system can be duplicated and defined as primary and standby units, allowing the system to keep running if one of the servers becomes inoperative.
- **Clusters** — Clustering allows you to group independent sets of Plant SCADA's server components within a single project, allowing multiple systems to be monitored and controlled simultaneously. The clusters offer the benefit of keeping a logical structure to the project during configuration.

Plant SCADA's configuration environment allows to you view a system topology as a computer-based view or a cluster-based view.

## I/O Devices

I/O devices can include any control or monitoring equipment with a communication port or data exchange interface, such as programmable logic controllers, loop controllers, bar code readers, scientific analyzers, remote terminal units, or distributed control systems.

To incorporate an I/O device into a system topology, you will need to install the relevant driver. A driver allows an I/O device to communicate with a Plant SCADA I/O server.

You also need to configure the various components required to enable runtime communication with the device. This can include boards, ports and modems. You will need to consider the following:

- **Transport medium** — the physical communications medium and the low-level logic necessary to drive it.
- **Protocol** — typically a device will support an industry-standard communications protocols, such as Modbus, OPC, DNP 3.0 or BACnet. This will, however, depend on the device.

Much of the functionality supported by the protocols and transports you select can be modified using the driver options and parameters. Each driver has its own specific section in the online help that will guide you through configuration options.

To help you get started, Plant SCADA includes a Device Communications Wizard that can assist with the configuration of the communications settings for a particular device.

## See Also

[Topology](#)  
[Server Processes](#)  
[I/O Devices](#)

## Build a System Model

You can use a System Model to organize a Plant SCADA project in a way that reflects the physical production system. This will simplify the configuration of a project, and deliver inherent logic when the project is executed at

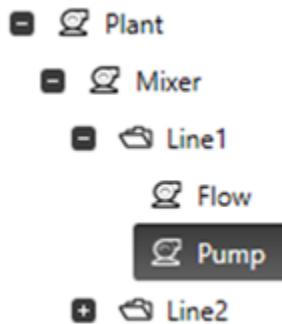
runtime.

You can build a System Model into a project through equipment definitions and tags.

## Equipment

Equipment facilitates the creation of an object-oriented System Model in a Plant SCADA project. This allows you to logically reference production system hardware based on its location in an equipment hierarchy.

Equipment definitions use dot notation to indicate levels within this hierarchy. For example, the equipment name "Plant.Mixer.Line1.Pump" refers to the current selection in the following equipment hierarchy:



Each piece of equipment become a parent to the tags that are associated with it. At runtime, the hierarchy can be used to search and display information in a way that is logically aligned with your System Model.

## Tags

In a Plant SCADA project, you use tags to label the inputs and outputs on your production system hardware. They can include **variables**, **trends**, **alarms**, **accumulators**, and so on.

A structured tag naming strategy that logically reflects geographical areas or processes within a plant is an example of a way you can use tags to implement a simple System Model. This type of approach means the purpose of each tag is inherently clear through its name.

To configure tags and equipment in a Plant SCADA project, you use the **System Model** activity in Plant SCADA Studio.

## See Also

- [Equipment](#)
- [Alarms](#)
- [Variable Tags](#)
- [Trends](#)
- [Accumulators](#)

## Create Graphics Pages

Graphics pages present the content that appears on a Plant SCADA display client at runtime. They create the visual interface an operator uses to monitor and control a production system.

With graphics pages, you can:

- Use animated objects to display the operating status and performance of a plant
- Provide operators with centralized or local control of production equipment using graphical tools and keyboard commands
- Implement historical trending of tag data in a graphical format
- Develop a multi-layered security system that controls user access according to functional or geographical areas.

Graphics pages are created and edited using **Graphics Builder** or the **AVEVA™ Industrial Graphics Editor**. You can also browse and manage pages and library items in the **Visualization** activity.

To help you create graphics pages, Plant SCADA supports page templates. Templates can incorporate standard navigation and support tools that are common across a set of functional page layouts. This delivers a consistent look and feel to a runtime system.

To create a project based on one of the template sets included with Plant SCADA, you can use a [Create a Project Using a Starter Project](#). Any project that is created from a starter project will also include a set of library objects that you can add to your graphics pages, such as **Genies** and **Super Genies** (controls that can be passed information at runtime).

## See Also

- [Visualization](#)
- [Graphics Builder](#)
- [AVEVA Industrial Graphics](#)

## Secure Your System

A Plant SCADA system is protected by implementing a user-based security model. When planning a project, you need to consider who will be using the system and the areas they will need to access.

There are two ways you can manage user accounts:

- **Integrated Windows security** (recommended) — Windows user accounts are used to access Plant SCADA computers. The security measures specified for the local network domain will apply.
- **Plant SCADA security** — user accounts are created within the Plant SCADA project. They can be organized into groups.

In both cases, access to the Plant SCADA system is managed by assigning user accounts to "roles" defined within a project. Roles specify for a particular type of user:

- The areas they will be able to access
- The privileges they will have.

Roles and Plant SCADA user accounts are configured using the **Security** activity.

You can also [Securing Runtime Computers](#), or run Plant SCADA as a [Windows service](#).

## See Also

[Security](#)

[Roles](#)

## Compile and Run a Project

Whenever you modify a Plant SCADA project, you need to compile it to incorporate the changes you have made. Compiling a Plant SCADA project assembles the included content (such as configuration databases, graphics and Cicode files) and prepares it for runtime. If required, error notifications will display in the Compile Messages area of Plant SCADA Studio.

Runtime activates a project and engages it with the production system. Staff can visually monitor the system, initiate production processes and respond to alarm conditions. Historical and trend data can also be collated and distributed to assess operational performance metrics such as production volume, efficiency, and maintenance requirements.

A set of runtime tools are available to support this:

- **Runtime Manager** — an application used to manage and control the CPU configuration of the project, and the running state of each component. See [Runtime Manager](#)
- **Process Analyst** — an Active X control that allows you to compare and analyze historical and real-time trend and alarm data during runtime. See [Process Analyst](#).
- **Scheduler** — a calendar-based tool that allows you specify how a piece of equipment will operate at a particular time. See [Scheduler](#).

## See Also

[Compile](#)

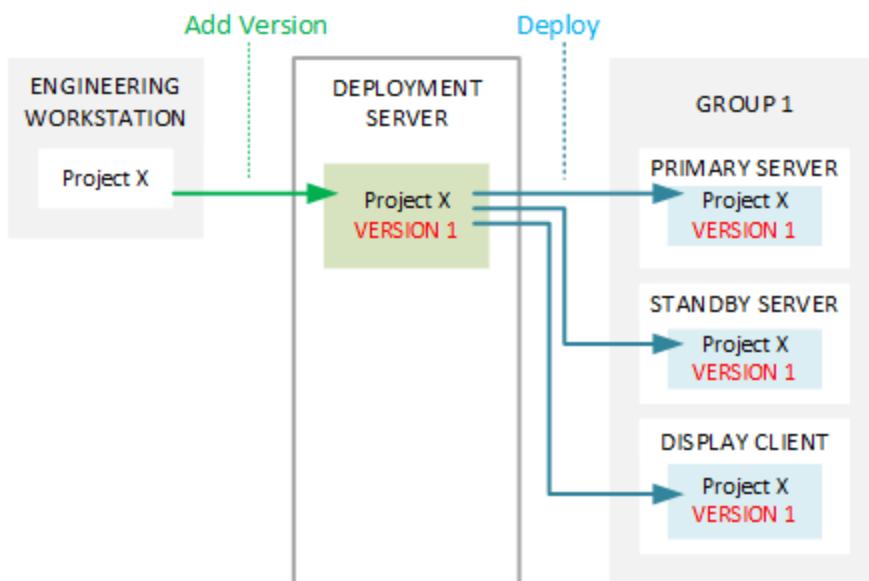
[Runtime](#)

## Deploy a Project

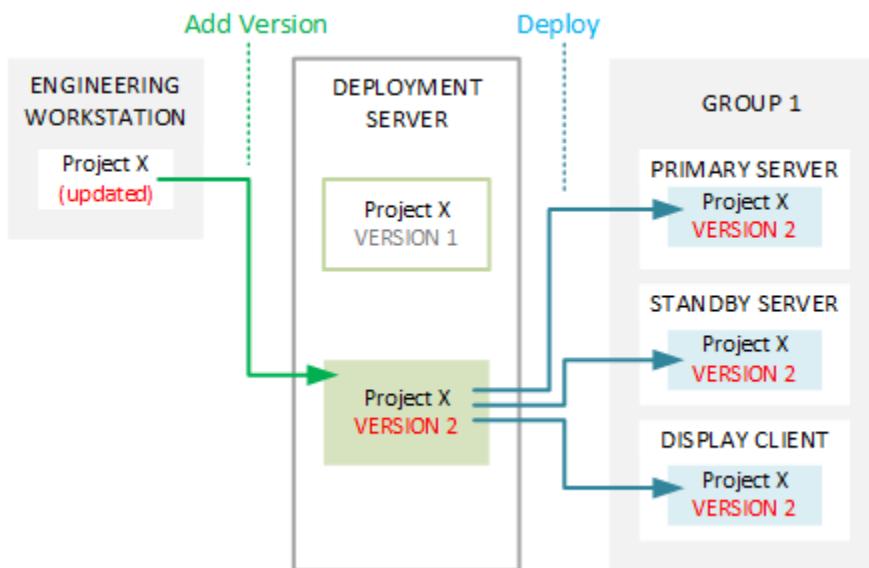
To engage in a runtime system, each computer in a Plant SCADA topology requires a copy of a project's runtime files. To help you distribute these files, Plant SCADA supports a deployment server.

The deployment server stores runtime files as "versions" and distributes them to deployment clients across an encrypted connection. Each version is created by adding a project to the deployment server.

In the diagram below, Project X has been configured on an engineering workstation. A version of the project is added to the deployment server as "Version 1". The project's runtime files can then be deployed to the computers included in a group named "Group 1".



If any changes are made to Project X, a new version of the project is simply added to the deployment server, allowing version 2 to be deployed.



To enable deployment, you firstly need to set up the deployment server and establish a connection to each deployment client using a tool called the Configurator.

You can then use the **Deployment** activity in Plant SCADA Studio to manage and distribute project versions.

## See Also

[Deployment](#)

## Typical System Scenarios

The scenarios described in this section of the help demonstrate how Plant SCADA can be used to support typical

processes found in primary production, utilities delivery, and manufacturing.

In reality, a project will incorporate a combination of the scenarios described here, with a high degree of customization and scalability. However, these examples have been simplified to demonstrate how Plant SCADA can be configured and deployed to meet the specific requirements of a production system.

System Type	Description
Standalone	Every component of a system runs on a single computer. See <a href="#">Standalone System</a> .
Distributed I/O	Plant SCADA is used to monitor and manage distributed devices that are each connected to remote I/O servers. See <a href="#">Distributed I/O System</a> .
Redundant Server	One or more of the servers associated with a system are duplicated and defined as primary and standby units, allowing the system to keep running in the event one of the servers becomes inoperative. See <a href="#">Redundant Server System</a> .
Client-server	The servers and clients associated with a system are independently distributed across a number of computers on a network, offering greater accessibility and performance benefits. See <a href="#">Client-Server System</a> .
Redundant and Distributed Control	Remote or geographically separate sections of a production system have fully operational sub-systems in place that are monitored and controlled locally. If such a sub-system becomes partially or wholly inoperative in a manner preventing local control, this arrangement allows remote Control Clients to take control of the affected sub-system. See <a href="#">Redundant and Distributed Control System</a> .
Cluster Controlled	A production system is organized into discrete areas being monitored by operators within each area. However, there is also a level of control that supervises every area of the system. See <a href="#">Clustered Control System</a> .
Load Sharing System	The system splits the load of an otherwise stressed system across multiple machines, better utilizing the available infrastructure. See <a href="#">Load Sharing System</a> .

## See Also

[Clusters](#)

[Redundancy](#)

## Standalone System

A standalone installation of Plant SCADA runs every server and client component of a system on a single computer. These include:

- I/O server
- Alarm server
- Trends server
- Reports server
- Control client.

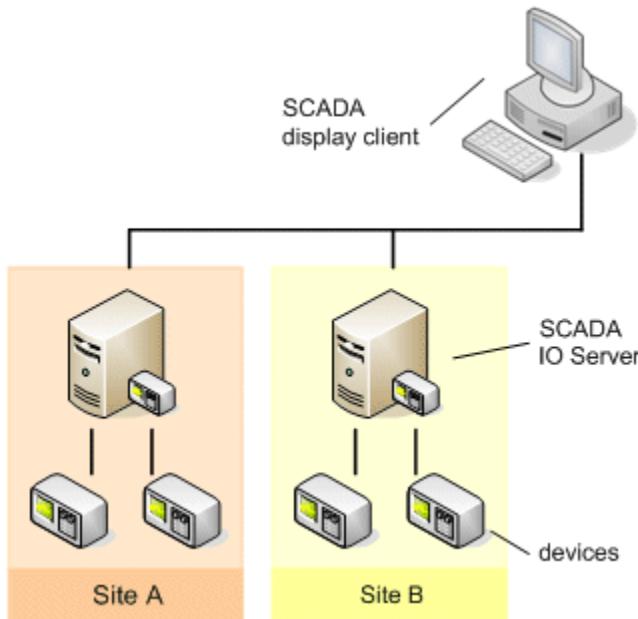
This allows Plant SCADA to be run as a small, self-contained system.

**Note:** You can run the server and client components of a standalone system as a single-process or multi-process system. It is recommended that a single- process setup only be used as a short term solution for your control system, or to run demonstrations and test projects. Adding redundancy to your system will make it more reliable and more efficient.

## Distributed I/O System

This scenario demonstrates a method of connecting Plant SCADA to a number of devices that are distributed across several sites over a wide geographical area.

Instead of attempting to connect devices directly via a remote connection, an I/O server is placed at each site, enabling communication to be managed within the system.

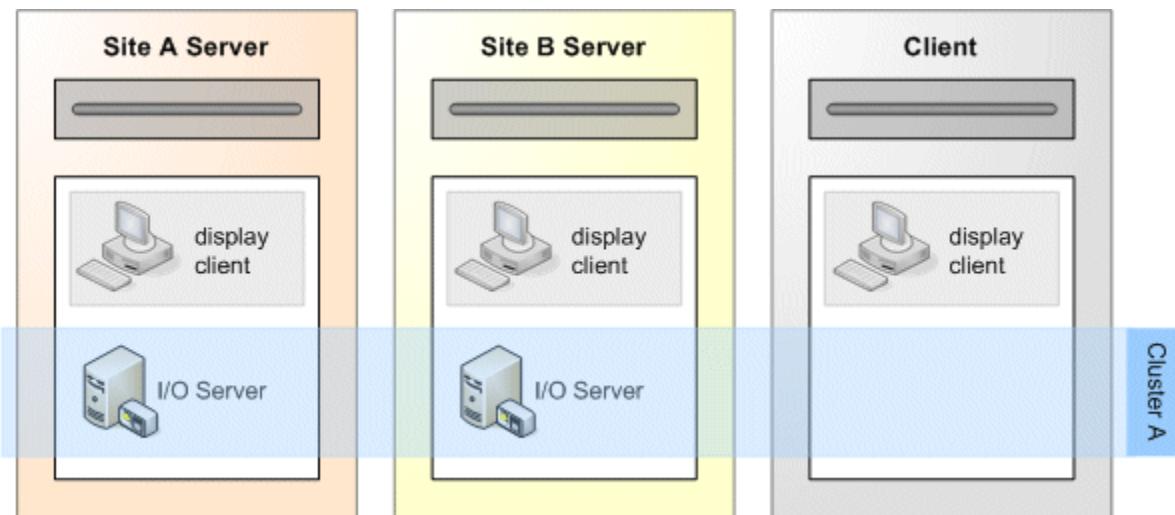


This model is also useful in plants that contain devices with a serial port or limited communications capabilities. By placing I/O servers on the factory floor to interface with these devices, you can optimize communications on slow or low-bandwidth networks and improve overall performance.

Despite the geographical distribution of I/O servers across many sites, this type of system can be configured as a

single cluster system, as a cluster is able to support many I/O servers.

The diagram below demonstrates how to approach the deployment of this type of system across the server machines using a single cluster.

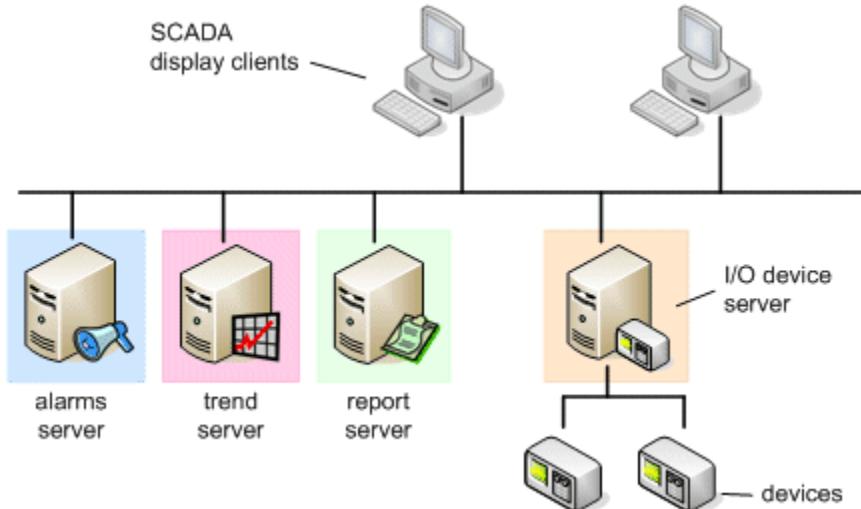


A second cluster will only become necessary if your project requirements call for more than one redundant pair of alarms, trends or reports servers.

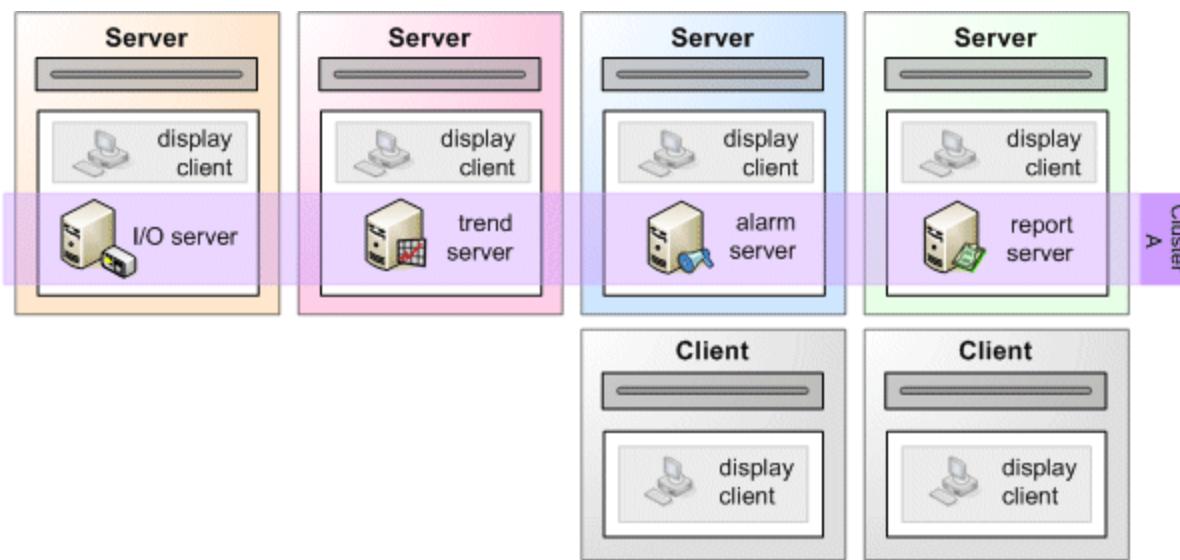
## Client-Server System

Plant SCADA's client-server architecture allows the components of a system to be distributed across a number of computers on a LAN, creating a system that offers geographical flexibility and performance benefits.

Each component is simply identified within the project by an address, allowing the location and hardware requirements for each to be considered independently.



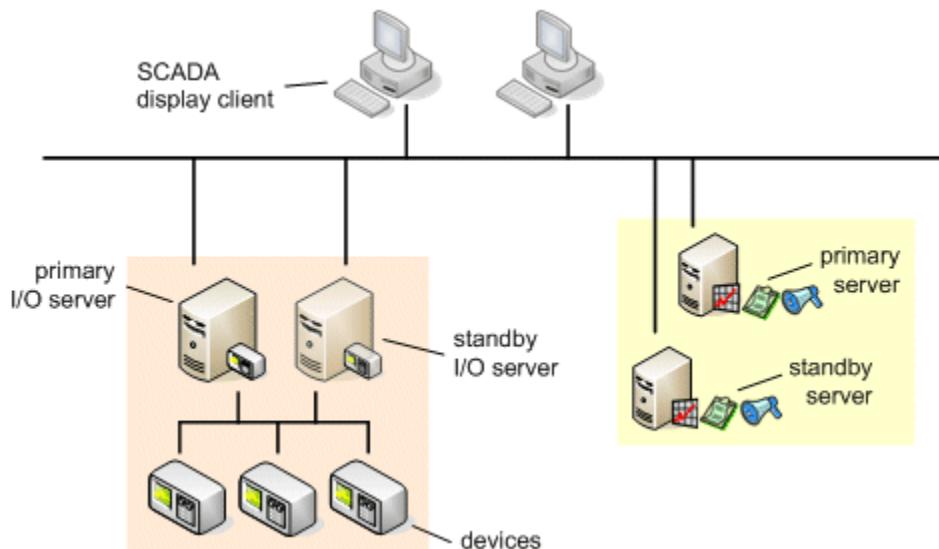
The diagram below demonstrates how this example can still be configured within a single cluster.



Each server also acts as a display client across the system architecture.

## Redundant Server System

The ability to define primary and standby servers within a project allows hardware redundancy to be built into your system infrastructure. This helps prevent situations where an error on one server results in the overall system becoming inoperative. Systems of this type are especially beneficial when service continuity and/or secure data collection are important.

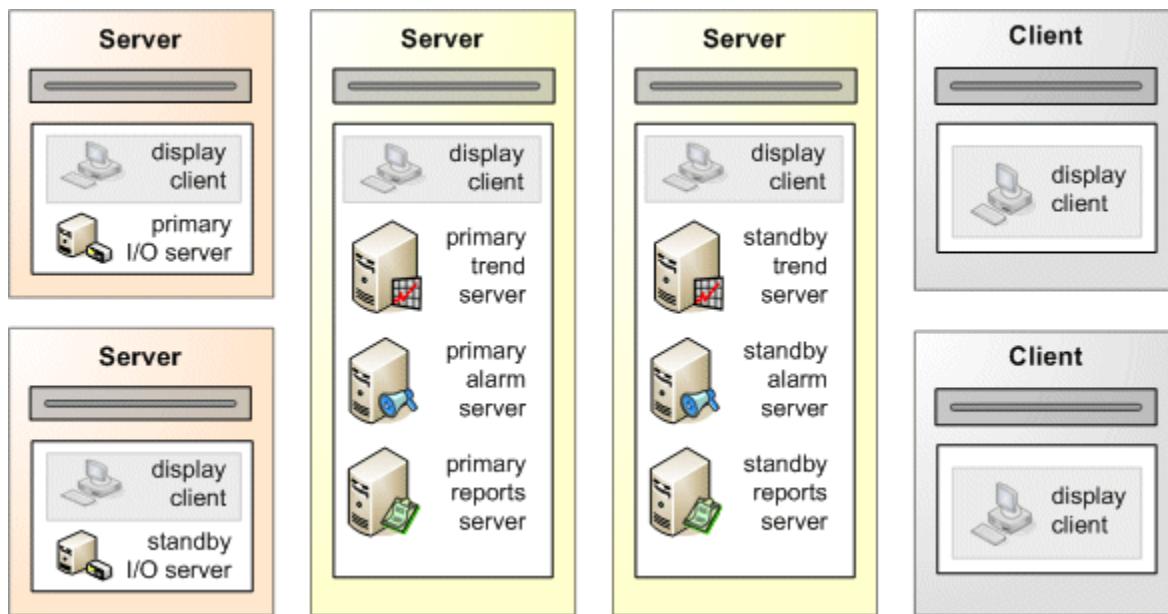


In the case of I/O server redundancy, a standby server is maintained in parallel to the primary server. If a hardware error is detected, the standby server can assume control of device communication with minimal interruption to the system. You can also use redundant I/O servers to split the processing load.

Alarm, report and trends servers can also be implemented as redundant servers. This improves the likelihood that clients will continue to have access to data from a standby server in the case a primary server becomes inoperative. Plant SCADA maintains identical data on both servers.

In the diagram below, the primary and standby I/O servers are deployed independently, while the alarms, trends

and reports servers are run as separate processes on common primary and standby computers. In this case, the entire system can be configured as a single cluster.

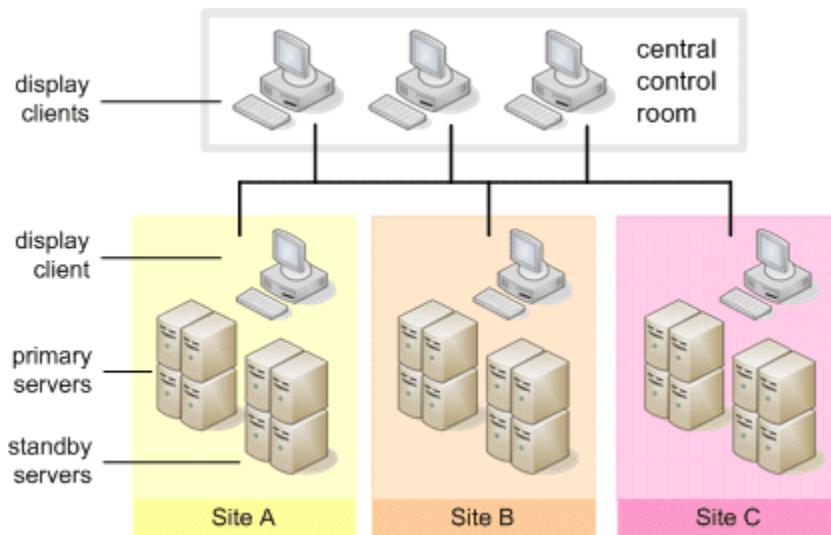


## See Also

[Redundancy](#)

## Clustered Control System

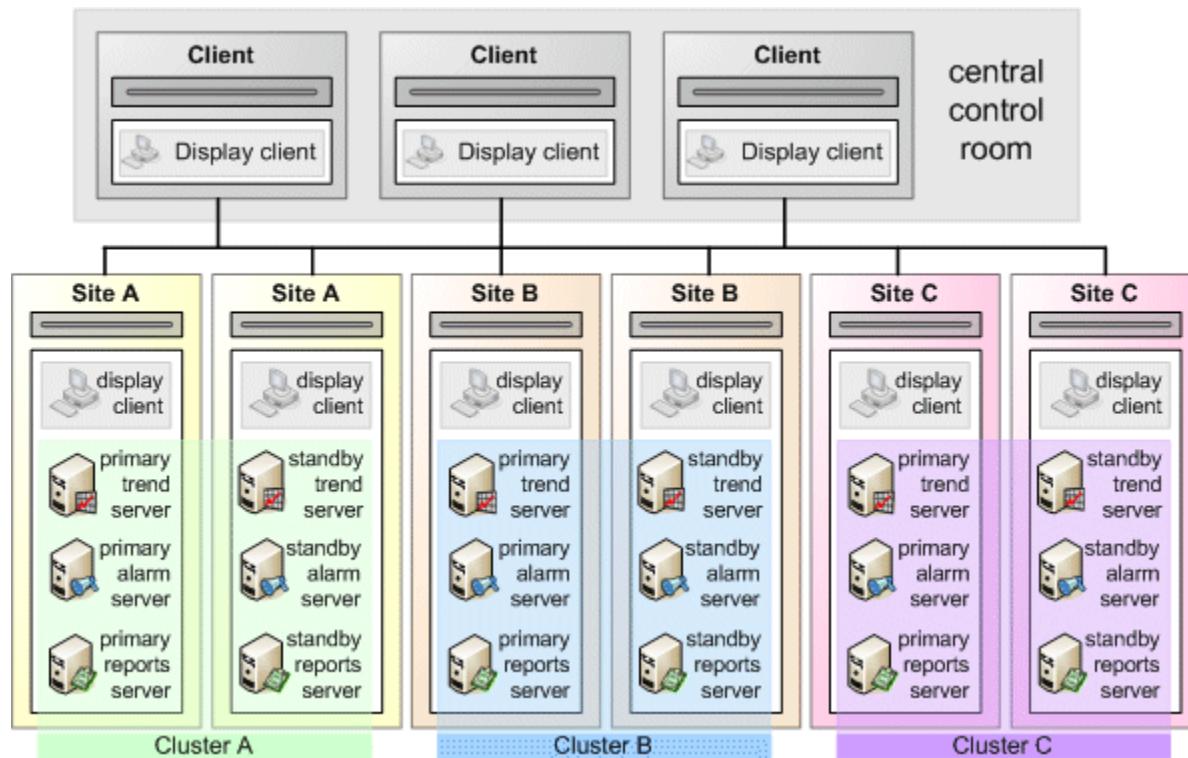
In this scenario, the system is organized into discrete sites being controlled by local operators, and supported by local redundant servers. At the same time, there is a level of management that requires sites across the system to be monitored simultaneously from a central control room.



Each site is represented in the project with a separate cluster, grouping its primary and standby servers. Clients at each site are only interested in the local cluster, whereas clients at the central control room are able to view every cluster.

The deployment of a control room scenario is straightforward, as each site can be addressed independently within its own cluster. The control room itself only needs control clients.

The deployment of servers could be mapped out as follows:



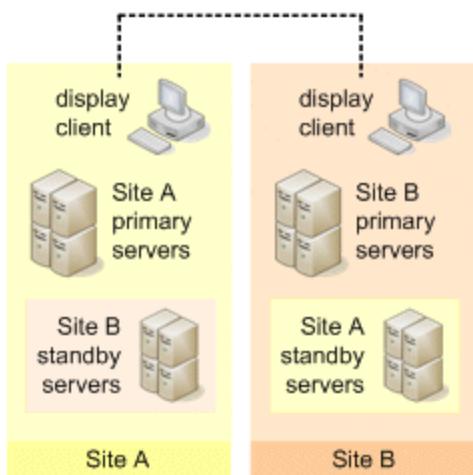
Plant SCADA's support for dynamic clustering means each site can be monitored and controlled from the central control room if necessary. For example, if an operator at a particular site only works during regular business hours, then the monitoring can be switched to the central control room after hours.

## Redundant and Distributed Control System

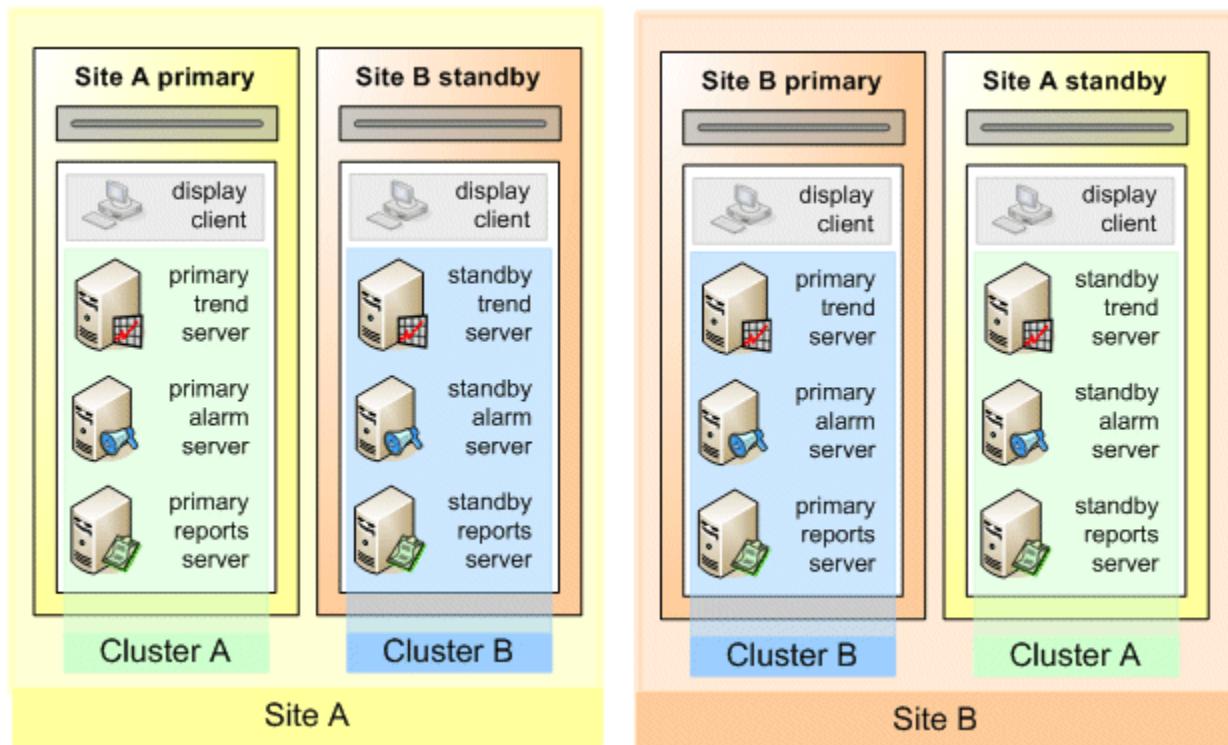
In this scenario, a project represents a number of locally operated sites each containing its own set of servers and clients. For example, a number of pumping stations across a water distribution system, or multiple production lines in a manufacturing facility. However, there is a requirement for monitoring to continue in the event the system at one of the sites becomes inoperative.

This is achieved by distributing the primary and standby servers across the different sites, or by placing the standby servers in a central location.

Clustering is used to define the role of the different servers at each site, which can be viewed in a common project running on every client. This means Site A can be monitored from Site B, and vice versa, if a system becomes inoperative at one of the sites.



The example above would require the creation of two clusters, so that the project can include two sets of primary and standby servers. The clusters represent the redundant pairs of servers, and would be deployed across the two sites as follows:



The clusters offer the benefit of keeping a logical structure to the project during configuration, despite the unusual distribution of redundant server pairs.

## Load Sharing System

Load sharing of system components across different computers and CPUs means the work load of a potentially stressed system can be split across multiple machines, better utilizing the available infrastructure.

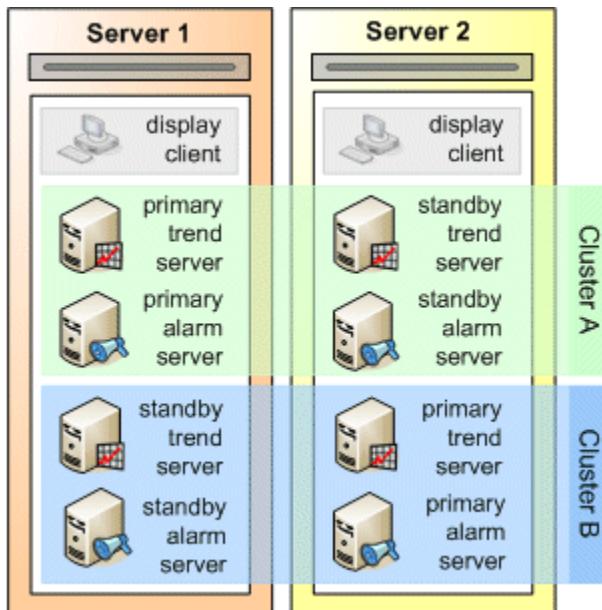
For example, managing alarms can draw heavily on a CPU's performance, while trending data can use a lot of disk space. By assigning your trends and alarm servers to different processes on a shared computer, an alarm

server can be used as a standby trends server, making practical use of idle disk space.

This approach can be used to improve network performance, data access times, and general system stability.

If you introduce clustering, you have the flexibility to run multiple servers of the same type on a single computer. As long as a client has access to every cluster configured in a project, it doesn't matter if a set of servers is distributed across a number of clusters.

In the diagram below, two servers have been configured to act as standby units for each other, supporting two sets of redundant trends and alarm servers.



Both machines have an even balance of trends and alarm servers, making effective use of the CPU and disk space. By distributing the servers across two clusters, the servers are also able to act as redundant units to each other. This has reduced the necessary number of computers from a maximum of eight down to just two.

## Licensing

To determine the licensing requirements for a Plant SCADA system, you firstly need to consider the roles assigned to your runtime computers. Roles are used to define how a computer will function within a runtime system. There are four options:

- Server
- Control client
- View-only client
- Standalone HMI (a system on a single computer with no networking).

A role is specified for a computer using the [Computer Role](#) page of the Setup Wizard, or via the Citect.ini parameter [\[Client\]ComputerRole](#).

The computer roles align with the different license types that are supported by Plant SCADA. When licenses are allocated at runtime, it is the role that determines the type of license a computer will attempt to retrieve. This process also takes into account the following:

- A server license also provides authorization for a single control client license.
- A computer running as a server only needs one license, even if it is running multiple server processes.

**Note:** Every runtime computer includes a client process that manages license acquisitions. If you shut down the client process, licensing will stop working for all other local processes.

---

This means you need to consider the following before you implement the licensing for a Plant SCADA system.

- The number of computers that are assigned to each role in your runtime system.
- The number of licenses you have for each computer type.
- Where the licenses will be stored.
- How the licenses will be retrieved by each computer.

This will largely be determined by the licensing technology you use; **AVEVA Enterprise Licensing** (software based) or **Sentinel Licensing** (USB keys).

**Note:** You cannot run both licensing systems on the same Plant SCADA computer.

---

- **AVEVA™ Enterprise Licensing**

The AVEVA Enterprise Licensing system is a common platform solution that allows you to manage AVEVA Enterprise Software product licenses.

The system is comprised of a browser-based License Manager and a License Server that allow you to share and deliver licenses for your installed AVEVA applications. This distributed architecture provides the following benefits:

- Centralized license management.
- Increase license security through the use of activated licenses.
- Flexible topologies for any size system.

You can use Plant SCADA Studio to launch Enterprise License Manager from any installed location within your Plant SCADA system. See [Launch Enterprise License Manager from Plant SCADA Studio](#).

**Note:** The AVEVA Enterprise Licensing system is required for Industrial Graphics applications.

---

For more information, see [AVEVA™ Enterprise Licensing](#).

- **Sentinel Licensing (using USB keys)**

Sentinel Licensing is a legacy licensing solution for Plant SCADA. It uses physical USB keys that plug in to each computer in your Plant SCADA system. The USB key contains details of your user license, such as its type and I/O point count.

When a USB key is detected on a computer, all of the available licenses are loaded into runtime. Any additional licenses that are not used by the local computer are then made available to other computers. This is enabled via a floating license mechanism that allows server computers to allocate licenses to clients.

You are occasionally required to update your Sentinel keys, for example, when you upgrade to a new version of Plant SCADA. To do this, you need to retrieve an authorization code from AVEVA's online License Generator. See [Update a Sentinel Key with CiUSAPE](#).

In both cases, Plant SCADA uses a [Dynamic Point Count](#) to determine if your system is operating within the limitations of your license agreement. This process tallies the number of I/O device addresses being used by the runtime system.

A point limit is allocated to each type of license included in your license agreement.

If required, you can specify how many points will be required by a particular computer (see [Specify the Required Point Count for a Computer](#)).

---

**Note:** With the release Plant SCADA 2023, Schneider Electric's Floating License Manager is no longer supported. Existing users of the Floating License Manager will need to reconfigure their system to use AVEVA Enterprise Licensing.

---

## See Also

[Demo Mode](#)

## AVEVA™ Enterprise Licensing

The AVEVA Enterprise Licensing system is a common platform that allows you to manage AVEVA Enterprise Software product licenses.

The system was introduced to allow centralized license management, remove the need for dongles, and increase license security by use of activated licenses. It supports flexible topologies for any size system.

The following describes the components of the AVEVA Enterprise Licensing system.

### **License Manager**

The License Manager allows you to access and maintain licenses for certain AVEVA Enterprise Software products in your different environments using its scalable, flexible design features:

- Browser-based for scalability and ease of use; can be remotely accessed by any of the supported web browsers.
- Light-weight, standalone software you can install on the same node as the License Server computer, or on any other node based on your deployment needs.
- Manage one or multiple License Servers to organize the licensing requirements for your environment.
- Access details about your licenses, such as usage information, from the License Manager interface.

You can use Plant SCADA Studio to launch Enterprise License Manager from any installed location within your Plant SCADA system. See [Launch Enterprise License Manager from Plant SCADA Studio](#).

### **License Server**

License Server provides the functionality to acquire, store, maintain, and serve licenses to your installed AVEVA Enterprise software.

- Licenses are hosted and maintained on the License Server.
- Securely serve any type of software applications being licensed, including Windows browsers, tablets, and mobile devices.
- Provide current license usage information.

---

**Note:** Plant SCADA 2023 R2 allows you to run your AVEVA Enterprise License Server in **Secure Mode**. This facilitates encrypted communication for any Plant SCADA computers that connect with your license server to acquire licenses. For more information, see *Running in Secure Mode* in the topic [Use Configurator to Set Up an Enterprise License Server](#).

---

### [Activation Server](#)

Activation Server is a cloud-based Internet-accessible server to which the License Manager connects for license activation. The License Manager connects to the AVEVA Activation Server in the Cloud only temporarily during the activation process.

### Installed Products

Once the AVEVA Enterprise Software product is installed it relies on the licensing system to enable its functionality. Following installation, you use Configurator to specify from which License Server you want the product to get its license.

- If a License Server is installed on the same computer as the product, it will be selected in Configurator by default.
- If the product is installed on a different computer than the License Server, you can use Configurator to point the product to the required License Server computer.

You need to configure the connection to the License Server for any computers that need to acquire licenses. See [Use Configurator to Set Up an Enterprise License Server](#).

All AVEVA products installed on the same computer need to point to the same License Server.

If you are implementing a distributed topology, we recommend creating a network diagram as an initial step. This will allow you to visually model your system before you decide where to install the License Manager and License Server components. For more information, see [AVEVA™ Enterprise Licensing Topologies](#).

---

**Note:** When working with virtual environments, you need to deactivate all licenses in a License Server before cloning a virtual machine. If this is not done, the cloned License Server will go into Grace Period mode of 15 days, after which it will cease serving licenses.

---

## See Also

[Enterprise License Server Installation Requirements](#)

[Enterprise License Server Installation Procedure](#)

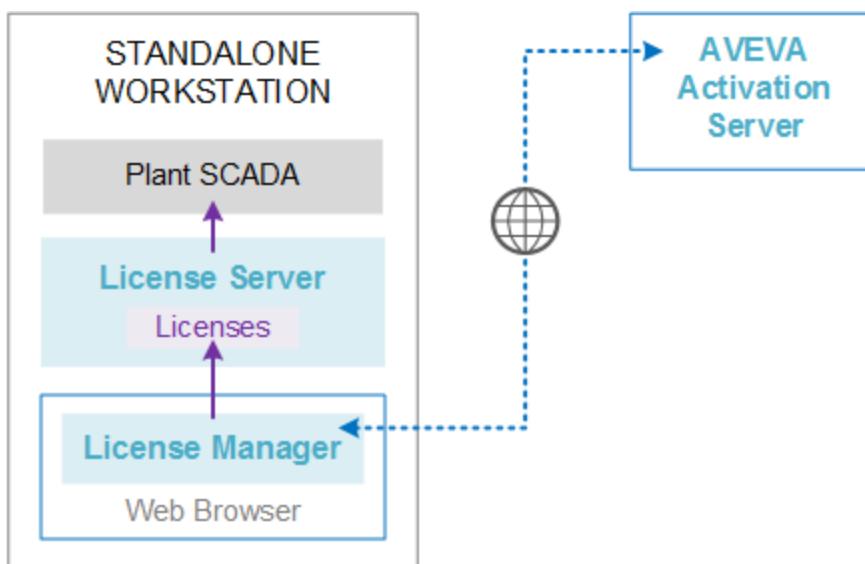
## AVEVA™ Enterprise Licensing Topologies

Use the models described in this topic to design the topology for your [AVEVA™ Enterprise Licensing](#) platform. This will help you decide where you need to install the License Manager and License Server components.

There are three topology types to consider.

### [Single Node Topology](#)

A single node topology involves a computer with Plant SCADA, License Manager, and License Server installed. In this configuration, the communication between the server components, License Server, and License Manager is local. Internet access is required to connect to the Activation Server.



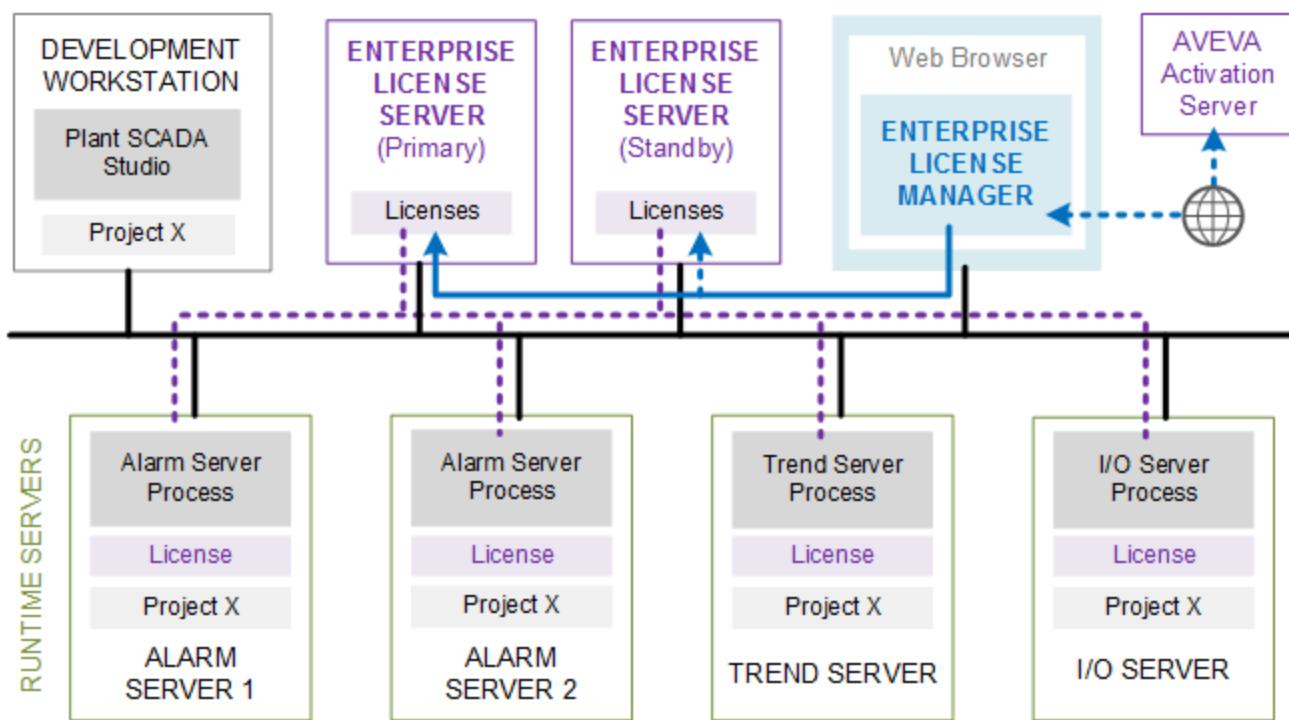
Typically a single node topology would be used to test a project on a development workstation. It can also be used for training or evaluation purposes.

### Distributed Topology

The licensing components can be installed on separate computers as long as there is reliable network connectivity between them. This type of topology is suitable for systems where you need to manage licenses for multiple installations of Plant SCADA across a distributed runtime system.

In the following example:

- The **Enterprise License Manager** components are installed on a computer with Internet access to the Activation Server. The interface runs locally in a web browser.
- The **Enterprise License Server** components are installed on two separate computers that are acting as a redundant pair.
- The Plant SCADA runtime computers each require network access to the License Servers.



This type of topology allows you to manage multiple License Servers for a runtime system from a single License Manager browser interface.

License Server redundancy is highly recommended when centralizing your licenses. It is available at no additional cost. A paired set of License Servers keeps licenses, reservations and the status of licenses synchronized to help maintain stability for the runtime system.

If you use clustering to manage a system that is physically distributed across multiple sites, it is recommended that you have a pair of redundant license servers at each location.

For information and procedures about implementing redundancy, see the topic *Working With License Server Redundancy* in the AVEVA Enterprise Licensing documentation in [AVEVA™ Help](#).

#### **Disconnected Topology**

This topology is required if an internet connection is not available. For example, a firewall or other constraints may exist between the License Manager components and the Activation Server.

In this scenario, you need to set up a separate computer with Internet access to the License Activation Web Page and Activation Server and then complete the activation process in "offline" mode. For more information, see the topic *Activating Licenses in Offline Mode* in the AVEVA Enterprise Licensing documentation in [AVEVA™ Help](#).

---

**Note:** If you are planning to run a system that includes a combination of client, server, and HMI licenses, you need to confirm that your Plant SCADA computers have the appropriate role configured. This will allow an Enterprise License Server to correctly allocate different license types. Computer roles are configured using the [Computer Role Setup](#) page of the Setup Wizard.

For an example of how to acquire, activate and distribute your licenses, see [AVEVA™ Enterprise Licensing Workflow](#) in the Plant SCADA documentation.

For more detailed information see the AVEVA Enterprise Licensing documentation in [AVEVA™ Help](#).

## AVEVA™ Enterprise Licensing Workflow

The following example of the AVEVA Enterprise Licensing workflow assumes that both License Manager and License Server are installed on the same computer, and that the computer can connect to the Internet.

### 1. Purchase one or more AVEVA Enterprise Software products

Along with your installation media or downloads, you will receive one or more entitlement files containing license details.

### 2. Start License Manager

It opens in a browser window and accesses the installed License Manager components.

### 3. Select the License Server

Use the License Manager to select an available server to host the licenses you want to activate. You can also add a new license server.

---

**Note:** You need to configure the connection to the License Server on any computer that needs to acquire licenses. See .

---

### 4. Import the entitlement file

Once you import the entitlement file, the License Manager displays the available licenses.

The entitlement file contains information for specific product licenses that are imported into the License Manager to activate purchased licenses. It is sent to you in an email upon purchase of your product licenses. It is in a zipped .xml file format.

### 5. Activate licenses

License Manager connects to the AVEVA Enterprise Activation Server over the Internet to activate the selected licenses on the License Server. The Activation Server activates licenses on the License Server.

### 6. Manage your licenses

Licenses are now available for their respective product to use. You can now:

- Activate new licenses
- Deactivate current licenses
- Reserve and unreserve licenses
- Checkout licenses.

In some cases, you may need to reserve a license for a specific computer using the device reservation feature. For example, it is recommended that you checkout a license for any critical runtime computers. For more information, see [Allocating AVEVA™ Enterprise Licenses](#).

---

**Note:** Plant SCADA only supports device reservations. You cannot reserve licenses for specific users.

---

### 7. Install and configure Plant SCADA

When running, Plant SCADA will acquire licenses from the License Server selected in Configurator (see Use Configurator to Set Up an Enterprise License Server). They will release them when no longer needed.

---

**Note:** For more detailed information see the AVEVA Enterprise Licensing documentation in [AVEVA™ Help](#).

---

## Allocating AVEVA™ Enterprise Licenses

After you have activated licenses on an AVEVA Enterprise License Server for your Plant SCADA servers, clients

and standalone HMI systems, you need to allocate them to the appropriate computers in your SCADA system.

When Plant SCADA runtime starts, it will request a license from the AVEVA Enterprise License Server that it is connected to. The type of license will be determined by the role configured for the computer in Plant SCADA's Computer Setup Wizard.

For example, Plant SCADA servers will request a server license, control clients will request a control client license and view-only clients will request a view-only client license. Plant SCADA Access Anywhere clients or remote desktop clients will also automatically retrieve a license directly from the AVEVA Enterprise License Server.

However, there are circumstances where you may need to reserve a license for a specific computer using the device reservation feature.

- **Critical runtime computers**

For any computers in your SCADA system that support critical runtime functionality (for example, an I/O server or an alarm server), it is recommended you reserve the required license and download it directly on to the computer using the **Checkout** feature.

---

**Note:** If you are using device reservations with a computer that hosts an Industrial Graphics Server, a Plant SCADA Access Anywhere client or an RDS client, you also need to use the **Checkout** feature. Be aware, however, that you should shut down runtime before you remove a checkout for a computer.

---

- **Specific license types**

If your servers, control clients and view only clients have the same point count, you will not need to allocate specific licenses to each computer.

However, if you have purchased licenses with different point counts, or any licenses similar to the following, you will need to allocate them to the relevant computers using Enterprise Licensing's device reservation feature.

- Plant SCADA Control Client 250 Pack
- Plant SCADA View-Only Client 250 Pack
- Plant SCADA Driver, PowerConnect
- Plant SCADA Driver, KNX
- Plant SCADA Driver, BACnet
- Plant SCADA Driver, S7TCP
- Plant SCADA Driver, PSDirect MPI
- Plant SCADA Driver, PSDirect ETH
- Plant SCADA Driver, IEC60870.

This will avoid a license with a specific point count or support for a specific driver being automatically retrieved by a computer that does not require it.

For more information, see *Reserving Licenses* in the AVEVA Enterprise Licensing documentation in [AVEVA™ Help](#).

If you need to access a license that has been allocated to a computer that is currently inaccessible, you can use a "drop device" operation in AVEVA License Manager to release licenses. For more information, see *Dropping Devices* in the AVEVA Enterprise Licensing documentation.

## Launch Enterprise License Manager from Plant SCADA Studio

AVEVA™ Enterprise License Manager is a browser-based application that manages the AVEVA Enterprise License

Server. It allows you to view and organize the licenses for a Plant SCADA system.

You can use Plant SCADA Studio to launch Enterprise License Manager from any installed location within your Plant SCADA system.

#### To launch Enterprise License Manager from Plant SCADA Studio:

1. Go to the **Licensing** activity.
2. On the AVEVA Enterprise Manager tile, select **Launch**.

If it is the first time you have launched the License Manager from Plant SCADA Studio, the **Configure Enterprise License Manager URL** dialog will appear. By default, the following address will be used:

`http://localhost/AELicenseManager`

This URL will launch the local installation of License Manager. If you want to use a remote installation, enter the following:

`http://<ComputerName>/AELicenseManager`

Where:

`<ComputerName>` is the name of the computer that hosts the installation you would like to use.

---

**Note:** If you have configured License Manager to use a port number other than the default (port 80), you need to include the port number in the License Manager URL:

`http://<ComputerName>:<PortNumber>/AELicenseManager`

---

3. Select **Update**.

License Manager will launch from the specified location.

The next time you select the **Launch** button, License Manager will automatically open in a browser from this specified location.

If you want to change this location to a different computer, select the **Configure** button and enter a new URL in the **Configure Enterprise License Manager URL** dialog.

---

**Note:** For more information about working with Enterprise License Manager, see the chapter *Using AVEVA Enterprise Licensing* in the *AVEVA Enterprise Licensing* documentation in *AVEVA™ Help*.

---

## Update a Sentinel Key with CiUSAFE

If your Plant SCADA system uses Sentinel Licensing, there may be times when you need to update your USB keys (for example, when you upgrade to a new version of Plant SCADA). To do this, you use the CiUSAFE dialog box.

#### To update a Sentinel USB key with CiUSAFE:

1. Plug the key you would like to update in a local USB port.
2. Open Plant SCADA Studio.
3. On the **Activity Bar**, select **Licensing** from the menu.  
Or:  
Click the Licensing icon.



4. On the **Sentinel Key Update** panel, click **Launch**.

The CiUSAPE dialog box will appear. (See below for a description of the CiUSAPE dialog box fields.)

5. Retrieve the **Serial Number** for the key from CiUSAPE.
6. Visit the **License Activation** page on the [AVEVA Knowledge and Support Center](#), and enter the serial number in the USB Key Serial Number field.
7. Click **Submit**.  
If the key is validated, an authorization code will be generated.
8. In CiUSAPE, enter the generated code in the **Authorization Code** field.
9. Click **Update**.

CiUSAPE will display a **Return Code** to confirm if the update was successful. See the table below for an explanation of the return code values.

### [CiUSAPE Dialog Box Fields](#)

#### *Serial Number*

The serial number of the attached hardware key. If it does not appear automatically in CiUSAPE, you can read the number from the label on the hardware key. You need to enter the serial number into AVEVA's online License Generator to update the key.

#### *KeyID*

Each time you launch CiUSAPE, a Key ID will display in the KEYID field. You might need to provide the Key ID plus the serial number when updating the hardware key. This depends on the status of the key in the Plant SCADA license database, and you are prompted if the Key ID is required. Click Save KeyID to save the Key ID and serial number to a text file, which you can refer to when visiting the AVEVA web site.

#### *Authorization Code*

To update the hardware key, enter the 106-character authorization code. You are asked for this code once you have entered the Key ID and serial number, and your license and Customer Service agreement have been verified. Click Update to update your hardware key.

#### *Return Code*

The Return Code indicates the result of the key update:

0	The key was updated successfully.
1.3	Either the KeyID or the Authorization code you entered is invalid.
2	Either the KeyID or the Authorization code you entered has been corrupted.
4,16	Either the KeyID or the Authorization code you entered is invalid.
9	No hardware key could be found.

### See Also

#### [Licensing](#)

## Dynamic Point Count

Plant SCADA counts I/O device addresses dynamically at runtime.

The client process keeps track of the dynamic point count. When in multi-process mode, the client component will accumulate its own point count, which will include all the variable tags that are used by the process. This includes variable tags associated with the following:

- Events
- OPC DA Server
- Pages and Super Genies
- Cicode functions (TagRead, TagWrite, TagSubscribe, TagGetProperty and TagResolve)
- Any tag referenced by Cicode
- Reads or writes using DDE, ODBC, CTAPI or external OPC DA clients.

When in multi-process mode, the machine point count will be the point count of the client component, or the point count added up from each server component, depending on whichever is bigger. If the server license point count is 500 or greater, the client component point count is disregarded.

---

**Note:** If you are running a control client using a full license ([Client]FullLicense=1) and the point count of the license is 500 or more, then the point count is unlimited on that client and is disregarded.

---

A particular variable tag is only counted towards your point count the first time it is requested. Even if you have configured a certain tag on a particular page in your project, the variable tag will not be counted towards your point count unless you navigate to that page and request the data.

You should also be aware of the following:

- A dynamic point count is tag based, not address based. For example, two tags that use the same PLC address will be counted twice.
- If two trend tags use the same variable tag, it will be counted once. If two server components use the same tag(s) (say alarm and trend), the tags will not be counted twice when the point count gets totaled in the client process.
- Reading properties of a tag with [TagGetProperty\(\)](#) or [TagSubscribe\(\)](#) will cause that tag to be included in the point count, even if the value is not read.
- Persisted I/O (memory devices), local variables and disk I/O variable tags will not count towards the dynamic point count, unless they are written to by an external source (via OPC, DDE, ODBC, or CTAPI). For example, if you use an OPC client to write to a local variable, each local variable will be counted once the first time it is used.

---

**Note:**

- You can use the [CitectInfo\(\)](#) Cicode function or the General page in the Plant SCADA Kernel to determine the point count status of a client process.
  - You can specify the point count required by a client computer by using the [\[Client\]PointCountRequired](#) INI parameter.
- 

## See Also

[Demo Mode](#)

## Specify the Required Point Count for a Computer

The available point count for a Plant SCADA computer is determined by the type of license to which it is entitled. This is based on the role assigned to the computer by the [\[Client\]ComputerRole](#) parameter (which is typically set via the [Computer Role](#) page of the Setup Wizard).

Normally, the computer will get the first available matching point count. However, you can specify the point count required by a client computer by using the [\[Client\]PointCountRequired](#) INI parameter.

When any remote clients disconnect, the corresponding licenses that have been served to them can be reclaimed.

---

**Note:** A INI parameter is also available to control IP address aging. It is used to indicate how long to reserve a license for a given IP address in cases when a remote client connection is lost. This does not apply to full server licenses. The parameter is [\[General\]LicenseReservationTimeout](#).

---

## See Also

[Dynamic Point Count](#)

## Demo Mode

You can run Plant SCADA without the hardware key in demonstration (demo) mode. Demo mode lets you use every Plant SCADA feature normally, but with runtime and I/O restrictions.

In demo mode, you can run multiple processes on a single computer (with the networking model selected as "stand alone"), or in single process mode.

The following demonstration modes are available:

- 15 minutes with a maximum of 50,000 real I/O or calculated variables.
- 10 hours with a maximum of one dynamic real I/O or one calculated variable. This is useful for demonstrations using memory and disk I/Os.

Plant SCADA starts in this mode if no other licensing is available. If the system detects that you are using more than one real I/O point at runtime, it will swap to the 15 minutes demo mode.

---

**Note:** Writing to any tag through DDE, CTAPI, or ODBC will cause that tag to contribute to the dynamic point count even if it is a memory or disk I/O point. So if you write to more than one point through these interfaces, it will swap to the 15 minute demo mode.

---

## OPC UA Server and Industrial Graphics

The Industrial Graphics Server and OPC UA Server can both support demo mode, as long as they run on same computer as your Plant SCADA servers. In both cases, the setup is slightly different as the system needs to run in network mode. Remote Plant SCADA processes cannot connect to a server running in demo mode.

### To set up demo mode for an OPC UA Server and/or an Industrial Graphics Server:

1. Use the Setup Wizard to set the **Network Model** to "Networked" (see [Network Model](#)).
2. Configure your system to use encrypted communications (see [Enable Encryption](#)).

This means you will need to have a System Management Server configured and the Runtime Manager

running as a service.

3. Confirm that all server processes use an enabled network address that points to the local computer (see [Add Network Addresses](#)).

## See Also

[Licensing](#)

# Install Plant SCADA

This guide provides instructions for installing AVEVA™ Plant SCADA.

The Plant SCADA installer initially requires you to specify the "role" a computer will play within your control system. You can select from the three following options:

- **Development Workstation** — the computer will be used to build and test Plant SCADA projects.
- **Server** — the computer will be responsible for running a Plant SCADA system. It will perform tasks such as processing I/O device communications and managing alarms and trends.
- **Runtime-only Client** — the computer will be a display client for monitoring and controlling a system.

When you specify a role for a computer, a set of default components are automatically selected for installation. If required, you can customize the list of default selections. For example, if you select "Server", you can specify which type of server(s) you want to install.

If you are not familiar with Plant SCADA, see [Installation Information](#) for guidance on the installation process and some specific requirements.

If you need help to determine the role for a computer, see [What Should I Install?](#)

---

**Note:** If you want your system to use secured features such as encryption or deployment, you should not install Plant SCADA on the computer that acts as the domain controller within your Windows® domain.

---

When installation is complete, a dialog will appear asking if you would like to launch Configurator. This is a tool that is used to prepare a computer for its role within a Plant SCADA system. For example, it can be used to manage the certificates required to enable encrypted communications, or to map the local Windows® user groups to Plant SCADA's security roles. For more details, see [Configurator](#).

---

**Note:** The installation experience is not complete until after you exit Configurator. A restart request is likely to appear at this point. To avoid permission issues, you should close Configurator and confirm if a reboot is required before you run Plant SCADA.

---

## Upgrading

If you already have Plant SCADA installed, the Upgrading section of this guide provides information to help you update to the latest version. It describes:

- Instructions for both online and offline upgrade methods
- Supported upgrade paths
- The Plant SCADA Migration Tool.

See [Upgrade Plant SCADA](#).

**Note:** Due to significant component organizational changes in this release, you need to fully uninstall any existing versions of Plant SCADA and its associated components before you install Plant SCADA 2023 R2. You do not need to uninstall Platform Common Services.

## Installation Information

If you are not familiar with Plant SCADA, the following topics include information you need to know about the installation process and some specific requirements.

### Licensing Tools

Plant SCADA supports two different software licensing models:

- **Sentinel Licensing**

Sentinel Licensing uses physical USB keys that plug in to each computer in your Plant SCADA system.

To use Sentinel Licensing, you need to install the **Sentinel USB Driver**. It can be selected for installation via the **Extensions** branch of the products/components page of the Plant SCADA installer.

- **AVEVA™ Enterprise Licensing**

The [AVEVA™ Enterprise Licensing](#) system is a common platform solution that allows you to manage AVEVA Enterprise Software product licenses.

The system is comprised of a browser-based License Manager and a License Server that allow you to share and deliver licenses for your installed AVEVA applications. This distributed architecture enables centralized management of activated licenses and flexible topologies to support systems of any size.

The AVEVA Enterprise Licensing system is required for Industrial Graphics applications.

In both cases, Plant SCADA uses a Dynamic Point Count to determine if your system is operating within the limitations of your license agreement. This process tallies the number of I/O device addresses being used by the runtime system.

You can run Plant SCADA without a key in demonstration (demo) mode. Demo mode lets you use every feature normally, but with runtime and I/O restrictions (15 minutes with a maximum of 50,000 real I/O, or 10 hours with a maximum of one dynamic real I/O).

**Note:** In Plant SCADA 2023 and 2023 R2, Schneider Electric's Floating License Manager is no longer supported. Existing users of the Floating License Manager will need to reconfigure their system to use AVEVA Enterprise Licensing.

### Windows User Groups and Security Roles

You can only perform certain operations within Plant SCADA's engineering and runtime environments with relevant permissions. A page in **Configurator** allows you to grant these permissions to local and domain user groups.

Plant SCADA creates a set of security roles and local Windows user groups on a computer during installation. By default the local Windows user groups are associated with these security roles as members. Each role provides a different level of access to features, applications and project resources.

If you are installing Plant SCADA on a clean computer (one that has not hosted an earlier version), the following security roles and local Windows user groups will be created and mapped as follows:

Security Role	Windows User Group	Description
Configuration Users	SCADA.ConfigUsers	<p>Members of this role can run configuration tools (such as Plant SCADA Studio or Computer Setup Wizard) and start the runtime display client and server processes.</p> <p><b>Note:</b> It is recommended that members of this role only start runtime for development purposes and not in a production system environment.</p>
Runtime Users	SCADA.RuntimeUsers	Members of this role can run the runtime display client and make local CtAPI connections.
Server Users	SCADA.ServerUsers	<p>Members of this role can run Plant SCADA as a server process.</p> <p>If you are not running Plant SCADA as a service, add a member to this role who needs to run a Plant SCADA server (including a display client with [CtAPI]Remote enabled).</p>

If you choose to install Deployment Server components on a clean computer, additional security roles and local Windows user groups will be created and mapped as follows:

Security Role	Windows User Group	Description
Deployment Administrators	SCADA.DeploymentAdmins	Members of this role can add or remove client computers to/from deployment server. They can also perform upload and deploy operations.
Deployment Uploaders	SCADA.DeploymentUploaders	Members of this role can upload a new project version to the deployment server.
Deployment Users	SCADA.DeploymentUsers	Members of this role can deploy a new project to a connected deployment client computer.

If you choose to install an Industrial Graphics Server on a computer, additional security roles and local Windows user groups will be created and mapped as follows:

Security Role	Windows User Group	Description
Industrial Graphics Users	AIGUsers	<p>Members of this role can connect and authenticate with the Industrial Graphics Server with read-only access.</p> <p><b>Note:</b> You should avoid adding individual users to this security role. To allow authorization in a distributed system, only add domain groups to an Industrial Graphics security role.</p>
Industrial Graphics R/W Users	AIGUsersRW	<p>Members of this role can connect and authenticate with the Industrial Graphics Server. They will also be able to write to variable tags in a Plant SCADA system, provided the tags have been configured to support writes via the Write Roles property.</p> <p>See <a href="#">Enable Tag Writes for Industrial Graphics Applications</a>.</p> <p><b>Note:</b> You should avoid adding individual users to this security role. To allow authorization in a distributed system, only add domain groups to an Industrial Graphics security role. Members of this role can upload a new project version to the deployment server.</p>

If you are upgrading Plant SCADA from an earlier version, equivalent user groups will already exist on the computer. If the installer detects these existing groups, they will be retained. Only the security roles will be created and mapped to the existing local Windows user groups.

Security Role	Windows User Group Name	
	Clean installation	Upgrade from earlier version
Configuration Users	SCADA.ConfigUsers	Citect.Engineers
Runtime Users	SCADA.RuntimeUsers	Citect.LocalUsers
Server Users	SCADA.ServerUsers	Citect.ServerUsers
Deployment Administrators	SCADA.DeploymentAdmins	Asb.Deployment.AdminRole
Deployment Uploaders	SCADA.DeploymentUploaders	Asb.Deployment.UploadRole

Deployment Users	SCADA.DeploymentUsers	Asb.Deployment.DeployRole
------------------	-----------------------	---------------------------

**Note:** In an upgrade scenario, the Asb.Deployment.ReadRole user group is obsolete and not associated with any security role.

If required, you can use [Configurator](#) to change the associated members with these security roles (see [Modifying the Members of a Security Role](#)).

### Windows Firewalls

Following installation, your Windows Firewall settings will need to be adjusted so that Plant SCADA and its components are included in the list of authorized programs.

Plant SCADA can automatically adjust these settings for you. If Windows Firewall is operational on a computer, the installer will display a **Firewall** page. To allow Plant SCADA to adjust these setting for you, select **Yes, please modify Windows Firewall settings**.

This will add the following applications to the list of authorized programs.

Name	Program	Local Port
Citect SCADA Runtime (x64)	C:\Program Files (x86)\AVEVA Plant SCADA\Bin\Bin (x64)\Citect.exe	All ports
Plant SCADA Runtime	C:\Program Files (x86)\AVEVA Plant SCADA\Bin\Citect32.exe	All ports
Configurator	C:\Program Files (x86)\Common Files\Archestra\configurator.exe	All ports
Configurator 443	All programs	443
Configurator 80	All programs	80
LicenseServerPort	All programs	55555
LicenseServerAgentPort	All programs	59200

If during installation you select **No, I will modify Windows Firewall settings later**, you will need to manually configure an **Inbound Rule** for these components in Windows Firewall Advanced Settings (if they do not already exist).

See the topic [Firewall Settings and Plant SCADA](#) in the Runtime section of the Plant SCADA documentation.

**Note:** Plant SCADA networking and redundancy needs the option "Plant SCADA Runtime" to communicate through a Windows Firewall.

### Communication Drivers

Plant SCADA communicates with I/O devices such as PLCs, loop controllers, bar code readers, scientific analyzers, remote terminal units (RTUs), and distributed control systems (DCS). This communication takes place with each device through the implementation of a communications driver.

The installation process allows you to select from a set of individual drivers that are specific to your system and its I/O devices. There are also certain drivers that are necessary for Plant SCADA to function correctly. These will be installed automatically.

**Note:** With the release of Plant SCADA 2020 R2, a classification system for the driver portfolio was introduced to indicate the level of ongoing maintenance that customers can reasonably expect for each driver. In line with

---

these classifications, the number of drivers included by default with the Plant SCADA installation media has been reduced to a set of 'core' drivers that have undergone recent updates. All other drivers continue to be available for download from the **Communication Drivers** page at the **AVEVA Knowledge and Support Center** located at <https://softwaresupport.aveva.com/>.

---

### OPC Factory Server

The Plant SCADA installer allows you to install a copy of Schneider Electric's OPC Factory Server.

Based on the OPC protocol, OPC Factory Server enables Windows-based OPC client applications to communicate with a range of Schneider Electric's PLCs.

For more information, see [Install OPC Factory Server](#).

### Anti-virus Software Setup

If a computer uses an anti-virus software product, you need to be aware of the following implications for a Plant SCADA installation.

#### **WARNING**

##### **SYSTEM PERFORMANCE DEGRADATION**

The "on access" scan in anti-virus products can lock files used by Plant SCADA, usually having the effect of slowing Plant SCADA down whilst it waits for the scan of that file to finish.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

#### **CAUTION**

##### **INOPERABLE SYSTEM OR LOSS OF DATA**

In some extreme cases, anti-virus software may (incorrectly) detect certain patterns within data files as being viruses. Depending on the anti-virus configuration, this may result in files being relocated or deleted, resulting in data being lost or the system being inoperable.

**Failure to follow these instructions can result in injury or equipment damage.**

It is recommended that the following directories are excluded from scanning by any anti-virus products:

- Program Files installation directory (including files and sub directories)
- Data and Logs directories
- Any alarm server archive paths.

The above exclusions are recommended for "on access" or "real time" scans that run continuously and scan each file that is read from or written to.

### Maintaining System Currency

After you have completed the installation and configuration of Plant SCADA and deployed it as your production system, it is recommended that you keep your software up to date.

AVEVA will periodically publish software updates for Plant SCADA and advisories relating to safety, security and functionality. These are available from the Product Hub page of the AVEVA Knowledge & Support Center website at <https://softwaresupport.aveva.com>. We especially recommend that you nominate a person in your organization to refer, and subscribe, to the RSS feeds for Safety and Security, as well as the latest articles on the web site.

### Virtualization Host Support

You can run components of your Plant SCADA system in a virtual environment. The following virtualization environments are supported:

- Microsoft Hyper-V: based on the version of Windows.
- VMware Workstation: basic virtualization without High Availability and Disaster Recovery.
- VMware vSphere.

## What Should I Install?

The Plant SCADA installer directs you through a set of pages that will help you determine which components you require on a computer. Before you begin this process, you need to consider the questions below.

**Note:** If you are building a large system, it is important to carefully consider the overall topology of the system. For example, a distributed architecture, server redundancy, load sharing and clustered control scenarios can help to address your process requirements and benefit the overall performance of a system.

### Will you use the computer to configure a Plant SCADA project?

To configure a Plant SCADA project, you need to use a development workstation. An installation of a development workstation includes two fundamental sets of components:

- **Configuration Environment** — includes Plant SCADA Studio and other tools used to create and configure a project.
- **Runtime** — includes components used to run a project.

See [Install a Development Workstation](#).

### Do you want to use AVEVA™ Industrial Graphics?

AVEVA Industrial Graphics is a thin client solution for the delivery of HTML5 graphics to desktop browsers and mobile devices. It provides a common graphics technology that can be used across AVEVA's SCADA and HMI solutions.

Clients access AVEVA Industrial Graphics via an Industrial Graphics Server, a dedicated computer that authenticates client details and provides access to the graphical and functional content.

If you want to use Industrial Graphics, you will need to set up an Industrial Graphics Server. You will also need to configure a Deployment Server in your network to deploy content to the Industrial Graphics Server.

An AVEVA Enterprise License Server also needs to be part of your network to provide licenses for client access.

See [Install an Industrial Graphics Server](#).

### Do you need to use AVEVA™ Enterprise Licensing?

AVEVA Enterprise Licensing is a centralized license management solution that removes the need for dongles and increases license security through the use of activated licenses.

If you are setting up a new Enterprise Licensing system, you should review the following topics to determine your installation requirements.

- [AVEVA™ Enterprise Licensing](#) describes the required components in a system.
- [AVEVA™ Enterprise Licensing Topologies](#) describes the recommended system topologies.

If you are implementing a distributed system, we recommend creating a network diagram to determine where

you need to install the **License Manager** and **License Server** components.

#### **Do you want Plant SCADA to act as a server for OPC UA client applications?**

Setting up an OPC UA Server allows Plant SCADA's runtime system to function as a server to OPC UA client applications. For more information, see [Install an OPCUA Server](#).

#### **Does your system require encrypted communications?**

Communications between Plant SCADA processes, computers and CtAPI can be encrypted. This is required if:

- You need to use trusted communications for your runtime system computers.
- You want to connect an OPC UA server to your system.
- You want to connect an Industrial Graphics Server to your system.

To enable encryption, a **System Management Server** is required to manage the certificates used to support trusted communications. A System Management Server is set up using a tool called (see [Set Up a System Management Server](#)).

---

**Note:** To use encrypted communication, your Plant SCADA servers need to be running as a service.

If you already have a System Management Server set up, you will need access to the certificate that will connect your new installation to the System Management Server. This can be also be configured using Configurator. See [Connect a Computer to a System Management Server](#).

#### **Do you need to run Plant SCADA as a service?**

Plant SCADA can be run as a Windows™ service, allowing for unattended operation of a system's servers.

To create an encrypted connection, your Plant SCADA servers need to be running as a service. This is required when:

- You are running with encryption enabled. This includes running in mixed mode.
- You want to connect an OPC UA server to your system.
- You want to connect an Industrial Graphics Server to your system.

If you are using Plant SCADA's **Deployment** feature, you also need to run as a service under the following circumstances:

- When a deployment client is configured on a computer, and you want to push project updates using "Force" mode.
- If you are using encrypted communications, and the current user on a deployment client does not have required privileges.

To configure Runtime Manager to run as a service, see [Configure a Runtime Computer for Encryption](#).

#### **Do you want to use Plant SCADA's Deployment feature?**

Plant SCADA's Deployment functionality allows you to send runtime files to specific computers in a Plant SCADA system. This simplifies the process of distributing project changes across multiple computers.

If you want to use this feature, a **Deployment Server** is required. This is where you store versions of a project for distribution. Specific components need to be installed for a Deployment Server (and any Deployment Clients that connect to it).

See [Install a Deployment Server](#).

---

**Note:** A Deployment Server uses encrypted communications, so you will need to connect the host computer

---

(and any connected clients) to an existing System Management Server.

#### Do you just need to install a runtime client?

If you are setting up a computer that will only operate as a runtime client, you have the option to install just the Plant SCADA runtime components.

See [Install a Runtime-only Client](#).

---

**Note:** If you want to install a standalone Plant SCADA system for training or evaluation purposes, [Install a Development Workstation](#).

---

## Configurator

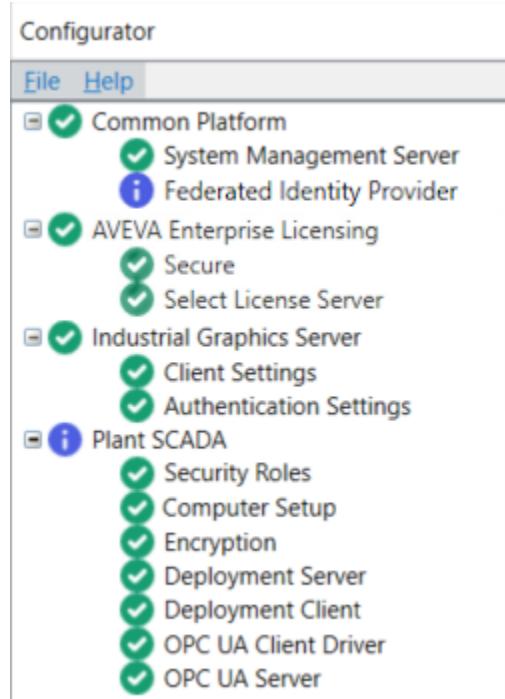
Configurator is a tool you can use to prepare a computer for its role within a Plant SCADA system.

When a Plant SCADA installation is complete, a dialog will appear asking if you would like to launch Configurator. In most cases, you will require Configurator to complete the setup of a computer.

To access Configurator at other times:

- Go to the Windows® **Start** menu and locate **AVEVA | Configurator**.  
Or:
- Go to the **Projects** activity in Plant SCADA Studio and select **Configurator** from the **Setup Wizard** drop-down menu.

Configurator presents a set of pages that can be accessed via a directory to the left of the interface.



For AVEVA **Common Platform** components, the following page is included:

- **System Management Server** — used to configure a System Management Server.  
See [Configure a System Management Server](#).

---

**Note:** The **Federated Identity Provider** will appear under the Common Platform components. This is not used by Plant SCADA. If required, you can set the **Identity Provider** field to "None". This will present a green tick for the page in the Configurator tree.

---

For **AVEVA Enterprise Licensing**, the following pages are included:

- **Secure** — indicates if the Enterprise License Server is running in Secure Mode.
- **Select License Server** — configures or connects to an Enterprise License Server.  
See [Use Configurator to Set Up an Enterprise License Server](#).

For an **AVEVA Industrial Graphics Server**, the following pages are included:

- **Client Settings**
- **Authentication Settings**  
See [Use Configurator to Set Up an Industrial Graphics Server](#) for more information.

For **Plant SCADA**, the following pages are included:

- **Security Roles** — used to map Plant SCADA Roles to Windows® user groups, Domain Groups and Individual users.  
See [Security Roles](#).
- **Computer Setup** — used to configure environment settings for a runtime computer, including a server password and some deployment settings.  
See [Use Configurator to Set Up a Runtime Server](#).
- **Encryption** — used to encrypt communications between Plant SCADA processes, computers and CTAPI applications.  
See [Enable Encryption](#).
- **Deployment Server** — used to set up a Deployment Server for the distribution of Plant SCADA project versions.  
See [Configure a Deployment Server](#).
- **Deployment Client** — used to set up a Deployment Client.  
See [Configure a Deployment Client](#).
- **OPC UA Client Driver** — used to manage secure communications with OPC UA devices connected to Plant SCADA via an external OPC UA server. This relates to an installed driver, not Plant SCADA's OPC UA Server.  
See [OPC UA Client Driver](#).
- **OPC UA Server** — allows Plant SCADA's runtime system to function as a server to third-party OPC UA client application.  
See [Use Configurator to Set Up an OPC UA Server](#).

The icons next to each page indicate the current status of the associated settings, according to the following legend.



Each page includes the **Configuration Messages** panel to advise you of the outcome of any adjustments you make. You can also [View the Configurator Messages List](#).

## View the Configurator Messages List

The Configurator Messages List allows you to view and export all of Configurator's current messages.

### To view the Configurator Messages List:

- On a Configurator page, click the **All Messages** button.

The Messages List dialog box displays all current messages in a tabular format. Click on a column heading to sort the table according to the column. Click again to toggle between ascending or descending order.

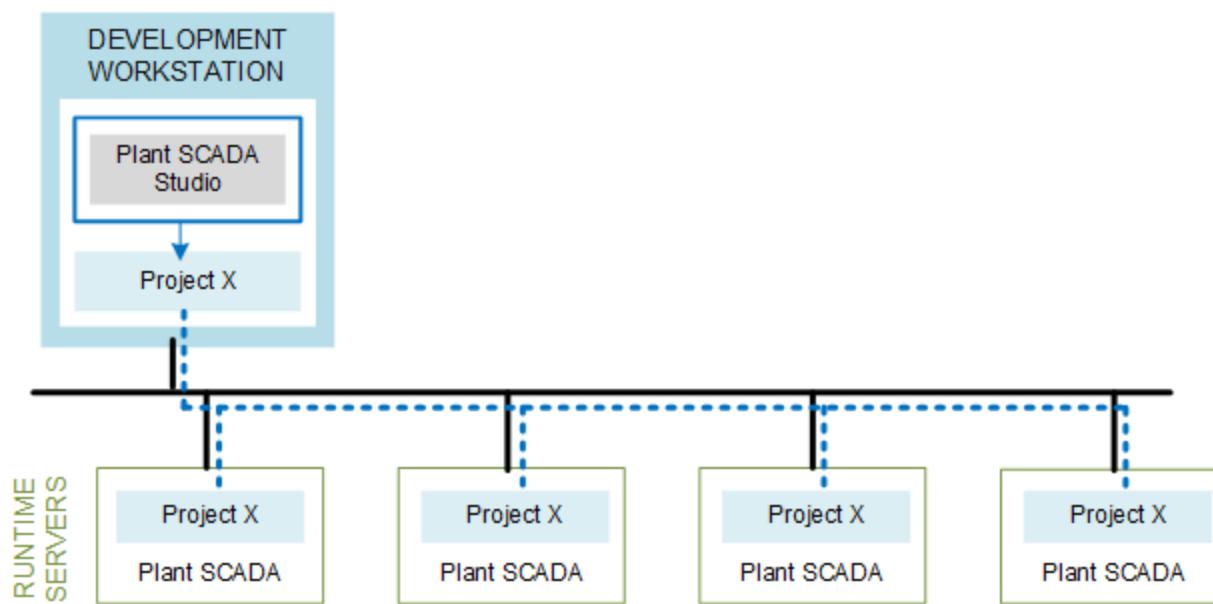
### To export the Configurator Messages List:

1. Click the **Export to file** button.
2. Use the Save As dialog to save the file.

## Install a Development Workstation

A development workstation uses configuration tools such as **Plant SCADA Studio** and **Graphics Builder** to create a Plant SCADA project.

A project includes runtime content and configuration information. You need to distribute a copy of the project to each computer in a Plant SCADA runtime system.



When you chose to install a development workstation, the following components are selected by default:

- AVEVA Plant SCADA
  - Configuration Environment
  - Runtime
  - Deployment Server
- AVEVA Plant SCADA Communications Drivers
  - A set of commonly used drivers
- AVEVA Industrial Graphics
  - Industrial Graphics Web Server
- Extensions
  - Software Update by Schneider Electric

The following licensing options are also selected by default.

- AVEVA Enterprise Licensing
  - AVEVA Enterprise License Manager
  - AVEVA Enterprise License Server.

See the section *Licensing Tools* in the topic [Installation Information](#) to determine your licensing requirements.

If required, you can use the **Customize the components that will be installed** page to modify the list of selected components.

## See Also

- [Development Workstation Installation Requirements](#)
- [Development Workstation Installation Procedure](#)
- [Use Configurator to Set Up a Development Workstation](#)

## Development Workstation Installation Requirements

This topic describes the hardware and software requirements for a Plant SCADA development workstation.

**Note:** Before you install Plant SCADA, it is recommended that you install the latest updates from Microsoft® for your operating system and system software.

### Hardware Requirements

The hardware requirements for a development workstation are as follows, based on the size of the system you are configuring.

	Compact System <1,500 pts	Small System <15,000 pts	Medium System <50,000 pts	Large System <500,000 pts	Huge System >500,000 pts
CPU PassMark®	2000	2000	4250	4250	8000
Cores	2	2	4	4	4
RAM	8 GB				
HDD	10 GB	20 GB	50 GB	50 GB	100 GB
Graphics	DirectX 11 or later with WDDM 1.0 Driver. 128 MB of dedicated VRAM.	DirectX 11 or later with WDDM 1.0 Driver. 128 MB of dedicated VRAM.	DirectX 11 or later with WDDM 1.0 Driver. 128 MB of dedicated VRAM.	DirectX 11 or later with WDDM 1.0 Driver. 128 MB of dedicated VRAM.	DirectX 11 or later with WDDM 1.0 Driver. 128 MB of dedicated VRAM.
Screen Resolution	1920 x 1080				
Network	100 Mb				

The suggested disk space requirements for the HDD is an estimate only and includes:

- A full Plant SCADA installation including optional components and documentation
- Project assets for the specified system size.

If the deployment feature is being used, the HDD needs to have the required space for the number of configured project versions + 2. You should remove unused project versions periodically to reduce HDD usage.

An SSD is recommended for engineering machines for better performance. If a non-SSD is used, select a minimum RPM of 7200.

## Operating System Requirements

The following operating systems are supported (64-bit only):

- Windows 11 21H2 (or later)
- Windows 10 21H2 (or later)
- Windows 10 LTSC 1607 (or later)
- Windows Server 2016
- Windows Server 2019
- Windows Server 2022

Microsoft .NET Framework 4.8 is required. It will be installed by Plant SCADA if not detected.

When .NET installation is complete, you may be prompted to restart your computer. If this occurs, you should initiate a restart before you continue.

## See Also

[Development Workstation Installation Procedure](#)

[Use Configurator to Set Up a Development Workstation](#)

## Development Workstation Installation Procedure

This table below describes the pages included in the Plant SCADA installer and the settings required to install a development workstation.

Use the **Back** and **Next** buttons to navigate through the pages.

Installer Page	Notes
Installation Documentation	If required, view the available documents.
License Agreement	Accept the terms of the License Agreement.
What is the role of this computer?	Select <b>Development Workstation</b> .
The selected products/components will be installed/upgraded.	By default, the following components are selected when you install a Development Workstation. <ul style="list-style-type: none"><li>• Platform Common Services</li><li>• AVEVA Plant SCADA<ul style="list-style-type: none"><li>• Configuration Environment</li><li>• Runtime</li><li>• Deployment Server</li></ul></li><li>• AVEVA Plant SCADA Communications Drivers<ul style="list-style-type: none"><li>A set of commonly used drivers.</li></ul></li></ul>

Installer Page	Notes
	<ul style="list-style-type: none"><li>• AVEVA Industrial Graphics<ul style="list-style-type: none"><li>• Industrial Graphics Web Server</li></ul></li><li>• AVEVA Enterprise Licensing<ul style="list-style-type: none"><li>• AVEVA Enterprise License Manager</li><li>• AVEVA Enterprise License Server.</li></ul></li><li>• Extensions<ul style="list-style-type: none"><li>• Software Update by Schneider Electric</li></ul></li></ul> <p>You can use this page to customize the list of selected components. For example, you can select any additional communication drivers, OPC UA components, or the Project DBF Add-In for Excel.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"><li>• The AVEVA Enterprise Licensing system only requires one License Server. If you are going to connect to an existing License Server, you do not need to select this option.</li><li>• Plant SCADA 2023 R2 installs version 4.0 of AVEVA Enterprise Licensing. If your system includes computers that are not upgraded to AEL 4.0, you will need to install <b>AVEVA Enterprise Licensing Server Legacy Support</b> on your Enterprise License Server. This enables backwards compatibility for clients nodes that are running an earlier version of AEL.</li></ul>
Access Restriction	<p>This page provides information about the local Windows groups the installer will create to manage security (see the section <i>Windows User Groups and Security Roles</i> in the topic <a href="#">Installation Information</a>).</p> <p><b>Note:</b> This page will only appear if you are installing Plant SCADA on a clean computer (one that has not hosted an earlier version). If you are upgrading from an earlier version, equivalent user groups will already exist on the computer.</p> <p>If you want your current Windows user account to be included in the groups that are being created, select <b>Add the current Windows user to these groups</b>.</p> <p>A dialog will let you know that the group membership changes will not occur until after a restart.</p>
Firewall	If the installer detects that the computer has Windows Firewall enabled, you will be asked if you would like the installer to modify your settings.

Installer Page	Notes
	<p>If you select <b>Yes</b>, Plant SCADA Runtime will be added to the list of authorized programs.</p> <p>See the section <i>Windows Firewalls</i> in the topic <a href="#">Installation Information</a>).</p>
Customize the installation location	<p>This page identifies the destination folders for:</p> <ul style="list-style-type: none"><li>• The selected Plant SCADA program files.</li><li>• The Plant SCADA User and Data folders.</li></ul> <p>You can modify the folder locations by clicking the <b>Change</b> buttons and selecting alternative locations.</p> <p>You can also <b>Reset</b> the default locations.</p> <p><b>Note:</b> The installation folder and the 'User and Data' folder cannot be the same, or a sub-directory of one another. You should also avoid selecting your Windows user folder.</p>
Ready to Install the Program	<p>This page lists the components that will be installed based on your selections.</p> <p>If required, use the <b>Back</b> button to make any modifications.</p> <p>Click <b>Install</b> to start the installation process.</p>

When installation is complete, a dialog will appear asking if you would like to launch [Configurator](#). This may take a few minutes if you have installed some external components (for example, AVEVA Enterprise Licensing components).

---

**Note:** The installation experience is not complete until after you exit Configurator. A restart request is likely to appear at this point. To avoid permission issues, you should close Configurator and confirm if a reboot is required before you run Plant SCADA.

---

## See Also

[Use Configurator to Set Up a Development Workstation](#)

## Use Configurator to Set Up a Development Workstation

When your Plant SCADA installation is complete, a dialog appears to inform you that some components require configuration before use. Click **Configure** to launch [Configurator](#).

The following descriptions of the Configurator's pages will help you determine which settings you need to consider.

## Computer Setup

The Computer Setup page allows you to specify three key settings for a Plant SCADA computer.

- **Server Authentication** — specifies the password the Plant SCADA server processes use to communicate with each other. If the computer will run a server process, you will need to configure the server password.
- **Project Run Path** — instructs Runtime Manager to run the project currently selected in Plant SCADA Studio, or a deployed project from the Deployment Server.
- **Runtime Manager Configuration** — allows you to run Runtime Manager as a service, which may be required for a Deployment Client.

## Security Roles

To manage the usage of Plant SCADA and restrict access to its sensitive components, only users with relevant permissions can perform certain security-related operations.

Each security role has specific permissions to access different features, software applications and project resources. Using the **Security Roles** page in the Configurator, you can assign the security roles to a user account or a group account. For more information, see [Security Roles](#).

## Encryption

If your Plant SCADA system uses encrypted communications, you will have to enable encryption for each new installation of Plant SCADA. This is achieved using Configurator's **Encryption** page (see [Enable Encryption](#)).

As a prerequisite, the computer needs to be registered with a System Management Server. This is achieved using Configurator's **System Management Server** page (see below).

## System Management Server

This page allows you to set up the computer as a System Management Server (see [Configure a System Management Server](#)).

If your Plant SCADA system already has a System Management Server, you can use this page to establish a trusted connection to it (see [Connect a Computer to a System Management Server](#)).

## Additional Components

Some pages will appear in the Configurator if you chose to install additional components. Use the links below for more information on how to configure these pages.

- **Deployment Server** — see [Configure a Deployment Server](#).
- **Deployment Client** — see [Configure a Deployment Client](#).
- **OPC UA Server** — see [Use Configurator to Set Up an OPC UA Server](#).
- **OPC UA Client Driver** — see [OPC UA Client Driver](#).
- **Industrial Graphics Server** — see [Use Configurator to Set Up an Industrial Graphics Server](#).

---

**Note:** The installation experience is not complete until after you exit Configurator. A restart request is likely

---

to appear at this point. To avoid permission issues, you should close Configurator and confirm if a reboot is required before you run Plant SCADA.

---

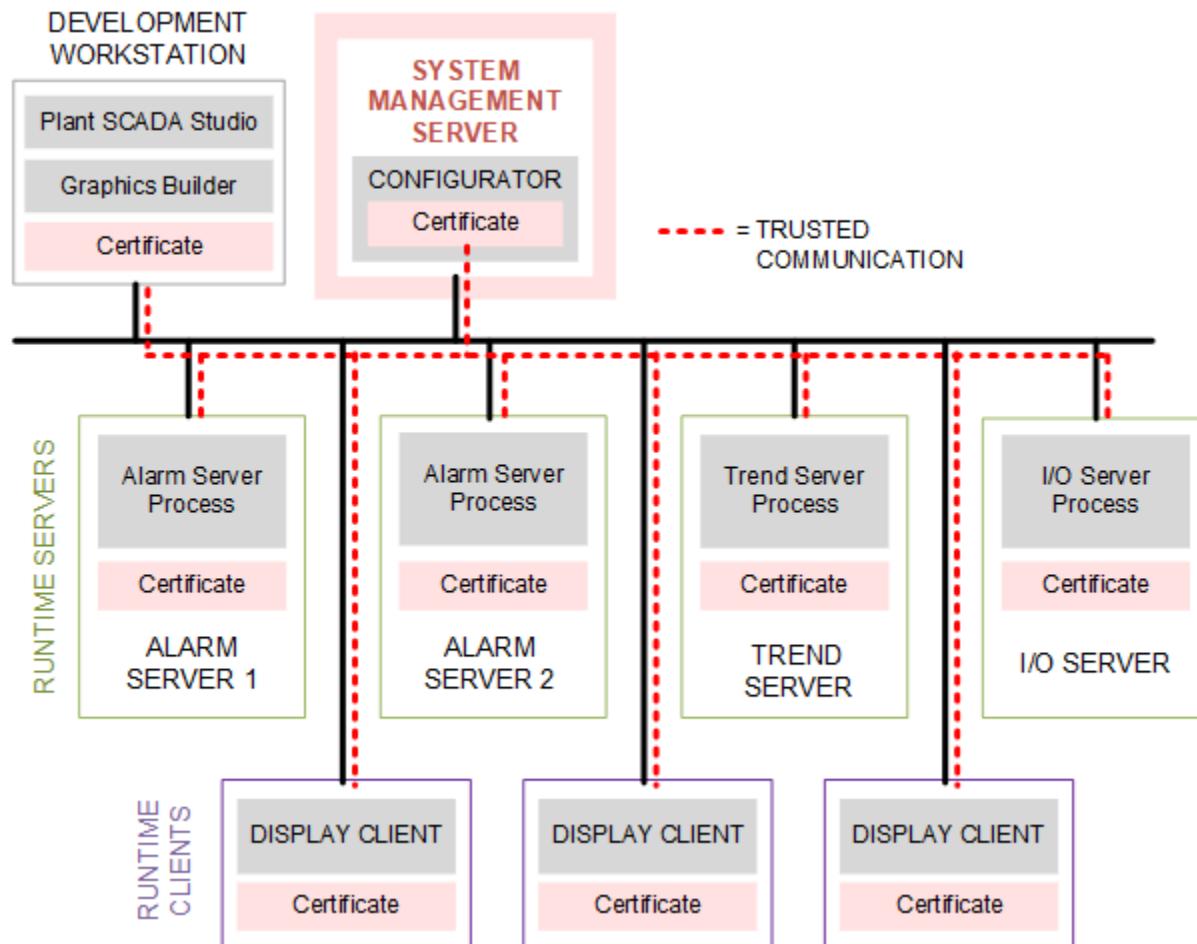
## See Also

- [Development Workstation Installation Requirements](#)
- [Development Workstation Installation Procedure](#)

## Set Up a System Management Server

A System Management Server ensures Plant SCADA components only work on trusted computers. It also enables support for end-to-end encryption (which is highly recommended).

Only one of the computers in a Plant SCADA system can be set up as a System Management Server.



A System Management Server does not need to be on a dedicated computer. In a typical Plant SCADA system that is also using deployment, it is recommended that the System Management Server is on the same computer as the Deployment Server.

The components required to enable encryption are included with every installation of Plant SCADA. You need to use [Configurator](#) to specify which computer in your system will act as the System Management Server.

The certificates required to enable trusted communications can then be created on this computer using

Configurator. Alternatively, you can also use your own third party certificates if required.

All other computers in your system then use this certificate to acquire validation from the System Management Server.

## See Also

[Configure a System Management Server](#)

## Configure a System Management Server

Plant SCADA needs post-installation configuration in order to use encrypted communications via a **System Management Server**. Encryption is a requirement for components such as a Deployment Server, an Industrial Graphics Server or an OPC UA Server.

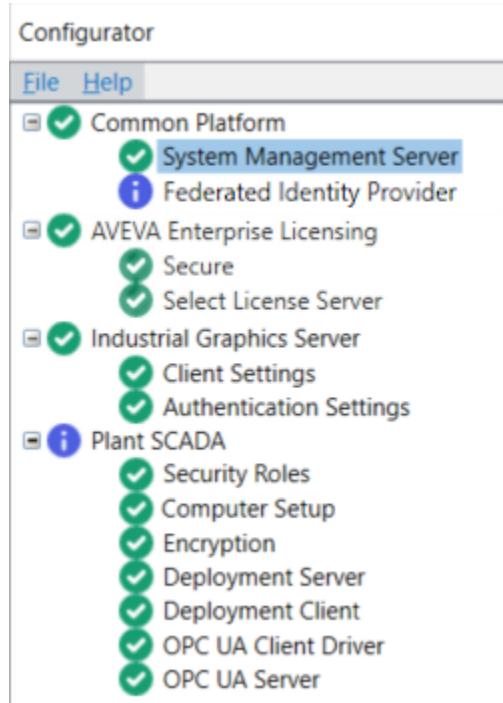
Only one of the computers in the network can be configured as a System Management Server. This is achieved using [Configurator](#).

You can then use Configurator to establish a trust relationship between one or more computers running Plant SCADA and its services. This is achieved by configuring a certificate on the System Management Server that can then be distributed to other computers to enable encrypted communications.

Certificates may be generated automatically on the System Management Server, or provided by a system administrator or IT department (see [Use Externally Provided Certificates for Encryption](#) in the Plant SCADA documentation).

### To configure the System Management Server:

1. Start the [Configurator](#).
2. In the left pane, select **Common Platform | System Management Server**.



The following page is displayed:

Machines running AVEVA software must be configured to trust each other so that encrypted communications can be utilized. This is done by connecting them to a System Management Server.

- Connect to an existing System Management Server.

- This machine is the System Management Server.

There should only be one System Management Server in your topology for all AVEVA products. All other machines should be configured to connect to this System Management Server.

When configuring other machines, you should validate that the security code shown in the Configurator matches:

[Details...](#)

- No System Management Server configured. (NOT RECOMMENDED)

This option also allows you to remove any existing certificates that were managed by the System Management Server.

You can connect to an existing System Management Server or configure a new System Management Server by selecting one of the first two options, respectively. When you click Configure, a certificate and the web ports to use for communication are configured. To modify these configurations, click Advanced.

Advanced

3. Select **This machine is the System Management Server**. Review the notes on the screen before you start the configuration.

**Note:** The machine on which you want to install the System Management Server needs to be configured to have a static IP address.

- #### 4. Click **Configure**.

**Note:** If you need to change the port used by a System Management Server see the topic *Advanced Configuration for a System Management Server* in the Plant SCADA documentation.

5. On successful configuration, the message "Device configuration completed" is displayed. The security code is displayed in the **Configurator** as shown below. To view more information about the certificate, click **Details**.

## Configuration Messages

Certificates are configured.

Connecting to the System Management Server 'https://[REDACTED].com:443'.

#### **Registering the device.**

The device is registered.

Device configuration completed.

If the configuration is unsuccessful, check the [Log Viewer](#).

You can access the Log Viewer by selecting **AVEVA | Operations Control Management Console** in the Windows **Start** menu.

Alternatively, view the error messages in the System Management Console.

6. Click **Close** to exit the **Configurator**.

---

**Note:** After you have configured a System Management Server, it is recommended that you either sign out or reboot the computer to ensure all settings are applied correctly.

---

## See Also

[Connect a Computer to a System Management Server](#)

## Connect a Computer to a System Management Server

Computers running Plant SCADA need to connect to the **System Management Server** to use encrypted communication enabled by a certificate.

If you do not have a System Management Server configured, refer to the section [Configure a System Management Server](#) for instructions. AVEVA applications need to be configured to trust each other to use encrypted communications.

To connect a computer to the System Management Server, you use the System Management Server page in **Configurator**. You can launch **Configurator** at the completion of the installation procedure, or from the Windows™ Start menu.

To establish a connection, the current user on the remote computer needs to be a member of either the "aaAdministrators" or the "Administrators" group on the machine where the System Management Server is installed. This is necessary to establish a trust relationship, it is not a runtime requirement.

---

**Note:** For a Deployment Server, you need to authorize the connection to a System Management Server from Configurator's Deployment Server page. See [Configure a Deployment Server](#).

---

### To connect a computer to the System Management Server:

1. In the panel on the left side of the **Configurator** select **Common Platform, System Management Server**.
2. Select the option **Connect to an existing System Management Server**.
3. In the field below, select the name of the System Management Server you want to use from the drop-down list.

The Configurator uses SSDP (Simple Service Discovery Protocol) to detect a System Management Server on a remote computer. If SSDP has been blocked on your network, nothing will appear in the drop-down list.

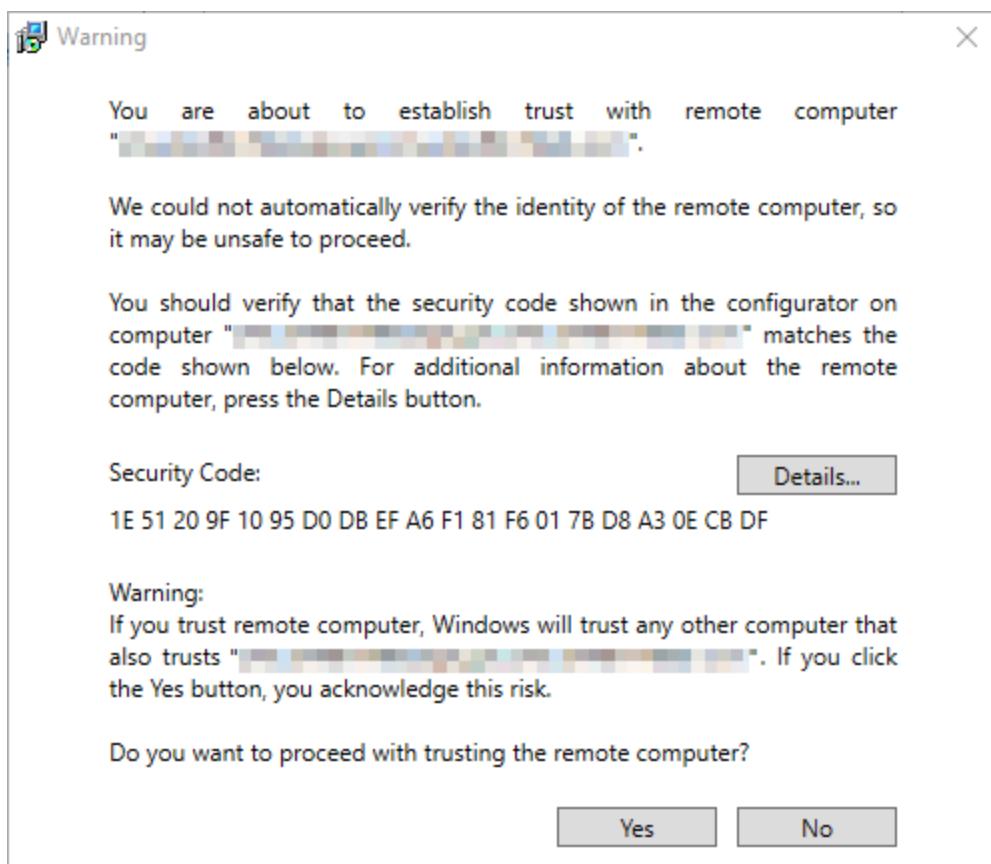
If the required computer name does not appear in the list, you can manually enter a hostname or Fully Qualified Domain Name (FQDN) for the computer on which the System Management Server is installed. Connecting via an FQDN will require a connection to a domain server.

---

**Note:** The Configurator interface refers to "redundant SSO servers". This feature is not supported by Plant SCADA, you cannot configure a computer as a redundant SSO server.

---

4. Click **Configure**. The root certificate is downloaded, and the following message is displayed.



5. Review the message carefully before you click Yes. If you select No, the configuration process will be canceled.

A dialog box may appear prompting you to log on to the System Management Server. This will occur if the current user is not a member of either the "aaAdministrators" or the "Administrators" group on the machine where the System Management Server is installed. If this happens, enter the credentials for a member of one of these groups and click OK.

The Configuration Messages area displays the steps in the configuration process and the progress.

If the configuration is unsuccessful, view details of the error messages in the System Management Console. See the topic [Troubleshooting Certificate Error Messages](#).

Alternatively, check the [Log Viewer](#).

You can access the Log Viewer by selecting **AVEVA | Operations Control Management Console** in the Windows **Start** menu.

6. Click **Close** to exit the **Configurator**.

**Note:** After you have connected a computer to a System Management Server, it is recommended that you either sign out or reboot the computer to ensure all settings are applied correctly.

Once you have connected to a System Management Server, you can modify the configuration settings by selecting **Advanced**. Refer to the topic [Advanced Configuration for a System Management Server](#) in the Plant SCADA documentation for more information.

**Note:** If you need to connect a client computer that is not on the same domain as the System Management Server (for example, it is part of a Workgroup), you need to configure the **Primary DNS Suffix** on the client to identify the domain on which the System Management Server resides. Refer to the Windows documentation for

---

instructions on how to specify a Primary DNS Suffix for a computer.

---

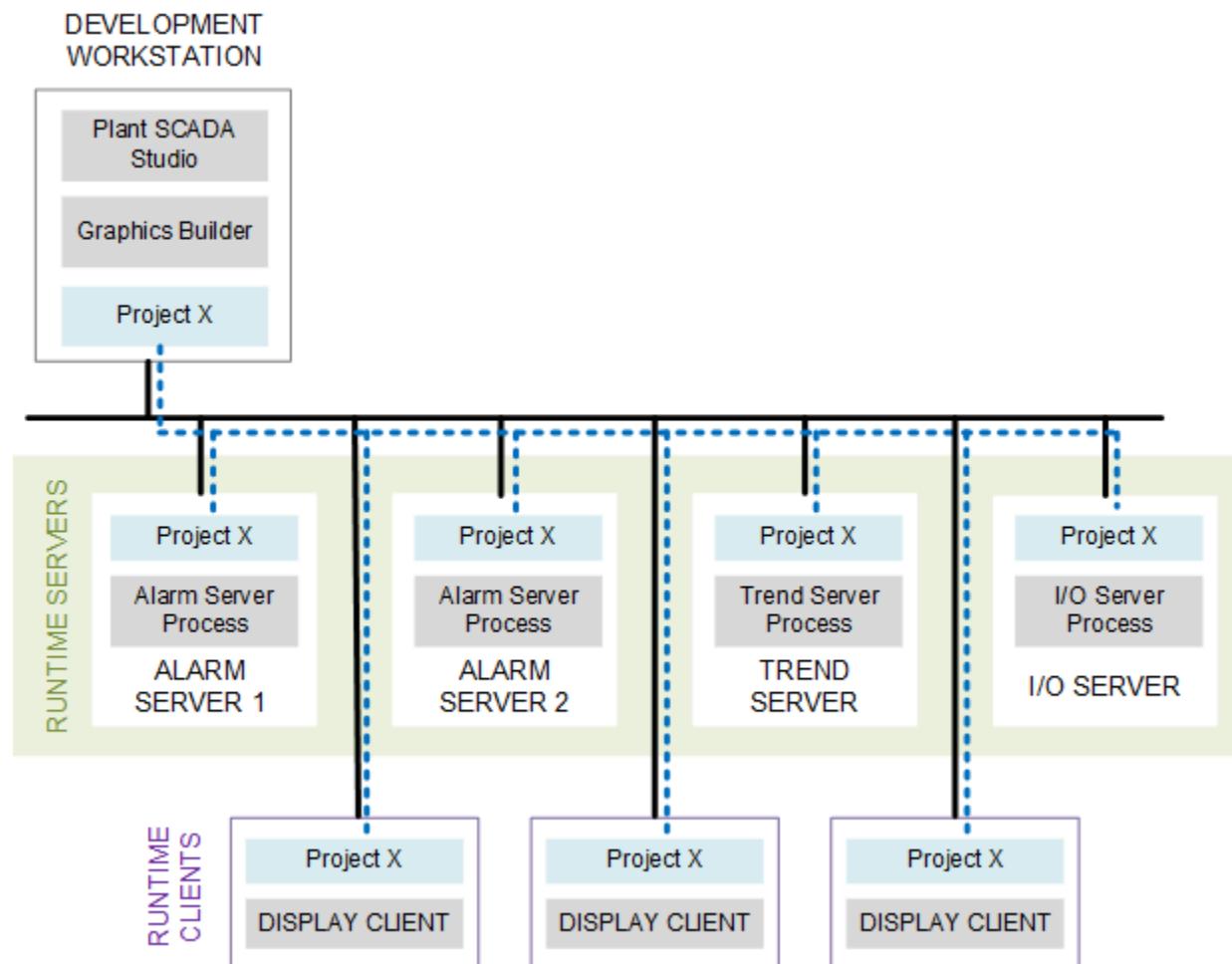
## Install a Runtime Server

A Runtime Server is any computer that hosts a server process in support of the runtime system. For example, a Runtime Server can perform one or more of the following functions:

- Alarms server - evaluates the conditions that define an alarm
- Trends server - manages the accumulation and logging of trend information
- Reports server - generates configured reports
- I/O device server - manages communication with field devices.

The server processes that are hosted by a runtime server are not configured during the installation process. These are defined in a project using the **Topology** activity in Plant SCADA Studio.

These processes can be installed on a single computer, or separated onto multiple computers across a distributed system. See the topic *Typical System Scenarios* in the Plant SCADA documentation for examples of how you can distribute server processes.



## See Also

- [Runtime Server Installation Requirements](#)
- [Runtime Server Installation Procedure](#)
- [Use Configurator to Set Up a Runtime Server](#)

## Runtime Server Installation Requirements

This topic describes the hardware and software requirements for a computer that will act as a Runtime Server.

Before you install Plant SCADA, it is recommended that you install the latest updates from Microsoft® for your operating system and system software.

## Hardware Requirements

The hardware requirements for a Runtime Server are as follows, based on the size of your system.

	<b>Compact System</b> <b>&lt;1,500 pts per server</b>	<b>Small System</b> <b>&lt;15,000 pts per server</b>	<b>Medium System</b> <b>&lt;50,000 pts per server</b>	<b>Large System</b> <b>&lt;200,000 pts per server</b>
CPU PassMark®	2000	2000	4250	8000
Cores	2	2	4	8
RAM	8 GB	8 GB	8 GB	16 GB
HDD	10 GB	20 GB	50 GB	100 GB
Graphics	DirectX 11 or later with WDDM 1.0 Driver. 128 MB of dedicated VRAM.	DirectX 11 or later with WDDM 1.0 Driver. 128 MB of dedicated VRAM.	DirectX 11 or later with WDDM 1.0 Driver. 128 MB of dedicated VRAM.	DirectX 11 or later with WDDM 1.0 Driver. 128 MB of dedicated VRAM.
Screen Resolution	1920 x 1080	1920 x 1080	1920 x 1080	1920 x 1080
Network	100 Mb	100 Mb	100 Mb	100 Mb

**Note:** The recommendations above are for a single cluster and server only running I/O, alarms, trends and reports. System resources for CPU and memory should be increased when using clustering, or if there is high rate of change of data (I/O or alarms).

The suggested disk space requirements for the HDD is an estimate only and includes:

- Runtime components.

- Compiled project.
- 20% of the I/O trending with a change on average every 10 seconds, 24 x 7 for 3 months.
- Alarm changes equal to the number of I/O changing per day.

If the deployment feature is being used, the HDD needs to have the required space for the number of configured project versions + 2. You should remove unused project versions periodically to reduce HDD usage.

An SSD is recommended for engineering machines for better performance. If a non-SSD is used, select a minimum RPM of 7200.

## Operating System Requirements

The following operating systems are supported (64-bit only):

- Windows 11 21H2 (or later)
- Windows 10 21H2 (or later)
- Windows 10 LTSC 1607 (or later)
- Windows Server 2016
- Windows Server 2019
- Windows Server 2022

Microsoft .NET Framework 4.8 is required. It will be installed by Plant SCADA if not detected.

When .NET installation is complete, you may be prompted to restart your computer. If this occurs, you should initiate a restart before you continue.

## See Also

[Runtime Server Installation Procedure](#)

[Use Configurator to Set Up a Runtime Server](#)

## Runtime Server Installation Procedure

This section lists the pages displayed by the Plant SCADA installer and the settings required for a Runtime Server. Use the **Back** and **Next** buttons to navigate through the included pages.

Installer Page	Notes
Installation Documentation	If required, view the available documents.
License Agreement	Accept the terms of the License Agreement.
What is the role of this computer?	Select <b>Server</b> .
Select the server(s) you want to install	Select <b>Runtime Server</b> . If required, you can also select other server types.

Installer Page	Notes
The selected products/components will be installed/upgraded.	<p>By default, the following components are selected when you install a Runtime Server.</p> <ul style="list-style-type: none"><li>• Platform Common Services</li><li>• AVEVA Plant SCADA<ul style="list-style-type: none"><li>• Runtime</li></ul></li><li>• AVEVA Plant SCADA Communications Drivers<ul style="list-style-type: none"><li>A set of commonly used drivers</li></ul></li><li>• Extensions<ul style="list-style-type: none"><li>• Software Update by Schneider Electric</li></ul></li></ul> <p>You can use this page to customize the list of components selected for installation.</p>
Access Restriction	<p>This page provides information about the local Windows groups the installer will create to manage security (see the section <i>Windows User Groups and Security Roles</i> in the topic <a href="#">Installation Information</a>).</p> <p><b>Note:</b> This page will only appear if you are installing Plant SCADA on a clean computer (one that has not hosted an earlier version). If you are upgrading from an earlier version, equivalent user groups will already exist on the computer.</p> <p>If you want your current Windows user account to be included in the groups that are being created, select <b>Add the current Windows user to these groups</b>.</p> <p>A dialog will let you know that the group membership changes will not occur until after a restart.</p>
Firewall	<p>If the installer detects that the computer has Windows Firewall enabled, you will be asked if you would like the installer to modify your settings.</p> <p>If you select <b>Yes</b>, Plant SCADA Runtime will be added to the list of authorized programs.</p> <p>See the section <i>Windows Firewalls</i> in the topic <a href="#">Installation Information</a>).</p>
Customize the installation location	<p>This page identifies the destination folders for:</p> <ul style="list-style-type: none"><li>• The selected Plant SCADA program files.</li><li>• The Plant SCADA User and Data folders.</li></ul> <p>You can modify the folder locations by clicking the <b>Change</b> buttons and selecting alternative locations.</p>

Installer Page	Notes
	You can also <b>Reset</b> the default locations. <b>Note:</b> The installation folder and the 'User and Data' folder cannot be the same, or a sub-directory of one another. You should also avoid selecting your Windows user folder.
Ready to Install the Program	This page lists the components that will be installed based on your selections. If required, use the <b>Back</b> button to make any modifications. Click <b>Install</b> to start the installation process.

When installation is complete, a dialog will appear asking if you would like to launch [Configurator](#). This may take a few minutes if you have installed some external components (for example, AVEVA Enterprise Licensing components).

**Note:** The installation experience is not complete until after you exit Configurator. A restart request is likely to appear at this point. To avoid permission issues, you should close Configurator and confirm if a reboot is required before you run Plant SCADA.

## See Also

[Runtime Server Installation Requirements](#)

[Use Configurator to Set Up a Runtime Server](#)

## Use Configurator to Set Up a Runtime Server

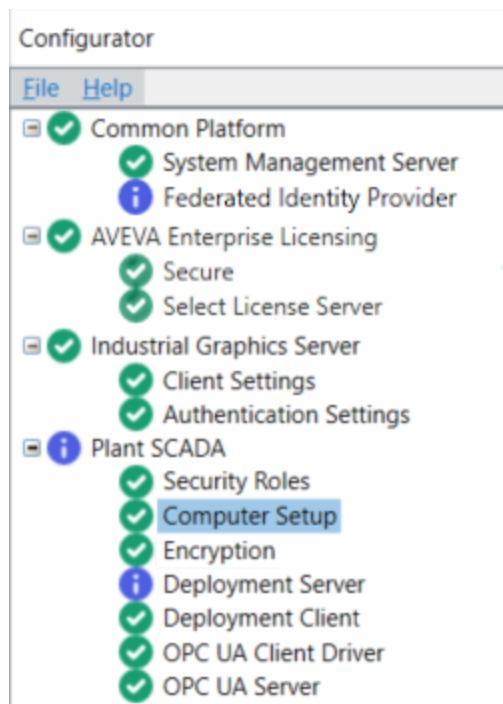
When your Plant SCADA installation is complete, a dialog will appear asking if you would like to launch [Configurator](#). You can use the **Configurator's Computer Setup** page to adjust the following runtime environment settings for a runtime server.

## Server Authentication

A server process requires the configuration of a password. This allows servers to authenticate and create trust with each other. If a computer will run a server process, you will need to create a new password, or use the password that is already used by other server computers in your Plant SCADA system.

[To configure server authentication for a runtime computer:](#)

1. In the panel on the left side of the **Configurator**, select **Plant SCADA | Computer Setup**.



2. The **Computer Setup** page appears. Navigate to the Server Authentication section. **Server Authentication** section.

#### Server Authentication

*Running a server process requires the configuration of a server password. Setting this password allows servers to authenticate and create trust with each other.*

Password	<input checked="" type="checkbox"/> Configure Server Password
<input type="password"/> <span style="font-family: monospace;">*****</span>	
Confirm Password	<input type="password"/> <span style="font-family: monospace;">*****</span>

3. Select the **Configure Server Password** check box.
4. Enter the required password and confirm it in the fields provided.

If the **Password** and **Confirm Password** fields already contain an entry, it means a server password has already been configured on the local computer. If required, you can enter a new password.

The password you use needs to match the password configured for the other server processes included in your Plant SCADA system. This password can be set on each computer using Configurator or the Setup Wizard.

5. To apply your settings, click the **Configure** button.

**Note:** If you change the server password while the runtime system is active, it will not disrupt any established connections as the password is only retrieved when authentication needs to occur. However, it is recommended that you apply the new password as soon as possible. To do this, you need to restart the runtime system.

## Project Run Path

If the computer will receive project updates from a Deployment Server, you need to indicate that you want to run the deployed project.

If you are not using deployment, you can use the default setting **Run the project selected in Plant SCADA Studio**.

### To set the Project Run Path:

1. In the panel on the left side of the **Configurator**, select **Plant SCADA | Computer Setup**.

The **Computer Setup** page appears.

2. Navigate to the **Project Run Path** section.

#### Project Run Path

*Runtime can run the project selected in Plant SCADA Studio, or the project that has been deployed by the Deployment Server*

- Run the project selected in Plant SCADA Studio.**  
 **Run the project deployed from the Deployment Server.**

3. Select one of the following options to determine which project will be launched by Runtime Manager:

- **Run the project currently selected in Plant SCADA Studio** — This is selected by default and will run the project from Plant SCADA Studio.
- **Run the project deployed from the Deployment Server** — Select this option to set up a computer for deployment. This option allows you to run the deployed projects from the deployment server.

4. To apply your settings, click the **Configure** button.

---

**Note:** If Plant SCADA Runtime is running as a service and the **Project Run Path** option in the Configurator is changed to **Run the project deployed from the Deployment Server**, you need to restart the Runtime Manager service for deployment to function correctly.

---

## Runtime Manager Configuration

This setting instructs Runtime Manager to run as a service. This allow a project to be deployed and run on a computer without having to change other settings.

You need to select **Run Runtime Manager as a service** under the following circumstances:

- When a deployment client is configured on a computer and you want to push project updates using "Force" mode.
- If you are using encrypted communications, and the current user on a deployment client may not have required privileges.

---

**Note:** The current user on a deployment client needs to be a member of **Configuration Users** security role to run the deployment client (see [Security Roles](#)).

---

These settings can be made using the **Configurator's Plant SCADA | Computer Setup** page. You can launch Configurator at the completion of the installation procedure, or go to the Windows® **Start** menu and locate AVEVA | **Configurator**.

### To run Runtime Manager as a service:

1. In the panel on the left side of the **Configurator**, select Plant SCADA | **Computer Setup**. The **Computer Setup** page appears. Navigate to the **Runtime Manager Configuration** section of the dialog.

#### Runtime Manager Configuration

*The Runtime Manager can be configured to run as a service or as a standard process. When run as a service, Runtime Manager allows projects to be deployed to the machine without having to change any other settings.*

Run Runtime Manager as a service

2. Select **Run Runtime Manager as a Service**.

---

**Note:** If you are using Deployment to run a project and do not select the **Run Runtime Manager as a Service** option, you will need to manually start the Plant SCADA Runtime Manager before deploying the project.

---

3. To apply your settings, click the **Configure** button.

---

**Note:** If you have configured a Plant SCADA OPC DA server on a computer that is running Plant SCADA as a Windows service, you will need to make the additional configuration changes. See the topic [Running an OPC DA Server as a Service](#) in the *Runtime* section of the Plant SCADA documentation.

---

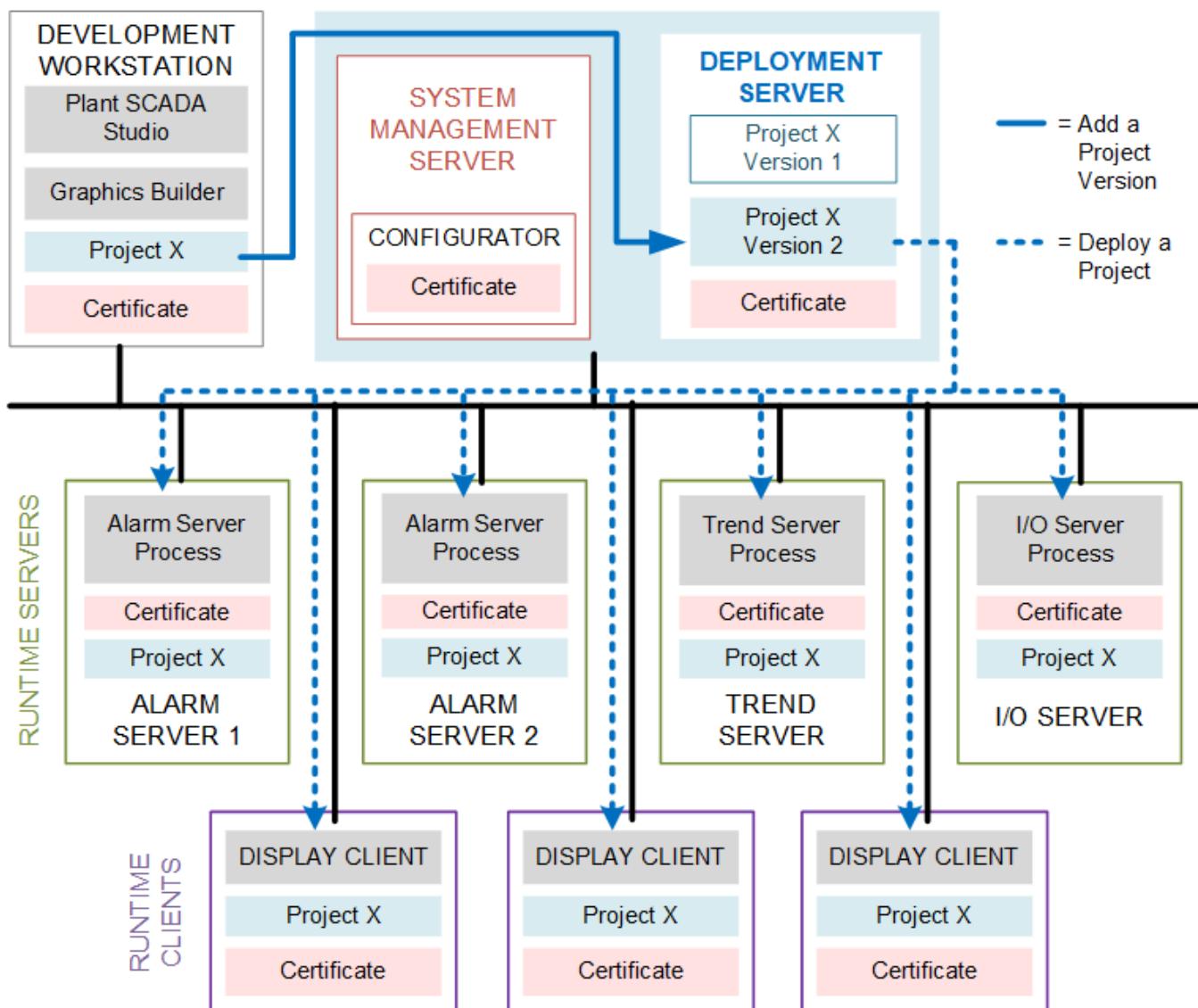
## See Also

[Runtime Server Installation Requirements](#)

[Runtime Server Installation Procedure](#)

## Install a Deployment Server

A Deployment Server allows you to distribute projects (and any subsequent updates) to the computers in your Plant SCADA system. It stores multiple "versions" of a project's runtime files, allowing you to deploy a specific version to any computer that has been configured as a deployment client.



A Deployment Server uses encrypted communications, which means it requires access to a System Management Server. Ideally the Deployment Server should be installed on the same computer as the System Management Server. See [Set Up a System Management Server](#).

#### Note:

- If you install the Deployment Server on a different computer to the System Management Server, you cannot use the default local user groups assigned to the Deployment Security Roles as they will not work in a distributed environment. You need to create domain user groups that align with the existing local groups, and add these to the Deployment Security Roles. The domain groups can then be used instead of the following default local user groups.

Deployment Administrators role: 'SCADA.DeploymentAdmins' or 'Asb.Deployment.AdminRole'.

Deployment Uploaders role: 'SCADA.DeploymentUploaders' or 'Asb.Deployment.UploadRole'.

Deployment Users role: 'SCADA.DeploymentUsers' or 'Asb.Deployment.DeployRole'.

See [Modifying the Members of a Security Role](#).

- When installing a Deployment Server as part of a workgroup, it must be installed on the same computer as the System Management Server. If you need the Deployment Server on a separate computer to the System Management Server, you will need to move your IT infrastructure to a domain.

When using Industrial Graphics, a Deployment Server is required to deliver project versions to the Industrial Graphics Server.

**Note:** The deployment architecture installed with Plant SCADA 2023 (and later) is no longer compatible with deployment servers running an earlier version. This means:

- You cannot use Plant SCADA Studio to add projects to a deployment server that is running a different version of Plant SCADA.
- You cannot deploy a project from a deployment server to a deployment client if they are running different versions of Plant SCADA.

If you have a distributed system that includes some computers on an earlier version of Plant SCADA, you should retain your existing deployment server to distribute projects to these computers until the roll out of the new version is complete. You will also need to configure a new deployment server to deploy new projects to your upgraded computers.

## See Also

[Deployment Server Installation Requirements](#)

[Deployment Server Installation Procedure](#)

[Configure a Deployment Server](#)

[Configure a Deployment Client](#)

## Deployment Server Installation Requirements

This section describes the hardware and software requirements for a computer that will host a Deployment Server.

Before you install Plant SCADA, it is recommended that you install the latest updates from Microsoft® for your operating system and system software.

## Hardware Requirements

The hardware requirements for a Deployment Server are as follows:

Cores	4
Memory	8 GB
HDD	Size of project backed up, multiplied by the number of versions expected to be stored.  An SSD is recommended for better performance. If a non-SSD is used, select a minimum RPM of 7200.
CPU PassMark®	4250
Network	1 GB

## Operating System Requirements

The following operating systems are supported (64-bit only):

- Windows 11 21H2 (or later)
- Windows 10 21H2 (or later)
- Windows 10 LTSC 1607 (or later)
- Windows Server 2016
- Windows Server 2019
- Windows Server 2022

Microsoft .NET Framework 4.8 is required. It will be installed by Plant SCADA if not detected.

When .NET installation is complete, you may be prompted to restart your computer. If this occurs, you should initiate a restart before you continue.

## See Also

[Deployment Server Installation Procedure](#)

## Deployment Server Installation Procedure

This section lists the pages displayed by the Plant SCADA installer and the settings required for a Deployment Server. Use the **Back** and **Next** buttons to navigate through the included pages.

Installer Page	Notes
Installation Documentation	If required, view the available documents.
License Agreement	Accept the terms of the License Agreement.
What is the role of this computer?	Select <b>Server</b> .
Select the server(s) you want to install	Select <b>Deployment Server</b> . If required, you can also select other server types.
The selected products/components will be installed/upgraded.	By default, the following components are selected when you install a Deployment Server. <ul style="list-style-type: none"><li>• Plant SCADA Deployment Server</li></ul> You can use this page to customize the list of selected components.
Access Restriction	This page provides information about the local Windows groups the installer will create to manage security (see the section <i>Windows User Groups and Security Roles</i> in the topic <a href="#">Installation Information</a> ).

Installer Page	Notes
	<p><b>Note:</b> This page will only appear if you are installing Plant SCADA on a clean computer (one that has not hosted an earlier version). If you are upgrading from an earlier version, equivalent user groups will already exist on the computer.</p> <p>If you want your current Windows user account to be included in the groups that are being created, select <b>Add the current Windows user to these groups</b>.</p> <p>A dialog will let you know that the group membership changes will not occur until after a restart.</p>
Firewall	<p>If the installer detects that the computer has Windows Firewall enabled, you will be asked if you would like the installer to modify your settings.</p> <p>If you select <b>Yes</b>, Plant SCADA Runtime will be added to the list of authorized programs.</p> <p>See the section <i>Windows Firewalls</i> in the topic <a href="#">Installation Information</a>).</p>
Customize the installation location	<p>This page identifies the destination folders for:</p> <ul style="list-style-type: none"> <li>The selected Plant SCADA program files.</li> <li>The Plant SCADA User and Data folders.</li> </ul> <p>You can modify the folder locations by clicking the <b>Change</b> buttons and selecting alternative locations.</p> <p>You can also <b>Reset</b> the default locations.</p> <p><b>Note:</b> The installation folder and the 'User and Data' folder cannot be the same, or a sub-directory of one another. You should also avoid selecting your Windows user folder.</p>
Ready to Install the Program	<p>This page lists the components that will be installed based on your selections.</p> <p>If required, use the <b>Back</b> button to make any modifications.</p> <p>Click <b>Install</b> to start the installation process.</p>

When installation is complete, a dialog will appear asking if you would like to launch [Configurator](#). This may take a few minutes if you have installed some external components (for example, AVEVA Enterprise Licensing components).

---

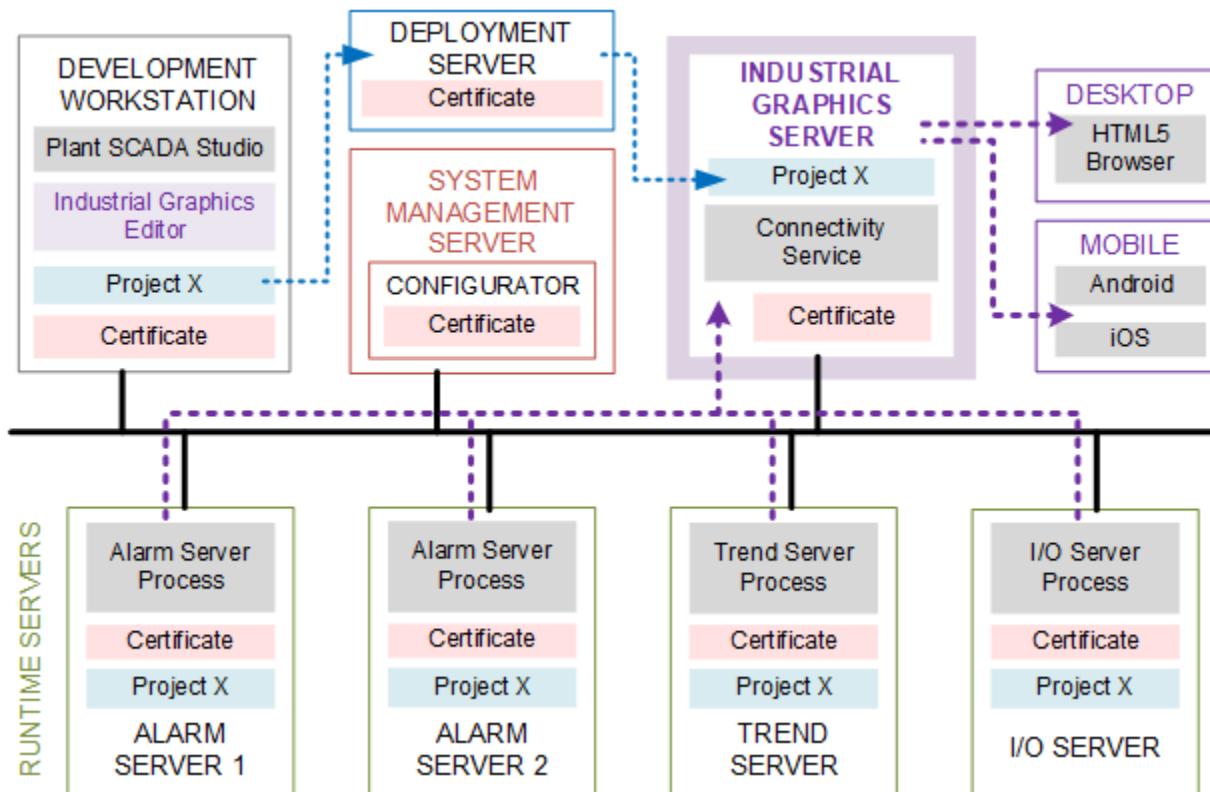
**Note:** The installation experience is not complete until after you exit Configurator. A restart request is likely to appear at this point. To avoid permission issues, you should close Configurator and confirm if a reboot is required before you run Plant SCADA.

To configure your system for deployment following installation, see the topics [Configure a Deployment Server](#)

and [Configure a Deployment Client](#).

## Install an Industrial Graphics Server

An AVEVA™ Industrial Graphics Server distributes HTML5 graphics to desktop and mobile browsers. It authenticates clients details and provides access to the graphical and functional content created using the Plant SCADA Industrial Graphics Editor.



**Note:** With this release of Plant SCADA, the Industrial Graphics Server has a number of functionality limitations. For example, it does not support alarms. For a full list of limitations, see [Industrial Graphics Server Functionality Limitations](#).

### ⚠️ WARNING

#### UNINTENDED EQUIPMENT OPERATION

A computer that hosts an Industrial Graphics Server must be configured to use an English regional locale. This ensures that any values with unit separators are interpreted correctly when entered by an operator.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

When you choose to install an Industrial Graphics Server, the following components are selected by default:

- Plant SCADA Industrial Graphics Server
- Plant SCADA Deployment Client.

The Deployment Client is required as Plant SCADA's Deployment functionality is used to deliver projects to the

Industrial Graphics Server.

**Note:**

- For runtime operation, it is recommended that you install an Industrial Graphics Server on a dedicated computer especially if it will be required to handle a large number of client requests.
- If you would like to review an Industrial Graphics application locally during development, you also need to have an Industrial Graphics Server installed locally on the development workstation.

Licensing for AVEVA Industrial Graphics is managed using the AVEVA Enterprise Licensing system, which means you may also need to install the Enterprise Licensing Server and/or the Enterprise Licensing Manager. For more information, see [AVEVA™ Enterprise Licensing](#).

If required, you can display the Industrial Graphics Web Client locally on the Industrial Graphics Server using the following address in a browser:

- `https://localhost/aig`

Be aware, however, that the connected System Management Server may be configured to use a non-default port (see [Advanced Configuration for a System Management Server](#)). This changes the port the AIG server uses to listen, which means you need to include the current **HTTPS Port** setting in the address as follows:

- `https://localhost:<HTTPS Port value>/aig`

**Note:** If the Plant SCADA compiler determines that your SCADA system tag names are not globally unique across all clusters, any tag subscriptions that do not specify a specific cluster will be rejected by the Industrial Graphics Server. If this occurs, you can use the parameter [\[General\]EnforceTagGlobalUniqueness](#) to diagnose which tags are not globally unique.

## See Also

[Industrial Graphics Server Installation Requirements](#)

[Industrial Graphics Server Installation Procedure](#)

[Use Configurator to Set Up an Industrial Graphics Server](#)

[Confirm the Settings for an Industrial Graphics Server](#)

## Industrial Graphics Server Functionality Limitations

With the current release of Plant SCADA, the Industrial Graphics Server has the following functionality limitations:

- Alarms are not supported.
- Only English locale is supported by the Industrial Graphics Server.
  - For decimal points you need to use "." (period).
  - For thousand separators you need to use "," (comma).
- The status of write operations is not reflected by the status element or quality styles applied to elements. Design your pages so that an operator can determine if a write has been set or rejected by displaying the current value.
- Local variables cannot be referenced.

- Alarm Properties as Tags are not exposed.
- Quality tag extensions are represented as a numeric value. Only OPC quality should be used for an Industrial Graphics application.
- Extended quality modes "Control Inhibit" and "Tag Override" are not available.
- Plant SCADA trends are not supported, however, instant trends via an I/O tag or AVEVA™ Historian are available.
- Web widgets are not supported.
- Industrial Graphics cannot be imported/exported between Plant SCADA and other AVEVA products.
- Native Plant SCADA graphics cannot be converted or imported into Industrial Graphics applications.
- Redundant Industrial Graphics Servers are not supported.

---

**Note:** It is recommended that you review the list of supported graphical elements and limitations before you start creating an Industrial Graphics application. See the topic [Supported Graphical Elements and Known Limitations](#) in the *AVEVA™ Industrial Graphics* section of the Plant SCADA documentation.

---

## See Also

- [Industrial Graphics Server Installation Requirements](#)
- [Industrial Graphics Server Installation Procedure](#)
- [Use Configurator to Set Up an Industrial Graphics Server](#)
- [Confirm the Settings for an Industrial Graphics Server](#)

## Industrial Graphics Server Installation Requirements

This topic describes the hardware and software requirements for a computer that will host an Industrial Graphics Server.

Before you install Plant SCADA, it is recommended that you install the latest updates from Microsoft® for your operating system and system software.

---

**Note:** It is recommended to install the Industrial Graphics Server on a dedicated computer.

---

## Hardware Requirements

The hardware requirements for an Industrial Graphics Server are as follows:

- CPU PassMark®: > 5200 pts
- RAM: 16 GB
  - Each client session requires ~200 Mb of memory, depending upon graphics complexity.
- HDD: 500 Mb and space for the project's Industrial Graphics.
- Client connections: 10
  - Additional clients can be supported by increasing the number of CPU's, CPU speed, and Memory.

---

**Note:** The recommended requirements are suitable for a system with ten clients, browsing pages with approximately 40 dashboard/charting components with ~250 I/O tags on the page. Pages may take a longer time to display on the first visit. The display time depends on graphics and script complexity.

---

## Operating System Requirements

The following operating systems are supported (64-bit only):

- Windows 11 21H2 (or later)
- Windows 10 21H2 (or later)
- Windows 10 LTSC 1607 (or later)
- Windows Server 2016
- Windows Server 2019
- Windows Server 2022

Microsoft .NET Framework 4.8 is required. It will be installed by Plant SCADA if not detected.

When .NET installation is complete, you may be prompted to restart your computer. If this occurs, you should initiate a restart before you continue.

**Note:** Licensing for AVEVA Industrial Graphics is managed using the AVEVA Enterprise Licensing system. If you do not have access to an existing Enterprise License Server, you will need to include the **AVEVA Enterprise License Manager** and **AVEVA Enterprise License Server** to your Plant SCADA installation. For more information, see [AVEVA™ Enterprise Licensing](#).

## See Also

[Industrial Graphics Server Installation Procedure](#)

## Industrial Graphics Server Installation Procedure

This section describes the pages displayed by the Plant SCADA installer and the settings required for an Industrial Graphics Server. Use the **Back** and **Next** buttons to navigate through the included pages.

Installer Page	Notes
Installation Documentation	If required, view the available documents.
License Agreement	Accept the terms of the License Agreement.
What is the role of this computer?	Select <b>Server</b> .
Select the server(s) you want to install	Select <b>Plant SCADA Industrial Graphics Server</b> . If required, you can also select other server types. However, it is recommended that you install an Industrial Graphics Server on a dedicated computer.
The selected products/components will be installed/upgraded.	By default, the following components are selected when you install an Industrial Graphics Server. <ul style="list-style-type: none"><li>• Plant SCADA Industrial Graphics Server</li></ul>

Installer Page	Notes
	<ul style="list-style-type: none"><li>• Plant SCADA Deployment Client.</li></ul> <p>You can use this page to customize the list of selected components.</p> <p>Licensing for AVEVA Industrial Graphics is managed using the AVEVA Enterprise Licensing system. If you do not have access to an existing Enterprise License Server, you will need to include the following components in your installation.</p> <ul style="list-style-type: none"><li>• AVEVA Enterprise License Manager</li><li>• AVEVA Enterprise License Server.</li></ul> <p>If an Enterprise License Server is already set up, you will still need the License Manager to access it.</p> <p><b>Note:</b> Plant SCADA 2023 R2 installs version 4.0 of AVEVA Enterprise Licensing. If your system includes computers that are not upgraded to AEL 4.0, you will need to install <b>AVEVA Enterprise Licensing Server Legacy Support</b> on your Enterprise License Server. This enables backwards compatibility for clients nodes that are running an earlier version of AEL.</p>
Access Restriction	<p>This page provides information about the local Windows groups the installer will create to manage security (see the section <i>Windows User Groups and Security Roles</i> in the topic <a href="#">Installation Information</a>).</p> <p><b>Note:</b> This page will only appear if you are installing Plant SCADA on a clean computer (one that has not hosted an earlier version). If you are upgrading from an earlier version, equivalent user groups will already exist on the computer.</p> <p>If you want your current Windows user account to be included in the groups that are being created, select <b>Add the current Windows user to these groups</b>.</p> <p>A dialog will let you know that the group membership changes will not occur until after a restart.</p>
Firewall	<p>If the installer detects that the computer has Windows Firewall enabled, you will be asked if you would like the installer to modify your settings.</p> <p>If you select <b>Yes</b>, Plant SCADA Runtime will be added to the list of authorized programs.</p> <p>See the section <i>Windows Firewalls</i> in the topic <a href="#">Installation Information</a>.</p>

Installer Page	Notes
	<a href="#">Installation Information</a> ).
Customize the installation location	<p>This page identifies the destination folders for:</p> <ul style="list-style-type: none"><li>• The selected Plant SCADA program files.</li><li>• The Plant SCADA User and Data folders.</li></ul> <p>You can modify the folder locations by clicking the <b>Change</b> buttons and selecting alternative locations. You can also <b>Reset</b> the default locations.</p> <p><b>Note:</b> The installation folder and the 'User and Data' folder cannot be the same, or a sub-directory of one another. You should also avoid selecting your Windows user folder.</p>
Ready to Install the Program	<p>This page lists the components that will be installed based on your selections.</p> <p>If required, use the <b>Back</b> button to make any modifications.</p> <p>Click <b>Install</b> to start the installation process.</p>

When installation is complete, a dialog will appear asking if you would like to launch [Configurator](#). This may take a few minutes if you have installed some external components (for example, AVEVA Enterprise Licensing components).

---

**Note:** The installation experience is not complete until after you exit Configurator. A restart request is likely to appear at this point. To avoid permission issues, you should close Configurator and confirm if a reboot is required before you run Plant SCADA.

---

## See Also

- [Industrial Graphics Server Installation Requirements](#)
- [Use Configurator to Set Up an Industrial Graphics Server](#)
- [Confirm the Settings for an Industrial Graphics Server](#)

## Use Configurator to Set Up an Industrial Graphics Server

You can use [Configurator](#) to enable authentication for an Industrial Graphics Server. You can also adjust the refresh rates for any connecting clients. These settings are available on the following pages within the **Industrial Graphics Server** branch of the Configurator directory.

### Authentication Settings

#### System Management Server

You need to connect a computer that hosts an Industrial Graphics Server to a System Management Server (SMS). This will allow authentication and authorization requests to be handled by the SMS's Identity Manager.

When connected, the **System Management Server** field will display the name of the computer hosting the SMS.

If "not connected" is displayed, you will need to configure the connection to an SMS (see [Connect a Computer to a System Management Server](#)).

### Secure Gateway

If required, you can use a secure gateway to avoid directly exposing the Industrial Graphics Server to client computers that are not a part of your SCADA system network.

To use a secure gateway, enter the Fully Qualified Domain Name for the computer that hosts the secure gateway server you want to use. The computer you use should be located in a DMZ.

For more information, see the topic [Secure the Industrial Graphics Web Client](#).

## Client Settings

The Client Settings page allows you to specify the following settings for clients that connect to the Industrial Graphics Server.

### Graphic Refresh Rate

Allows you to specify (in milliseconds) how regularly graphics are refreshed in a connected Industrial Graphics Client. You can specify a **Refresh Rate** between 250 and 60000 milliseconds.

### Alarm Refresh Rate

This feature is not supported by Industrial Graphics applications in this version of Plant SCADA.

---

**Note:** Plant SCADA's Deployment functionality is used to deliver projects to the Industrial Graphics Server. For this reason, you will need to use Configurator's **Deployment Client** page to connect the Industrial Graphics Server to your system's Deployment Server. See [Configure a Deployment Client](#) for further instructions.

---

## Confirm the Settings for an Industrial Graphics Server

To confirm that an Industrial Graphics Server is ready for operation, you should check that you have completed the following steps.

### Connect all your Plant SCADA computers to a System Management Server

To connect a computer to a System Management Server, use the System Management Server page in **Configurator**. See [Connect a Computer to a System Management Server](#).

### Configure the authentication settings

The required settings are available on the **Authentication Settings** page within the **Industrial Graphics Server** branch of Configurator. See [Use Configurator to Set Up an Industrial Graphics Server](#).

### Set up deployment

If your Industrial Graphics Server is running on a remote computer, set it up for deployment.

- Set up a deployment server in your system (see [Configure a Deployment Server](#)).
- Configure a deployment client on the Industrial Graphics Server so that it can receive project versions from the deployment server (see [Configure a Deployment Client](#)).

## Enable encryption

An Industrial Graphics Server will only connect to your Plant SCADA servers over an encrypted connection. You need to confirm the following:

- Plant SCADA is configured to run with encryption enabled. This is achieved via the **Encryption** page in Configurator (see [Enable Encryption](#) in the Plant SCADA documentation).  
Running in mixed mode is acceptable as long the remaining prerequisites are also met.
- Runtime Manager needs to be configured to run as a service.  
This is required to enable encryption. The required settings are available on **Configurator's Computer Setup** page (see [Configure a Runtime Computer for Encryption](#) in the Plant SCADA documentation).
- In your project settings, confirm the **DNS Name** field is configured for each entry in the **Computers** activity. This is also required to enable encryption.
- Run the Computer Setup Wizard and confirm on the **Network Model** page that networking is enabled. This is required even if you are running the Industrial Graphics Server and the Plant SCADA servers on a single computer.

## Set up user authentication

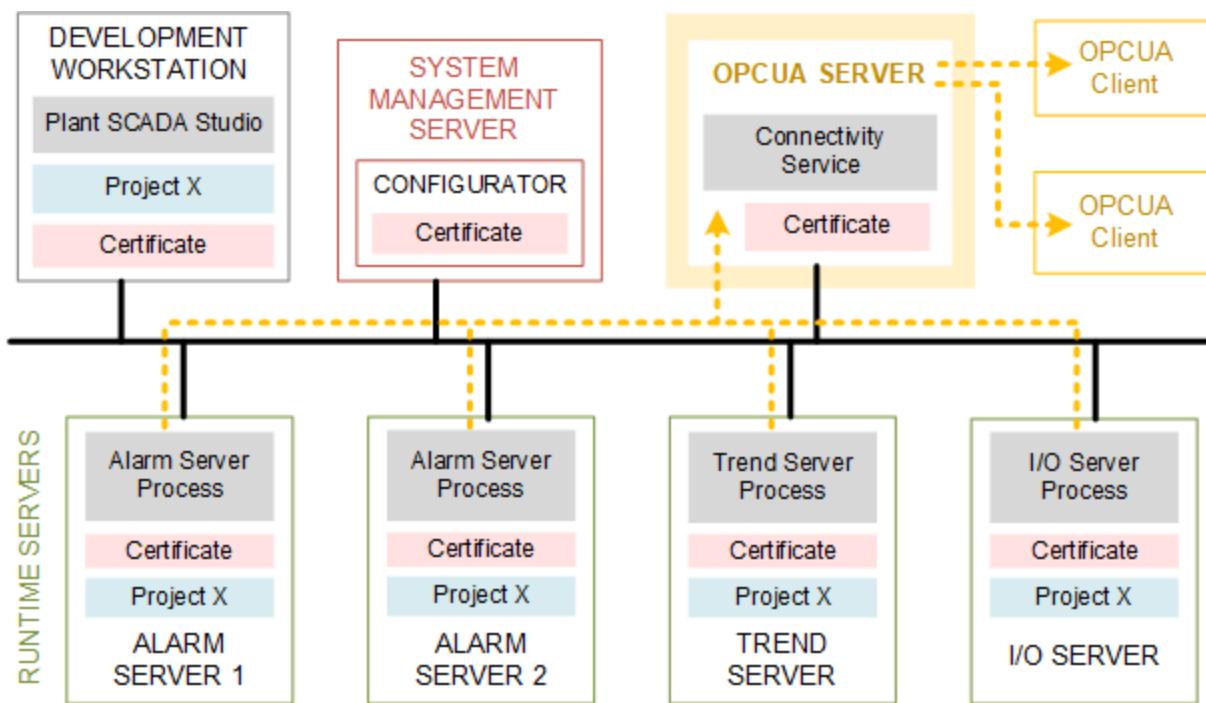
The Industrial Graphics Server only supports authentication with Windows® users. Plant SCADA users are not supported.

Authentication occurs in two places:

- To establish a connection to the Industrial Graphics Server, a user must be a member of a Windows group that is mapped to the "Industrial Graphics Users" security role (see [Security Roles](#)).
- The user must also be a member of a Windows group that is mapped to a valid Role within your Plant SCADA project. See the topic [Roles](#) within the *Runtime System Security* section of the Plant SCADA documentation.

## Install an OPC UA Server

An OPC UA server allows Plant SCADA's runtime system to function as a server to OPC UA client applications.



**Note:** In this release of Plant SCADA, the OPC UA server has the following functionality limitations:

- The server supports DA only.
- Quality tag extensions are represented as a numeric value. Only the OPC quality is exposed.
- Alarm Properties as Tags are not exposed.
- Extended quality modes "Control Inhibit" and "Tag Override" are not available.
- Redundant OPC UA servers are not supported.

Tag extensions are included in the OPC UA browse hierarchy. Of those presented, a client can subscribe to the following:

- Field
- Control Mode
- Override Mode
- Valid
- Status
- Override

Following installation, you can configure the OPC UA server via Configurator's **OPC UA Server** plugin under Plant SCADA. This has four configuration options:

- **Port number** – the port number the OPC UA server will listen on.
- **URL** – the URL to use to connect an OPC UA client to the OPC UA server (read only).
- **Enable Encryption** – This option allows encrypted communications between the OPC UA server and a connected client.
- **Allow Anonymous access** – Checking this option allows anonymous users to connect to the OPC UA server.

See [Use Configurator to Set Up an OPC UA Server](#).

If the OPC UA server is installed on a separate computer, you will need to configure a deployment server and

client to deploy the relevant project information to the OPC UA server. Once the project files have been deployed, the OPC UA server will start using the new configuration automatically.

To communicate with the Plant SCADA runtime system, you also need to connect the OPC UA server to a System Management Server to enable encrypted communication with the Plant SCADA servers.

**Note:** The OPC UA server requires the purchase of a specific license. Please contact your Plant SCADA distributor for more information.

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

If you update the Equipment Name or Equipment Item for a variable tag and restart the I/O server, any OPC UA clients that are subscribed to this tag via its old equipment name may show outdated values. After restarting the I/O server, perform the rebrowse operation in the OPC UA client and resubscribe to those variables tags.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** The OPC UA Server's browse functionality will always retrieve a tag name with the cluster prefix. However, if you manually subscribe to a Plant SCADA tag using its tag name, the OPC UA Server may reject the subscription request if a cluster is not explicitly defined. This will occur if the compiler determines that your SCADA system tags are not globally unique across all clusters. If this occurs, you can use the parameter [\[General\]EnforceTagGlobalUniqueness](#) to diagnose which tags are not globally unique.

## **See Also**

[OPC UA Server Installation Requirements](#)

[OPC UA Server Installation Procedure](#)

[Use Configurator to Set Up an OPC UA Server](#)

[Confirm the Settings for an OPC UA Server](#)

## **OPC UA Server Installation Requirements**

This topic describes the hardware and software requirements for a computer that will host an OPC UA Server.

Before you install Plant SCADA, it is recommended that you install the latest updates from Microsoft® for your operating system and system software.

It is also recommended that you install the OPC UA Server on a dedicated computer.

**Note:** The OPC UA server requires the purchase of a specific license. Please contact your Plant SCADA distributor for more information.

## **Hardware Requirements**

The hardware requirements for a OPC UA Server are as follows:

- CPU PassMark®: > 4000 pts
- RAM: 16 GB
- Changes per second: < 25,000

OPC UA in Plant SCADA supports a minimum sampling rate of 500 milliseconds, which means it can capture value changes for a tag at one second interval.

A higher number of changes per second can be supported, but this will require additional memory and CPU performance.

- Subscribed tags: < 100,000
- Maximum OPC UA clients in the system: < 10

Adding each OPC UA client to the connected OPC UA server increases the CPU and memory usage on both the OPC UA server and the I/O sever.

---

**Note:**

- Several systems with a large number of tags arranged in a flat architecture, or a hierarchy with many child nodes, will significantly impact the browse performance of a connected OPC UA client.
  - Plant SCADA now includes a connectivity service that manages tag subscriptions for the OPC UA Server and Industrial Graphics Server. This places an additional load on your I/O servers, particularly if a subscribed tag also specifies a property or extension. If this impacts I/O performance, you should consider distributing your I/O servers across a number of computers. You can also disable support for properties and extensions with the parameter [\[IOServer\]DisableConnectivityPropertiesAndExtensions](#).
- 

## Operating System Requirements

The following operating systems are supported (64-bit only):

- Windows 11 21H2 (or later)
- Windows 10 21H2 (or later)
- Windows 10 LTSC 1607 (or later)
- Windows Server 2016
- Windows Server 2019
- Windows Server 2022

Microsoft .NET Framework 4.8 is required. It will be installed by Plant SCADA if not detected.

When .NET installation is complete, you may be prompted to restart your computer. If this occurs, you should initiate a restart before you continue.

---

**Note:** The OPC UA Server does not support browse filtering by OPC UA Clients.

---

## See Also

[OPC UA Server Installation Procedure](#)

[Use Configurator to Set Up an OPC UA Server](#)

[Confirm the Settings for an OPC UA Server](#)

## OPC UA Server Installation Procedure

This section describes the pages displayed by the Plant SCADA installer and the settings required for an OPC UA Server. Use the **Back** and **Next** buttons to navigate through the included pages.

Installer Page	Notes
Installation Documentation	If required, view the available documents.
License Agreement	Accept the terms of the License Agreement.
What is the role of this computer?	Select <b>Server</b> .
Select the server(s) you want to install	<p>Select <b>Plant SCADA OPC UA Server</b>. If required, you can also select other server types.</p>
The selected products/components will be installed/upgraded.	<p>By default, the following components are selected when you install an OPC UA Server.</p> <ul style="list-style-type: none"> <li>• Platform Common Services</li> <li>• Plant SCADA OPC UA Server</li> </ul> <p>You can use this page to customize the list of selected components.</p>
Access Restriction	<p>This page provides information about the local Windows groups the installer will create to manage security (see the section <i>Windows User Groups and Security Roles</i> in the topic <a href="#">Installation Information</a>).</p> <p><b>Note:</b> This page will only appear if you are installing Plant SCADA on a clean computer (one that has not hosted an earlier version). If you are upgrading from an earlier version, equivalent user groups will already exist on the computer.</p> <p>If you want your current Windows user account to be included in the groups that are being created, select <b>Add the current Windows user to these groups</b>. A dialog will let you know that the group membership changes will not occur until after a restart.</p>
Firewall	<p>If the installer detects that the computer has Windows Firewall enabled, you will be asked if you would like the installer to modify your settings.</p> <p>If you select <b>Yes</b>, Plant SCADA Runtime will be added to the list of authorized programs.</p> <p>See the section <i>Windows Firewalls</i> in the topic <a href="#">Installation Information</a>).</p>
Customize the installation location	<p>This page identifies the destination folders for:</p> <ul style="list-style-type: none"> <li>• The selected Plant SCADA program files.</li> <li>• The Plant SCADA User and Data folders.</li> </ul>

Installer Page	Notes
	<p>You can modify the folder locations by clicking the <b>Change</b> buttons and selecting alternative locations.</p> <p>You can also <b>Reset</b> the default locations.</p> <p><b>Note:</b> The installation folder and the 'User and Data' folder cannot be the same, or a sub-directory of one another. You should also avoid selecting your Windows user folder.</p>
Ready to Install the Program	<p>This page lists the components that will be installed based on your selections.</p> <p>If required, use the <b>Back</b> button to make any modifications.</p> <p>Click <b>Install</b> to start the installation process.</p>

When installation is complete, a dialog will appear asking if you would like to launch [Configurator](#). This may take a few minutes if you have installed some external components (for example, AVEVA Enterprise Licensing components).

**Note:** The installation experience is not complete until after you exit Configurator. A restart request is likely to appear at this point. To avoid permission issues, you should close Configurator and confirm if a reboot is required before you run Plant SCADA.

## See Also

[OPC UA Server Installation Requirements](#)

[Use Configurator to Set Up an OPC UA Server](#)

[Confirm the Settings for an OPC UA Server](#)

## Use Configurator to Set Up an OPC UA Server

When your Plant SCADA installation is complete, a dialog will appear asking if you would like to launch [Configurator](#).

To set up an OPC UA Server, open Configurator and adjust the following settings on the **OPC UA Sever** page under Plant SCADA.

**Note:** The settings on this page will not encrypt communication with Plant SCADA runtime. You need to connect the OPC UA server to a System Management Server and have encryption enabled to allow communication with a runtime system. See [Enable Encryption](#).

## Address Configuration

- Enter the **Port Number** that you want to use for the OPC UA server. The value you enter will be reflected in the URL.
- The **URL** field shows the address that you need to use to connect an OPC UA client to the OPC UA server. You cannot edit this field. It defaults to:

opc.tcp://<computer name>:48031/plantscada.

The OPC UA server does not support OPC UA's discovery feature, so you need to manually enter this URL into a client application to connect to an OPC UA server.

## Encryption

Select **Enable encrypted communications** to protect the connection between the OPC UA server and any connected clients. This will set the security policy to Basic256Sha256 and the security mode to SignAndEncrypt.

For information on how to use certificates to enable encrypted communications between a Plant SCADA OPC UA server and an OPC UA client, see [Configure Client Certificates for an OPC UA Server](#).

---

**Note:** It is recommended that you enable encryption to help secure the OPC UA server.

## Authentication

Select **Allow anonymous connections** if you want to allow users to connect anonymously to the OPC UA Server.

If this option is not selected, a user will only be able to access the server if they are part of a Windows group that is associated with one of the [Security Roles](#) defined in your project. Plant SCADA users are not supported.

---

**Note:** It is recommended that you disable anonymous access to facilitate user-based security for the OPCUA server. Also, tag writes from an OPC UA client application are not supported if anonymous access is enabled.

## Browsing

Many OPC UA clients will only allow writes to occur on the leaf nodes in the OPC UA browse result. For OPC UA clients that support this functionality, you will need to select **Disable properties and extensions**.

## Configure Client Certificates for an OPC UA Server

You can use certificates to enable encrypted communications between a Plant SCADA OPC UA Server and an OPC UA client. To achieve this, both computers require access to the following certificates:

- The "<Computer Name> ASB OPC UA Server" certificate from the OPC UA Server
- The client certificate from the OPC UA client.

---

**Note:** The information provided in this topic is intended for a scenario where SMS certificates are used on computers contained within your SCADA system network. If you want to deliver information across multiple domains or externally via the internet, we recommend you seek professional advice on setting up an external web server.

To complete this setup, you will need to perform the following procedures.

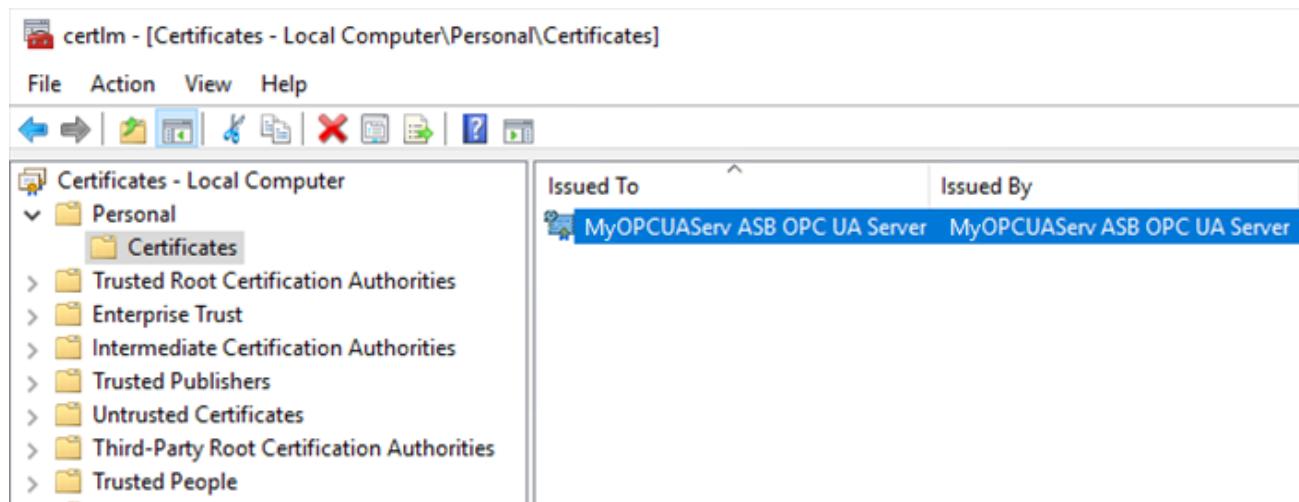
### Copy the OPC UA Server certificate to the OPC UA client

You initially need to export a copy of two certificates created by the SMS.

1. Open Windows™ Certificate Manager.

To do this, you can type "manage computer certificates" into the Windows **Search** bar.

2. In the tree view, go to the **Personal** branch then select **Certificates**.
3. Locate the certificate named "<computer name> ASB OPC UA Server".



4. Right-click on the certificate and select **All Tasks**, then **Export**. This will open the Certificate Export Wizard. Depending on the type of certificates that your client application uses, you will need to export the certificate as a CER file (if the client uses the Windows™ Certificate Store) or a DER file (if the client uses file-based certificates). These two formats are the same, only the file extension is different.
5. Use the following settings to export the certificate.
  - On the **Export Private Key** page, select **No, do not export the private key**.
  - On the **Export File Format** page, select **DER encoded binary X.509 (.CER)** or **Base-64 encoded X.509 (.CER)** (depending on the requirements of your client application).
  - On the **File To Export** page, enter a path and file name (for example, "c:\temp\<machine name> OPC UA Server.cer").
6. When you reach the Finish page, review the settings and click **Finish** to export a copy of the certificate.

Once the certificate has been exported, you may need to rename the file extension on the certificate from ".cer" to ".der" (if the client uses file-based certificates).

## Import the OPC UA Server Certificate on the Client Computer

OPC UA client applications will generally use one of the following mechanisms to manage certificates:

- Using the Windows™ Certificate Store
- Storing certificates in a specified folder (that is predefined by the OPC UA client application).

### Import a certificate into the Windows Certificate Store

Importing a certificate into the Windows Certificate Store requires administrator rights.

1. Copy the "<machine name> OPC UA Server.cer" file to an appropriate location on the client computer.
2. Right click on the CER file and select **Install Certificate**. This will open the Certificate Import Wizard.
3. Use the following settings.

- Under **Store Location**, select **Local Machine**.
  - On the **Certificate Store** page, select **Personal**.
4. When you reach the last page, review the settings and click **Finish**.

### Import a certificate into a specified location

Copy the "<machine name> OPC UA Server.der" file to the folder that is used by your OPC UA client application.

The table below includes the location of the certificate folder for some common OPC UA clients.

OPC UA Client	Manufacturer	Certificate Folder
Datafeed OPC UA Client	Softing	C:\ProgramData\Softing\OpcClient\pki\trusted\certs
UaExpert	Unified Automation	C:\Users\Admin\AppData\Roaming\unifiedautomation\uaexpert\PKI\trusted\certs
UA Client Getting Started	Unified Automation	C:\ProgramData\unifiedautomation\UaSdkNetBundleEval\pkiclient\trusted\certs
Matrikon	Matrikon	C:\Users\Admin\AppData\Local\Matrikon\OPCUAExplorer\pki\DefaultApplicationGroup\trusted\certs
KEPServer	Kepware	C:\ProgramData\Kepware\KEPServerEX\V6\UA\Client Driver\cert
Top Server	Software Toolbox	C:\ProgramData\Software Toolbox\TOP Server\V6\UA\Client Driver\cert

For more information, please refer to the documentation provided with your OPC UA client application.

### Configure the OPC UA client certificate on the OPC UA Server

The next step in this process is trust the OPC UA client certificate on the computer the OPC UA Server is running on.

The easiest way to do this is to attempt to connect an OPC UA client to the server. As the server will not trust the client certificate yet, the connection is expected to be unsuccessful. However, after this happens, any OPC UA client certificates that are not installed on the OPC UA Server computer will be put into the "Rejected Certificate" folder on the OPC UA Server. By default, this is in the following folder:

C:\ProgramData\AVEVA\PCS\OPC UA Rejected Client Certificates\certs"

Accessing this folder requires administrator rights.

## Import certificates from the Rejected Certificates folder

1. Browse to the rejected certificate folder.
2. Right click on the certificate for the OPC UA client that you want to trust and select **Install Certificate**. This will open the Certificate Import Wizard.
3. Use the following settings.
  - Under **Store Location**, select **Local Machine**.
  - On the **Certificate Store** page, select **Trusted People**.
4. When you reach the last page, review the settings and click **Finish**.

## Configure the OPC UA Server ports

OPC UA communicates via a single TCP port. This is specified in the **Endpoint Connection** setting for the OPC UA Server.

This defaults to port 48031.

You will need to confirm that this port is not blocked by any firewall software installed on your computer.

## See Also

[Confirm the Settings for an OPC UA Server](#)

## Confirm the Settings for an OPC UA Server

To confirm that an OPC UA Server is ready for operation, you should check that you have completed the following steps.

## Connect all your Plant SCADA computers to a System Management Server

The System Management Server facilitates encrypted communications for your Plant SCADA system. To connect a computer to a System Management Server, use the System Management Server page in **Configurator**. See [Connect a Computer to a System Management Server](#).

## Set up deployment

If your OPC UA Server is running on a remote computer, set it up for deployment.

- Set up a deployment server in your system (see [Configure a Deployment Server](#)).
- Configure a deployment client on the OPC UA Server so that it can receive project versions from the deployment server (see [Configure a Deployment Client](#)).

## Confirm the Configurator settings for the OPC UA Server

The required settings are available on the **OPC UA Server** page within the **Plant SCADA** branch of Configurator. See [Use Configurator to Set Up an OPC UA Server](#).

## Enable encryption

An OPC UA Server will only connect to your Plant SCADA servers over an encrypted connection. You need to confirm the following:

- Plant SCADA is configured to run with encryption enabled. This is achieved via the **Encryption** page in Configurator (see [Enable Encryption](#)).

Running in mixed mode is acceptable as long the remaining prerequisites are also met.

- Runtime Manager needs to be configured to run as a service.

This is required to enable encryption. The required settings are available on **Configurator's Computer Setup** page (see [Configure a Runtime Computer for Encryption](#)).

- In your project settings, confirm the **DNS Name** field is configured for each entry in the **Computers** activity. This is also required to enable encryption.
- Run the Computer Setup Wizard and confirm on the **Network Model** page that networking is enabled. This is required even if you are running the OPC UA Server and the Plant SCADA servers on a single computer.

## Set up user authentication

If **Enable Anonymous Access** is disabled for the OPC UA Server, a user name and password will be required to connect to the OPC UA server. Only authentication with Windows® users is supported (Plant SCADA users are not supported).

The user must also be a member of a Windows group that is mapped to a valid Role within your Plant SCADA project. See the topic [Roles](#) within the *Runtime System Security* section of the Plant SCADA documentation.

Anonymous access needs to be disabled to support tag writes from an OPC UA client application.

## Enable tag writes for your variable tags

Your variable tags can be configured to support writes from an OPC UA client application.

---

**Note:** Tag writes will only work if anonymous access is disabled for the OPC UA Server.

To enable tag writes, you need to set the **Write Roles** property for each variable tag that will have writes enabled. The Property Grid allows you to select one of the roles configured in your Plant SCADA project, or you can manually enter a comma-separated list to include multiple roles.

To write to a variable tag, the user that is currently logged in to the OPCUA client application needs to be part of a Windows™ domain group that is associated with the role specified in the **Write Roles** property.

---

**Note:**

- Plant SCADA runtime does not support online changes for variables, which means any changes you make to the Write Roles setting for a variable tag will not be implemented until you restart the I/O server.
- If you have renamed any Roles, you should also restart the I/O server to synchronize the changes at runtime, particularly if you use the Cicode function [UserUpdateRecord](#) to recompile a local project configuration.

## OPC UA Client Driver

The **OPC UA Client Driver** page appears under Plant SCADA in [Configurator](#), when the Plant SCADA OPCUA driver

is installed. This is one of the communication drivers that is installed with Plant SCADA by default (see [Install Communication Drivers](#)).

---

**Note:** The Configurator's OPC UA Client Driver page is used to manage secure device communications via an external OPC UA server. It is not related to the Plant SCADA OPC UA Server, which delivers runtime data to third-party OPC UA clients.

Plant SCADA's OPCUA driver supports the use of user credentials when connecting to a device. It allows you to establish a secure connection between the driver and an external OPC UA server using X509 client and server certificates. The driver needs access to the server certificate and the server needs access to the driver's client certificate.

To support this, the OPC UA Client Driver page under Plant SCADA in Configurator allows you to:

- Create security settings.
- Select the client certificate from the Windows® certificate store.
- Import the client certificate from a file into the Windows certificate store.
- Issue a Plant SCADA OPC UA client certificate, store it in the Windows certificate store and export it to a file.
- Import the server certificate from the file into the Windows certificate store.
- Test connection to the OPC UA server.

For further instructions, see the topic *Security Configuration* in the OPCUA driver documentation. You can access the driver documentation via the **Driver Reference Help** toolbar button in Plant SCADA Studio's **Topology | I/O Devices** view.

## Install a Runtime-only Client

You can use the Plant SCADA installer to set up a runtime-only client. This will create dedicated runtime display client that does not include the components required to configure a project.

### See Also

- [Runtime-only Client Requirements](#)
- [Runtime-only Client Installation Procedure](#)
- [Use Configurator to Set Up a Runtime-only Client](#)

## Runtime Only Client Installation Requirements

This section describes the hardware and software requirements for a Runtime Only Client.

Before you install Plant SCADA, it is recommended that you install the latest updates from Microsoft® for your operating system and system software.

### Hardware Requirements

The hardware requirements for a Runtime Only Client are as follows.

CPU PassMark®	2000
Cores	2
RAM	4 GB
HDD	10 GB
Graphics	DirectX 11 or later with WDDM 1.0 Driver. 128 MB of dedicated VRAM.
Screen Resolution	1920 x 1080
Network	100 Mb

The complexity of your pages such as the number of graphical animations and Cicode running in the background will impact your client CPU choice. It is recommended to use a higher performing PC with high clock speed when building complex user interfaces.

As a guide, the following will require high clock speed to maintain a client CPU load of less than 25% on a single core:

- HD user interface with 50 complex Genies
- UHD4K user interface with 100 complex Genies.

If the deployment feature is being used, the HDD needs to have the required space for the number of configured project versions + 2. You should remove unused project versions periodically to reduce HDD usage.

The suggested screen resolution supports lower and higher resolutions including 4K UHD (3840 x 2160). However, 4K UHD will require a high clock speed CPU. A multi-monitor client may also require a higher clock speed CPU and more memory.

## Operating System Requirements

The following operating systems are supported (64-bit only):

- Windows 11 21H2 (or later)
- Windows 10 21H2 (or later)
- Windows 10 LTSC 1607 (or later)
- Windows Server 2016
- Windows Server 2019
- Windows Server 2022

Microsoft .NET Framework 4.8 is required. It will be installed by Plant SCADA if not detected.

When .NET installation is complete, you may be prompted to restart your computer. If this occurs, you should initiate a restart before you continue.

## See Also

- [Runtime-only Client Installation Procedure](#)
- [Use Configurator to Set Up a Runtime-only Client](#)

## Runtime Only Client Installation Procedure

This section describes the pages displayed by the Plant SCADA installer and the setting required for a Runtime Only Client. Use the **Back** and **Next** buttons to navigate through the included pages.

Installer Page	Notes
Installation Documentation	If required, view the available documents.
License Agreement	Accept the terms of the License Agreement.
What is the role of this computer?	Select <b>Runtime Only Client</b> .
The selected products/components will be installed/upgraded.	<p>By default, only the following component is selected when you install a Web Server.</p> <ul style="list-style-type: none"><li>• AVEVA Plant SCADA<ul style="list-style-type: none"><li>• Plant SCADA Runtime</li><li>• Extensions</li><li>• Software Update Tool</li></ul></li></ul> <p>You can use this page to customize the list of components selected for installation.</p>
Access Restriction	<p>This page provides information about the local Windows groups the installer will create to manage security (see the section <i>Windows User Groups and Security Roles</i> in the topic <a href="#">Installation Information</a>).</p> <p><b>Note:</b> This page will only appear if you are installing Plant SCADA on a clean computer (one that has not hosted an earlier version). If you are upgrading from an earlier version, equivalent user groups will already exist on the computer.</p> <p>If you want your current Windows user account to be included in the groups that are being created, select <b>Add the current Windows user to these groups</b>.</p> <p>A dialog will let you know that the group membership changes will not occur until after a restart.</p>
Firewall	If the installer detects that the computer has Windows Firewall enabled, you will be asked if you would like the installer to modify your settings.

Installer Page	Notes
	<p>If you select <b>Yes</b>, Plant SCADA Runtime will be added to the list of authorized programs.</p> <p>See the section <i>Windows Firewalls</i> in the topic <a href="#">Installation Information</a>).</p>
Customize the installation location	<p>This page identifies the destination folders for:</p> <ul style="list-style-type: none"><li>• The selected Plant SCADA program files.</li><li>• The Plant SCADA User and Data folders.</li></ul> <p>You can modify the folder locations by clicking the <b>Change</b> buttons and selecting alternative locations.</p> <p><b>Note:</b> The installation folder and the 'User and Data' folder cannot be the same, or a sub-directory of one another. You should also avoid selecting your Windows user folder.</p>
Ready to Install the Program	<p>This page lists the components that will be installed based on your selections.</p> <p>If required, use the <b>Back</b> button to make any modifications.</p> <p>Click <b>Install</b> to start the installation process.</p>

When installation is complete, a dialog will appear asking if you would like to launch [Configurator](#). This may take a few minutes if you have installed some external components (for example, AVEVA Enterprise Licensing components).

---

**Note:** The installation experience is not complete until after you exit Configurator. A restart request is likely to appear at this point. To avoid permission issues, you should close Configurator and confirm if a reboot is required before you run Plant SCADA.

---

## See Also

[Use Configurator to Set Up a Runtime-only Client](#)

## Use Configurator to Set Up a Runtime-only Client

When your Plant SCADA installation is complete, a dialog will appear asking if you would like to launch [Configurator](#). For a runtime-only client, you can use the Configurator's **Computer Setup** page under Plant SCADA to adjust the following runtime environment settings for a computer.

## Project Run Path

This setting allows you to specify whether or not the computer will receive project versions from a Deployment Server.

**To set the Project Run Path:**

1. In the panel on the left side of the **Configurator**, select **Plant SCADA | Computer Setup**.  
The **Computer Setup** page appears.
2. Navigate to the **Project Run Path** section.

**Project Run Path**

*Runtime can run the project selected in Plant SCADA Studio, or the project that has been deployed by the Deployment Server*

- [Run the project selected in Plant SCADA Studio](#).  
 [Run the project deployed from the Deployment Server](#).

3. Select one of the following options to determine which project will be launched by Runtime Manager:
  - **Run the project currently selected in Plant SCADA Studio** — This is selected by default and will run the project from Plant SCADA Studio.
  - **Run the project deployed from the Deployment Server** — Select this option to set up a computer for deployment. This option allows you to run the deployed projects from the deployment server.
4. To apply your settings, click the **Configure** button.

**Note:** If Plant SCADA Runtime is running as a service and the **Project Run Path** option in the Configurator is changed to **Run the project deployed from the Deployment Server**, you need to restart the Runtime Manager service for deployment to function correctly.

If you intend to use Plant SCADA's Deployment feature to distribute project versions to this client, you will need to use Configurator to connect the computer to a Deployment Server. See [Configure a Deployment Client](#).

## Runtime Manager Configuration

The Runtime Manager can be configured to run as a Windows® service. This allow a project to be deployed and run on a computer without having to change other settings.

You need to select **Run Runtime Manager as a service** under the following circumstances:

- When a deployment client is configured on a computer and you want to push project updates using "Force" mode.
- If you are using encrypted communications, and the current user on a deployment client may not have required privileges.

**Note:** The current user on a deployment client needs to be a member of the **Configuration Users** security role to run the deployment client (see [Security Roles](#)).

**To run Runtime Manager as a service:**

1. In the panel on the left side of the **Configurator**, select **Plant SCADA | Computer Setup**. The **Computer Setup** page appears. Navigate to the **Runtime Manager Configuration** section of the dialog.

### Runtime Manager Configuration

The Runtime Manager can be configured to run as a service or as a standard process. When run as a service, Runtime Manager allows projects to be deployed to the machine without having to change any other settings.

Run Runtime Manager as a service

#### 2. Select **Run Runtime Manager as a Service**.

**Note:** If you are using Deployment to run a project and do not select the **Run Runtime Manager as a Service** option, you will need to manually start the Plant SCADA Runtime Manager before deploying the project.

#### 3. To apply your settings, click the **Configure** button.

**Note:** If you have configured a Plant SCADA OPC DA server on a computer that is running Plant SCADA as a Windows service, you will need to make the additional configuration changes. See the topic [Running an OPC DA Server as a Service](#) in the *Runtime* section of the Plant SCADA documentation.

## AVEVA™ Enterprise Licensing

The AVEVA Enterprise Licensing system is a common platform that allows you to manage AVEVA Enterprise Software product licenses.

The system was introduced to allow centralized license management, remove the need for dongles, and increase license security by use of activated licenses. It supports flexible topologies for any size system.

The following describes the components of the AVEVA Enterprise Licensing system.

### License Manager

The License Manager allows you to access and maintain licenses for certain AVEVA Enterprise Software products in your different environments using its scalable, flexible design features:

- Browser-based for scalability and ease of use; can be remotely accessed by any of the supported web browsers.
- Light-weight, standalone software you can install on the same node as the License Server computer, or on any other node based on your deployment needs.
- Manage one or multiple License Servers to organize the licensing requirements for your environment.
- Access details about your licenses, such as usage information, from the License Manager interface.

You can use Plant SCADA Studio to launch Enterprise License Manager from any installed location within your Plant SCADA system. See [Launch Enterprise License Manager from Plant SCADA Studio](#).

### License Server

License Server provides the functionality to acquire, store, maintain, and serve licenses to your installed AVEVA Enterprise software.

- Licenses are hosted and maintained on the License Server.
- Securely serve any type of software applications being licensed, including Windows browsers, tablets, and mobile devices.

- Provide current license usage information.

**Note:** Plant SCADA 2023 R2 allows you to run your AVEVA Enterprise License Server in **Secure Mode**. This facilitates encrypted communication for any Plant SCADA computers that connect with your license server to acquire licenses. For more information, see *Running in Secure Mode* in the topic [Use Configurator to Set Up an Enterprise License Server](#).

### Activation Server

Activation Server is a cloud-based Internet-accessible server to which the License Manager connects for license activation. The License Manager connects to the AVEVA Activation Server in the Cloud only temporarily during the activation process.

### Installed Products

Once the AVEVA Enterprise Software product is installed it relies on the licensing system to enable its functionality. Following installation, you use Configurator to specify from which License Server you want the product to get its license.

- If a License Server is installed on the same computer as the product, it will be selected in Configurator by default.
- If the product is installed on a different computer than the License Server, you can use Configurator to point the product to the required License Server computer.

You need to configure the connection to the License Server for any computers that need to acquire licenses. See [Use Configurator to Set Up an Enterprise License Server](#).

All AVEVA products installed on the same computer need to point to the same License Server.

If you are implementing a distributed topology, we recommend creating a network diagram as an initial step. This will allow you to visually model your system before you decide where to install the License Manager and License Server components. For more information, see [AVEVA™ Enterprise Licensing Topologies](#).

**Note:** When working with virtual environments, you need to deactivate all licenses in a License Server before cloning a virtual machine. If this is not done, the cloned License Server will go into Grace Period mode of 15 days, after which it will cease serving licenses.

## See Also

[Enterprise License Server Installation Requirements](#)

[Enterprise License Server Installation Procedure](#)

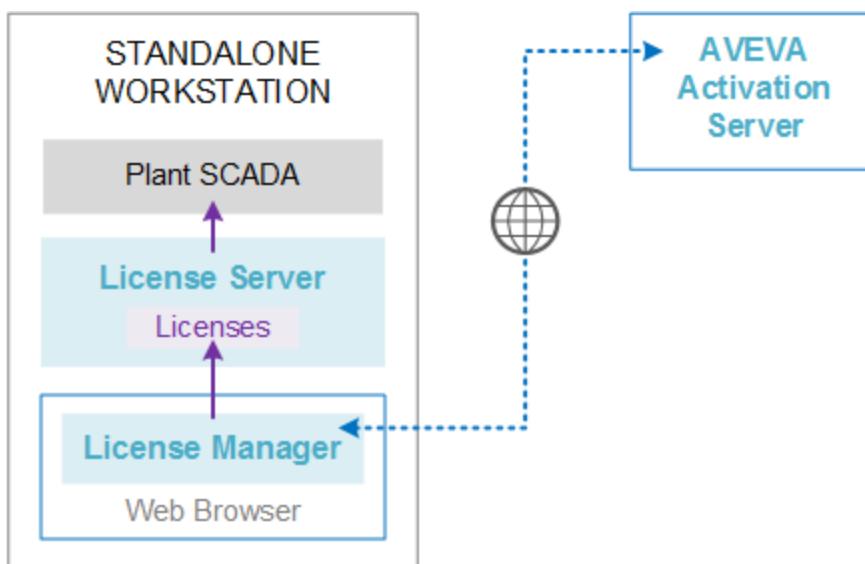
## AVEVA™ Enterprise Licensing Topologies

Use the models described in this topic to design the topology for your [AVEVA™ Enterprise Licensing](#) platform. This will help you decide where you need to install the License Manager and License Server components.

There are three topology types to consider.

### Single Node Topology

A single node topology involves a computer with Plant SCADA, License Manager, and License Server installed. In this configuration, the communication between the server components, License Server, and License Manager is local. Internet access is required to connect to the Activation Server.



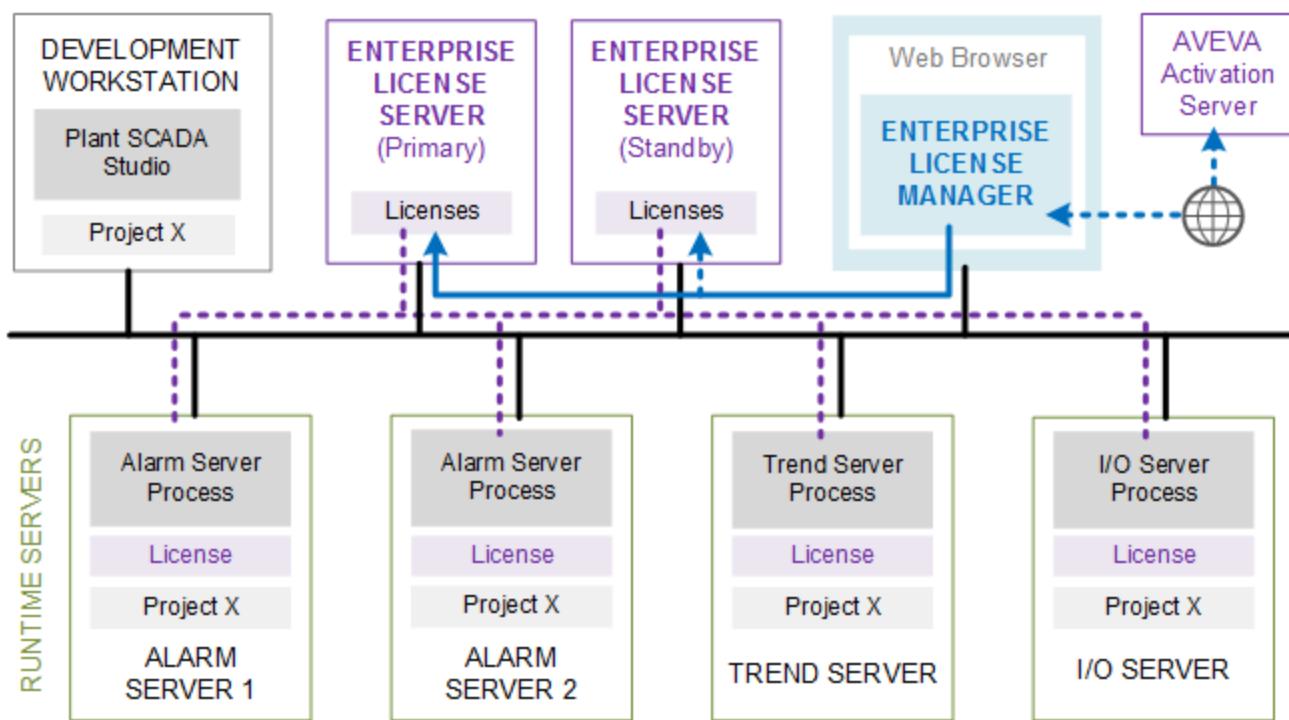
Typically a single node topology would be used to test a project on a development workstation. It can also be used for training or evaluation purposes.

### Distributed Topology

The licensing components can be installed on separate computers as long as there is reliable network connectivity between them. This type of topology is suitable for systems where you need to manage licenses for multiple installations of Plant SCADA across a distributed runtime system.

In the following example:

- The **Enterprise License Manager** components are installed on a computer with Internet access to the Activation Server. The interface runs locally in a web browser.
- The **Enterprise License Server** components are installed on two separate computers that are acting as a redundant pair.
- The Plant SCADA runtime computers each require network access to the License Servers.



This type of topology allows you to manage multiple License Servers for a runtime system from a single License Manager browser interface.

License Server redundancy is highly recommended when centralizing your licenses. It is available at no additional cost. A paired set of License Servers keeps licenses, reservations and the status of licenses synchronized to help maintain stability for the runtime system.

If you use clustering to manage a system that is physically distributed across multiple sites, it is recommended that you have a pair of redundant license servers at each location.

For information and procedures about implementing redundancy, see the topic *Working With License Server Redundancy* in the AVEVA Enterprise Licensing documentation in [AVEVA™ Help](#).

#### [Disconnected Topology](#)

This topology is required if an internet connection is not available. For example, a firewall or other constraints may exist between the License Manager components and the Activation Server.

In this scenario, you need to set up a separate computer with Internet access to the License Activation Web Page and Activation Server and then complete the activation process in "offline" mode. For more information, see the topic *Activating Licenses in Offline Mode* in the AVEVA Enterprise Licensing documentation in [AVEVA™ Help](#).

---

**Note:** If you are planning to run a system that includes a combination of client, server, and HMI licenses, you need to confirm that your Plant SCADA computers have the appropriate role configured. This will allow an Enterprise License Server to correctly allocate different license types. Computer roles are configured using the [Computer Role Setup](#) page of the Setup Wizard.

For an example of how to acquire, activate and distribute your licenses, see [AVEVA™ Enterprise Licensing Workflow](#) in the Plant SCADA documentation.

For more detailed information see the AVEVA Enterprise Licensing documentation in [AVEVA™ Help](#).

## AVEVA™ Enterprise Licensing Workflow

The following example of the AVEVA Enterprise Licensing workflow assumes that both License Manager and License Server are installed on the same computer, and that the computer can connect to the Internet.

### 1. Purchase one or more AVEVA Enterprise Software products

Along with your installation media or downloads, you will receive one or more entitlement files containing license details.

### 2. Start License Manager

It opens in a browser window and accesses the installed License Manager components.

### 3. Select the License Server

Use the License Manager to select an available server to host the licenses you want to activate. You can also add a new license server.

---

**Note:** You need to configure the connection to the License Server on any computer that needs to acquire licenses. See .

---

### 4. Import the entitlement file

Once you import the entitlement file, the License Manager displays the available licenses.

The entitlement file contains information for specific product licenses that are imported into the License Manager to activate purchased licenses. It is sent to you in an email upon purchase of your product licenses. It is in a zipped .xml file format.

### 5. Activate licenses

License Manager connects to the AVEVA Enterprise Activation Server over the Internet to activate the selected licenses on the License Server. The Activation Server activates licenses on the License Server.

### 6. Manage your licenses

Licenses are now available for their respective product to use. You can now:

- Activate new licenses
- Deactivate current licenses
- Reserve and unreserve licenses
- Checkout licenses.

In some cases, you may need to reserve a license for a specific computer using the device reservation feature. For example, it is recommended that you checkout a license for any critical runtime computers. For more information, see [Allocating AVEVA™ Enterprise Licenses](#).

---

**Note:** Plant SCADA only supports device reservations. You cannot reserve licenses for specific users.

---

### 7. Install and configure Plant SCADA

When running, Plant SCADA will acquire licenses from the License Server selected in Configurator (see Use Configurator to Set Up an Enterprise License Server). They will release them when no longer needed.

---

**Note:** For more detailed information see the AVEVA Enterprise Licensing documentation in [AVEVA™ Help](#).

---

## Enterprise License Server Installation Requirements

This section describes the hardware and software requirements for a computer that will host an AVEVA™

Enterprise License Server.

Before you install this component, it is recommended that you install the latest updates from Microsoft® for your operating system and system software.

## Hardware Requirements

The hardware requirements for a Enterprise License Server are as follows:

Cores	4
Memory	4 GB

## Operating System Requirements

The following operating systems are supported (64-bit only):

- Windows 11 21H2 (or later)
- Windows 10 21H2 (or later)
- Windows 10 LTSC 1607 (or later)
- Windows Server 2016
- Windows Server 2019
- Windows Server 2022

Microsoft .NET Framework 4.8 is required. It will be installed by Plant SCADA if not detected.

**Note:** Plant SCADA 2023 R2 installs version 4.0 of AVEVA Enterprise Licensing. If your system includes computers that are not upgraded to AEL 4.0, you will need to install **AVEVA Enterprise Licensing Server Legacy Support** on your Enterprise License Server. This enables backwards compatibility for clients nodes that are running an earlier version of AEL. You can select this option on the products/components page on the Plant SCADA installer. You should also check the **Legacy Server Port** setting in Configurator. See [Use Configurator to Set Up an Enterprise License Server](#).

## See Also

[Enterprise License Server Installation Procedure](#)

## Enterprise License Server Installation Procedure

This section describes the pages displayed by the Plant SCADA installer and the settings required for an AVEVA™ Enterprise License Server. Use the **Back** and **Next** buttons to navigate through the included pages.

Installer Page	Notes
Installation Documentation	If required, view the available documents.
License Agreement	Accept the terms of the License Agreement.

Installer Page	Notes
What is the role of this computer?	Select <b>Server</b> .
Select the server(s) you want to install	Select <b>AVEVA Enterprise License Server</b> . If required, you can also select other server types.
The selected products/components will be installed/upgraded.	<p>By default, the following components are selected when you install an Enterprise License Server.</p> <ul style="list-style-type: none"><li>• AVEVA Enterprise License Manager</li><li>• AVEVA Enterprise License Server.</li></ul> <p>If you are going to connect the computer to a License Server on a remote computer, you only need to install the License Manager.</p> <p><b>Note:</b> Plant SCADA 2023 R2 installs version 4.0 of AVEVA Enterprise Licensing. If your system includes computers that are not upgraded to AEL 4.0, you will need to install <b>AVEVA Enterprise Licensing Server Legacy Support</b> on your Enterprise License Server. This enables backwards compatibility for clients nodes that are running an earlier version of AEL.</p>
Access Restriction	<p>This page provides information about the local Windows groups the installer will create to manage security (see the section <i>Windows User Groups and Security Roles</i> in the topic <a href="#">Installation Information</a>).</p> <p><b>Note:</b> This page will only appear if you are installing Plant SCADA on a clean computer (one that has not hosted an earlier version). If you are upgrading from an earlier version, equivalent user groups will already exist on the computer.</p> <p>If you want your current Windows user account to be included in the groups that are being created, select <b>Add the current Windows user to these groups</b>.</p> <p>A dialog will let you know that the group membership changes will not occur until after a restart.</p>
Firewall	<p>If the installer detects that the computer has Windows Firewall enabled, you will be asked if you would like the installer to modify your settings.</p> <p>If you select <b>Yes</b>, Plant SCADA Runtime will be added to the list of authorized programs.</p> <p>See the section <i>Windows Firewalls</i> in the topic <a href="#">Installation Information</a>).</p>

Installer Page	Notes
Customize the installation location	<p>This page identifies the destination folders for:</p> <ul style="list-style-type: none"><li>• The selected Plant SCADA program files.</li><li>• The Plant SCADA User and Data folders.</li></ul> <p>You can modify the folder locations by clicking the <b>Change</b> buttons and selecting alternative locations.</p> <p>You can also <b>Reset</b> the default locations.</p> <p><b>Note:</b> The installation folder and the 'User and Data' folder cannot be the same, or a sub-directory of one another. You should also avoid selecting your Windows user folder.</p>
Ready to Install the Program	<p>This page lists the components that will be installed based on your selections.</p> <p>If required, use the <b>Back</b> button to make any modifications.</p> <p>Click <b>Install</b> to start the installation process.</p>

When installation is complete, a dialog will appear asking if you would like to launch [Configurator](#). This may take a few minutes if you have installed some external components (including AVEVA Enterprise Licensing components).

---

**Note:** The installation experience is not complete until after you exit Configurator. A restart request is likely to appear at this point. To avoid permission issues, you should close Configurator and confirm if a reboot is required before you run Plant SCADA.

---

## See Also

[Use Configurator to Set Up an Enterprise License Server](#)

## Use Configurator to Set Up an Enterprise License Server

You need to configure a connection to a License Server on any computer that needs to acquire licenses.

To configure an Enterprise License Server, open [Configurator](#) and go to the **Select License Server** page located under **AVEVA Enterprise Licensing**.

---

**Note:** Before running the Configurator, make sure that ports 55559 and 59200 are open. If you have also installed **AVEVA Enterprise Licensing Server Legacy Support** (see **Legacy Server Port** below), enable port 55555.

---

- **Primary Server Name**

By default, the installation process identifies the local computer as the primary server and enters the computer name as the Primary Server Name.

If you want to point the local computer a different License Server on your network, you will need to manually enter the computer name as Plant SCADA does not support network discovery for Enterprise License Servers.

- **Server Port**

The default Server Port is 55559. You can specify a different port, if necessary.

- **Agent Port**

The default Agent Port is 59200. You can specify a different port, if necessary.

- **Legacy Server Port**

Plant SCADA 2023 R2 installs version 4.0 of AVEVA Enterprise Licensing. If your system includes computers that are not upgraded to AEL 4.0, you can enable backwards compatibility by installing **AVEVA Enterprise Licensing Server Legacy Support** on your Enterprise License Server. This is achieved via the products/components page of the Plant SCADA installer.

The **Legacy Server Port** represents the port used by legacy client nodes to acquire licenses from the license server. The default value is 55555.

You can test your connection to the License Server by selecting **Test Connection**.

Press **Configure** to save all your changes.

---

**Note:** If you disconnect a computer from an Enterprise License Server, the licenses that have been acquired from the server will be released.

## Running in Secure Mode

Plant SCADA 2023 R2 allows you to run your AVEVA Enterprise License Server in **Secure Mode**. This facilitates encrypted communication for any Plant SCADA computers that connect with your license server to acquire licenses.

Secure Mode is enabled by default when your Enterprise License Server is connected to a System Management Server. You can check if Secure Mode is enabled by viewing the **Secure** page in the **AVEVA Enterprise Licensing** branch of Configurator.

This page will indicate if the license server is currently configured to support Secure Mode. If it is not, it will provide instructions on how to set it up.

## Install OPC Factory Server

Plant SCADA allows you to install a copy of Schneider Electric's OPC Factory Server.

Based on the OPC protocol, OPC Factory Server enables Windows-based OPC client applications to communicate with the following Schneider Electric PLCs:

- TSX Compact
- Micro
- TSX Momentum
- TSX/PCX Premium
- Quantum
- M340
- TSX Series 7
- TSX S1000.

---

**Note:** A Plant SCADA server license includes an OFS license. This means you need to install Plant SCADA before you install OPC Factory Server so it can be licensed using the Plant SCADA license. This will allow the correct part and serial number combination to be registered.

---

OPC Factory Server can be installed on the following operating systems (64-bit only):

- Windows® Server 2022
- Windows® Server 2019
- Windows® Server 2016
- Windows 11 21H2 (or later)
- Windows 10 21H2 (or later)
- Windows 10 LTSC 1607 (or later).

To install OPC Factory Server, run the **Setup.bat** file located in the following folder in Plant SCADA's installation media:

...InstallFiles\Extras\OFS vX.XX [SPn]

Details on the installation options for OPC Factory Server can be found in the **Documentation** folder within this directory.

## Installation Troubleshooting

Log files are available for a Plant SCADA installation. Each installed component has one or more log files in their respective "ID" folder. For example, Plant SCADA creates the following installation log files:

C:\Program Files (x86)\Common Files\ArchestrA\Install\{692B4E4F-CC1F-4AA4-AEE8-1C22F7BA336B}\ILog<DateTimeStamp>.log

C:\Program Files (x86)\Common Files\ArchestrA\Install\{692B4E4F-CC1F-4AA4-AEE8-1C22F7BA336B}\MSI<DateTimeStamp>.log

The log files will notify you if Plant SCADA's security configuration was not successful during installation. You should take the action described below if one of the following warning messages is displayed.

- **Installation successfully created the local Windows groups but was not able to add the current user to them. You can manually add the user to the groups.**

The account that was logged in during installation was not added to the Windows groups created by Plant SCADA (see *Windows User Groups and Security Roles* in the topic [Installation Information](#)).

If required, you can use the **Security Roles** page in Configurator to associate this user with the required security roles (see [Modifying the Members of a Security Role](#)).

- **One or more of the members of a security role could not be validated. Please check the Security Roles tab in the Configurator.**

This indicates the users and local user groups associated with Plant SCADA's Security Roles could not be validated. For example, the domain server may have been unreachable or a user name was not valid.

To fix this, go to the **Security Roles** page in Configurator. Verify the users and local user groups associated with each security role, then click **Configure**. The Configurator Messages panel will indicate if any further configuration changes are required. See [Modifying the Members of a Security Role](#).

- **An error occurred applying security to resources. Please check the Security Roles tab in the Configurator.**

Plant SCADA will attempt to restrict access to some folders and services that need to be secured. This message indicates that the required security settings were not successfully applied.

To fix this, go to the **Security Roles** page in Configurator. Select **Configure** to re-apply the security settings.

- **Could not initialize security model because AVEVA Data Store service is not running. Please check the AVEVA Data Store service is running in the Windows Services Manager.**

Locate **AVEVA Data Store** in Windows Services Manager and confirm that its **Status** indicates it is "Running". You can **Restart** the service if required. Its **Startup Type** should also be set to "Automatic".

## Unattended Silent Installation

The installation of Plant SCADA can be executed via a command line with the required components specified via a configurable response file.

**Note:** A silent installation will only work on a clean computer. If you have any existing dependent components on a computer from an earlier installation (such as Operations Control Logger or Platform Common Services), a silent install will not be successful. If this occurs, you can complete your installation by manually running the Plant SCADA Installer.

### Response File

A response file specifies the set of components that will be installed during a silent installation. It represents the forms normally seen when interacting with the user interface of the Plant SCADA installer. The available options for each form are configured via a set of properties.

Four preconfigured response files are available for use:

- **All.txt** – installs all components
- **DevelopmentWorkstation.txt** – installs components for a development workstation
- **RuntimeClient.txt** – installs components for a runtime client
- **Server.txt** – installs components for a runtime server.

These can be used to install a typical set of components, or they can be used as reference for your own customized response file (see [Customize a Response File](#)).

The four files are available from the following directory within the Plant SCADA installation media:

#### InstallFiles\ResponseFiles

The following example shows the content of the Server.txt response file.

```
<responsefile>
  <install>
    ComputerRoleSelectionForm.SelectedComputerRoleName=Server
    ServerRoleSelectionForm.ServerRolesList=RuntimeServer,DeploymentServer,
    IndustrialGraphicsServer,OPCUAServer,EnterpriseLicenseServer
    FeaturesForm.SFeatureList=AVEVA Plant SCADA.Configuration,AVEVA Plant SCADA.Runtime,
    AVEVA Plant SCADA.InTouchWebClient,AVEVA Plant SCADA.OpcUaServer,
    AVEVA Plant SCADA.DeploymentServer
    SecurityGroupForm.AddUserToGroups=true
    FirewallForm.ModifyFireWall=true
    CustomDestinationSelectionForm.InstallDirPath=C:\Program Files (x86)\AVEVA Plant
```

```
SCADA
CustomDestinationSelectionForm.ProgramDataPath=C:\Program Data\AVEVA Plant SCADA 2023
R2
</install>
</responsefile>
```

You can also specify Configurator settings within a response file. See [Add Configurator Settings to a Response File](#).

## Silent Install/Uninstall Commands

The following commands can be executed from a command prompt:

- **Command for installation:**

```
Setup.exe /silent "<Response File Path>"
```

The specified response file needs to be text-based. A full absolute path needs to be provided. For example:

```
Setup.exe /silent "D:\PlantSCADA\MyResponseFiles\All.txt"
```

You can also use the "silentnoreboot" option to suppress a machine reboot after installation completes, for example:

```
Setup.exe /silentnoreboot "<Response File Path>"
```

- **Command for uninstallation:**

```
Setup.exe /silentuninstall <ID of Plant SCADA>
```

For example:

```
Setup.exe /silentuninstall {XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}
```

The ID for Plant SCADA can be retrieved from the **Setup.xml** file, located in the **InstallFiles** directory within the installation media.

Search for the ID value within the **<mainproduct>** tags.

**Note:** Do not manually uninstall any non-core components (such as AVEVA Enterprise Licensing, the Sentinel Driver, PCS, Operation Control Logger or AVEVA Help) prior to running a silent uninstall of Plant SCADA as the silent uninstall may not be successful.

When you run a silent uninstallation, you need to follow the procedure below when also uninstalling any non-core components.

1. Uninstall AVEVA Plant SCADA's core components (using the silent uninstall command).
2. Uninstall AVEVA Enterprise Licensing components:
  - AVEVA Enterprise License Server
  - AVEVA Enterprise License Manager
  - AVEVA Enterprise Licensing Platform.
3. Uninstall Schneider Electric Software Update.
4. Uninstall Sentinel System Driver Installer.
5. Uninstall AVEVA Platform Common Services.
6. Uninstall Operations Control Logger.
7. Uninstall AVEVA Help.

**Note:** You can also use a command to silently install a Plant SCADA update. See [Silently Install a Plant SCADA Update](#).

## Customize a Response File

To create a customized set of options for a silent installation of Plant SCADA, you should start by creating a copy of one of the provided response files. Four are available for use:

- **All.txt** – installs all components (including all communication drivers)
- **DevelopmentWorkstation.txt** – a typical set of components for a development workstation
- **RuntimeClient.txt** – a typical set of components for a runtime client
- **Server.txt** – a typical set of components for runtime server.

These are available from the following directory within the Plant SCADA installation media:

### InstallFiles\ResponseFiles

You can then use the following set of properties to specify the settings for the forms normally seen when interacting with the Plant SCADA installer.

Form Name	Properties	Allowable Values
ComputerRoleSelectionForm	SelectedComputerRoleName	One of the following values: <ul style="list-style-type: none"><li>• DevelopmentWorkstation</li><li>• Server</li><li>• RuntimeOnlyClient</li></ul>
ServerRoleSelectionForm	ServerRolesList	Multiple server names separated by commas: <ul style="list-style-type: none"><li>• RuntimeServer</li><li>• DeploymentServer</li><li>• IndustrialGraphicsServer</li><li>• OPCUAServer</li><li>• EnterpriseLicenseServer</li></ul>
FeaturesForm	SFeatureList	ALL or multiple feature names separated by commas (see below).
SecurityGroupForm	AddUserToGroups True adds the current Windows user to created groups.	True or False
FirewallForm	ModifyFireWall	True or False
CustomDestinationSelectionForm	InstallDirPath	A valid absolute folder path

	ProgramDataPath	A valid absolute folder path
--	-----------------	------------------------------

**Note:** A response file needs to include an entry for each of the six forms it represents. If you remove any of the forms, installation will not succeed.

To specify the features that are installed via the **FeaturesForm.SFeaturesList** property, you can enter the following as a comma separated list.

Components	Description	SFeaturesList setting
All components	Includes all components listed below.	ALL
Runtime client	Provides runtime client display and server level processing of I/O, trends, alarms, reports, and CtAPI.	AVEVA Plant SCADA.Runtime
Configuration Environment	Plant SCADA Studio, graphics, Cicode Editor and other configuration tools for building your system.  This option also requires the Runtime and Platform Common Services components.	AVEVA Plant SCADA.Configuration
Deployment Server	Provides server functionality for managing the runtime versions of your system and deployment to runtime computers.	AVEVA Plant SCADA.DeploymentServer
Industrial Graphics Web Server	Provides server functionality for managing the runtime versions of your system and deployment to runtime computers.	AVEVA Plant SCADA.InTouchWebClient
OPC UA Server	Provides server functionality that allows OPC UA clients to browse and subscribe to Plant SCADA I/O data.	AVEVA Plant SCADA.OpcUaServer
Project DBF Add-in for Microsoft Excel	Provides bulk editing of configuration databases in Microsoft Excel.	Extensions.BinFiles
Software Update by Schneider Electric	Provides notification of updates, new versions and telemetry information.	Extensions.SEP_Features
AVEVA License Server	Provides server functionality to acquire, store, maintain, and serve licenses to your installed AVEVA	AVEVA Common Components.AELicenseServer

	software.	
AVEVA License Manager	Browser-based client application for accessing and maintaining licenses for AVEVA Enterprise Software products.	AVEVA Common Components.AELicenseManager
Platform Common Services  <b>Note:</b> You only need to include this component in a response file if you are setting up a computer as a standalone System Management Server. It will automatically be installed as a prerequisite for any other Plant SCADA components.	Provides the following services: System Management Server Identity Management Service Data Storage Service Diagnostic Portal	AVEVA Common Components.ASBRuntime
Plant SCADA Communications Drivers  These can include:  ABCLX ABMLXEIP ABTCP BACNET CTICMP DNPR GETCP IEC870IP KNX MELSECQ MODBUS MODBUA MODNET OFSOPC OMFINS OPC OPCLX OPCUA S7TCP SNMPII	Installs software to enable communication with I/O devices using a specific protocol.	AVEVA Plant SCADA Communication Drivers.<driverName>Driver  For example: AVEVA Plant SCADA Communication Drivers.BACNETDriver

**Note:** You can also specify Configurator settings within a response file. See [Add Configurator Settings to a Response File](#).

#### Example

The following demonstrates the response file you could use to install a development workstation with all available features.

```
<responsefile>
<install>
ComputerRoleSelectionForm.SelectedComputerRoleName=DevelopmentWorkstation
```

```
FeaturesForm.SFeatureList=ALL
SecurityGroupForm.AddUserToGroups=true
FirewallForm.ModifyFireWall=true
CustomDestinationSelectionForm.InstallDirPath=C:\Program Files (x86)\AVEVA Plant SCADA
CustomDestinationSelectionForm.ProgramDataPath=C:\Program Data\AVEVA Plant SCADA 2023 R2
</install>
</responsefile>
```

**Note:** You can determine if the silent configuration of plugins has been successful at the end of an installation using the XML log file located at:

**C:\Program Files (x86)\Common Files\ArchestrA\SilentConfigurator**

## See Also

[Unattended Silent Installation](#)

## Add Configurator Settings to a Response File

You can add a section to a response file that allows you to preconfigure the settings that are available in [Configurator](#) for Plant SCADA (and other AVEVA components).

To do this, define a new section within a response file using the following tags:

```
<configurator> <configurator/>
```

You can then use the information in the following tables to define the settings for each page in Configurator when running a silent installation.

**Note:** You cannot configure the authentication settings for an Industrial Graphics Server during a silent installation. If you silently install an Industrial Graphics Server, you will need to manually adjust its settings in Configurator. See [Use Configurator to Set Up an Industrial Graphics Server](#).

You can determine the success of a silent installation using the XML log file located at:

**C:\Program Files (x86)\Common Files\ArchestrA\SilentConfigurator**

### Security Roles page

Configurator Setting	Options	Response file syntax
Security Role Members	<p>Role:User,User; Role:User,User;</p> <p><b>Role</b> needs to be one of the following:</p> <ul style="list-style-type: none"><li>• ConfigUsers</li><li>• ServerUsers</li><li>• RuntimeUsers</li><li>• DeploymentAdmins</li><li>• DeploymentUsers</li><li>• DeploymentUploaders</li><li>• AIGUsers</li><li>• AIGUsersRW</li></ul> <p><b>User</b> needs to be:</p> <ul style="list-style-type: none"><li>• A local user: ".\&lt;UserName&gt;" or</li></ul>	<p>Plant SCADA.</p> <p>SecurityConfiguration.RoleMembers=</p> <p><b>Examples</b></p> <p>To add UserA and UserB to ConfigUsers:</p> <pre>Plant SCADA. SecurityConfiguration.RoleMembers= ConfigUsers:MyComputer\UserA,MyComputer\UserB;</pre> <p>To add UserA and UserB to ConfigUsers and ServerUsers:</p>

	"<ComputerName>\<UserName>" <ul style="list-style-type: none"> <li>• A local Windows group: ".\&lt;GroupName&gt;" or</li> <li>"&lt;ComputerName&gt;\&lt;GroupName&gt;"</li> </ul> <ul style="list-style-type: none"> <li>• A domain group user: "&lt;DomainName&gt;\&lt;UserName&gt;"</li> <li>• A domain group: "&lt;DomainName&gt;\&lt;GroupName&gt;"</li> </ul>	Plant SCADA. SecurityConfiguration.RoleMembers= ConfigUsers:MyComputer\ UserA,MyComputer\ UserB;ServerUsers:MyComputer\ UserA,MyComputer\UserB;
--	---	--

Using a response file to add members to a security role has the following limitations:

- Adding NT Service accounts is not supported.
- If you modify a role, all existing users will be removed (except for any existing NT Service accounts).
- If no valid users are specified for a role, the role will not be modified.
- If the limit of 10 users per role is exceeded in the response file, only the first 10 users will be added. This means if you try to add 12 users, the last two will be ignored.

#### Computer Setup page

Configurator Setting	Options	Response file syntax
Configure Server Password	True or false.	Plant SCADA. ComputerSetup.PartOfTrustedNetwork=
Password / Confirm Password	A valid password.	Plant SCADA.ComputerSetup.ServerPassword=
Project Run Path	True = Run the project deployed from the Deployment Server. False = Run the project selected in Plant SCADA Studio.	Plant SCADA.ComputerSetup.DeploymentEnabled=
Run Runtime Manager as a Service	True or false.	Plant SCADA.ComputerSetup.RuntimeAsService=

#### Encryption page

Configurator Setting	Options	Response file syntax
Enable Encryption	True or false.	Plant SCADA.Encryption.IsEnabled=
Accept encrypted and non-encrypted connections	True or false.	Plant SCADA.Encryption.AcceptLegacyCo

		nnections=
--	--	------------

**Deployment Server page**

Configurator Setting	Options	Response file syntax
AUTHORIZE User Name	A valid user name.	Plant SCADA.DeploymentServer.UserNameValue=
AUTHORIZE Password	A valid password.	Plant SCADA.ComputerSetup.ServerPassword=
SETTINGS Password	A valid password.	Plant SCADA.DeploymentServer.DatabasePasswordValue=
SETTINGS Confirm Password	A valid password.	Plant SCADA.DeploymentServer.ConfirmDatabasePasswordValue=

**Deployment Client page**

Configurator Setting	Options	Response file syntax
CONNECT Deployment Server	A valid Windows computer name for the deployment server.	Plant SCADA.DeploymentClient.SelectedServer=
AUTHORIZE User Name	A valid user name.	Plant SCADA.DeploymentClient.UserNameValue=
AUTHORIZE Password	A valid password.	Plant SCADA.DeploymentClient.UserPasswordValue=
SETTINGS Unpack Rate	Zero (0).	Plant SCADA.DeploymentClient.UnpackRateValue=
SETTINGS Deployed project location	A valid path to the deployed project location.	Plant SCADA.DeploymentClient.DeployedProjectPath=

**OPCUA Server page**

Configurator Setting	Options	Response file syntax
Port Number	A valid port number. The default is 48031	Plant SCADA.OpcUaServer.Port=

Enable encrypted communications	True or false.	Plant SCADA.OpcUaServer.EnableEncryption=
Allow anonymous access	True or false.	Plant SCADA.OpcUaServer.EnableAuthentication=
Disable property and extension	True or false.	Plant SCADA.OpcUaServer.DisablePropertyAndExtensionBrowsing=

See [Use Configurator to Set Up an OPC UA Server](#).

#### AVEVA Enterprise License Server page

Configurator Setting	Options	Response file syntax
Primary Server Name	A valid Windows computer name	AVEVA Enterprise Licensing. LicAPI2.NewServerName=
Server Port	A valid port number. The default is 55559.	AVEVA Enterprise Licensing. LicAPI2.NewPortNumber=
Agent Port	A valid port number. The default is 59200.	AVEVA Enterprise Licensing. LicAPI2.NewAgentPortNumber=
Legacy Port	A valid port number. The default is 55555.	AVEVA Enterprise Licensing. LicAPI2.LegacyPortNumber=

See [Use Configurator to Set Up an Enterprise License Server](#).

#### System Management Server page

Configurator Setting	Options	Response file syntax
System Management Server location	0 = None (No SMS configured) 1 = Remote (Connect to an existing SMS) 2 = Local (This machine is the SMS)	Common Platform. ASBRuntime.ManagementServerLocation=
Connect to an existing System Management Server	A valid Windows computer name.	Common Platform. ASBRuntime.ManagementServerName=
Configure this machine as a redundant SSO Server	True or false.	Common Platform. ASBRuntime.IsRedundantSsoServer=
Advanced Configuration - Authentication	0 = None 1 = Authenticate using embedded	Common Platform. ASBRuntime.DisplayLoginMode=

	browser using popup dialog 2 = Authenticate using your default browser	
--	---	--

See [Configure a System Management Server](#).

#### System Management Server - Advanced Configuration dialog

Configurator Setting	Options	Response file syntax
Certificates tab - Certificate Source	True or false.	Common Platform. ASBRuntime.AsbManagedCertificates=
Certificates tab - System Management Server	A valid Windows computer name	Common Platform. ASBRuntime.ManagementServerName=
Certificates tab - Port	Port number. Default is 443.	Common Platform. ASBRuntime.ManagementServerPort=
Ports tab - HTTP Port	A valid port number. The default is 80	Common Platform.ASBRuntime.HttpPort=
Ports tab - HTTPS Port	A valid port number. The default is 443.	Common Platform.ASBRuntime.HttpsPort=

#### Example

The following is an example of the provided All.txt response file with Configurator settings included.

```
<responsefile>
  <install>
    ComputerRoleSelectionForm.SelectedComputerRoleName=DevelopmentWorkstation
    FeaturesForm.SFeatureList=ALL
    SecurityGroupForm.AddUserToGroups=true
    FirewallForm.ModifyFireWall=true
    CustomDestinationSelectionForm.InstallDirPath=C:\Program Files (x86)\AVEVA Plant
    SCADA
    CustomDestinationSelectionForm.ProgramDataPath=C:\ProgramData\AVEVA Plant SCADA 2023
    R2
  </install>
  <configurator>
    AVEVA Enterprise Licensing.LicAPI2.NewServerName=
    AVEVA Enterprise Licensing.LicAPI2.NewPortNumber=55559
    AVEVA Enterprise Licensing.LicAPI2.LegacyPortNumber=55555
    AVEVA Enterprise Licensing.LicAPI2.NewAgentPortNumber=59200
    Common Platform.ASBRuntime.HttpPort=80
    Common Platform.ASBRuntime.HttpsPort=443
    Common Platform.ASBRuntime.ManagementServerPort=443
    Common Platform.ASBRuntime.ManagementServerName=<ServerName>
    Common Platform.ASBRuntime.AsbManagedCertificates=true
```

```
Common Platform.ASBRuntime.IsRedundantSsoServer =false
Common Platform.ASBRuntime.SuitelinkMixedModeEnabled=false
Common Platform.ASBRuntime.NmxAllowAllUsers=false
Plant SCADA.ComputerSetup.RuntimeAsService=false
Plant SCADA.ComputerSetup.PartOfTrustedNetwork=true
Plant SCADA.ComputerSetup.DeploymentEnabled=true
Plant SCADA.ComputerSetup.ServerPassword=<ServerPassword>
Plant SCADA.Encryption.IsEnabled=true
Plant SCADA.Encryption.AcceptLegacyConnections=true
Plant SCADA.DeploymentServer.UserNameValue=<UserName>
Plant SCADA.DeploymentServer.UserPasswordValue=<Password>
Plant SCADA.DeploymentServer.DatabasePasswordValue=<Password>
Plant SCADA.DeploymentServer.ConfirmDatabasePasswordValue=<Password>
Plant SCADA.DeploymentClient.SelectedServer=<ServerName>
Plant SCADA.DeploymentClient.UserNameValue=<UserName>
Plant SCADA.DeploymentClient.UserPasswordValue=<Password>
Plant SCADA.DeploymentClient.UnpackRateValue=0
Plant SCADA.DeploymentClient.DeployedProjectPath=C:\ProgramData\AVEVA Plant SCADA
2023 R2\Deployment\Client\Projects
Plant SCADA.OpcUaServer.Port=48031
Plant SCADA.OpcUaServer.EnableEncryption=false
Plant SCADA.OpcUaServer.EnableAuthentication=true
Plant SCADA.OpcUaServer.DisablePropertyAndExtensionBrowsing=false
Plant SCADA.SecurityConfiguration.RoleMembers=
ConfigUsers:.\SCADA.ConfigUsers;
RuntimeUsers:.\SCADA.RuntimeUsers;
ServerUsers:.\SCADA.ServerUsers;
DeploymentAdmins:.\SCADA.DeploymentAdmins;
DeploymentUsers:.\SCADA.DeploymentUsers;
DeploymentUploaders:.\SCADA.DeploymentUploaders;
AIGUsers:.\AIGUsers;
AIGUsersRW:.\AIGUsersRW;
</configurator>
</responsefile>
```

## See Also

[Customize a Response File](#)

## Uninstall or Change Components

You can uninstall or change Plant SCADA's core components and its supporting applications using the **Programs and Features** page in Windows® Control Panel.

Following a Plant SCADA installation, the list of programs can include any of the following components:

- AVEVA Enterprise License Manager
- AVEVA Enterprise License Server
- AVEVA Enterprise Licensing Platform
- AVEVA Help
- AVEVA Plant SCADA 2023 R2

- AVEVA Platform Common Services
- Sentinel System Driver Installer
- Schneider Electric Software Update.

---

**Note:** The entry for AVEVA Plant SCADA provides access to all core components including the configuration environment, runtime components, communications drivers, OPC UA components, Industrial Graphics components and extensions.

---

### To uninstall or change a Plant SCADA installation:

1. From the Windows **Start** menu open **Control Panel**.
2. Go to **Programs** then **Programs and Features** to display the **Uninstall or Change a Program** page.  
**Note:** The change/modify option is not available via the **Apps & Features** settings in Windows. You can only use the Programs and Features page in Control Panel.
3. Locate the program you want to remove or modify and select **Uninstall/Change**.
4. Select one of the following options from the dialog that appears:
  - **Modify** — this option allows you to add components that were not installed during the original installation, or remove selected components.
  - **Repair** — this will reinstall all non-customizable files in the same location as the previous installation. If any of the files were accidentally deleted or modified, then this option will restore the software back to its original state.
  - **Remove** — This will remove any files associated with the selected component and its registry entries.
5. Click **Next**.
6. If you chose **Modify**, select the features that you would like to add or remove and click **Next..** The following page will confirm the changes you have requested. Select **Modify** to continue.

If you chose **Repair** or **Uninstall**, the following page will ask you to confirm your selection.

The installer will indicate when the selected process is complete.

---

**Note:** Only reinstall PCS if the support team advises it, or if you get a notification from the troubleshooting tool in the Common Services portal. Do not install PCS on a computer where it was not previously installed.

---

## Manage Updates

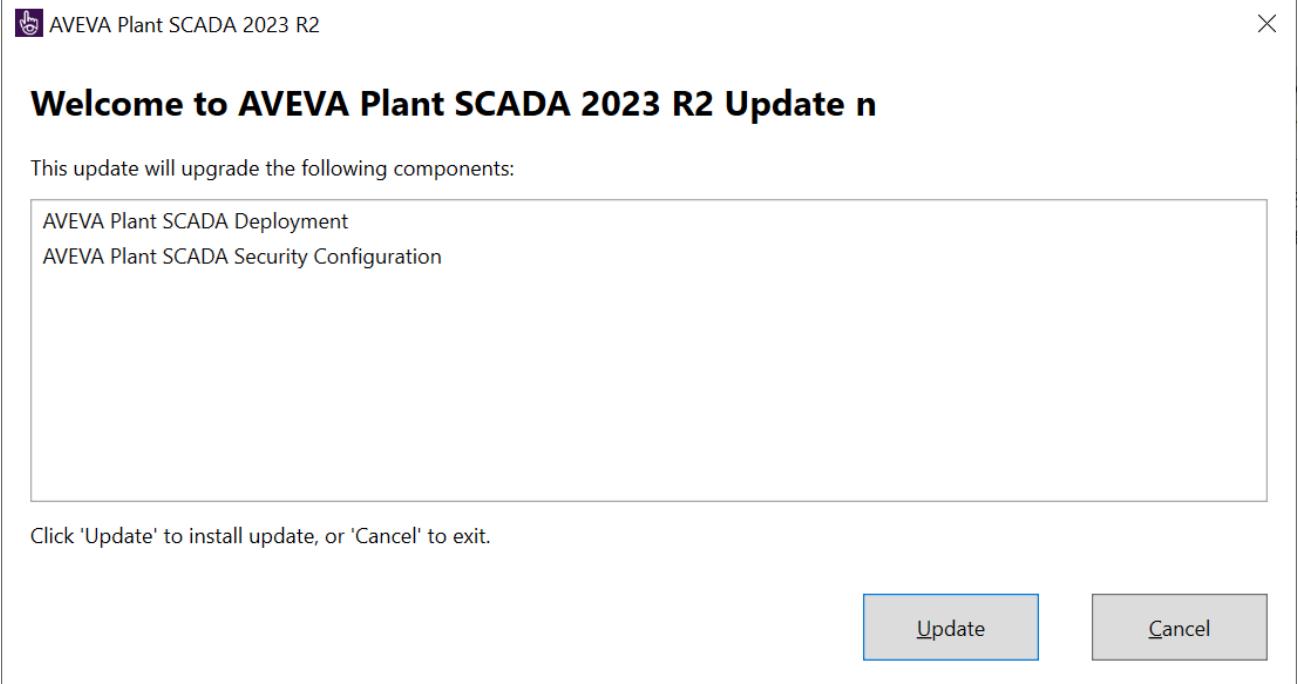
AVEVA will periodically publish software updates for Plant SCADA. You can download these updates from the **Product Hub** page of the [AVEVA™ Knowledge & Support Center](#) website at:

<https://softwaresupport.aveva.com/#/producthub>.

### To install a Plant SCADA software update

1. Go to the location where the update media was saved.
2. Run the executable file named "Plant SCADA <VersionNumber> Update *n*.exe".

The following dialog will appear. It lists the components that will be updated.



3. Select **Update** to proceed.

If any Plant SCADA services or processes are using files that need to be updated, an additional dialog will appear asking if you want to automatically shut everything down or reboot after the update is installed.

4. Make your selection and click **OK** to complete the installation.

A monthly update may include prerequisite upgrades to one or more AVEVA common components (such as Platform Common Services).

If an update contains such prerequisites, and the component has not already been upgraded on your computer, the following dialog will be shown (before the dialog from step 2 above).



AVEVA Plant SCADA 2023 R2



## Welcome to AVEVA Plant SCADA 2023 R2 Update n

This update requires the following components to be upgraded first.

Note: Once upgraded, these components can not be reverted to the previous version without uninstalling and reinstalling them.

Component	Version
AVEVA Platform Common Services	8.1.23456.1
AVEVA Enterprise License Server	4.0.2

Press 'Upgrade' to upgrade all required components, or 'Cancel' to exit.

[Upgrade](#)[Cancel](#)

Select **Upgrade** to proceed with upgrading the components. If all prerequisites are upgraded successfully, the update will display the dialog from step 2, otherwise an error will be displayed.

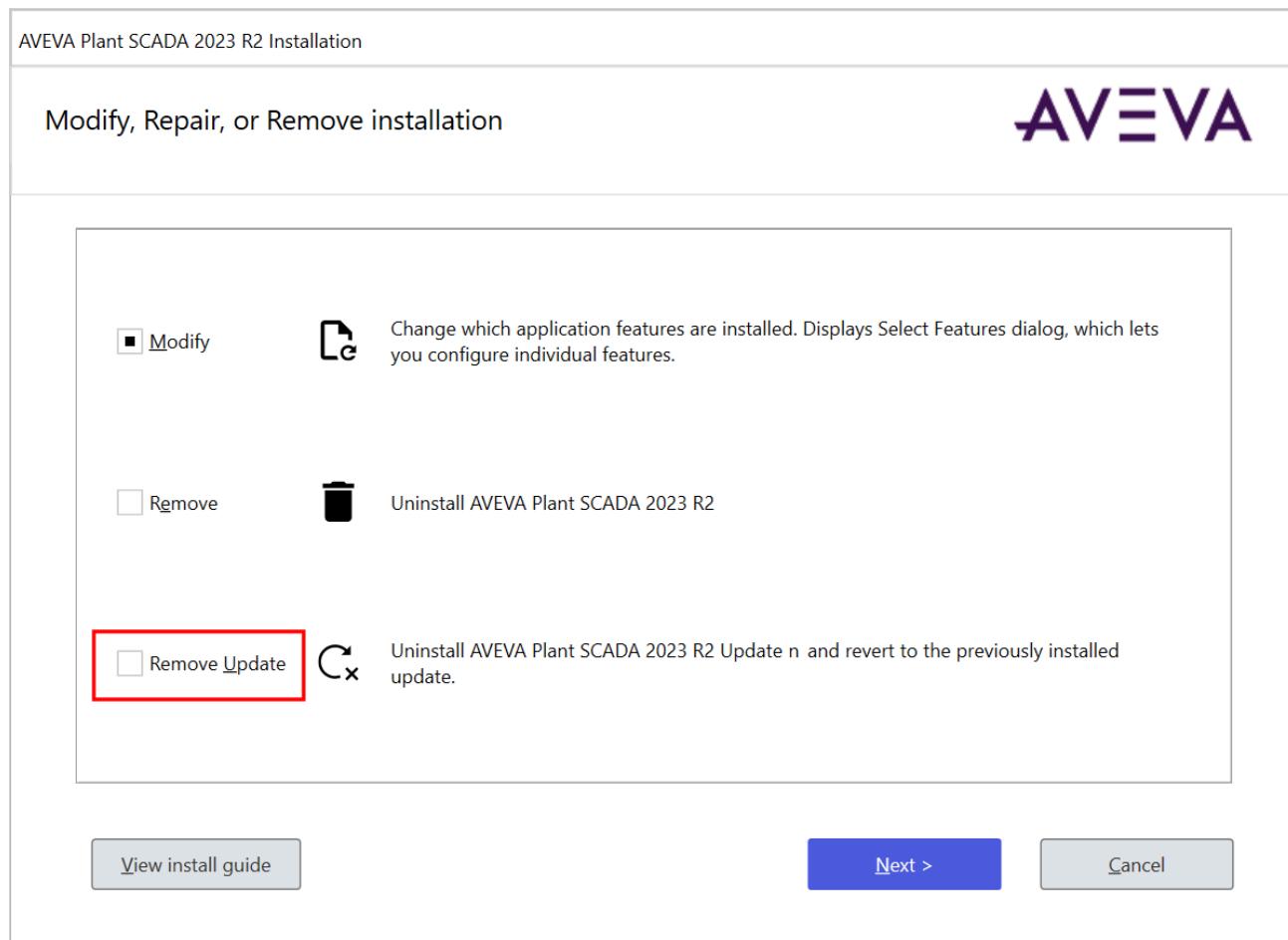
**Note:** Unlike updates to Plant SCADA components, upgrades of common components cannot be reverted without a full uninstall and reinstall of the component.

### To uninstall a Plant SCADA software update

You can uninstall Plant SCADA updates from the **Uninstall or change a program** page of Windows® Control Panel. The entry for Plant SCADA will display the most recent update that has been applied.

AVEVA Help	AVEVA Software, LLC
AVEVA Plant SCADA 2023 R2 Update n	AVEVA Software Australia Pty Ltd
AVEVA Platform Common Services 8.0	AVEVA Software, LLC

1. Select the **Uninstall/Change** option for AVEVA Plant SCADA.
2. When the **Modify, Repair, or Remove** page appears, select the **Remove Update** option.



### 3. Click **Next**, then **Uninstall**.

When the uninstallation process has completed, the entry for Plant SCADA in the Control Panel programs list will be updated to reflect the changes that have been made. If an earlier update was installed, it will indicate that the previous update is now the most recent.

**Note:** You can also use a command to silently install a Plant SCADA update. See [Silently Install a Plant SCADA Update](#).

## Silently Install an Update

To silently install a monthly update, run the following command from the command line (assuming the update executable is in the current directory):

```
"Plant SCADA 2023 R2 Update <n>.exe" -sp/silent
```

Replace <n> with the update number and date to match the name of the update executable.

This command will return immediately without waiting for the update to finish installing. To wait for the update to install (for example, when installing from a batch file), use the built-in start command:

```
start /w "" "Plant SCADA 2023 R2 Update <n>.exe" -sp/silent
```

**Note:** The empty "" are important to tell the start command to interpret the quoted update name as the executable to run (instead of the window title). The quotes are required on the update name as it contains spaces.

Any required prerequisites (for example, Platform Common Services) will be installed.

Any running Plant SCADA services will be automatically stopped by the update and restarted again when complete. Any other running Plant SCADA programs, such as Plant SCADA Studio or Plant SCADA Runtime in non-service mode, will not be stopped. If this occurs, the update will not be successful.

## Return Values

The update executable will return 0 to indicate the update was installed successfully.

If 3010 is returned, it indicates that the update was installed successfully but a reboot is required to complete installation.

Any other non-zero value will indicate an error (see the table below).

Use the %ERRORLEVEL% environment variable to check the return code. The start command must be used to retrieve the return code properly.

Return value	Description
1603	<ul style="list-style-type: none"><li>• One or more non-service Plant SCADA processes are running.</li><li>• An error occurred launching Windows Installer or setup.exe to install a prerequisite or patch file.</li></ul>
1605	The version of Plant SCADA this update is intended for is not installed.
1638	A newer monthly update is already installed.
Other	The error code returned by Windows Installer or setup.exe if an error occurs when installing a prerequisite or patch file.

## Upgrade Plant SCADA

This chapter describes the process required to migrate from an existing installation of Citect SCADA or Plant SCADA to the latest version of Plant SCADA.

**Note:** Cross version compatibility is not available for alarms version v7.20 onwards.

When updating the computer with a new product version, backup the existing projects and uninstall the existing installation. For instructions, consult the documentation for the existing version. You can then install the new version and restore your projects.

**Note:** The new version you are installing may have updates available. The update may have a fix for the automatic upgrade and may be required to be installed before restoring the project. Please refer to the **Product Hub** page of the **AVEVA Knowledge & Support Center** (<https://softwaresupport.aveva.com>).

Before you review this information, check that you have the necessary hardware and software required to run this version. See the requirements specified for the type of SCADA system computer you are upgrading in the relevant section of the Install Guide.

When upgrading to Plant SCADA 2023 R2 you need to consider the following:

- **Upgrade Method:** Depending on whether your system can afford downtime and loss of data, choose an upgrade method: offline or online.
- **Upgrade Path:** Upgrade path refers to the number of versions to which you need to upgrade to get from your current version of Citect SCADA/Plant SCADA to the latest version. For upgrading to intermediate versions specified in the upgrade path (for example, v2015 or v2016), refer to the documentation for those versions.

---

**Note:** With the release of Plant SCADA 2023, Schneider Electric's Floating License Manager is no longer supported. Existing users of the Floating License Manager will need to migrate their systems to [AVEVA™ Enterprise Licensing](#).

---

## See Also

- [What's New in Plant SCADA](#)
- [Offline Upgrade](#)
- [Online Upgrade](#)

## Upgrade Method

Before you plan to upgrade Plant SCADA, consider whether your SCADA system can afford downtime and whether your historical information needs to be available. The upgrade method you choose will depend upon this.

Upgrade methods are of the following types:

- This method requires your system to be shut down for the duration of the upgrade. If your system can afford downtime and depending on whether historical information needs to be available, this method is suitable for you. This is the basic upgrade process that will be required even if you use the online upgrade method.
- If you need your system to be available, this method is suitable for you. To be able to conduct an online upgrade, you need to have at least one pair of redundant servers (for details and other pre-requisites, see [Pre-requisites for Online Upgrade](#)).

## See Also

- [Upgrade Path](#)
- [Offline Upgrade](#)
- [Online Upgrade](#)

## Upgrade Path

Upgrade path refers to the versions you need to install to get from your current version of Citect SCADA to the latest version of Plant SCADA.

Some versions of Citect SCADA included substantial changes to the product, which require incremental upgrades involving intermediate steps between versions. The number of necessary steps will depend on whether you do an offline or online upgrade.

- For an [Offline Upgrade](#), you can upgrade a project directly from any version of Citect SCADA from 5.21

onwards.

- For an [Online Upgrade](#), you can go directly from the following versions:
  - Citect SCADA v2020 R2 Update 2 (March 2022) or later.
  - Plant SCADA v2023 (or later).

If you would like to perform an online upgrade from an earlier version, you will firstly need to temporarily upgrade your system to one of the versions listed above. For more information, see the Installation Guide for the version you choose.

You can download the installers for earlier versions from the **Product Hub** page of the **AVEVA Knowledge & Support Center** (located at <https://softwaresupport.aveva.com/>).

For both offline and online scenarios you need to:

- Apply the latest patch to the version you are upgrading from.
- Perform the upgrade to Plant SCADA 2023.
- Apply the latest updates to Plant SCADA 2023.

---

**Note:** Any patches released from July to October 2020 should not be used in an online upgrade scenario.

---

## Offline Upgrade

---

**Note:** This is the basic upgrade process. You will need to perform these steps even if you choose to use the Online Upgrade method.

---

An Offline Upgrade to Plant SCADA comprises the following steps:

### 1. Backup your current project and relevant files.

Perform a backup of your project and other relevant files. For the upgrade to complete smoothly, you need to back up a number of files/folders from your system other than your project files. The number of files you need to back up depends on your system configuration. For more information about performing a backup, refer to the Backing Up a Project section in the online help of your current version.

---

**Note:** With the release of Plant SCADA 2023 R2, the CSV\_Include project is no longer supported as a system project. If you have any projects based on the CSV\_Include templates, you will need to create a separate backup of the CSV\_Include project and restore manually it in version 2023 R2 (or later). You can then include it as a user project.

---

The following files need to be backed up:

File	Description
Project backup (.ctz file)	This is the main file to back up. For information about backing up a project, refer to your current version's online help. You need to have the <b>Save sub-directories</b> and <b>Save configuration files</b> options selected in the Backup dialog.
Citect.ini	This file is located in the config folder.

File	Description
Deployment configuration files	<p>If you have deployment configured, back up the following files:</p> <ul style="list-style-type: none"> <li>• SE.Asb.Deployment.Server.WindowsService.exe.config</li> <li>• SE.Asb.Deployment.Node.WindowsService.exe.config.</li> </ul> <p>These are located in the path [CtEdit]Config.</p>
Data directory	This file is found on the path [CtEdit]Data.
Deployment database	<p>This is located in the Deployment directory. For example:</p> <p>%PROGRAMDATA%\AVEVA Plant SCADA 2023\Deployment.</p>
Alarm Database (for v2015 or later)	<p>The Alarm Database is located in the Data directory: [Data]\&lt;Project Name&gt;\&lt;ClusterName.AlarmServerName&gt;.</p> <p>For each alarm server you have in your system, a corresponding Alarm Database will exist. You need to backup all alarm databases.</p>
Trend files: *.HST and *.00X	<p>The path and names of these files are defined on the trend tag itself and created in the Data directory defined in [CtEdit]Data. The files will be named after the trend name and number of files. For example, if the trend name is CPU, file names will be CPU.HST, CPU.001, CPU.002 and so on.</p>
Report Files	<p>These files contain the code that is executed on your reports. They are located in the [CtEdit]User\&lt;Project Name&gt; folder.</p>
Custom ActiveX Controls (.OCX)	<p>Plant SCADA includes some ActiveX controls, which will be available with the 2023 installation. However, you need to take a backup of your custom ActiveX controls.</p> <p>Check your ActiveX.dbf file in the [CtEdit]User\&lt;Project Name&gt; folder. This file contains a list of the ActiveX controls in your project and their GUID. Using the GUID, find the path of an ActiveX control using the Windows Registry key:</p> <p>KEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\"GUID"\InProcServer32\.</p> <p>The default value for this key is a path to the .DLL or</p>

File	Description
	.OCX file you need to back up.
Process Analyst files	Backup the main <Project Folder>\Analyst Views and <Project Folder>\Dictionary folders.
Device logs	These files contain any logging (alarm logs, report logs) you have configured in your project. You will find their location in the Devices dialog. Refer to your online help for more information.
Additional Files	Check your Citect.ini file or use the <b>Setup Editor, Paths</b> section as it could contain runtime files used by custom code in the project.
Driver Hotfixes	<p>If you are aware of any driver hotfix in your system, backup this driver DLL which is located in the Bin directory where Plant SCADA is installed.</p> <p><b>Note:</b> The fixes contained in this hotfix may have been included in the drivers which ship with Plant SCADA 2023.</p>

**Note:** Before you backup a project from version 2016 (or earlier) for upgrade to version 2018 R2 (or later), you should check if the Citect.ini file uses the parameter [CtEdit]ANSIToOEM. If so, confirm that the parameter is set to the required value. Then perform an **Update Pages** with the **Fast Update Pages** option disabled. This will back up the correct ANSI character values. Be aware that the Fast Update Pages option is not available from version 2018. If you are using a later version and have noticed inconsistencies due to the misconfiguration of [CtEdit]ANSIToOEM prior to an upgrade, contact technical support.

## 2. Upgrade your licenses

In order to do this, you will either need to have a valid support agreement or you will need to purchase a license upgrade. You can upgrade your key or soft license on the **License Activation** page of the AVEVA Knowledge and Support Center website at <https://softwaresupport.aveva.com>. You can also check the support status.

If your license is out of support, contact your local authorized AVEVA distributor. If you don't know who your local distributor is, send an email to [scada.orders@aveva.com](mailto:scada.orders@aveva.com) with your license and site ID details. For more information about licensing in Plant SCADA, refer to the [Licensing](#) section of the Plant SCADA documentation.

**Note:** With the release of Plant SCADA 2023, Schneider Electric's Floating License Manager is no longer supported. Existing users of the Floating License Manager will need to migrate their systems to [AVEVA™ Enterprise Licensing](#).

## 3. Install Plant SCADA

Uninstall the current version of Citect SCADA completely and install Plant SCADA 2023.

**Note:** Do not uninstall Platform Common Services (PCS) if upgrading from Plant SCADA 2018 R2.

## 4. Configure the Server Password

Configure the Server Password using the Configurator's **Computer Setup** page under Plant SCADA. For more information, see [Use Configurator to Set Up a Runtime Server](#).

## 5. Configure the System Management Server

Configure the System Management Server in the Configurator's System Management Server page under Plant SCADA. For more information, see [Configure a System Management Server](#) in the Plant SCADA documentation.

## 6. Restore your project

Restore your project. For instructions on restoring your project, refer to the Plant SCADA documentation.

## 7. Upgrade your project

As a default, when you restore your project from a previous version, Plant SCADA will force an update, and you will get a warning message. Click **Yes** to proceed with project upgrade.

If this message is not displayed, you can force an update of all projects by setting the [\[CtEdit\]Upgrade](#) INI parameter to 1. Close and re-open Plant SCADA Studio. Once you re-open Plant SCADA Studio, you will get a warning message. After clicking **Yes** all projects will be upgraded.

## 8. Migrate your project

The automatic project upgrade does not fully upgrade your projects and needs to be followed by the Migration Tool. The [Project Migration Tool](#) is a separate application that runs automatically after the project upgrade has been executed. It adds computers from the existing topology. You may need to run the Migration Tool separately for other components.

## 9. Merge your INI file

If you have defined any project-specific parameters in your Citect.ini file, merge them into the new version's INI file.

Merge any driver parameters from your old INI file as they will most likely be necessary to interface with your I/O network. For a list of changes to .INI parameters, refer to the topic [Citect.ini Parameters in Plant SCADA 2023](#) in the Plant SCADA documentation.

## 10. Compile your project

After upgrading your project and running the Migration tool, compile your project to ascertain that runtime functionality works as expected. It is very likely that you may encounter errors when you compile your project. One of the common sources of errors when upgrading is Cicode functions. This is because functions may have changed, deprecated or simply because the compiler code has been updated to prevent runtime errors. You can find a list of updates to Cicode functions in the *What's New* section of the Plant SCADA documentation.

Refer to the Plant SCADA documentation for instructions on compiling your project.

---

**Note:** If you are upgrading from version 2018 R2, the compiler warning message W1041 will be generated if the **Address** field for your network address is an IP address. If this occurs and you want to use encryption, you need to set the **DNS Name** field in the **Topology|Computers** view for the matching network address to the **DNS Name** of the computer.

---

## 11. Run the Setup Wizard

Before running your project, run the Setup Wizard to configure the Runtime Manager and other settings that are relevant to the runtime process. The Setup Wizard will automatically determine the role of your computer based on the network addresses defined in your project. After finishing the Setup Wizard, restore your historic data and other files, and run your project.

## 12. Restore runtime files

After compiling your project, place the files necessary for runtime in the correct directories. Refer to point 1 in this topic for the list of files you need to place in the corresponding directories as defined in your Citect.INI file and project configuration.

## 13. Restore historical data files

Restore the historical data files before running your upgraded projects.

### Alarms

Convert your alarm database in the Data directory with the following steps:

1. Check that you have placed your backed-up alarm database in the directory defined by the [CtEdit]Data parameter.
2. Delete any legacy database files.

Prior to version 7.30, Citect SCADA used the following alarm database files:

- <project>\_<cluster>\_ALMSAVE.DAT
  - <project>\_<cluster>\_ALMINDEXSAVE.DAT
- Or:
- ALMSAVE.DAT
  - ALMINDEXSAVE.DAT.

If you are upgrading from a version prior to 7.30, these legacy files are converted to a new alarm database format when the alarm server is launched. Once this occurs, these files are no longer required.

If you are upgrading from a version later than 7.30 (2015 or later), you should delete these files prior to starting the alarm server. Confirm that the directory specified by the [Alarm]SavePrimary parameter does not contain the legacy alarm database files listed above.

---

**Note:** If you do not delete these files, in some circumstances the legacy data from these files may replace or merge into your current alarm database when you launch the alarm server.

---

### Trends

Follow these steps to convert the files:

1. Create the same file hierarchy on the new system.
2. Place the files in the same folders.
3. If you want to change the folder location or you cannot replicate the same file hierarchy, edit the trend tag definitions to match the new file paths.

## 14. Run your project

Run your project to check that the functionality works as intended:

- Check any Cicode that you needed to modify in order to compile your project.
- Test communications to your I/O devices, alarm triggering and trend capture.

### See Also

[Migrate to Production](#)

[Troubleshooting an Offline Upgrade](#)

[Online Upgrade](#)

## Migrate to Production

Review the following information to complete your Offline Upgrade process and apply the changes to your production system.

### Testing Considerations

After the upgrade and configuration changes to the project are complete, it is recommended to perform system testing of the new project version. This is to check that functionality and operation behaves as expected before applying the new project to the production environment.

### Licensing

When changing to use a newer product version, the hardware/software key may need to be updated. The hardware key is a physical key that plugs into either the parallel port or USB port of your computer. To upgrade the key, a new authorization code is required which can be created by using the AuthCode Generator. To prepare the system, it is recommended to update the production machine keys before the project is updated on the production machines as the updated key will still license the previous version. You can update the hardware/software key in the Licensing activity in Plant SCADA Studio.

---

**Note:** With the release of Plant SCADA 2023, Schneider Electric's Floating License Manager is no longer supported. Existing users of the Floating License Manager will need to migrate their systems to [AVEVA™ Enterprise Licensing](#).

---

### Prepare Configuration [INI] Files

Before beginning any changes to the production computers, it is recommended that you back up the configuration [INI] files for each machine as they may be required for reference.

The current configuration file can be used with the new product version after the path parameters have been updated to the new version file locations. Refer to the setup of the development environment section of the specific version for further parameter information.

The Setup Editor and Setup Wizard can be used to finalize the configuration of the computer setup.

## Server Addresses

During a migration with an existing system, it may be useful to use a new set of IP addresses and computer names for the new version. This is typically done when there is a need to provide isolation between the system project versions to allow the two systems to individually co-exist on the network for a period of time. When isolated, the systems will be independent and not cross communicate or synchronize between the existing and new versions. This type of upgrade would have the new version start with a snapshot of the historical data from the previous system and then run in parallel.

## Communication Drivers

The project may be using specialty drivers. If so, it is recommended that you back up the driver files located in the product 'bin' directory. Existing specialty drivers that are used may be required for the new version. The Connectivity Hub on the [AVEVA Knowledge & Support Center](#) can be checked for driver availability and compatibility.

**Note:** Plant SCADA 2020 R2 introduced a classification system for the driver portfolio that reduced the number of drivers included by default with the Plant SCADA installation media. Some drivers may need to be downloaded from the **Connectivity Hub** at the AVEVA Knowledge and Support Center.

## Specialty Software

The project may be using specialty software to provide certain system functionality. These applications may be required to be updated or re-installed during the upgrade process and considered in the context of the upgrade.

## Format File

The project may be using custom configuration forms in the product. This configuration is located in the FRM file which may be required in the new installation. For further information, check the Tech Note "Upgrading with a modified CITECT.FRM file" (TN3795) on the AVEVA Knowledge & Support Center.

## Trend and Alarm Data

A project upgrade may also require the trend and alarm data to be updated based on the new product features. It is recommended to keep a backup of the existing production trend data files and the alarm save data file from the original.

Once the data files have been upgraded, the updated data files may not be compatible with the previous version. It is not recommended to change the directory path of the trend data files during the project upgrade as this may affect the trend operation. The default data directory may be changed between product versions and may need to be considered in the context of the install and upgrade with regards to the trend file location.

## See Also

[Troubleshooting Offline Upgrade](#)

## Troubleshooting an Offline Upgrade

This section lists common issues you might encounter during your [Offline Upgrade](#), which may be compiling errors or any other pre-runtime issues.

### Not able to upgrade license key

- Check that you have correctly installed the latest versions of **CiUSafe** and the **Sentinel Driver** via the [License Activation](#) page of the AVEVA Knowledge and Support Center.
- Check that the Authorization code matches the Key you are trying to upgrade. If you still cannot upgrade your license, please check the Tech Note "CIUSafe Error Return Codes" (TN5882) on the [Knowledge Base](#) page of the [AVEVA Knowledge and Support Center](#).

### Compiler errors and warnings not related to deprecated functions

As Plant SCADA evolves, the compiler feature becomes stricter in order to maintain project quality and runtime success. The fact that you are getting compiling errors that were not appearing before is because of stricter compilation, which will result in more predictable and stable runtime. Refer to the error code in the error message to resolve any errors and warnings. You can search the online help using the error code for more information about a specific error code.

## Online Upgrade

An online upgrade takes advantage of Plant SCADA's native server redundancy to avoid loss of data and minimize downtime on a production system.

---

**Note:** An online upgrade is only supported from versions that are currently in mainstream support.

---

There are two ways to perform an online upgrade:

- Upgrade one by one — firstly primary servers, then clients and then standby servers.
- Upgrade side by side — where a new set of server and clients runs in parallel.

If the primary server has a lower version of AVEVA Plant SCADA than the standby server, then the server communication will be lost. So, make sure to avoid the following scenarios during online upgrade:

- Shutting down the upgraded primary server before the standby is upgraded.
- Stopping both the servers when the upgrade is in progress.
- Starting up the non-upgraded standby server, when the upgraded primary server is running.

Similar to the [Offline Upgrade](#), you will need to follow the upgrade path, and repeat the process as many times as the number of steps in your upgrade path.

---

**Note:** With the release of Plant SCADA 2023, Schneider Electric's Floating License Manager is no longer supported. Existing users of the Floating License Manager will need to migrate their systems to [AVEVA™ Enterprise Licensing](#).

---

The required patches for an online upgrade are:

- For Plant SCADA 2020 R2, Update 2 (March 2022) or later is required.

For Plant SCADA 2023, you can upgrade from an RTM installation or a later installed update.

Refer to the following sections for specific instructions on how to upgrade from your current version.

- [Upgrading from v2020 R2](#)
- [Upgrading from v2023](#).

## See Also

[Pre-requisites for Online Upgrade](#)

[Troubleshooting an Online Upgrade](#)

## Pre-requisites for Online Upgrade

As mentioned earlier, an online upgrade will allow you to avoid downtime and loss of data. It is important that you take into consideration the complexity and size of your project when planning for this upgrade. It is recommended that you review the following pre-requisites before you start an online upgrade:

1. **At least one pair of redundant servers:** This is to upgrade one server at the time while the redundant server assumes primary operation, avoiding downtime and loss of data.
2. **Upgraded project:** Check that your project runs and works properly on Plant SCADA 2023 before migrating to production and starting the online upgrade. If your project is complex, it is recommended that you have a test environment as the offline upgrade could be complex and could involve a long server downtime if done on your production system.
3. **Restore runtime files:** Check that you have restored the necessary files for runtime onto the appropriate directories to avoid any disturbances on the upgraded live system.
4. **Capture data files:** To allow historic data to be restored into the new version, you need to assess and move data files to the required location during the upgrade process. This is described in detail in the online upgrade steps in the relevant sections.
5. **Configure your running system for online upgrade:** We recommend leveraging your current redundant system and adding the following Citect.ini parameters before the online upgrade:
  - **[LAN]EarliestLegacyVersion:** Use 8300 for v2020 R2, or 8400 for v2023. This will allow your upgraded servers to accept connections from the older version.
  - **[Alarm]EnableStateLogging:** Set this parameter to 1 to allow logging the alarm synchronization messages into the syslog file.
  - **[Debug]Kernel = 1 (optional):** This allows you to use the Kernel to monitor progress during the upgrade.

---

**Note:** Before you perform an online upgrade from Plant SCADA version 2020 R2, you need to install Update 2 (March 2022) or later. Updates for Plant SCADA are available from the **Product Hub** page of the AVEVA Knowledge & Support Center website at <https://softwaresupport.aveva.com>.

---

## Upgrading from v2020 R2

**Note:** For an online upgrade from Plant SCADA 2020 R2, Update 2 (March 2022) or later is required.

### To upgrade from v2020 R2:

1. Check that you have added the following parameter in the .INI file to all your server nodes before you start the online upgrade.

**[LAN]EarliestLegacyVersion = 8300.**

Restart the servers after adding the parameter for the changes to take effect.

2. Shutdown SCADA runtime on the primary server.
3. Upgrade Plant SCADA on this server according to the [Offline Upgrade](#).

**Note:** Do not uninstall Platform Common Services (PCS) if upgrading from Plant SCADA 2020 R2.

4. Set up the Server Password in Configurator's **Computer Setup** page. See [Use Configurator to Set Up a Runtime Server](#).

5. Configure your System Management Server and encryption settings based on your requirements.

**Note:** The encryption settings in version 2023 R2 need to be configured to align with the settings as they were in 2020 R2, otherwise communications may not be successful.

The following table indicates the encryption settings that will align successfully across version 2023 R2 and 2020 R2.

		Encryption setting on Plant SCADA 2023 R2		
		Unencrypted	Mixed Mode	Encrypted
Encryption setting on 2020 R2	Unencrypted	Communication OK	Communication OK	Communication unsuccessful
	Mixed Mode	Communication OK	Communication OK	Communication OK
	Encrypted	Communication unsuccessful	Communication OK	Communication OK

6. Place the backed-up Alarm database in the [CtEdit]Data directory. This will allow a quicker synchronization of alarm servers.
7. Restart the primary server, which is now upgraded.
8. The upgraded server will synchronize its alarm database with the running v2020 R2 server. You need to wait for the synchronization process to finish, and this will depend on the size of your alarm database. The synchronization information is available from the main kernel window of the Alarm Process as well as the syslog.
9. Upgrade your client nodes one by one.
10. Configure your System Management Server and encryption settings based on your requirements.
11. Check the **Server Authentication** section of the Computer Setup page in Configurator, under Plant SCADA. If you have selected **Configure Server Password**, make sure you use the same password as the one defined above in step 5.
12. Shutdown runtime on the standby server.

13. When the newly upgraded server assumes the primary server role it will migrate the entire alarm database to the new format, and you should now be able to see Alarm Summary data on all migrated clients.
14. Upgrade Plant SCADA on this server according to the [Offline Upgrade](#).
15. Set up the Server Password in the Configurator, Plant SCADA, Computer Setup page.
16. Configure your System Management Server and encryption settings based on your requirements.
17. Restart the standby server, which is now upgraded.
18. Check functionality of the system as a whole.
19. Test redundancy by switching off the primary server and checking that the standby takes over and clients switch over.

## Upgrading from v2023

### To upgrade from v2023:

1. Check that you have added the following parameter in the .INI file to all your server nodes before you start the online upgrade.

**[LAN]EarliestLegacyVersion = 8400.**

Restart the servers after adding the parameter for the changes to take effect.

2. Shutdown SCADA runtime on the primary server.
3. Upgrade Plant SCADA on this server according to the [Offline Upgrade](#).

---

**Note:** Do not uninstall Platform Common Services (PCS) if upgrading from Plant SCADA 2023.

---

4. Set up the Server Password in Configurator's **Computer Setup** page. See [Use Configurator to Set Up a Runtime Server](#).
5. Configure your System Management Server and encryption settings based on your requirements.

---

**Note:** The encryption settings in version 2023 R2 need to be configured to align with the settings as they were in 2023, otherwise communications may not be successful.

---

The following table indicates the encryption settings that will align successfully across version 2023 R2 and 2023.

		Encryption setting on Plant SCADA 2023 R2		
		Unencrypted	Mixed Mode	Encrypted
Encryption setting on 2023	Unencrypted	Communication OK	Communication OK	Communication unsuccessful
	Mixed Mode	Communication OK	Communication OK	Communication OK
	Encrypted	Communication unsuccessful	Communication OK	Communication OK

6. Place the backed-up Alarm database in the [CtEdit]Data directory. This will allow a quicker synchronization of alarm servers.
7. Restart the primary server, which is now upgraded.
8. The upgraded server will synchronize its alarm database with the running v2023 server. You need to wait for

the synchronization process to finish, and this will depend on the size of your alarm database. The synchronization information is available from the main kernel window of the Alarm Process as well as the syslog.

9. Upgrade your client nodes one by one.
10. Configure your System Management Server and encryption settings based on your requirements.
11. Check the **Server Authentication** section of the Computer Setup page in Configurator, under Plant SCADA. If you have selected **Configure Server Password**, make sure you use the same password as the one defined above in step 5.
12. Shutdown runtime on the standby server.
13. When the newly upgraded server assumes the primary server role it will migrate the entire alarm database to the new format, and you should now be able to see Alarm Summary data on all migrated clients.
14. Upgrade Plant SCADA on this server according to the [Offline Upgrade](#).
15. Set up the Server Password in the Configurator, Plant SCADA, Computer Setup page.
16. Configure your System Management Server and encryption settings based on your requirements.
17. Restart the standby server, which is now upgraded.
18. Check functionality of the system as a whole.
19. Test redundancy by switching off the primary server and checking that the standby takes over and clients switch over.

## Troubleshooting an Online Upgrade

This section lists common issues you might encounter during your Online Upgrade, which may be related to runtime issues and redundancy connectivity.

### Redundant servers do not communicate

I cannot make my redundant servers communicate and I keep getting the hardware alarm "Redundant Server not found".

1. Check that you have set your [LAN]EarliestLegacyVersion parameter correctly.
  - If upgrading v2020 R2 use [LAN]EarliestLegacyVersion=8300.
  - If upgrading v2023 use [LAN]EarliestLegacyVersion=8400.
2. Check that you have run the Setup Wizard and set both servers to Networked mode.
3. Set the same server password on both servers in the Setup Wizard.
4. If security is enabled, check that Plant SCADA Runtime Manager is running as a Windows service.
5. Check that the **Accept encrypted and non-encrypted connections (mixed mode)** option is selected in the Configurator, Plant SCADA, Encryption page.

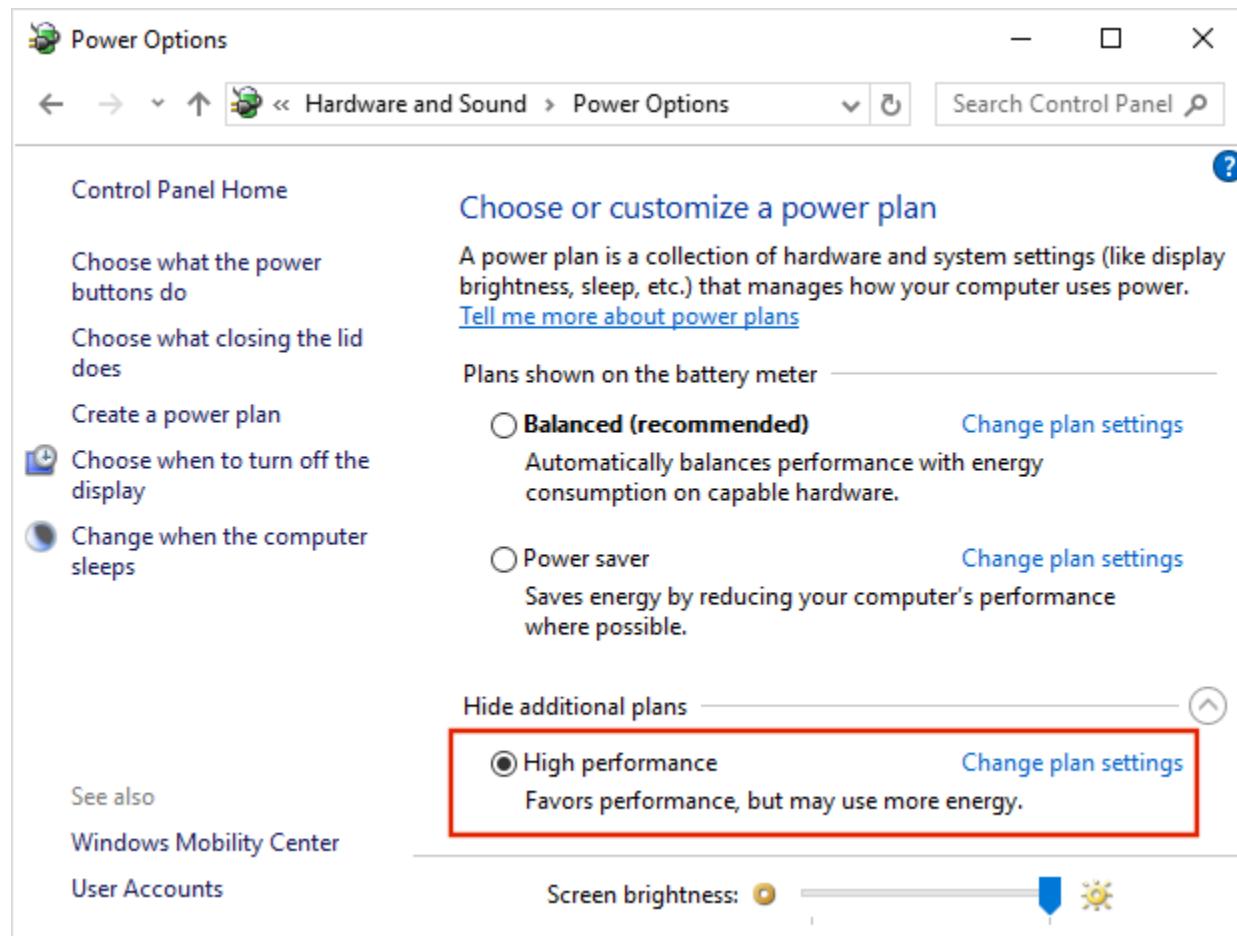
---

**Note:** The current version should not have encryption enabled with **Accept encrypted and non-encrypted** not selected, otherwise the servers will not be able to communicate. **Mixed Mode** should be used, or encryption should be disabled.

---

## My system is performing slowly even though Hardware and software requirements are met

Check your system's power options: **Control Panel, Power Options**.



## Remove Upgrade related parameters

After finalizing the upgrade process and confirming that runtime is fully functional, we recommend removing or updating the following .INI parameters. You will need to restart the servers after changing the parameters for the changes to take effect.

- **[Debug]Kernel = 0:** this is to enhance security and keep operators out of the kernel.
- **[LAN]EarliestLegacyVersion:** remove this parameter.

## Upgrade Deployment from Plant SCADA 2020 R2

The deployment architecture installed with Plant SCADA 2023 and 2023 R2 is no longer compatible with deployment servers running an earlier version. This means:

- You cannot use a 2023 R2 version of Plant SCADA Studio to add projects to a deployment server that is running Plant SCADA 2020 R2 (or earlier).

- You cannot deploy a project from a deployment server running 2023 R2 to a deployment client that is running Plant SCADA 2020 R2 (or earlier).

If you have a distributed system that includes some computers on Plant SCADA 2020 R2 (or earlier), you should retain your existing deployment server to distribute projects to these computers until the roll out of 2023 R2 is complete. You will also need to configure a new deployment server to deploy 2023 R2 projects to any upgraded computers.

---

**Note:** If you want to maintain your existing deployment server and client configurations from 2020 R2 (or earlier), you need to confirm that you have a backup of the Plant SCADA \Config folder (for example, C:\ProgramData\AVEVA Plant SCADA 2020 R2\Config) prior to upgrading to version 2023 R2.

---

Additional steps are required if you are upgrading deployment directly from Citect SCADA 2018 to Plant SCADA 2023 R2. For more information, see *Upgrade deployment directly from Citect SCADA 2018* below.

### To migrate your deployment configuration from 2020 R2 (or earlier) to a 2023 R2 deployment server.

1. Confirm the System Management Server configuration has been upgraded correctly.
2. In Configurator, confirm that Computer Setup is configured correctly.
3. On Configurator's Deployment Server page, select **Import Configuration file from previous version of Deployment Server**.

If this option is not available, you need to close Configurator and delete the following files from Plant SCADA's \Config folder:

- For a deployment client, delete SE.Asb.Deployment.Node.WindowsService.exe.config
- For a deployment server, delete SE.Asb.Deployment.Server.WindowsService.exe.config.

You then need to restart Configurator and attempt step 3 again.

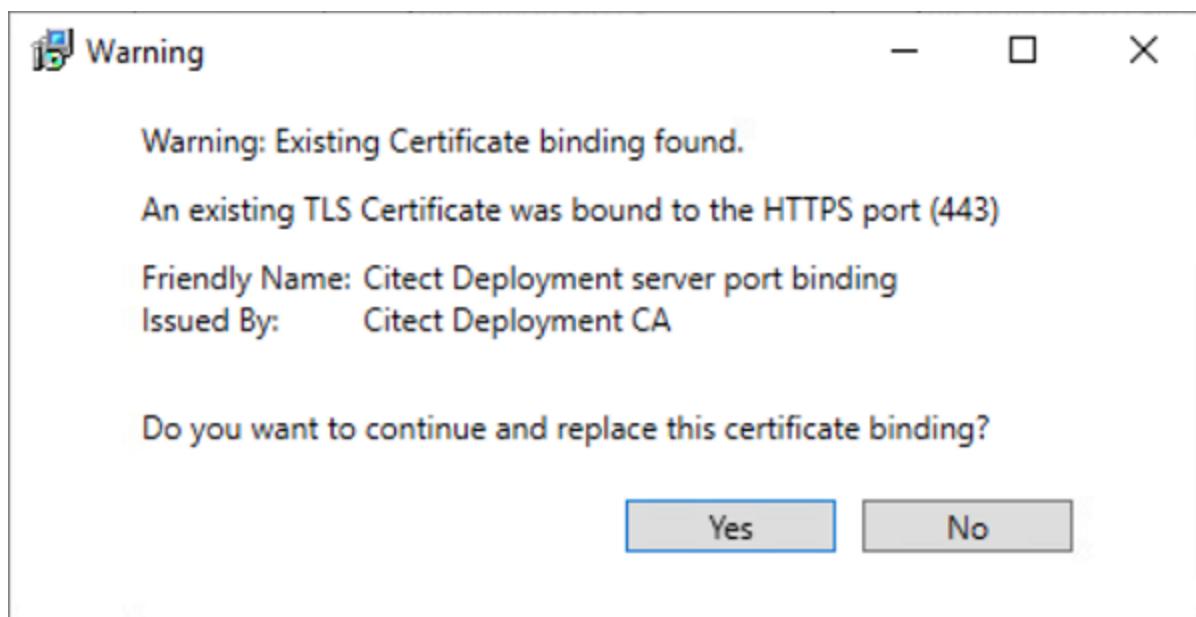
4. Using the browse dialog, locate the deployment server configuration file from your previous version. For example:  
C:\ProgramData\AVEVA Plant SCADA 2020 R2\Config\  
SE.Asb.Deployment.Server.WindowsService.exe.config
5. Open the file.
6. Click the **Configure** button to complete configuration of the deployment server.

The same steps should be taken for configuration of the new deployment clients.

When you check the Computer Setup settings for a deployment client in Configurator, confirm that the **Run the project deployed from the Deployment Server** setting is selected, otherwise the deployment client will not appear in Plant SCADA Studio's Deployment view.

### Upgrade deployment directly from Citect SCADA 2018

In this scenario, a deployment server from Citect SCADA 2018 will have taken port 443 by default. The following warning message displays if the 2023 R2 System Management Server is also configured with the default port 443.



If this occurs, you need to select **Yes** to enable the new System Management Server to take over the deployment certificate authority role. This also allows your deployment history to be persisted.

You can then follow the step in the procedure above.

## See Also

[Install a Deployment Server](#)

## Project Migration Tool

The automatic update that occurs when you initially launch Plant SCADA does not fully upgrade your projects, and needs to be followed by the use of the Project Migration Tool (if migrating from v7.x this is particularly noteworthy). The automatic update is a passive action which updates the database field definition for any database that has been changed between the two versions and copies new files that are required.

The Migration Tool is a separate application which has to be run manually after the automatic upgrade has been executed. It can be initiated after you have prepared the project for final migration. This tool will accommodate the changes in project functionality that are incorporated in 7.x and subsequent versions.

### **⚠ WARNING**

#### **UPGRADE ALTERS COMMUNICATIONS CONFIGURATIONS**

After upgrading, confirm and adjust the configuration of I/O devices in your project.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## See Also

[Using the Project Migration Tool](#)

## Using the Project Migration Tool

**Note:** Before you use the Project Migration Tool, is recommended that you familiarize yourself with the process that it performs, and the preparatory steps that you need to carry out with your existing projects.

### To run the Project Migration Tool:

1. Backup the projects that you need to migrate.
2. Navigate to the Project activity in Plant SCADA Studio, select **Home, Migration Tool** to display the Project Migration dialog.
3. Either accept the project displayed in the edit box, or browse for the project that you wish to upgrade.
4. Specify the changes you would like to implement during the migration process by selecting from the options described in the following table.

Option	Description
Remove obsolete Memory and Alarm devices	<p>Select this check box if you wish to delete these types of devices after successful migration (see <a href="#">Remove Obsolete Memory and Alarm Devices</a>).</p> <p><b>Note:</b> Do not select this check box when you run the tool for the first time on a project that contains any included projects which are shared with more than one master project. If you want to delete obsolete devices under these circumstances, you can run the tool a second time using this option if the migration is successful after it is run the first time.</p>
Append to existing log file	<p>Use this option to append information about the migration process to the existing Migration Tool log file (located in Plant SCADA's User directory). If this option is not selected, a new log file will be created when migration is complete.</p>
Create unique IDs for Variables and Equipment	<p>Select this option to add a unique ID to the variable tags, equipment and equipment types in your project.</p> <p>Unique IDs were added to variable tags in Plant SCADA 2023 and equipment in 2023 R2. This setting should be applied to any projects that you migrate.</p> <p>You can also use it to repair a project if you are seeing compiler warning W1050 (Unique ID expected).</p>
Create roles from User security information	<p>Select this option if you wish to migrate the users database from an existing project (see <a href="#">Creation of Roles for Existing Users</a>).</p>
Copy XP_Style menu into Tab_Style menu	<p>Select this option to convert legacy menu entries to</p>

Option	Description
	the format necessary for the new menu configuration system. By default, this option is unchecked to avoid potential compile errors that may occur if the legacy menu.dbf contains functions which have been removed.
Migrate included projects	Select this option to migrate the included projects associated with the selected project (see <a href="#">Migrate Included Projects</a> ).
Migrate Equipment database, Types and Parameters	<p>You should check this option to migrate your equipment to use configuration parameters, introduced in Plant SCADA 2023 to replace custom parameters. Configuration parameters support the same functionality as the legacy custom parameters, but they are stored in a dedicated configuration parameter database that removes the 256 character limitation. See <a href="#">Migrate Custom Parameters to the Configuration Parameters Database</a>.</p> <p>You should also select this option if you have an existing database that you want to migrate into this version. When upgrading from an earlier version, and the "PARENT" field of the equipment table was used, you should select this check box. Otherwise existing data from the PARENT field will be ignored. If runtime browsing is used, the PARENT field will return the equipment parent (the substring of the equipment name without the last '.' and anything after that).</p> <p>To retrieve information that was stored in the previous "PARENT" field the "COMPOSITE" field should be used.</p>
Migrate ABCLX to OPCLX	<p>Select this option if you want to migrate devices that currently use the ABCLX driver to the OPCLX driver. Select the <b>Configure</b> button to indicate which I/O devices you would like to migrate.</p> <p><b>Note:</b> You should confirm that the OPCLX driver is installed before you use this option.</p>
Migrate Trend/SPC storage method	If you select this option, the storage method will be set to scaled (2-byte samples) for all trends that have no storage method defined. Use this option to stop the compiler error message "The Storage Method is not defined". In previous versions, a blank storage method would default to scaled.

Option	Description
	However, this is no longer supported, resulting in the compile error message.
Create computers from Network Addresses	If you select this option, computers will be created from the servers and network addresses that you have configured for a project and its include projects. This option distinguishes whether a computer has multiple IP addresses.

**Note:** If 'Copy XP Style menu into Tab\_Style Menu' and 'Migrate Included Projects' are both selected when the migration tool runs, the following message will be displayed: "Copying menus of included projects may lead to conflicts. Any conflicts will need to be manually corrected". To avoid this from occurring, it is recommended you run the migration tool twice. In the first instance just select the option 'Copy XP\_Style menu into Tab\_Style Menu', and in the second instance just select the option 'Migrate Included Projects'.

- Click **Migrate** to begin the migration process.

A progress dialog will display indicating the stage of the conversion and the name of the project being migrated. If you wish to cancel the migration at this point click the **Abort** button.

**Note:** Aborting a migration will stop the migration process, and any changes already completed will not be rolled back. You will have to restore your project from the backup created in the first step.

When the migration process is concluded, a confirmation dialog box will display indicating the number of variables converted and the number of I/O devices deleted (if device deletion was selected at the start of migration).

- Click the **Close** button to close the dialog.

## Remove Obsolete Memory and Alarm Devices

When you use Plant SCADA's Migration Tool, the **Remove obsolete Memory and Alarm devices** option adjusts the following:

**Memory tags to local variables:** tags that are on an I/O device that are configured to use a 'memory' port.

**Note:** If there are real I/O devices in your project that have been set to use a 'memory' port during testing, these can be changed before running the migration tool to avoid those tags getting adjusted.

**Alarm devices:** can remove I/O devices that have a protocol set to 'Alarm', which was needed in earlier versions to enable alarm properties as tags. In version 7.x, the alarm properties are enabled via a setting on the alarm server configuration form.

## See Also

[Memory Devices](#)

[Alarm Devices](#)

[Converting Memory Variables](#)

[Inserting New Local Variables](#)

[Deleting Variable Tags](#)

[Deleting Obsolete I/O Devices](#)

## Memory Devices

In previous versions of Plant SCADA an I/O Device could be defined as a memory device by setting the port value to "Memory". This was generally done for one of the following purposes:

- To provide for future devices that were not currently connected to the system, but their points needed to be configured at this stage of project.
- For virtual devices where there was no corresponding physical I/O Device and you needed data storage with the entire functionality normally associated with I/O variables such as alarms.
- To act as a variable which was local to the process being used in place of Cicode global variables.

You can still use I/O Devices for future or virtual devices in version 7.0, but manually set the Port parameter to an unused value other than Memory, and set the Memory property of the device to True to indicate that it is an offline in-memory device before running the Migration Tool.

You need to review your project to identify which memory I/O Devices are local variable holders and which ones need to be changed to non-memory so that the Migration tool does not convert their variables.

The Migration Tool will set any I/O Device's port which is identified as a Memory device to the new Local Variable, and the original device record will be deleted.

## See Also

[Alarm Devices](#)[Converting Memory Variables](#)

## Alarm Devices

In previous versions of Plant SCADA Alarm devices were defined as devices with their Protocol property set to "Alarm". In version 7.0 the function of configuring such a device is now replaced by setting the Publish Alarm Properties property to True on the Alarm Server.

Alarm devices with their Protocol property set to "Alarm" will be deleted from I/O Devices table by the Migration Tool.

The Migration tool can delete memory and alarm device records. If you want to delete the devices at a later time, deselect the "Remove obsolete Memory and Alarm Devices" option.

---

**Note:** Alarm devices with their Protocol property set to "Alarm" are no longer used and will be removed by the Migration Tool. The Alarm Servers will now publish Alarm Properties.

---

## See Also

[Converting Memory Variables](#)

## Converting Memory Variables

A memory variable is a variable with its I/O Device Port property set to either "Memory" or "MEM\_PLA".

If there are multiple I/O Devices with the same name, possibly on different I/O Servers, the device would not be considered as a memory device regardless of its port value. In other words, the Migration tool will not process the variables for memory devices with duplicate names.

## See Also

[Inserting New Local Variables](#)  
[Deleting Variable Tags](#)

### Inserting New Local Variables

When the Migration Tool runs, a local variable record will be inserted for each identified memory variable, and the variable data will be copied into the new local variable.

Local variables have fewer fields than variables; the following table shows the mapping from variable to local variable when copying their data.

Variable Tag Parameter or Constant Value	Local Variable Parameter
Variable Tag name	Name
Data Type	Date Type
(Empty)	Array Size
Eng. Zero Scale	Zero Scale
Eng. Full Scale	Full Scale
Comment	Comment

With the exception of the Array Size, which has been introduced in version 7.0 exclusively for local variables, every field receives its value from the same or similar field.

## See Also

[Deleting Variable Tags](#)

### Deleting Variable Tags

Once the Migration Tool has created the local variable records it will insert those variable tag records that have been converted in the previous step, and delete the original variable tag.

If an error is detected during the insertion of the local variables, the deletion of the variable tags will not be performed. If this occurs it is possible to have two records with same name and data, one in the local variable (the newly inserted record) and one in the variable tags (the original record that has not been deleted). You need to delete either of the variables manually, or restore the backed up project after removing the cause of the error then run the Migration Tool again.

## See Also

[Deleting Obsolete I/O Devices](#)

### Deleting Obsolete I/O Devices

Deleting obsolete I/O Devices is an optional step in the Migration Tool and will be performed after the memory variables are converted. If the delete option is chosen, obsolete Memory devices and Alarm devices will be deleted as the final step of the Migration Tool operation.

## Creation of Roles for Existing Users

When upgrading an existing project using the migration tool, a new role will be created (if needed) for every existing user. The new role will have the same security settings that were defined for that user and be given a generic name such as Role\_1, Role\_2 etc. During the upgrade process, if a role exists with the same security settings as the user, then the existing role will be assigned to the user being upgraded. For example; If Role\_1 exists and matches the security settings of the upgraded user then that user will be assigned Role\_1 also.

If you do not want to migrate users from an existing project, clear the option **Create Roles from User security information** from the migration tool dialog before running it.

## See Also

[Migrate Included Projects](#)

### Migrate Included Projects

Each project may contain multiple included projects. Additionally, any included project may contain its own included project so creating a cascading project.

The Migration Tool needs to process the original project and included projects in a single step. The reason for this is that variables can be defined in one project that refer to I/O Devices defined in another included project.

The Migration Tool performs this procedure sequentially on the "master" project then each included project.

In the case where two master projects share the same project as an included project, you should not select the "Remove obsolete Memory and Alarm devices" check box when you process a project that contains shared included projects. This is because the removal is performed at the conclusion of the migration process on each master and included projects sequentially. This could cause the deletion of an I/O Device in the first master project which is referenced by a tag in a shared included project which is processed in a later step.

If two separate "master" projects contain the same included project, run the Migration Tool on each "master" project without selecting to delete obsolete devices.



#### UPGRADE ALTERS COMMUNICATIONS CONFIGURATIONS

After upgrading, confirm and adjust the configuration of all I/O devices in your project.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

To remove obsolete devices, it is recommended that once the Migration Tool has completed successfully (without the check box being selected), run it a second time with the check box selected. This will remove the devices since every tag conversion were completed in the first pass of the Migration Tool.

## Default Scale

The Scale properties in both variable tags and local variables are optional. If a Scale value is not specified, the default value is indicated by a parameter in the Citect.ini file. The parameter name is "DefaultSliderScale" under the [General] section in the Citect.ini file. The default values for Scale is 0-32000, unless the default slider scale is true in which case the default value depends on the type, for example, Integer, String, or so on.

The Migration Tool will read this parameter and if it is not set, or set to false, then it will explicitly set any empty Scale property to a value into the range of 0 to 32000. This will be done even if either of the Zero Scale or Full Scale parameters has a value, in which case the empty Scale parameter will receive the default value.

If the DefaultSliderScale in the Citect.ini file set to True, the Scale parameters will not be populated with a default value if they are empty, rather they will be interpreted at runtime.

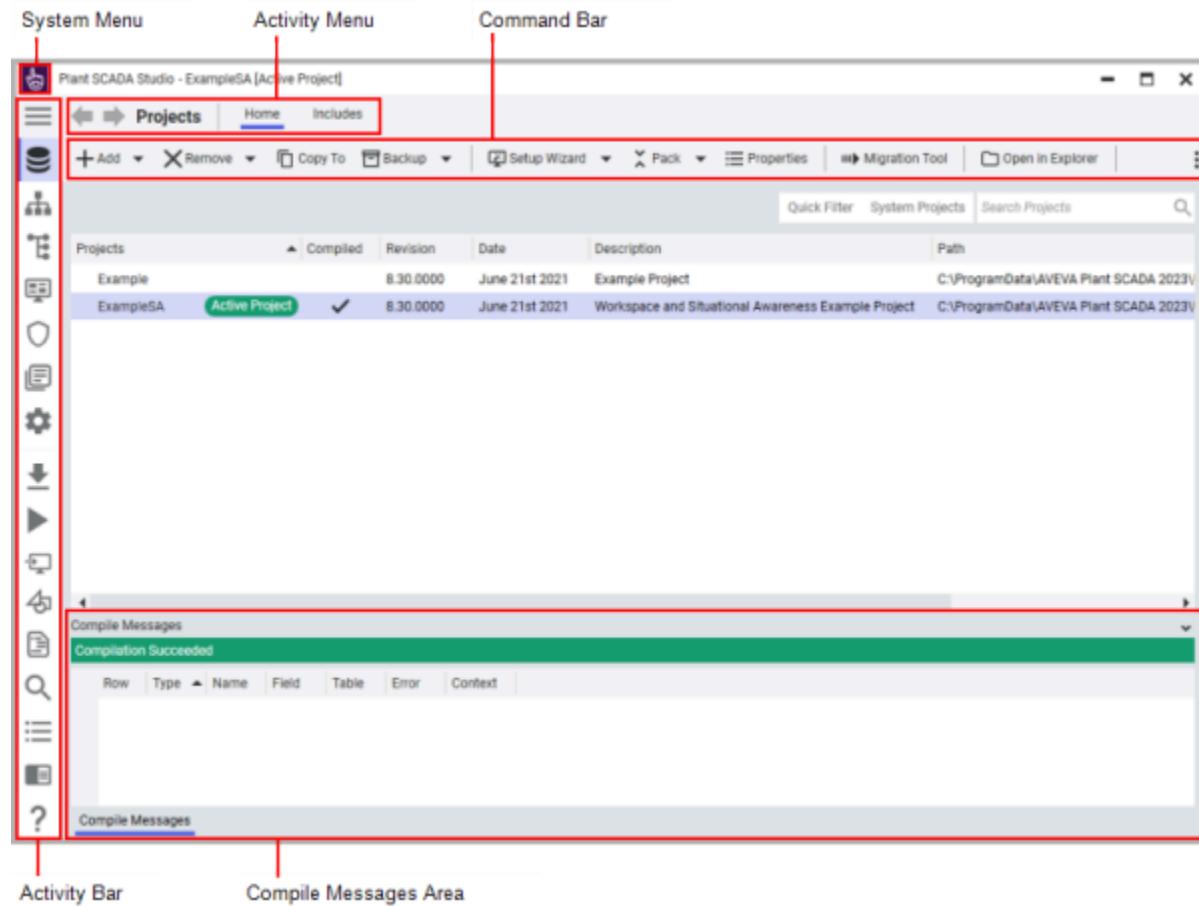
# Plant SCADA Studio

Plant SCADA Studio allows you to create and set up a logical representation of your plant environment. The configured environment can then be used to supervise and control the plant at runtime.

See [About Plant SCADA Studio](#).

## About Plant SCADA Studio

Plant SCADA Studio is divided into several [Plant SCADA Activities](#) (such as **Projects**, **Topology**, **System Model** and **Deployment**), allowing you to access different SCADA functions through a single interface.



- **Activity Bar** — allows you to navigate between the different activities, and also provides access to important global commands like **Compile** and **Run**.
- **Activity Menu** — within each activity, the Activity Menu allows you to configure different aspects of the associated functionality via a set of Activity Tabs. There may be multiple menus in an activity.
- **Command Bar** — each Activity Menu displays a Command Bar with several options for performing specific tasks within that activity. Frequently used Command Bar options (such as **Save** and **Discard**) are available in

more than one activity.

- [Compile Messages Area](#) — this area is available to view at any time, and shows messages pertaining to errors encountered when a project is compiled.
- [The System Menu](#) — this menu can be accessed by clicking the following icon in the top left corner:



Plant SCADA Studio displays the **Projects** activity when it is launched. All projects in your system are displayed, and the name of the selected project is displayed on the title bar of the Plant SCADA Studio. To select a different project, click on the required project name.

## See Also

[Plant SCADA Activities](#)

[Activity Bar](#)

[Activity Menu](#)

[Command Bar](#)

## Plant SCADA Activities

This section provides an overview of the activities that you can access from .

Plant SCADA activity	Use this activity to...
<a href="#">Projects</a>	Create and manage your projects. The Project activity lists all your projects. You can rename, back up, restore, run, compile or delete a project in this activity.
<a href="#">Topology</a>	Configure computers, clusters, I/O devices or run the Profile Wizard.
<a href="#">System Model</a>	Configure and view equipment, variable tags, alarms, trend tags, SPC tags and accumulators.
<a href="#">Visualization</a>	Configure keyboard commands, and edit menu configuration properties and report format.
<a href="#">Security</a>	Configure and view security information such as user roles, and user and group permissions.
<a href="#">Standards</a>	Configure and view reusable assets across your projects including parameters, labels and fonts.
<a href="#">Setup</a>	Configure alarm categories, devices, groups, languages, events, reports and keyboard keys.

Plant SCADA activity	Use this activity to...
Deployment	Manage and distribute versions of the runtime projects to computers in your network.
License	Launch the AVEVA™ Enterprise License Manager or update a Sentinel Protection Key (see <a href="#">Licensing</a> ).

## See Also

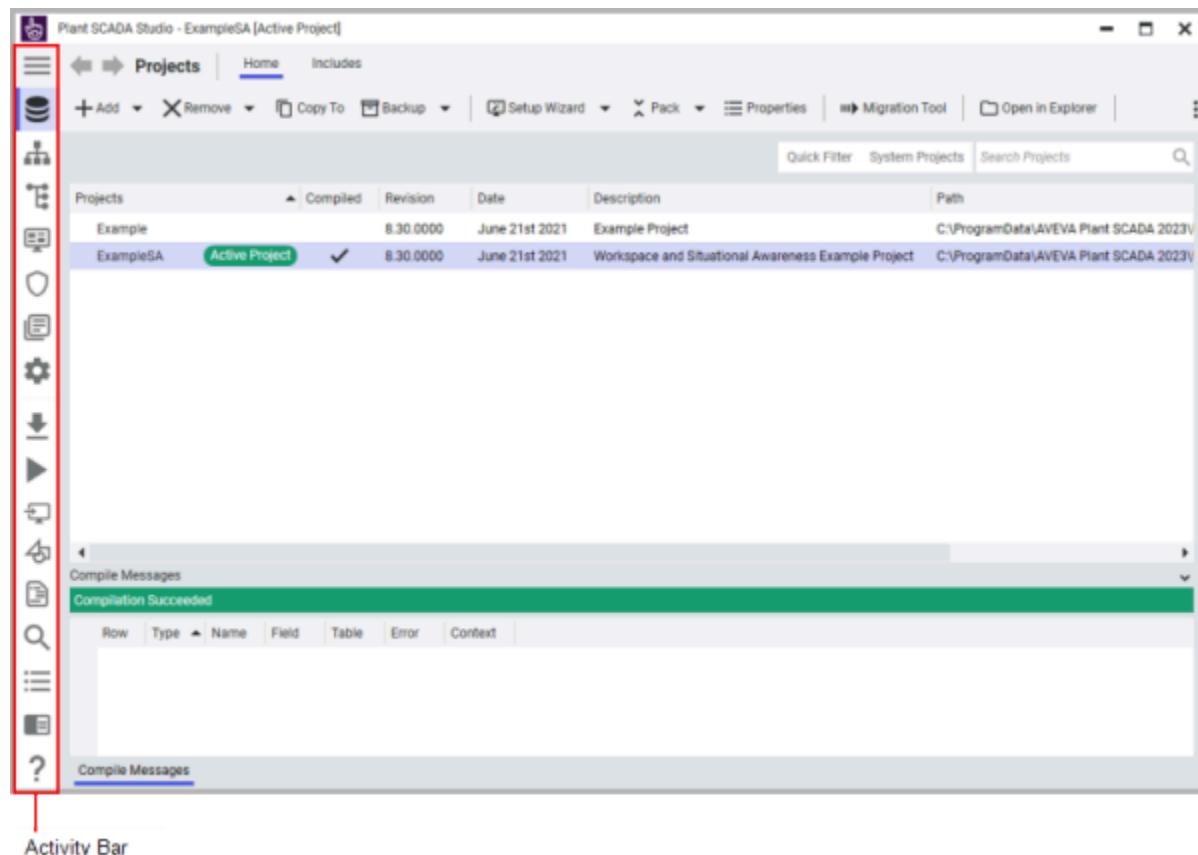
[Activity Bar](#)

[Activity Menu](#)

[Command Bar](#)

## Activity Bar

The Activity Bar is located on the left hand side of Plant SCADA Studio. It displays a set of icons that provide access to various project engineering activities such as **Projects**, **Topology**, **Security**, and so on.

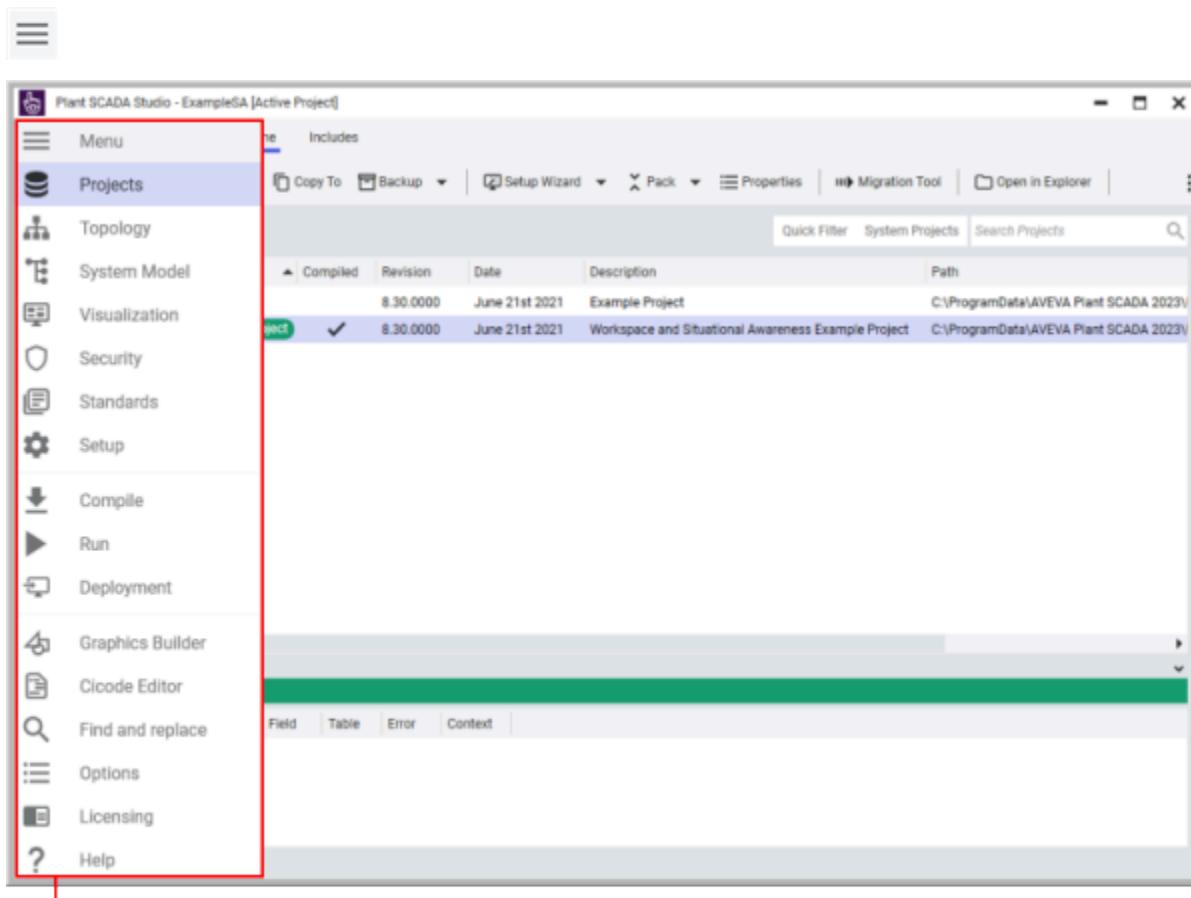


For example, clicking on the following icon on the Activity Bar displays the **Projects** activity.



Each icon displays a tooltip for easy identification.

Selecting the following icon displays the Activity Bar as a menu, which provides a description of each icon.



The Activity Bar is always visible. Selecting an option in the Activity Bar displays the corresponding [Plant SCADA Activities](#) and [Activity Menu](#).

You can also use the Activity Bar to perform the following tasks:

- Run, compile and deploy the current project
- Switch to an editor, for example, the Graphics Builder or Cicode Editor
- Set options for Plant SCADA
- Access the Plant SCADA help.

## See Also

[Plant SCADA Studio](#)

[Plant SCADA Activities](#)

[Activity Menu](#)

[The Grid Editor](#)

[Command Bar](#)

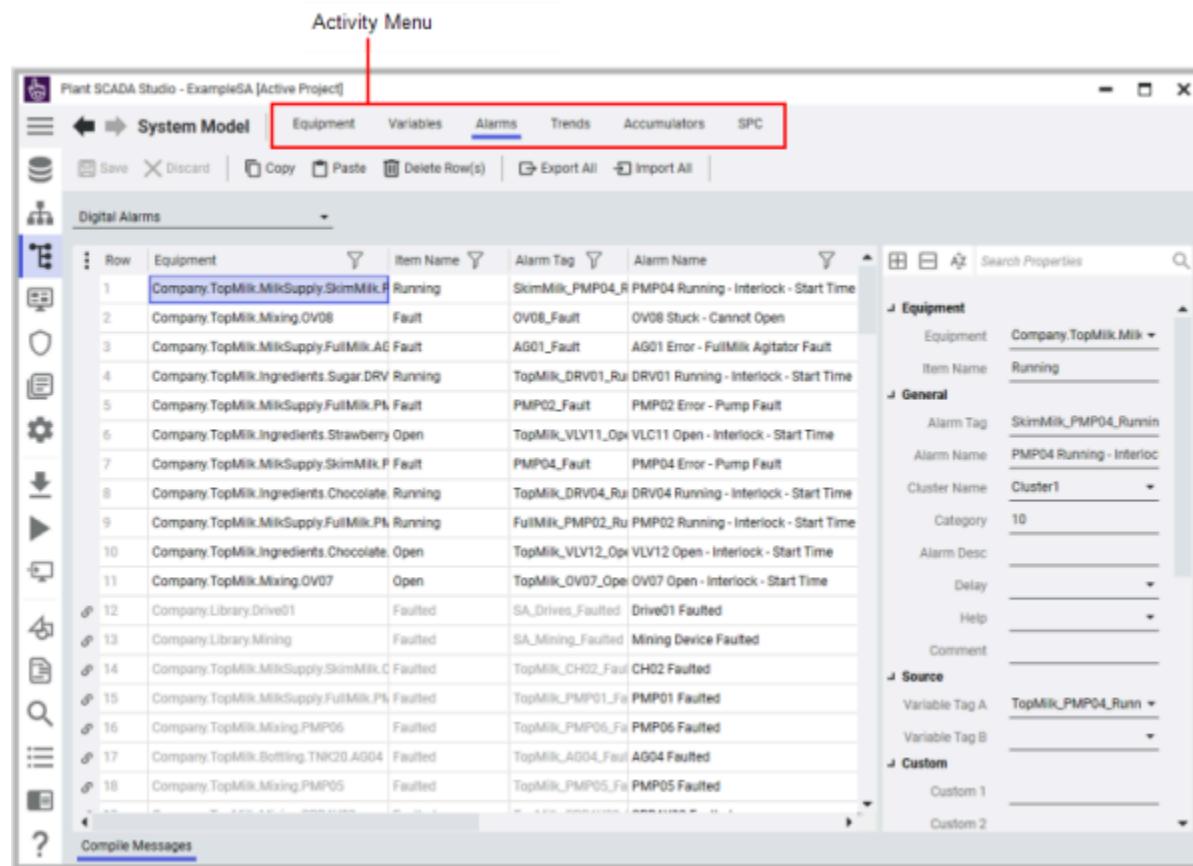
[Plant SCADA Studio Options](#)

## Activity Menu

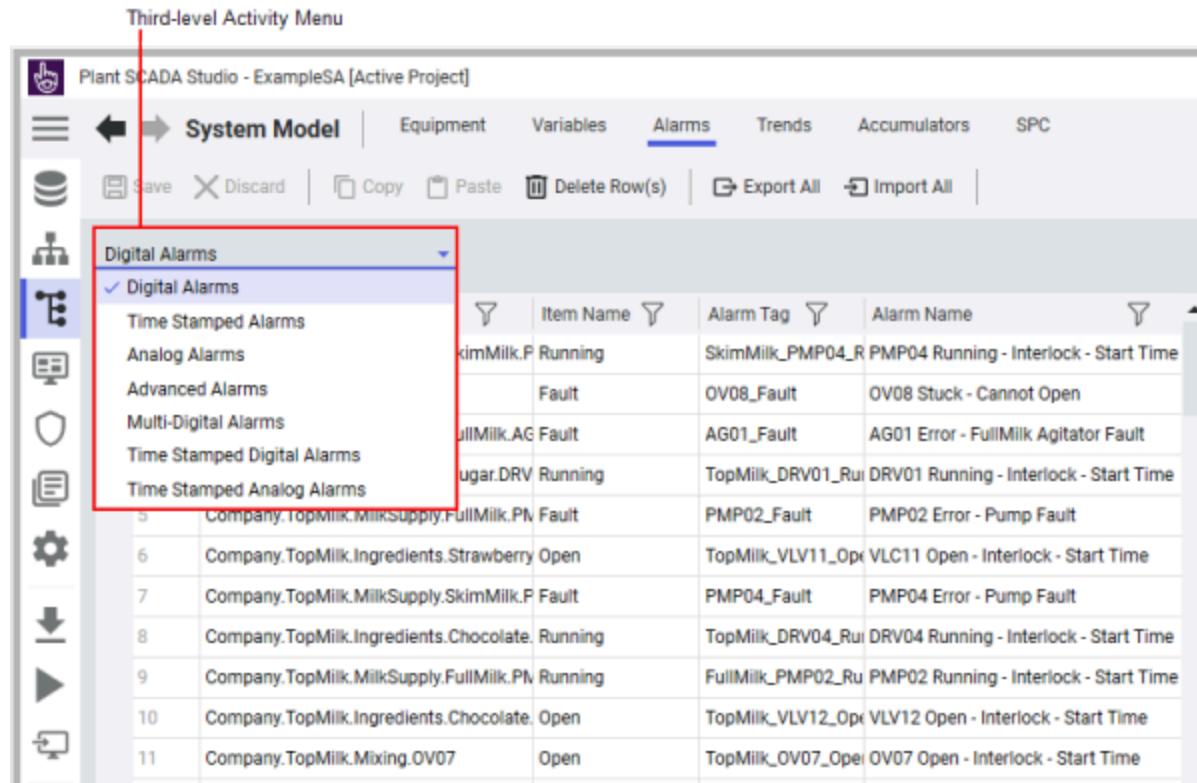
The Activity Menu displays at the top of the window in an activity. It includes tabs for each function associated with the selected activity.

The name of the currently selected activity is displayed on the left. Next to it, **Back** and **Forward** buttons are provided to navigate directly to activities and views that have recently displayed. The selected Activity Menu tab is displayed with an underline.

For example, when you select **System Model** on the Activity Bar, the **System Model** activity is displayed with its associated Activity Menu. In the example below, the **Alarms** tab is selected on the Activity Menu.



An Activity Menu tab may have a third-level of navigation displayed below the **Command Bar**. The third-level menu is displayed as a drop-down list from which you can select an option. For example, **System Model | Alarms** displays the following menu, which can be used to select the type of alarms to configure.



## See Also

[Plant SCADA Studio](#)

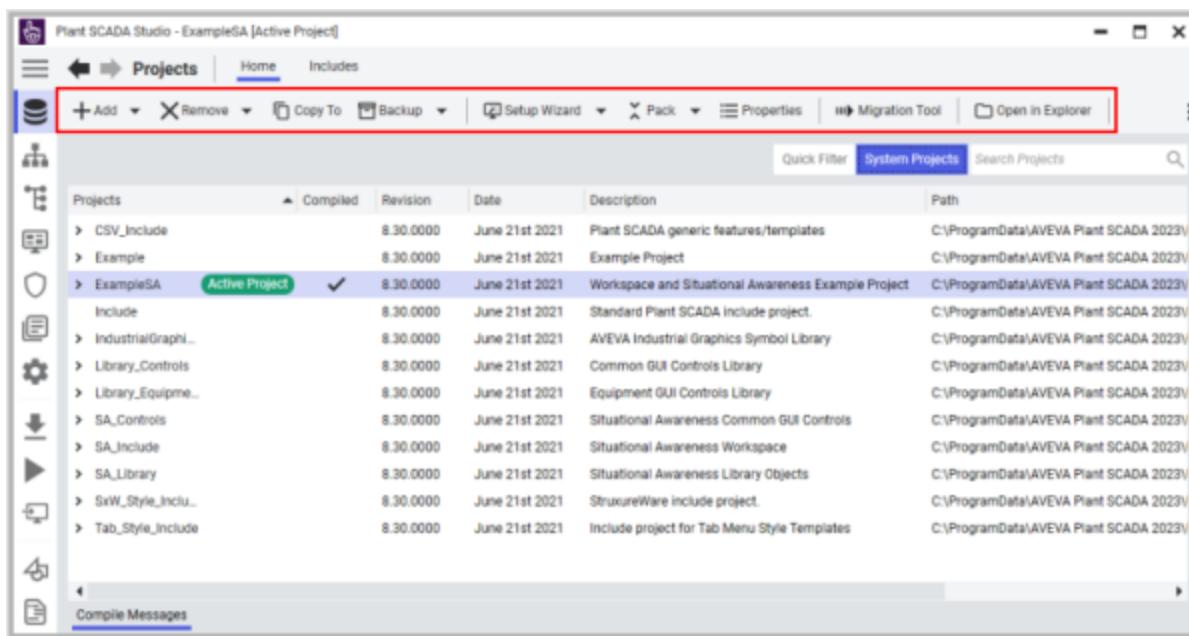
[Activity Bar](#)

[Command Bar](#)

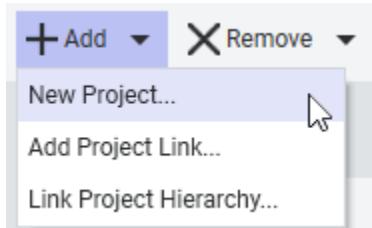
[The Grid Editor](#)

## Command Bar

The Command Bar is located below the [Activity Menu](#). It displays commands that represent one or more actions that can be performed within the selected activity.



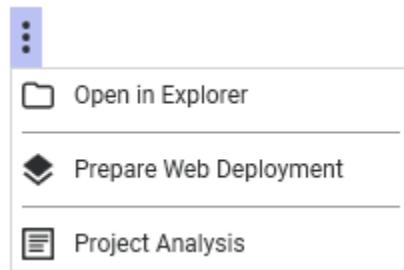
Each option (button) on the Command Bar performs a specific task. Command Bar options may have one or more sub-options accessed via a drop down control. For example, the **Add** option in the **Home** tab consists of the following tasks/commands as shown below.



If Plant SCADA Studio's current size does not allow enough space to display all the commands for an activity, the following icon will display on the right-hand side of the Command Bar.



Clicking on this icon will provide access to any commands that are currently not displayed.



## See Also

[Plant SCADA Studio](#)

[Activity Menu](#)

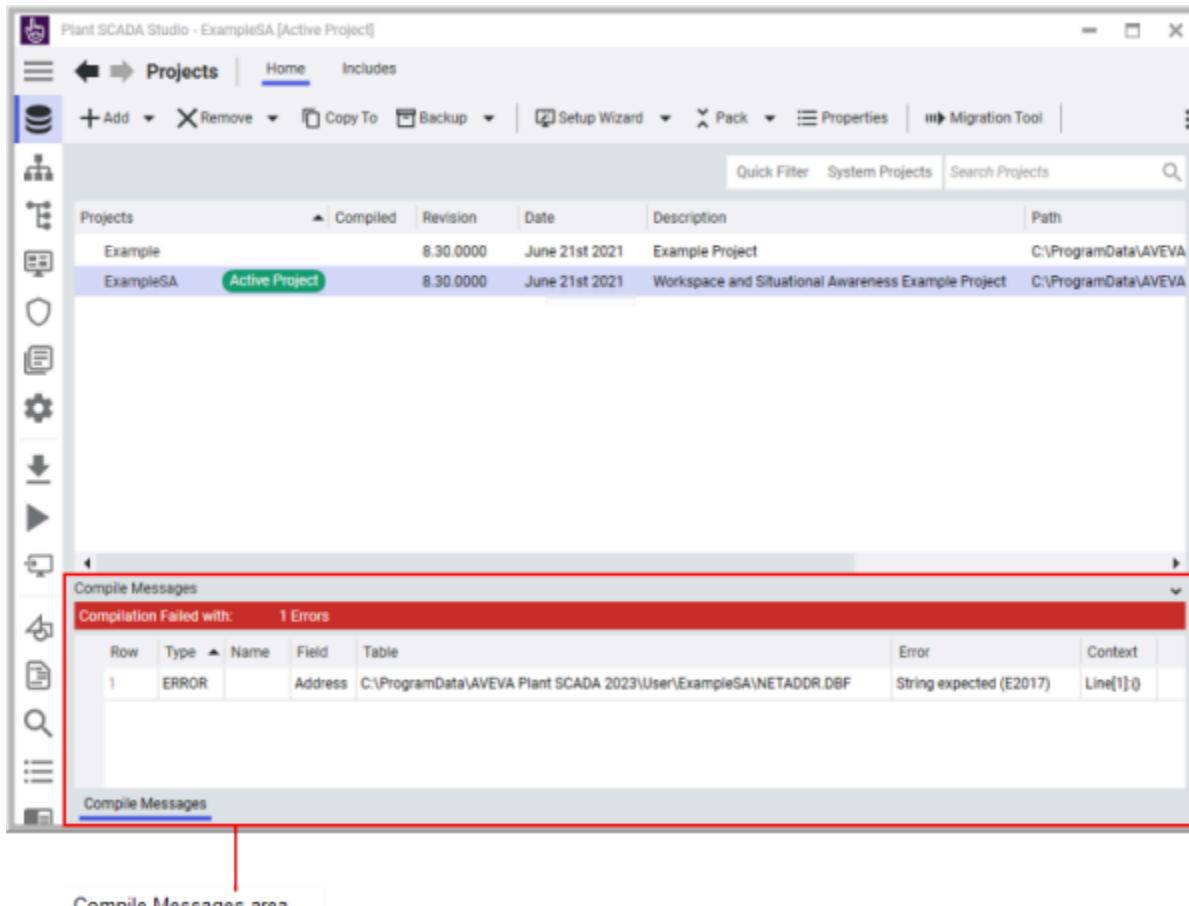
[Activity Bar](#)

## The Grid Editor

# Compile Messages Area

The Compile Messages area is located at the bottom of every view in Plant SCADA Studio. It displays error messages when compiling a project.

To view these messages, click **Compile Messages**. Messages are displayed in a tabular format. To go to a record that relates to a compile message, double click within the message row. The Grid Editor will display with the associated field selected.



To close the **Compile Messages** area, click the arrow located in the top right corner of the Compile Messages area.

**Note:** You can sort the messages by clicking on one or more column names.

## See Also

[Compile](#)

[Compile Messages](#)

## The System Menu

The System Menu can be accessed by clicking the following icon in the top left corner of Plant SCADA Studio.



It provides the following options:

Option	Description
<b>Restore</b>	Standard Microsoft Windows functionality for restoring, moving, sizing, minimizing, maximizing and closing the current window.
<b>Move</b>	
<b>Size</b>	
<b>Minimize</b>	
<b>Maximize</b>	
<b>Close</b>	
<b>Data Folder</b>	Opens the folder that contains the runtime data in a new Windows Explorer window.
<b>Logs Folder</b>	Opens the folder that contains the runtime log files in a new Windows Explorer window.
<b>Config Folder</b>	Opens the folder that contains the configuration files for the local computer (such as citect.ini) in a new Windows Explorer window.
<b>Runtime Manager</b>	Launches the Runtime Manager.
<b>About</b>	Opens the About dialog that displays detailed information about the version of Plant SCADA you are running.

### See Also

[Plant SCADA Studio](#)

## Plant SCADA Studio Options

Plant SCADA Studio is supported by a set of options that allow you to change the way the project configuration environment operates.

**To set the Plant SCADA Plant SCADA Studio options:**

1. Open Plant SCADA Plant SCADA Studio.

2. On the [Activity Bar](#), select **Options**.

The **Options** dialog box appears. Select the required options (see below for a description of the options).

3. Make any required changes.

4. Click **OK**.

**Plant SCADA Studio Options**

Option	Description
<b>Incremental compile</b>	Enables the incremental compilation of the project.
<b>Incremental equipment update</b>	Enables an incremental equipment update that only includes equipment and equipment types that have been modified since the last update occurred. See <a href="#">Update Equipment in Plant SCADA Studio</a> .
<b>Disable user functions search</b>	When you use a combo box to select a function (for a command or expression field), a list of built-in Cicode functions and user-written functions displays. If you disable user functions, only the built-in functions are displayed in the list.
<b>Confirm on project packing</b>	Enables the "Packing databases may take a long time" message window to appear when packing a database.
<b>Compile enquiry message</b>	Enables the "Do you want to compile?" message window to appear when the project has been modified and Run is selected from the File menu. Normally, Plant SCADA compiles the project automatically (if the project has been modified) when Run is selected.
<b>Prompt on tag not exist</b>	Enables the "Some variable tags were not found" message window to appear when a variable tag or tag reference is specified that does not exist in the database.
<b>Prepare for Web deployment</b>	Automatically runs the Web Deployment Preparation tool every time you compile a project. Please be aware that this dramatically increases the amount of time taken for each compile, particularly for large projects.
<b>Log deprecated warnings during compile</b>	If you select this option, the compiler will generate an alert message to identify any deprecated elements it

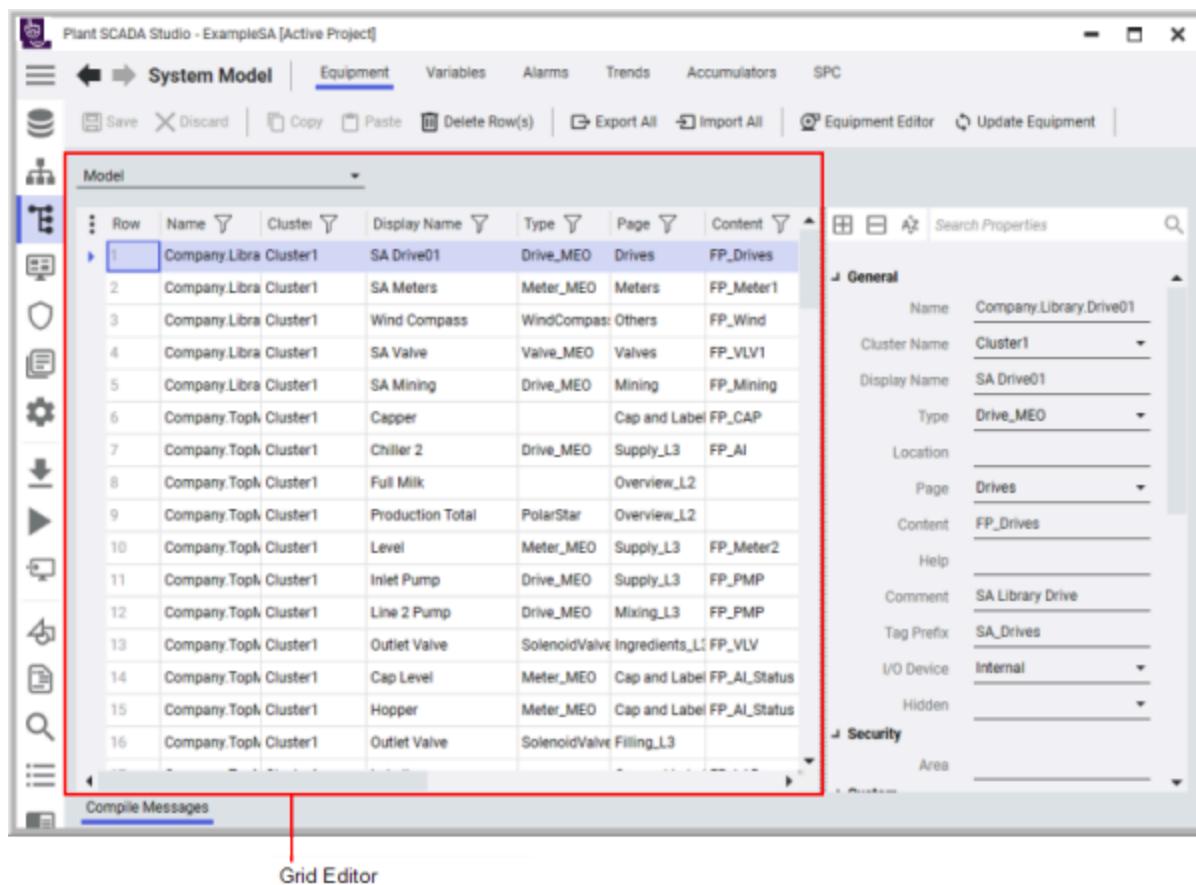
Option	Description
	<p>detects in a project, that is any functions, parameters, or Kernel commands that are no longer supported.</p> <p>By deselecting this option, the alert messages are still included in the displayed alert count, but they are not added to the error log.</p>
<b>Check Computer Setup status before running project</b>	<p>When you run a project, Plant SCADA can check if the local computer was previously set up for a project that is different to the one you have chosen to run. If this is the case, a dialog will prompt you to run the Computer Setup Wizard. Select this option if you want this check to occur.</p>
<b>Info popup time</b>	<p>The delay (in seconds) from the beginning of a database search until a search information window displays. The search information window displays the number of the traced records and allows you to cancel the search. You can cancel the search by selecting the Cancel button in the information window.</p>
<b>Report Editor</b>	<p>The editor that is used for editing Report Format Files. You need to enter the name of the executable file in this field. The default editor is Write (write.exe). If you are using Rich Text Format (RTF) reports, verify that your editor is RTF capable.</p>
<b>Maximum list box items</b>	<p>The maximum number of records that are displayed in drop-down combo boxes.</p>
<b>Warn about unused tags during full compile</b>	<p>Enables the generation of alert entries for unused tags that are not used directly in a project. The alert entries are included in the Compile Messages panel when a full compile is run. By default this option is not selected.</p> <p><b>Note:</b> For this option Alert entries are generated only for a full compile, not an incremental compile.</p>
<b>Log "tag not defined" warnings during compile</b>	<p>If you select this option, the compiler will generate a 'tag not defined' alert in the error log for any tags detected that are not defined in the variable database. As Plant SCADA allows you to include undefined tags on your graphic pages, this alert may be redundant and impractical. By deselecting this option, the alerts are still included in the displayed count, but they are not added to the error log.</p>
<b>Display equipment items when populating tag list</b>	<p>This option is selected by default. Equipment.Item tag</p>

Option	Description
	<p>references will populate 'insert tag' list when adding objects to a graphics page. If not set the variable tag list will be populated.</p> <p><b>Note:</b> If this option is selected and no equipment has been configured, the list will be empty.</p>
<b>Cascade changes across the system</b>	<p>Select <b>Always</b> to cascade field value changes through to all associated references within the Grid Editor. For example, if you change the <b>Tag Name</b> for a variable, any alarms that reference the variable as a trigger will also be updated with the new name.</p> <p>Select <b>Never</b> to limit a field value change to only the current view.</p> <p>Select <b>Prompt</b> to display the Cascade Changes message window where you can choose to cascade changes or limit it to the current view.</p> <p><b>Note:</b> This option will be disabled if the [General]ClusterReplication citect.ini parameter is enabled. This is to prevent changes from being cascaded across clusters and replacing values incorrectly.</p>

## The Grid Editor

The Grid Editor is an interface that displays information in a tabular format in numbered rows. Use the Grid Editor to [Add Rows to the Grid](#), [Edit Values in the Grid Editor](#) or [Delete Rows from the Grid](#) values individually or in bulk.

For example, the image below shows the Grid Editor with the equipment defined for the active project.



You can change the way data is displayed by changing the column on which the information is sorted or by changing the sort order. You can also [Customize Grid Editor Columns](#) by adding columns to or removing columns from the grid. To revert to the default display of the Grid Editor, select **Reset to Default Columns**.

You can also [Export to Microsoft™ Excel](#) and [Import from Microsoft Excel](#) data rows in the Grid Editor.

## Grid Editor Operations

The following table lists operations that are allowed in the Grid Editor.

Operations Allowed in the Grid Editor	Comments
<a href="#">Change the Default Sorting</a>	
<a href="#">Customize Grid Editor Columns</a>	
<a href="#">Filtering</a>	
<a href="#">Add Rows to the Grid</a>	
<a href="#">Edit Values in the Grid Editor</a>	
<a href="#">Delete Rows from the Grid</a>	Rows or columns that need to be selected should be contiguous.
<a href="#">Cut/Copy and Paste Rows and Columns</a>	

Operations Allowed in the Grid Editor	Comments
Export to Microsoft™ Excel	
Save Changes	
Cascade Changes	
Import from Microsoft Excel	

## See Also

[Plant SCADA Studio](#)

[Activity Bar](#)

[Activity Menu](#)

[Command Bar](#)

## Change the Default Sorting

By default, information displayed in the Grid Editor is sorted by the project name. Information in the grid can be sorted in one of three ways:

- Unsorted (the way the information was added)
- Ascending
- Descending.

---

**Note:** Sorting may take longer depending upon the volume of data in your system.

---

### To change the sorting:

1. Click the column header of the field that you want to use for sorting. For example, if you wish to sort data by **Name**, click the header **Name**. Data is sorted in ascending order by Name.

The screenshot shows the 'Equipment' tab selected in the top navigation bar of the 'System Model' interface. Below the toolbar, there are buttons for Save, Discard, Copy, Paste, Delete Row(s), Export All, Import All, and Equipment Editor. A dropdown menu labeled 'Model' is open. The main area is a grid table with columns: Row, Name, Cluster Name, Display Name, and Type. The 'Name' column header has a dropdown arrow pointing down, with a tooltip saying 'Click to change sort order. Click and drag to re-order.' The data rows are:

Row	Name	Cluster Name	Display Name	Type
1	Company.TopMilk.Production.Recipe.Sugar	Cluster1	Sugar	
2	Company.TopMilk.Production.Recipe.Straw	Cluster1		
3	Company.TopMilk.Production.Recipe.SkimMilk	Cluster1	Skim Milk	
4	Company.TopMilk.Production.Recipe.PreMixTime	Cluster1		
5	Company.TopMilk.Production.Recipe.MixTime	Cluster1		
6	Company.TopMilk.Production.Recipe.MixSpeed	Cluster1		
7	Company.TopMilk.Production.Recipe.Malt	Cluster1	Malt	

2. To reverse the sort order, click the column header again. Data is now sorted in descending order.
3. Click the header again to return the column to "unsorted".

**Note:** You can also sort by multiple columns by holding the **SHIFT** key down and clicking on the columns you want.

## See Also

[The Grid Editor](#)

[Plant SCADA Studio](#)

## Customize Grid Editor Columns

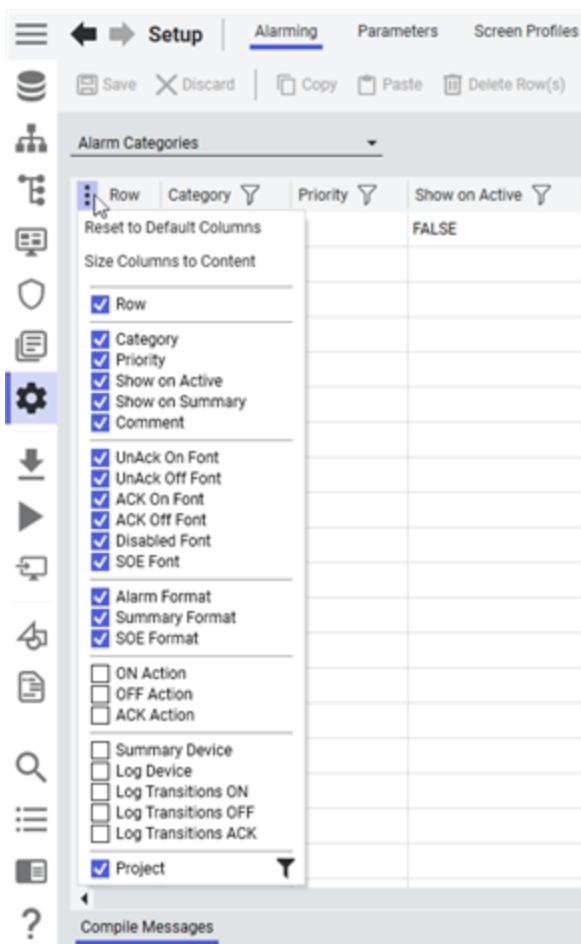
The Grid Editor displays a default selection of columns in the grid. However, this selection can be customized to suit your requirements.

### To customize the Grid Editor columns:

1. Click the icon displayed on the top left-hand side of the Grid Editor.



A list of available columns is displayed.



2. To select a column for display, select the check box next to the column name. To hide a column, clear the check box next to column name. Repeat this step until you have chosen the required columns. Columns are displayed or hidden as soon as you select or clear the selection of a check box.

**Note:** To hide row numbers, clear the check box next to the column name "Row".

3. To return to the default display, select **Reset to Default Columns**. Selecting this option will remove any filters applied to the grid.
4. To re-size columns according to their content, select **Size Columns to Content**.

To change the order in which the columns are displayed in the Grid Editor, click the column header, hold the left-mouse button down and drag the column to the desired location. Release the left-mouse button.

**Note:** The "Row" column cannot be moved.

## See Also

[The Grid Editor](#)

[Change the Default Sorting](#)

[Filtering](#)

[Add Rows to the Grid](#)

[Export to Microsoft™ Excel](#)

[Import from Microsoft Excel](#)

## Filtering

By default, the Grid Editor displays data for the selected project and its include projects. You can filter data displayed in a grid by one or more columns. The Filter icon on each column indicates the filter status:

Filter Button	Filter Status
	A filter has <b>not</b> been applied to the column.
	A filter has been applied to the column. This icon is also displayed next to the column name in the list of columns if a filter has been applied to a column.
	A change has been made to filtered data in the grid, which prevents you from making changes to the filtering and sorting until you save the changes.

### To filter data:

1. Click in the column by which you wish to filter. The Filter control is displayed and the "Contains" operator is selected by default.
  2. Select the required operator (the table below describes the available operators).
  3. In the area below the operator, type the text for the filter.
  4. Click **Apply**.
- To reset the filter, click **Clear**.
5. Click the **x** button to close the Filter control.

Filtering may take longer depending upon the volume of data in your system.

---

**Note:** On filtering data in the grid, row numbers are re-calculated.

---

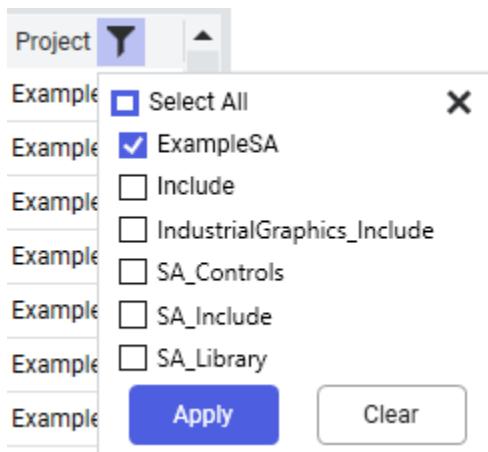
## Filter Operators

Filter Operator	Use Operator to Search for...
Contains	A numeric or alphanumeric value that includes the search text.
Does not contain	A numeric or alphanumeric value that does not include the search text.
Ends with	A numeric or alphanumeric value that ends with the search text.
Is contained in	A numeric or alphanumeric value that appears within

Filter Operator	Use Operator to Search for...
	the search text.
Is not contained in	A numeric or alphanumeric value that does not appear within the search text.
Is empty	Values that have not been specified (empty) within the column.
Is not empty	Any value within the column that is not empty.
Is equal to	A numeric or alphanumeric value that is identical to the search term.
Is not equal to	A numeric or alphanumeric value that is not identical to the search text.
Starts with	A numeric or alphanumeric value that begins with the search text.

### To filter data by project:

1. Click  in the Project column. A list of all projects is displayed.
2. Select the check box next to the project(s) for which you want to view data.



3. Click **Apply**.
- To reset the filter, click **Clear**. The Grid Editor displays data for the selected projects. If no project is selected, the grid will not display any data.
4. Click the **x** button to close the list of projects.

### See Also

[The Grid Editor](#)  
[Save Changes](#)

## Add Rows to the Grid

You can add as many rows as you wish to a grid in Plant SCADA Studio.

### To add rows to a grid:

1. Scroll to the last row in the grid.
2. Type the information in each column (see [Edit Values in the Grid Editor](#)), or in the boxes provided for each field in the Property Grid (see [Edit Values in the Property Grid](#)).  
Alternatively, you can [Cut/Copy and Paste Rows and Columns](#) into grid rows.
3. Click **Save**. For more information, see [Save Changes](#).

## See Also

[The Grid Editor](#)

[Edit Values in the Grid Editor](#)

[Delete Rows from the Grid](#)

[Cut/Copy and Paste Rows and Columns](#)

## Edit Values in the Grid Editor

Values in a grid can be edited directly in the grid or through the Property Grid.

### To edit values in the Grid Editor:

1. Select the field that contains the value you would like to edit
2. Click on the field a second time to highlight the value.
3. To select multiple fields, select the first field. Then, hold the SHIFT key and use the arrow keys or click the left mouse button to select additional fields.
4. Make the required change to the value. If a field is read-only, you will not be able to edit it.

---

**Note:** If the selected field values are different, they will be overwritten by the value you specify in the field that you edit.

---

5. Click **Save** to keep the change, or **Discard** to undo the change.

---

**Note:** Some changes you save will cascade through to all associated references within the Grid Editor. For more information, see [Cascade Changes](#).

---

Grid Editor values are also updated when changes are made in the Property Grid (see [Edit Values in the Property Grid](#)). You can also use the [Property Grid](#) to edit multiple rows at the same time (see [Edit Multiple Values](#)).

## See Also

[The Grid Editor](#)

[Add Rows to the Grid](#)

[Delete Rows from the Grid](#)

## Cut/Copy and Paste Rows and Columns

### Delete Rows from the Grid

Rows in a grid may be deleted if required.

#### To delete rows from the grid:

1. Click in the first column on the left hand side of the row you want to delete. The current row is selected.
2. To delete multiple rows, hold down the **Shift** key, and use the **Up/Down** arrow keys or the **Shift** key to select additional rows for deletion.
3. Press the **Delete** key or click the **Delete Row(s)** button on the Command Bar. A strike through appears in the selected row(s).

The screenshot shows the 'System Model' tab selected in the top navigation bar. Below it is a toolbar with buttons for Save, Discard, Copy, Paste, Delete Row(s), Export All, Import All, and Equipment Editor. The main area is a grid titled 'Model' with columns: Row, Name, Cluster Name, Display Name, and Page. Rows 1 through 4 are highlighted with a blue selection box, indicating they are selected for deletion. The other rows (5, 6, 7) are visible but not selected.

Row	Name	Cluster Name	Display Name	Page
1	Company.TopMilk.Production.Recipe.Sugar	Cluster1	Sugar	Overview_L2
2	Company.TopMilk.Production.Recipe.Strawberry	Cluster1		Overview_L2
3	Company.TopMilk.Production.Recipe.SkimMilk	Cluster1	Skim-Milk	Overview_L2
4	Company.TopMilk.Production.Recipe.PreMixTime	Cluster1		Overview_L2
5	Company.TopMilk.Production.Recipe.MixTime	Cluster1		Overview_L2
6	Company.TopMilk.Production.Recipe.MixSpeed	Cluster1		Overview_L2
7	Company.TopMilk.Production.Recipe.Malt	Cluster1	Malt	Overview_L2

4. Click **Save** to complete the deletion.

### See Also

[The Grid Editor](#)

[Add Rows to the Grid](#)

[Cut/Copy and Paste Rows and Columns](#)

## Cut/Copy and Paste Rows and Columns

Rows in a grid can be copied and pasted in a different grid.

#### To copy a subset of columns or rows and paste it in the grid:

1. Select the required row by clicking the box to the left of the row. To select multiple rows, select the first row and hold the **Shift** key while you select additional rows with the **Up/Down** arrow keys or click to select additional rows.

To select a column value, click in the column cell. To select multiple values, hold the left mouse button down

and drag up or down and/or right/left to select additional values.

2. Press **Ctrl + C** on the keyboard to copy the selected columns or rows. To move the columns or rows to a different location in the grid, press **Ctrl + X**.
3. Navigate to the column or row in the grid where you wish to paste the rows.
4. Press **Ctrl + V** to paste the rows.

**Note:** You can move rows of data between projects by using the **Cut** operation in the source project, which you can **Paste** into a destination project. When you do this, the rows that you selected to move will be deleted from the source project.

## See Also

[The Grid Editor](#)

[Add Rows to the Grid](#)

[Delete Rows from the Grid](#)

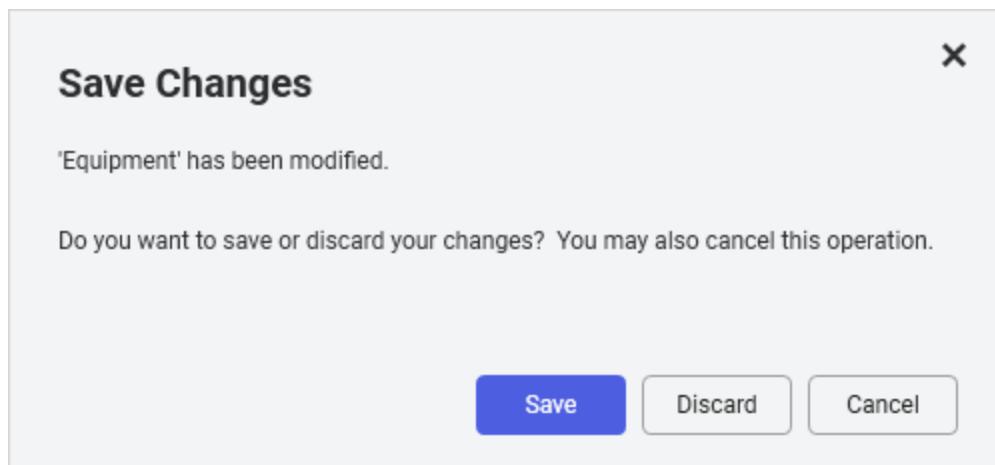
## Save Changes

After updating information in the Grid Editor, click **Save** on the Command Bar. To revert to the previous value(s) without saving your updates, click **Discard** on the Command Bar.

The screenshot shows the AVEVA Plant SCADA Grid Editor interface. At the top is a navigation bar with icons for back, forward, and search, followed by the title "System Model". Below the title is a toolbar with buttons for Save (highlighted with a red box), Discard, Copy, Paste, Delete Row(s), Export All, Import All, and Equipment Editor. The main area is a data grid titled "Model" with the following columns: Row, Name, Cluster Name, Display Name, and Page. The data rows are:

Row	Name	Cluster Name	Display Name	Page
1	Company.TopMilk.Production.Recipe.Sugar	Cluster1	Sugar	Overview_L2
2	Company.TopMilk.Production.Recipe.Strawberry	Cluster1		Overview_L2
3	Company.TopMilk.Production.Recipe.SkimMilk	Cluster1	Skim Milk	Overview_L2
4	Company.TopMilk.Production.Recipe.PreMixTime	Cluster1		Overview_L2
5	Company.TopMilk.Production.Recipe.MixTime	Cluster1		Overview_L2
6	Company.TopMilk.Production.Recipe.MixSpeed	Cluster1		Overview_L2
7	Company.TopMilk.Production.Recipe.Malt	Cluster1	Malt	Overview_L2

If you attempt to navigate away from the view without saving changes, the following message is displayed.



Click **Save** to keep your changes, and remain in the current activity.

Click **Discard** to cancel your changes, and navigate away from the current activity.

Click **Cancel** to stay in the same view without saving your changes. The changes you have made will continue to be displayed.

## See Also

[The Grid Editor](#)

[Cascade Changes](#)

[Edit Values in the Grid Editor](#)

[Edit Values in the Property Grid](#)

[Edit Multiple Values](#)

## Cascade Changes

Changes can be cascaded through to all associated references within the Grid Editor.

For example, if you change the cluster or tag name for a variable, any alarms or expressions that reference the variable as a trigger will also be updated with the new name.

**Clusters** context is a key determinant for changes being cascaded. When you change a tag name, the cluster associated with it will be used as a filter, and changes will be cascaded to alarms or expressions for that cluster only. If a cluster is not defined for a tag, the cluster name will be determined by the cluster specified for the associated I/O device, or the I/O server if the I/O device does not have a cluster specified. If you change the name of an I/O device in the grid, the cluster name of linked I/O server will be used for cascading changes to linked references.

Plant SCADA does not consider case sensitivity when it determines what needs to be changed.

Cascading changes are not supported for the following:

- Labels, groups, equipment states and remapping
- Alarm categories
- Equipment.item references used in expressions.

**Note:** Cascading changes are only passed through to associated references within the Grid Editor. References

that occur in locations such as graphics pages and Cicode files will not be automatically updated. Changes will not be cascaded if grid data is manipulated externally, for example through a tag import or modifications made through the Equipment Editor.

**To cascade changes on update:**

1. Open the Options dialog from the Activity Bar.
2. Set the Cascade changes across the system option to Always.

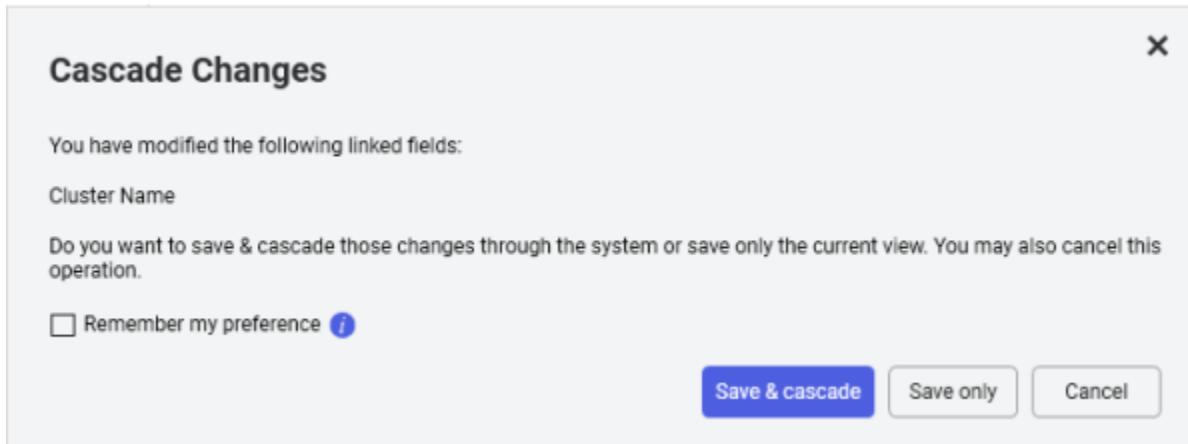
**To disable cascading changes:**

1. Open the Options dialog from the Activity Bar.
2. Set the Cascade changes across the system option to Never.

**To receive a message before you cascade changes:**

1. Open the Options dialog from the Activity Bar.
2. Set the Cascade changes across the system option to Prompt.

If you select the Prompt option, the following message will be displayed when you save your changes.



- Click **Save & cascade** to save the change and update the field value across the system.

**Note:** Details of unsuccessful cascaded changes are logged in a Change Log, which is located in the %PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Logs\ChangeLogs. A cascading change may be unsuccessful if you update a field with a value that exceeds the maximum number of characters the field can hold, if duplicated tag references occur, or if the database file that needs to be updated is inaccessible.

- Click Save only if you want to save the updated field value for the current view only.
- Click Cancel to stay in the same view without saving your changes. The changes you have made will continue to be displayed.
- Select the Remember my preference option to automatically use the currently selected option for subsequent save operations. Whenever required, you can switch to a different option using the Options window.

**Note:** Changes that are cascaded across the system are not applied to system projects.

## See Also

- [The Grid Editor](#)
- [Save Changes](#)
- [Edit Values in the Grid Editor](#)
- [Edit Values in the Property Grid](#)
- [Edit Multiple Values](#)

## Export to Microsoft™ Excel

You can export all the information in the current grid to Microsoft Excel. Any filters or grid selections will not apply, all data will be exported.



### WARNING

#### LOSS OF DATA

If you make an incorrect selection when you chose to import a CSV file into Plant SCADA Studio, you may overwrite your data with outdated or inappropriate values.

To avoid this situation, it is recommended that you use a naming convention for exported CSV files that clearly describes the content included in the file.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

#### To export all records:

1. On the Command Bar, click **Export All**. The Export data to CSV file dialog box is displayed.
2. Select the folder in which you want to save the CSV file, and specify a name for the file.
3. Click **Save**. The data is exported to a Comma Separated Values (CSV) file. A message is displayed on successful export. You can now open the CSV file in Microsoft Excel.

## See Also

- [The Grid Editor](#)
- [Add Rows to the Grid](#)
- [Delete Rows from the Grid](#)

## Import from Microsoft Excel

Data from Microsoft Excel can be imported into your Plant SCADA project. When you import a CSV file, data in the grid will be overwritten only for those projects that have corresponding data in the CSV file. Data for other projects will not be affected.

**WARNING****LOSS OF DATA**

If you make an incorrect selection when you chose to import a CSV file into Plant SCADA Studio, you may overwrite your data with outdated or inappropriate values.

To avoid this situation, it is recommended that you use a naming convention for exported CSV files that clearly describes the content included in the file.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

---

**Note:** Read-only records will remain read-only after an import even if these values are modified prior to an import. Modified values will result in a mismatch between values generated by the Equipment Editor or a tag import and those in the Grid Editor.

**To import all data from a CSV file:**

1. Save the data from in Microsoft Excel file to a Comma Separated Values (CSV) file.
2. Click **Import** in Plant SCADA Plant SCADA Studio. The Import Data from CSV file dialog box appears.
3. Select the file you want to import, and click **Open**. A warning message is displayed indicating that the entire database will be overwritten on import.
4. Click **OK** to proceed with the import. A message is displayed on successful import.

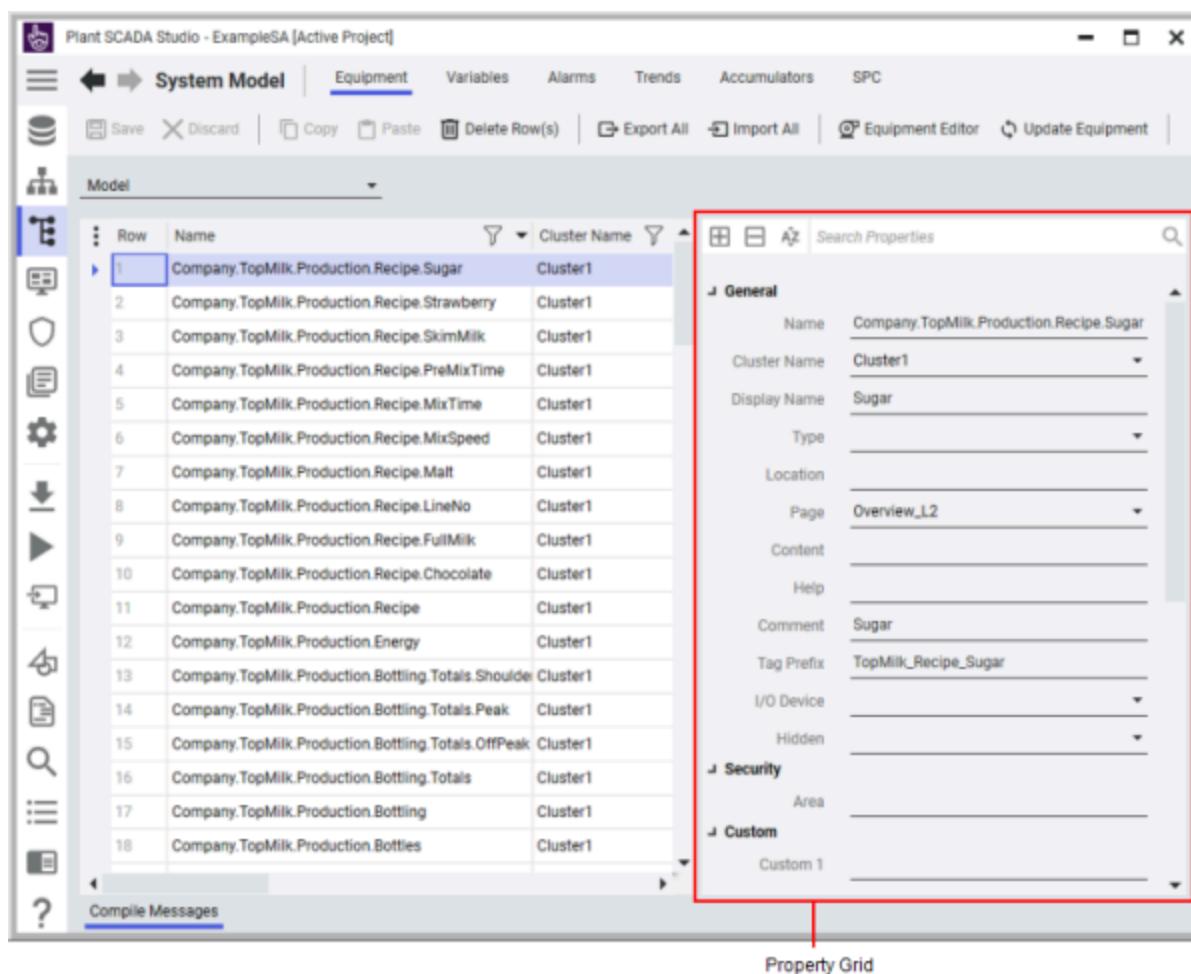
**Note:**

- If the data in the imported file is not valid (for example, it contains entries for a project that does not exist in your active project hierarchy), an error message is displayed and data import does not go through.
- If the imported file does not contain any entries for a particular project in your active project hierarchy, the data for that particular project will not be overwritten.

**See Also**[The Grid Editor](#)[Add Rows to the Grid](#)[Delete Rows from the Grid](#)

## Property Grid

The Property Grid is displayed to the right of a grid or browsing view. It displays details of the selected item(s). For example, if you have selected a piece of equipment in the System Model activity, the associated properties will display in the Property Grid.



By default, properties of the selected item are displayed as a categorized list.

- To collapse the categories, click
- To expand the categories, click
- To view properties as a flat list, click
- To return to the categorized list view, click

You can [Edit Values in the Property Grid](#), or you can use the Property Grid to edit multiple items at the same time (see [Edit Multiple Values](#)).

## See Also

[The Grid Editor](#)

## Edit Values in the Property Grid

### To edit values in the Property Grid:

1. In the Grid Editor, select the row that you would like to edit.

Or

In the **Visualization | Pages** view, select the page you would like to edit (see [Browse Pages in Plant SCADA Studio](#)).

The values associated with the selected item will appear in the Property Grid to the right.

2. Click on the field you want to edit in the Property Grid and make the required changes. If a field is read-only, you will not be able to edit it.
3. Press the **Enter** key (or click away from the field) to load the changes.
4. Click **Save** to keep the changes, or **Discard** to remove them.

Some saved changes will cascade through to all associated references. For more information, see [Save Changes](#).

**Note:** You can also use the Property Grid to simultaneously [Edit Multiple Values](#). If multiple items are selected and their property values differ, the Property Grid will display "Values differ" in the associated field.

## See Also

[Property Grid](#)

## Edit Multiple Values

### To simultaneously edit multiple values in the Property Grid:

1. In the Grid Editor, select the rows you would like to edit.

To select multiple rows, select the first row and hold the **Shift** key while you select additional rows with a mouse-click or with the up or down arrow keys.

Row	Name	Cluster Name	Page
4	SteelMill.Welder	Cluster1	SteelMill
5	SteelMill.EntryAccumulator	Cluster1	SteelMill
6	Loops.Level	Cluster1	LoopPage
7	Loops.Temperature	Cluster1	LoopPage
8	Loops.Density	Cluster1	LoopPage
9	Building	Cluster1	
10	Building.External	Cluster1	
11	Building.External.Light1	Cluster1	
12	Building.External.Light2	Cluster1	
13	Building.External.Light3	Cluster1	
14	Building.External.Light4	Cluster1	
15	Building.External.Light5	Cluster1	
16	Building.Level1	Cluster1	

**General**

Name	Values differ
Cluster Name	Cluster1
Display Name	
Type	Building light
Location	
Page	
Content	
Help	
Comment	Values differ
Tag Prefix	Values differ

If you are using the **Visualization | Pages** view, select the pages you would like to edit (see [Browse Pages in Plant SCADA Studio](#)).

If any corresponding fields in the selected items share common values, these values will appear in the Property Grid. For example, if you selected the three rows displayed above, "Cluster1" would appear in the **Cluster** field, and "Building light" would appear in the **Type** field.

If the selected field values are different, the text "Values differ" will be displayed for the corresponding fields in the Property Grid. If a field is read-only, you will not be able to edit it.

- To edit the values for the selected items, enter the changes in the fields in the Property Grid and press the **Enter** key.
- Click **Save** to keep the changes, or **Discard** to undo them.

---

**Note:** By default, any changes you save will cascade through to all associated references. For more information, see [Save Changes](#).

---

## See Also

[Property Grid](#)

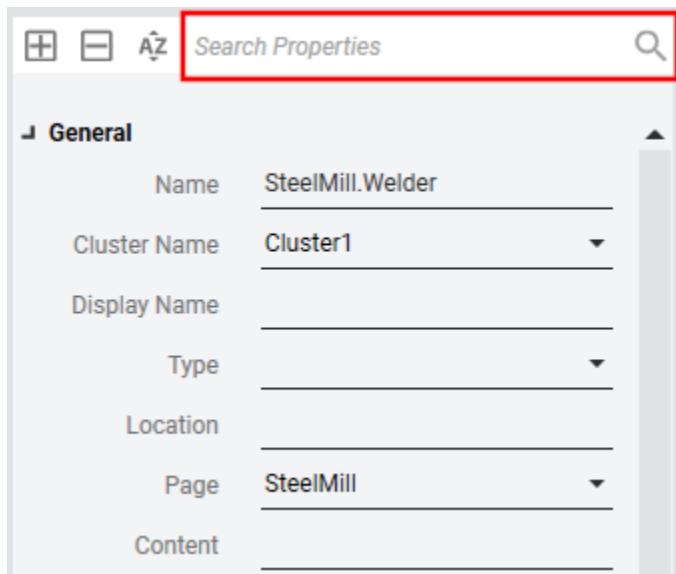
## Search for a Property

You can search for properties in the Property Grid. Instead of going through a long list of properties to edit a single property, you can search for the required property by name.

### To search for a property

- Click anywhere in the [The Grid Editor](#) or Pages view.

2. Type the entire property name or part of the property name in the **Search Properties** box.



All properties that contain the search term will be displayed. For example, to locate the Cluster Name property, you can type "clu", "clus" or "cluster" in the **Search Properties** box. If you search for "at", the properties Data Type and Format will be displayed.

## See Also

[Property Grid](#)

## Using Find and Replace

You can use the **Find and Replace** functionality to locate specified text in your projects, graphics and views. You can perform global text replaces as well as export search results.

This is explained in the following topics:

- [The Find and Replace Window](#)
- [Specify Search Coverage](#)
- [Use the Results List](#)
- [Remove Results](#)
- [Export Results](#)
- [Jumping to a Result \(Go To\)](#)
- [Replace Results](#)
- [Find and Replace Error Messages](#)

There is also a topic on [Troubleshooting Searches](#) to help determine if a search has been correctly configured when unexpected results are returned.

## The Find and Replace Window

You can open the **Find and Replace** window from:

- **Plant SCADA Studio** — used to find and replace text strings in your projects, included projects, graphics and Plant SCADA Studio views.
- **Graphics Builder** — used to find and replace text within a single graphics page, template, or Genie (including fields on the Metadata and Associations tabs, apart from the "In Use" field).

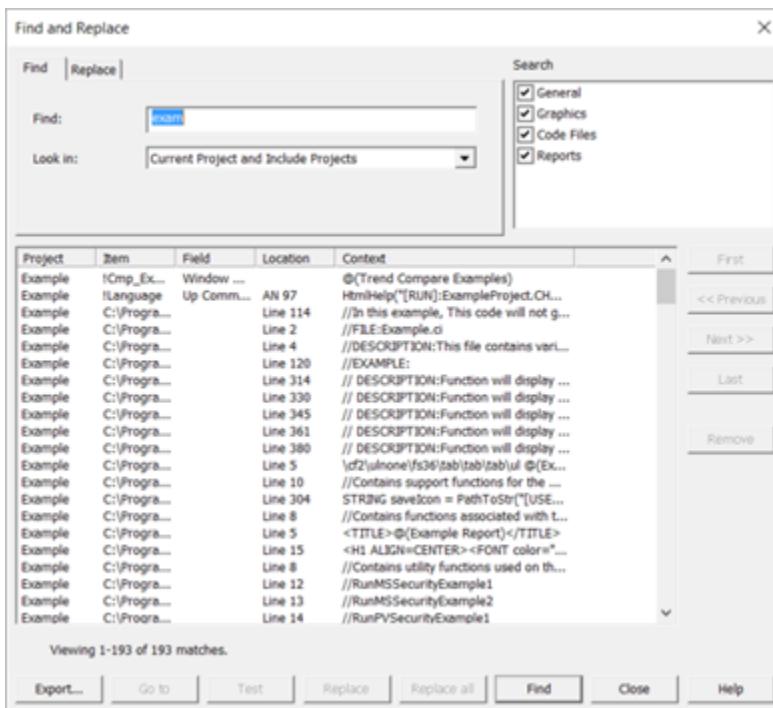
You can configure your search coverage, view your results, replace results, or open a search result for more information.

### To display the Find and Replace window:

1. From Plant SCADA Studio, click the following icon on the Activity Bar.



2. From Graphics Builder, click **Edit | Find or Edit | Replace**. The window box appears with either the **Find** tab or **Replace** tab selected, depending on which command you selected.
3. From Plant SCADA Studio or Graphics Builder, press **CTRL + F** on the keyboard.



### To search for text:

1. In the **Find** box, type the text string you want to search for. The search is not case-sensitive, so it doesn't matter whether you enter lower- or uppercase letters.

You can enter an entire string or a portion of the string you want to find. For example, typing BIT will return any string containing BIT, such as BIT\_1, BITE, HABIT, HABITS, and so on. You cannot enter wildcard

characters, but you can include special characters, as well as spaces if you want.

2. Specify your [Specify Search Coverage](#) using the **Look in** and **Search options** lists.
3. Click **Find**. Search results appear in the [Use the Results List](#) when the search completes. The status text under the results list indicates the progress of the search.

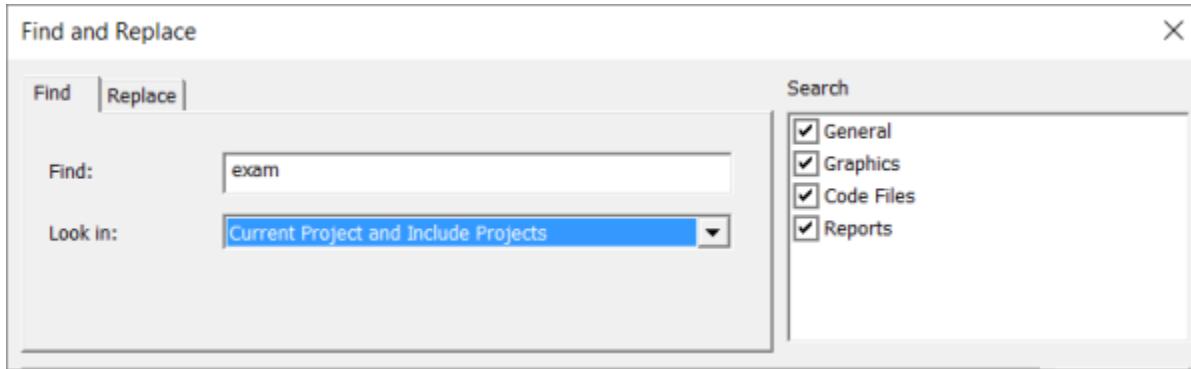
**Note:** When you start a search, the **Find** button changes to a **Stop** button you can use to exit the search. If you stop a search, a partial list of the results is displayed.

#### To replace text:

1. Click the Replace tab.
2. In the **Find** box, type the text you want to search for.
3. Specify your search coverage test
4. Specify your [Specify Search Coverage](#) using the **Look in** and **Search options** lists.
5. In the **Replace with** box, enter the replacement text.
6. Click **Find**.
7. View the [Use the Results List](#).
8. Replace Results using **Replace** or **Replace All**.

## Specify Search Coverage

You can specify search coverage using the **Look in** and **Search options** lists.



When you choose a **Look in** option, the **Search options** change. Each time you select a **Look in** option, associated **Search conditions** are selected by default. The **Look in** options work with the **Search options** as follows.

- Selecting **Current Project** or **Current Project and Include Projects** displays the options described below.
  - **General:** Searches configuration databases associated with a project as well as included projects (if that option is selected).
  - **Graphics:** Performs an "express search" for graphics pages only (the graphics page does not have to be currently open). *This search will not find text in symbols, Genies, or templates if they have not been used on a page.* If you don't find the text you want, try the more comprehensive graphics search by opening a page, Genie, symbol, or template and selecting **Current Graphic** as the **Look in** option (see below).
  - **Code Files:** Searches Cicode/VBA files in the current project (and included projects, if that option is selected).

- **Reports:** Searches report files within the project folder and included projects (if that option is selected).
- Selecting **Current Graphic** makes the options described below available. (This search is a more extensive search than that performed by the **Current Project:Graphics** search described above, and will search graphics documents that are currently open.)
  - **Inside Genies and Symbols** includes graphics objects contained within a Genie or symbol.
  - **Inside Templates** includes objects contained within a page template.
- Selecting **Current Unfiltered View:** Searches the current Grid Editor view. For example, if you are viewing variable tags, the search will be conducted within all variable tags for the Active project and all its includes.

**Note:** Searching the current filtered view does not take into account any project or column filters that have been applied to the view.

## Use the Results List

As matches are found they are listed in the results list. The results list shows an overview of items that match the string entered in the search.

The results list can display a maximum of 200 results per page, sorted by project and then item (you cannot change the sort order).

Project	Item	Field	Location	Context
Example	!Cmp_Ex...	Window ...		@(Trend Compare Examples)
Example	!Language	Up Comm...	AN 97	HtmlHelp("[RUN]:ExampleProject.CH...")
Example	C:\Progra...		Line 114	//In this example, This code will not g...
Example	C:\Progra...		Line 2	//FILE:Example.ci
Example	C:\Progra...		Line 4	//DESCRIPTION:This file contains vari...
Example	C:\Progra...		Line 120	//EXAMPLE:
Example	C:\Progra...		Line 314	// DESCRIPTION:Function will display ...
Example	C:\Progra...		Line 330	// DESCRIPTION:Function will display ...
Example	C:\Progra...		Line 345	// DESCRIPTION:Function will display ...
Example	C:\Progra...		Line 361	// DESCRIPTION:Function will display ...
Example	C:\Progra...		Line 380	// DESCRIPTION:Function will display ...
Example	C:\Progra...		Line 5	\cf2\ulnone\fs36\tab\tab\tab\ul @({Ex...
Example	C:\Progra...		Line 10	//Contains support functions for the ...
Example	C:\Progra...		Line 304	STRING saveIcon = PathToStr("[USE...
Example	C:\Progra...		Line 8	//Contains functions associated with t...
Example	C:\Progra...		Line 5	<TITLE>@({Example Report})</TITLE>
Example	C:\Progra...		Line 15	<H1 ALIGN=“CENTER”><FONT color="...
Example	C:\Progra...		Line 8	//Contains utility functions used on th...
Example	C:\Progra...		Line 12	//RunMSSecurityExample1
Example	C:\Progra...		Line 13	//RunMSSecurityExample2
Example	C:\Progra...		Line 14	//RunPVSecurityExample1

The results list contains the following columns:

Column	Description
<b>Project</b>	The name of the project in which the found text occurs.
<b>Item</b>	Depends on the type of document in which the item occurs. If the document type is a: <b>Database</b> - User-friendly name of the database.

Column	Description
	<b>Page</b> - Name of the page. <b>Cicode/VBA</b> - Name and path of the Cicode/VBA file. <b>Report</b> - Name of the report.
<b>Field</b>	Identifies that portion of the document/database in which the found item occurs in. For example, if the found item appears in a database, this refers to the column name in the database. Be aware that the search covers both expression/command as well as numeric properties.
<b>Location</b>	Shows the specific record number, AN, or line number on which the found item occurs within the document/database.
<b>Context</b>	An example of the context in which the found item occurs within the project. For example, if the document type is a: <b>Database</b> - a search result of BIT* might have a context of BIT_!. <b>Page</b> - BIT* might have a context of Toggle(BIT_!). <b>Cicode/VBA</b> - UserName might have a context of FUNCTION GetUserName() <b>Report</b> - PUMP* might have a context of @(Pump A)

You can toggle between the Find and Replace functionality without losing the search results, but if you close the Results page, your search results are lost.

---

**Note:** You can re-size list columns by moving your mouse cursor onto the separator between the list columns. When the mouse cursor changes shape to a black bar with arrows, drag the column to the new size. You can also double-click the vertical bar between fields to re-size that field to fit the widest item.

---

## Remove Results

You can remove a search result from the Results window. Results that are removed are not included in exports or in replacement operations. Removing a result does not delete it, but merely removes it from the Results window.

**To remove a result:**

1. Click to select the result you want to remove and click **Remove**.  
The result is removed from the Results window.

## Export Results

You can export search results in a tab-delimited format to a specified location. Results are exported in the format:

<Project> <Item> <Field> <Location> <Context>

If the Results window contains more than 200 results, every result is exported, not just the ones currently displayed. If you remove an item from the results list, it will not be exported. (For details on removing results, see [Remove Results](#).)

If you export an item that has a context, the context string is stripped of tabs and new line characters.

Results exported are in Unicode format. Because of this, two leading characters and two trailing characters are added to the file, but in most cases will remain hidden. When exporting results, you need to use a viewing tool that supports Unicode.

#### To export results:

1. With the search results you want to export listed in the Results window, click **Export**.
2. Specify the location in the window and then click **Save**. If the file already exists, you're given the option to overwrite the file. Status text under the results list indicates the progress of the export.

---

**Note:** If you want to stop the export, click **Stop**. You cannot perform a partial export, so clicking **Stop** cancels the export entirely.

---

## Jumping to a Result (Go To)

You can jump to an individual result to see where the result occurs. Depending on the type of document that contains the search result, the following occurs:

- **Grid:** The relevant grid is displayed. If you attempt to go to a result that is hidden by a filter, a message that you cannot view the result is displayed.
- **Cicode/VBA:** The document opens within the Cicode Editor and the text string is highlighted.
- **Graphic:** The page opens in the Graphics Builder and the Properties dialog box appears for the object that contains the text string. The property containing the text string is displayed. If the string occurs on or inside a Genie, the Genie form also appears.

#### To navigate to a result:

1. With the search result you want to navigate to selected in the Results window, click **Go To**.  
The document or form containing the occurrence opens.

## See Also

[Replace Results](#)

## Replace Results

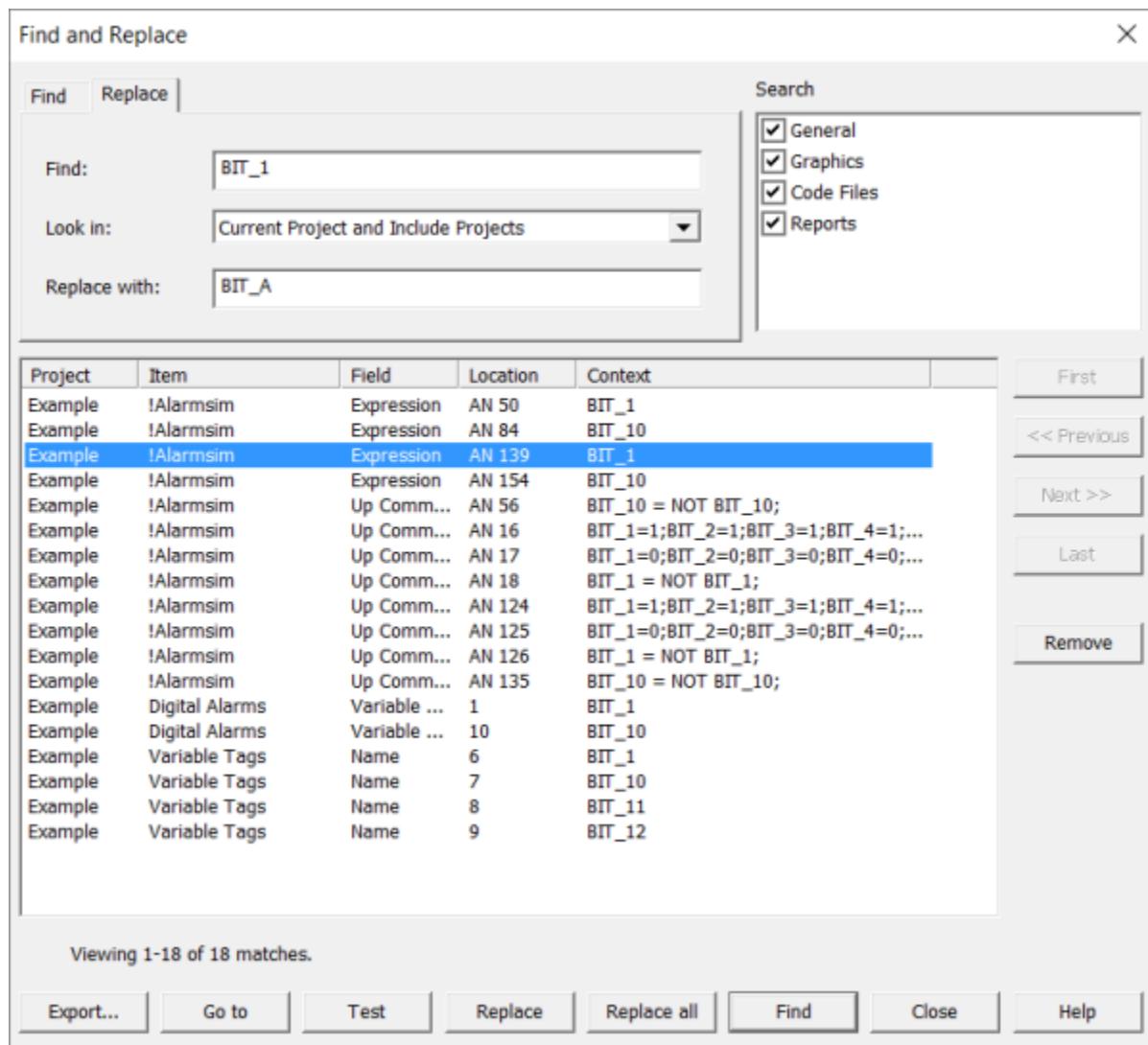
You can replace single results or multiple results with the replacement text string you specified. You can also test a single result before replacing it. Depending on the type of document that contains the search result, the following occurs when a replacement is made:

- **Database:** The result is replaced with the replacement text and the database record updated. The form containing the search result is not opened; to see the location of the search result before or after the

replacement is made, use the .

- **Cicode/VBA:** The Cicode file containing the matched text loads (if it is not loaded already), the replacement is made, and the file saved.
- **Graphic:** The page opens in the Graphics Builder (if it is not already) and the replacement made. If the page is open and contains unsaved changes, you're instructed to save or discard the changes before making the replacement. If there are multiple changes to be made to the same graphics page, the page remains open until every change has been made.
- **Report:** The found text is replaced with the replacement text and the file is saved.

**Note:** Replacements cannot be undone once performed. Take care to check your replacements before making them, especially when working with multiple replacements.



### To test a result:

1. With the result you want to test highlighted, click **Test**. A dialog box appears showing the result of the text replace.
2. Click **Accept** to accept the text replacement, or click **Cancel**.

**To replace a single result:**

1. With the result you want to replace highlighted, click **Replace**. The replacement is made and the result removed from the Results window. The next result in the list is then selected.

**To replace multiple results:**

1. With the search results you want to replace listed in the Results window, click **Replace All**. A confirmation dialog appears.
2. The replacements are made and removed from the Results window. (Replacements that are not made remain in the results list. This will occur if, for example, you try to replace a property that is read-only.)

---

**Note:** Clicking Stop during this process does not undo any replacements already made.

---

When attempting to make a replacement, you might encounter an alert message. For details, see [Find and Replace Error Messages](#).

## Find and Replace Error Messages

Find and Replace will display one of the following errors messages if it cannot replace a text string:

### File in Use

This alert message appears if the database or file that is necessary for writing to has become unavailable. This may be the case if the database/file is being used by a third-party application.

Do one of the following:

- Click **Try Again** (Default) to repeat the operation on the database/file.
- Click **Ignore** to skip the operation on this file.
- Click **Ignore All** button to skip any operations on files that are currently in use; this option causes this message not to reappear.

### Replaced Text Truncated

This alert message appears if performing the text replacement in a DBF field exceeds the field width limits. For example, if an Engineering Units field containing % is replaced with "%LONGTEXTSTRING", a "replaced text truncated" alert message appears because this field has a max width of 8. The replacement text would therefore be "%LONGTEX".

This alert message does not occur if searching the current graphics page.

Do one of the following:

- Click **Yes** to commit the truncated text to the database. Usually this will generate a compilation alert message when the project is compiled.
- Click **Yes to All** to commit every change to fields regardless if truncation exists without displaying the alert again.
- Click **No** (default) to leave the text as is.

- Click **No to All** button to leave truncated fields as is.

## Original Text Not Found

This alert message appears when attempting to replace an item on the current graphics page when the animation could not be found or the text could not be found. This would occur if the animation was deleted, or if the text found in an animation's field was changed after the find but before replacing the item.

In the example below, the Fill Level Maximum contained a value of 23, and the search text was 23. Before replacing the record, it was changed to 66.

Do one of the following:

- Click **Ignore** to skip this operation, leave the entry in the list, and move on to the next replacement if it exists.
- Clicking **Ignore All** acts like the **Ignore** button, except that it skips all occurrences.
- Click **Stop** to stop the replacement at the current record.

## Replaced Text Out of Range

This alert message appears when carrying out a replacement on the current graphics page in two different circumstances:

1. The field is text and is too long for the allowable field width.
2. The field is a numeric field, and would be out of the allowable range for a replacement value.

In the example below, the Fill Level Maximum allows a range of 0-100, and the value was 23 and is being replaced with 101, which would be out of range.

Do one of the following:

- Click **Ignore** to skip this operation, leave the entry in the list, and move on to the next replacement if one exists.
- Clicking **Ignore All** acts like the **Ignore** button, except that it skips all occurrences.
- Click **Stop** to stop the replacement at the current record.

## Replacement Text Not Numeric

This alert message will appear when carrying out a replacement on the current graphics page when the field being replaced is a numeric field, and the replacement text contains a non-numeric value.

In the example below, the Fill Level Maximum contained a value of '23', and the replacement text was 'fred'.

Do one of the following:

- Click **Ignore** to skip this operation, leave the entry in the list, and move on to the next replacement if it exists.
- Clicking **Ignore All** acts like the **Ignore** button, except that it skips all occurrences.
- Click **Stop** to stop the replacement at the current record.

## Field is Read-only

This alert message appears when replacing an item on the current graphics page when the field being replaced is part of a linked object like a Genie or template.

In the example below, the Expression field was part of an object that was part of a genie.

Do one of the following:

- Click **Ignore** to skip this operation, leave the entry in the list, and move on to the next replacement if it exists.
- Clicking **Ignore All** acts like the **Ignore** button, except that it skips all occurrences.
- Click **Stop** to stop the replacement at the current record.

## Undetermined error

This alert message appears when carrying out a replacement on the current graphics page when a general error is detected, and not happen in normal operation.

Do one of the following:

- Click **Ignore** to skip this operation, leave the entry in the list, and move on to the next replacement if it exists.
- Clicking **Ignore All** acts like the **Ignore** button, except that it skips all occurrences.
- Click **Stop** to stop the replacement at the current record.

## Troubleshooting Searches

If you do not find a result that you expected to find, check the following points and then perform your search again.

- Did you spell the text string correctly?
- Did you include the correct number of spaces?
- Are you using the appropriate Look in option?
- Are you using the appropriate Search options?
- Are you searching in the correct project?
- Are you using the correct graphics search?
- If you are using the graphics page search, do you have the correct graphics page open?

## Projects

Plant SCADA projects are repositories that hold the configuration information for your system that includes information such as I/O devices, tags, alarms and graphic pages that are used to build a runtime system.

The configuration for a runtime system can be spread across multiple projects depending upon the scale of operations. Small, simple operations may require only a single project that houses all components required for runtime. For larger, complex operations or multi-site operations, several projects can be created based on specific plant areas, engineering processes or libraries, which are "included" together to form a single merged configuration used at runtime.

## Project Types

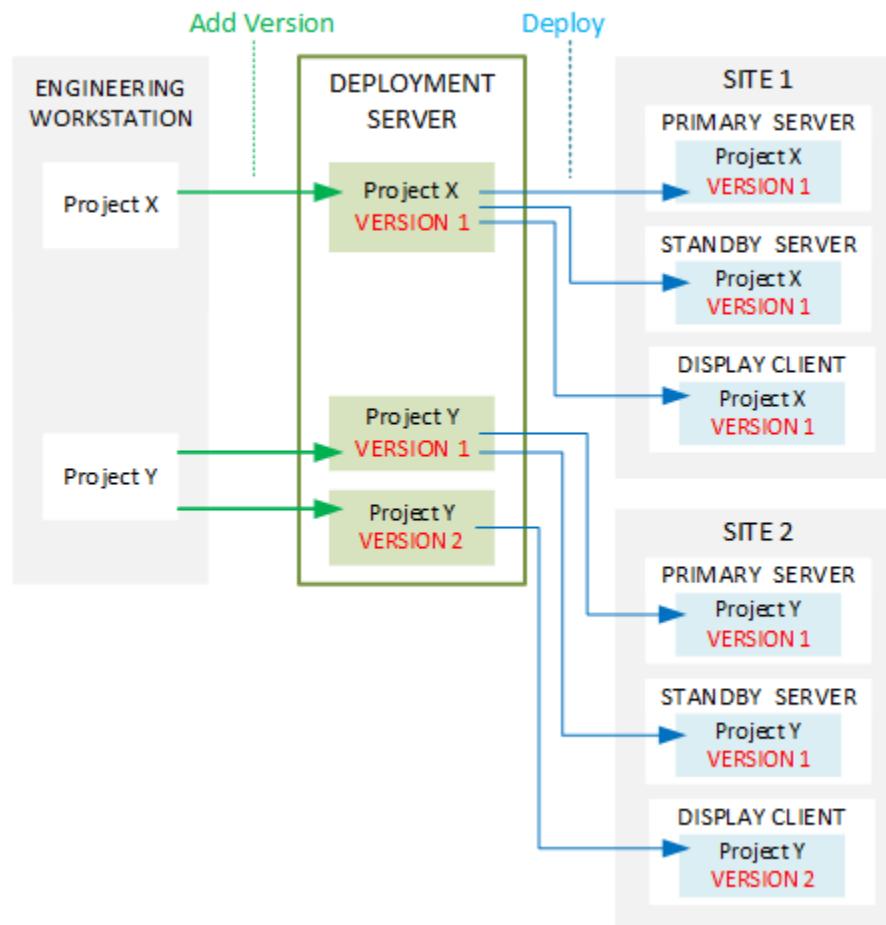
Projects created by you are referred to as "user" projects. In addition, Starter and System projects are also available for use.

- Starter projects initialize configuration in a newly created project to get you started quickly and can be run and/or deployed straight away. A Starter project can be selected when you [Create a Project Using a Starter Project](#).
- System projects provide internal, out-of-the-box content such as pages, settings and in-built functions that can assist with the building of your projects. System projects should not be modified as they are updated and replaced with each new version of Plant SCADA.

See [Project Types](#).

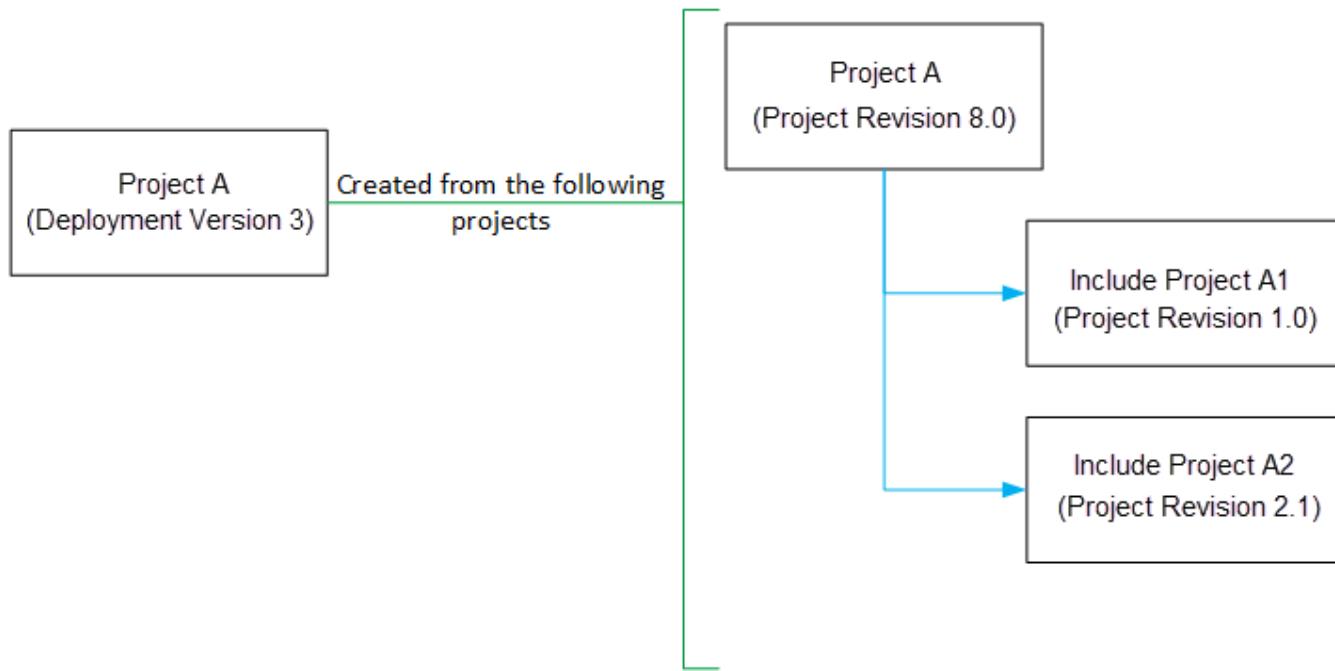
## Project Deployment

Multiple projects can be used to segregate different operational functions and plant areas, as well as simplifying the process of deployment. Deployment in a complex SCADA system typically spans several computers in a network, and may require different projects to be deployed on different computers for runtime. Minor and major project changes can be rolled out quickly and efficiently. For more information, refer to the Deployment section.



## Project Revisions

You can track changes to your projects through the revision fields in the [Edit a Project's Properties](#). A good practice would be to assign a major revision number for significant project changes, or a minor revision number for trivial updates. The **Date/Time** field can be used in conjunction with the revision fields to track the revision details of projects.

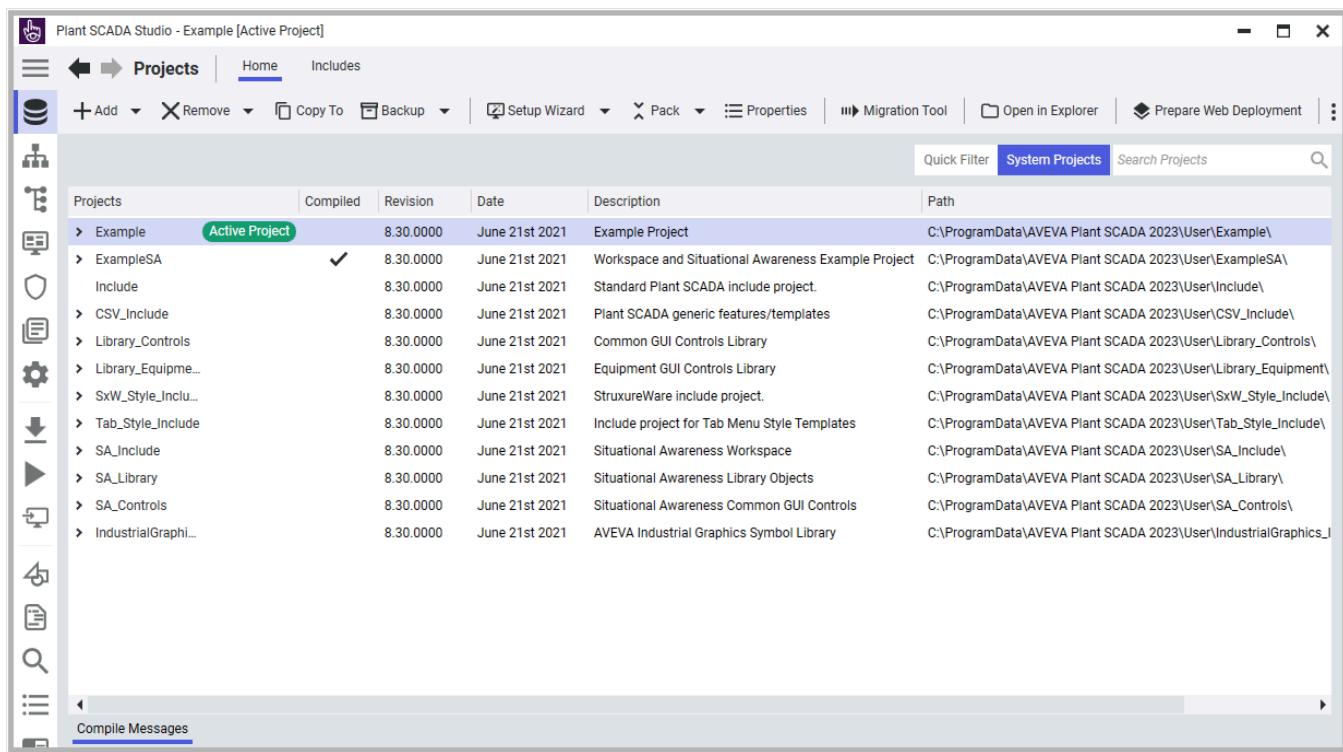


When you create a deployment version of your project, you can see the individual project revisions from any included projects that make up that deployment version.

## The Projects Activity

Projects can be created and managed in Plant SCADA Studio via the **Project** activity.

To access the **Project** activity, select **Project** on the Activity Bar.



**Note:** With the release of Plant SCADA 2023, unique IDs are applied to every Plant SCADA project. This provides consistent identification across the lifetime of a project. When you [Copy a Project](#) or [Restore a Project](#) a project, you may need to consider how you want the project IDs to be implemented.

## See Also

[Create a Project Using a Starter Project](#)

[Create a Standard Project](#)

[Make Active](#)

[Back Up a Project](#)

[Restore a Project](#)

[Edit a Project's Properties](#)

## Project Types

Plant SCADA projects can be one of the following types:

### User Projects

User projects are projects that you create to store customized content for a particular runtime system. They can include graphics components, tags and alarms, equipment definitions, Cicode files, and so on.

User projects are typically created using a template that provides the basic components required to compile and run the project. This allows you to create content that you can immediately test in a runtime environment.

To create a new project based on a template, you can use a Starter Project (see [Create a Project Using a Starter Project](#)).

You can also create a user project by taking a copy of an existing project (see [Copy a Project](#)).

---

**Note:** Due to the nature of their content, it is recommended that you regularly archive your user projects (see [Back Up a Project](#)).

### [Starter Projects](#)

When you create a new project, you have the option to base the project on a "starter project". This generates content that includes the basic components required to compile, run and navigate the project.

A project created using a starter project will also incorporate included projects that provide access to a page templates and library objects that you can use to create your own customized content.

You can choose from the following template styles when you create a project with a starter project:

- **Situational Awareness** — a workspace-based project that is designed to support abnormal situation management for operators.
- **SxW** — designed for use with systems that conform to Schneider Electric's StruxureWare specification.
- **Tab Style** — features navigation that uses a tabbed ribbon menu.

All styles are available in a variety of resolutions to suit the screen size of a client computer.

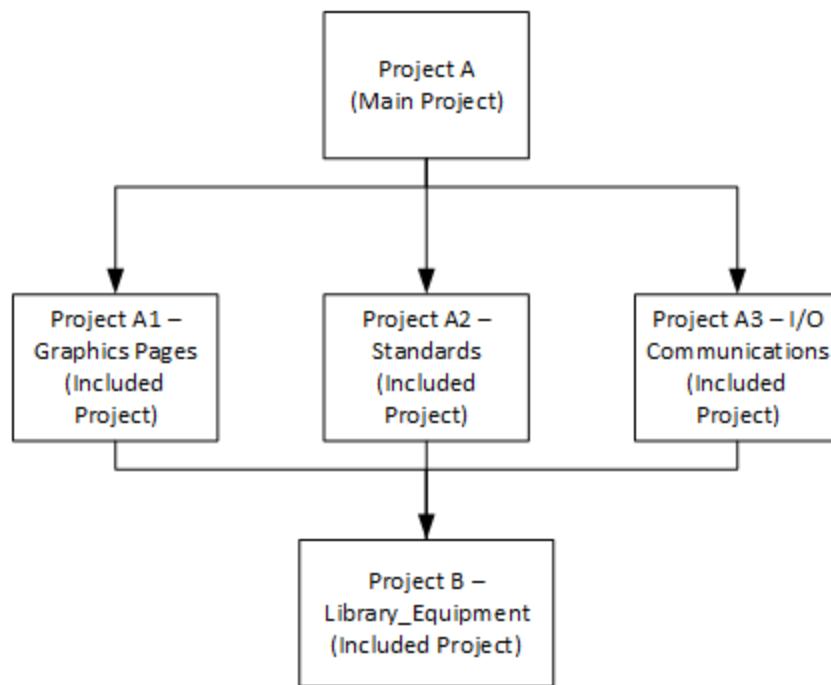
For more information about the contents of a starter project, see the following topics:

- [Situational Awareness Projects](#)
- [StruxureWare \(SxW\) Templates Project](#)
- [Tab Style Templates Project](#).

### [Included Projects](#)

Plant SCADA allows you to "include" a project within a project. This means you can separate configuration or share common configuration with multiple runtime systems.

Included projects are especially useful when you are configuring a large system as they allow you to work with a series of smaller projects that you can combine into a main project.



For example, you could use one project for common configuration for standardization and then a separate project for each section of a plant, or for each production process. This way, you can develop and test each of the smaller projects and system standards before including them in the main project. This also allows multiple users to work simultaneously in the configuration environment with each user working on a separate project.

To add a project within a project, see [Add an Included Project](#)

### System Projects

System projects are installed with Plant SCADA. They are intended for use as included projects or starter projects. They provide content that can assist with the configuration of a runtime system. This can include page templates, symbols, genies, and generic Cicode functions. The following system projects are installed with Plant SCADA.

Name	Type	Description
Include	Include	A template project with standard trending and alarm pages. This project cannot be compiled.
Library_Controls	Include	Contains a set of genies that are common interface controls, such as a tree-view, data table and scroll bars. This project cannot be compiled.
Library_Equipment	Include	Contains the "equip_motors" genie library, and a pop up template library called "equipment". Also contains diagnostic and status information pop up templates.

Name	Type	Description
		This project cannot be compiled.
IndustrialGraphics_Include	Include	Contains a default set of Industrial Graphics which includes charting and dashboard symbols, as well as ISA symbols. See <a href="#">AVEVA™ Industrial Graphics</a> .
SA_Include	Include	A Situational Awareness Include Project which contains public and Genie functions.
SA_Controls	Include	Contains a set of Genies for common user interface controls for Situational Awareness projects.
SA_Library	Include	Contains a set of Genies for common user interface controls for meter, drive and valve objects including output bar, numeric PV and tracker. This includes a set of Composite Genies that provide a template to easily create multiple objects with a consistent look and feel.
SA_Style_1_MultiRes	Starter	A Situational Awareness starter project with two screen resolutions of HD 1080 and UHD 4K.
SxW_Style_1_HD1080_titlebar	Starter	A SxW starter project with a screen resolution of 1080 HD (Full HD). Wide screen only. Includes the SxW_Include and Library_Equipment projects.
SxW_Style_1_HD768_titlebar	Starter	A SxW starter project with a screen resolution of 768 HD (Full HD). Wide screen only. Includes the SxW_Include and Library_Equipment projects.
SxW_Style_Include	Include	Includes a set of SxW templates styled for wide screen only.
Tab_Style_Include	Include	Includes a set of Tab Style templates that use a tabbed ribbon menu.

Name	Type	Description
		This project cannot be compiled.
Tab_Style_1_HD1080_titlebar	Starter	A Tab Style starter project with a screen resolution of 1080 HD (Full HD). Includes the Tab_Style_Include project.
Tab_Style_1_SXGA_titlebar	Starter	A Tab Style starter project with screen resolution for SXGA. Includes the Tab_Style_Include project.
Tab_Style_1_WUXGA_titlebar	Starter	A Tab Style starter project with screen resolution for WUXGA. Includes the Tab_Style_Include project.
Tab_Style_1_XGA_titlebar	Starter	A Tab Style starter project with screen resolution for XGA. Includes the Tab_Style_Include project.

You can view system include projects in the Project activity by clicking **System Projects**. Starter projects will not appear in the Project activity.

---

**Note:** You should avoid making configuration changes to a system project, as each time Plant SCADA is upgraded it installs new versions of the system projects. This means any changes you make to a system project will be lost.

See [System Projects](#).

### Example Projects

Example projects are installed with Plant SCADA to demonstrate some of the functionality that can be supported in an operational runtime system.

Two example projects are installed:

- **ExampleSA** – demonstrates a workspace-based [Situational Awareness Project](#).
- **Example** – demonstrates a project based on the [SxW Templates Project](#).

### The ExampleSA Project

A Situational Awareness project is designed to support abnormal situation management for operators. It provides accurate information to an operator in a way that can be perceived and acted upon quickly, without overwhelming their cognitive abilities. A consistent look and feel is applied to all graphical content, and standardized colors settings are used to emphasize important process data and alarms.

A Situational Awareness project differs to other Plant SCADA projects as it uses a master page on each client screen that is comprised of a set of "panes". Each pane displays a page that can be updated independently at runtime based on navigational or contextual changes. It creates a workspace that supports contextual updates that are managed via a project's equipment hierarchy, which creates an association between the location of a

detected change, and the type of content each pane is configured to display.

The ExampleSA project demonstrates the default layout applied to projects created using a Situational Awareness Starter project. This layout includes common elements such as a header bar, content area, and a dashboard that displays contextual information and navigational tools.

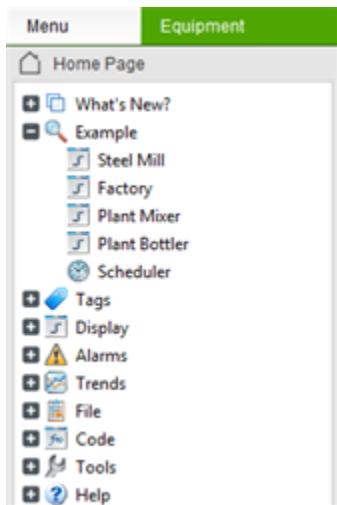
The project pages are built using objects that are available in an included library project (named "SA\_Library"). The included objects (and associated faceplates) support the autofill functionality of the workspace, and maintain a consistent look and feel across a project's display pages. See [Situational Awareness Library Project](#).

### The Example Project

This project comprises a set of pages based on the SxW templates (accessible via a starter project). These pages display tag and alarm values for a set of I/O devices that operate in memory. An equipment hierarchy is also included.

The graphical content included in the Example project pages demonstrates how Plant SCADA can be used to represent complex production processes such as a steel mill, a plant mixer, and a bottling process.

You can select and run the Example project from the **Project** activity in Plant SCADA Studio. To run this project, you need to make this the Active Project using the **Make Active** button. When runtime launches, you can access the project's pages via the **Menu** panel to the right.



The project is supported by its own help file that describes each of the included pages. This help file is available from the **Menu** panel.

---

**Note:** You should avoid making configuration changes to an Example project, as each time Plant SCADA is upgraded it installs a new version of the project. This means any changes you make to the Example project will be lost.

---

## See Also

[Create a Project Using a Starter Project](#)

[Create a Standard Project](#)

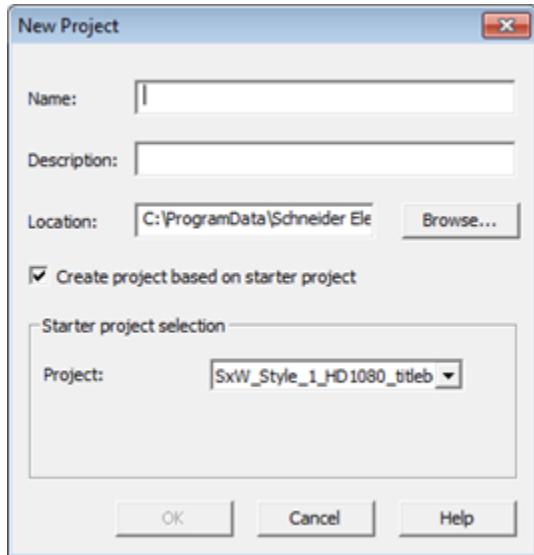
## Create a Project Using a Starter Project

A [Project Types](#) allows you to create a new project that includes the basic components required to compile, run and navigate the project.

**To create a project using a starter project:**

1. In the **Project** activity, select **Home**.
2. On the Command Bar, click **Add | New Project**.

The New Project dialog appears.



3. Complete the required fields (see below for a description of the fields).
4. Click **OK**.

The new project will appear in the **Project** activity.

## New Project Properties

Field	Description
<b>Name</b>	A unique name for the project. The project name is restricted to 64 characters. It can contain any characters other than characters in the Windows file naming rules "* \{}:<>?/'; Since the project name is a unique identifier, Plant SCADA does not permit you to create or restore a project with a duplicate name.
<b>Description</b>	A description of the project. This field is useful for giving an explanation of the role of the project. It is recommended that you complete this field.
<b>Location</b>	The directory path where the project files are stored. As the <b>Name</b> field is entered, the directory is automatically generated in the <b>Location</b> field. You can override this by manually

Field	Description
	entering the location or clicking <b>Browse</b> .
<b>Create project based on starter project</b>	<p>Select this option if you want to create a project based on the built-in starter projects. Choose the project style from the <b>Project</b> drop down list that is displayed when this option is selected.</p> <p>You can create custom starter projects by placing *.ctz backup files in the &lt;User&gt;/&lt;Data&gt;/Starter folder (where &lt;User&gt;/&lt;Data&gt; is the directory you chose during installation). The <a href="#">[CtEdit]Starter</a> parameter can be used to change this default path.</p>
<b>[Starter project selection] Project</b>	<p>Select the starter project you wish to use as a template for your project. You can change the style of existing pages and templates using the Page Properties, accessed through the Graphics Builder.</p>

## See Also

[Create a Standard Project](#)

[Make Active](#)

## Create a Standard Project

Use this procedure to create a new project that does not include the content generated by a starter project.

---

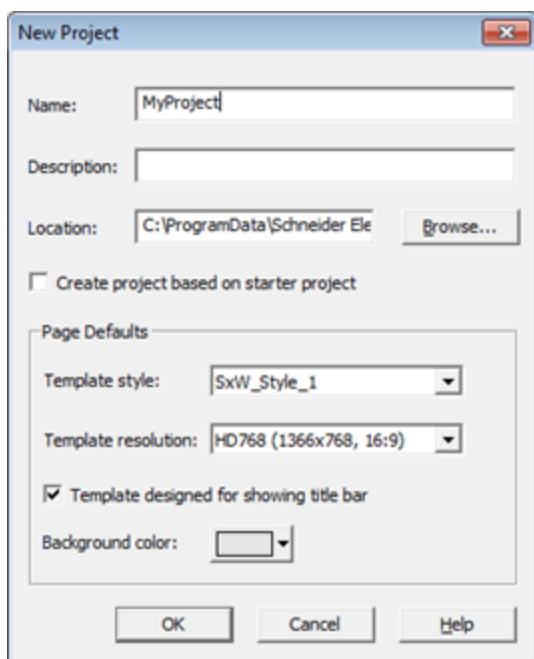
**Note:** If you use the steps described below to create a new project, you will not be able to compile the project until you have defined at least one cluster, one user and an I/O device.

---

### To create a standard project:

1. In the **Project** activity, select **Home**.
2. On the Command Bar, select **Add | New Project**.

The New Project dialog appears.



3. Complete the required fields (see below for a description of the fields).
4. Click **OK**.

The new project will appear in the **Project** activity.

### New Project Dialog Box

Field	Description
<b>Name</b>	A unique name for the project. The project name is restricted to 64 characters. It can contain any characters other than characters in the Windows file naming rules "*   \{}:<>?/;' Since the project name is a unique identifier, Plant SCADA does not permit you to create or restore a project with a duplicate name.
<b>Description</b>	A description of the project. This field is useful for giving an explanation of the role of the project. It is recommended that you complete this field.
<b>Location</b>	The directory path where the project files are stored. As the <b>Name</b> field is entered, the directory is automatically generated in the <b>Location</b> field. You can override this by manually entering the location or clicking <b>Browse</b> .
<b>Create project based on starter project</b>	To create a standard project, clear this check box.
<b>[Page defaults] Template style</b>	The style (appearance) of the graphics pages in the runtime system. The style you select is the default

Field	Description
	<p>style for any new pages you add to the project. You can change the style of existing pages and templates using the Page Properties, accessed through the Graphics Builder. You can choose from:</p> <ul style="list-style-type: none"> <li>• Top</li> <li>• Standard</li> <li>• Bottom</li> <li>• Tab_Style_1</li> <li>• Situational Awareness</li> <li>• SxW_Style_1</li> </ul>
<b>[Page defaults] Template resolution</b>	<p>The default screen resolution of the standard graphics pages (such as alarms pages and standard trend pages):</p> <ul style="list-style-type: none"> <li>• VGA (640 x 480, 4:3)</li> <li>• SVGA ( 800 x 600, 4:3)</li> <li>• XGA (1024 x 768, 4:3)</li> <li>• SXGA (1280 x 1024, 5:4)</li> <li>• HD768 (1366 x 768, 16:9)</li> <li>• HD1080 (1920 x 1080, 16:9)</li> <li>• WUXGA (1920 x 1200, 16:10)</li> <li>• 4K (3840 x 2160, 16:9)</li> <li>• User-defined</li> </ul>
<b>[Page defaults] Template designed for showing title bar</b>	<p>Determines whether to display the Windows title bar (at the top of each graphics page). To display a page in fullscreen (without a title bar), the size of the page needs to be the same size as the display (or larger). If the page is smaller than the display, the title bar still displays, even if fullscreen mode is enabled. Standard templates styles are available for both page sizes.</p>
<b>[Page defaults] Background color</b>	<p>The background color that will be displayed in newly created graphics pages.</p>

## See Also

[Create a Project Using a Starter Project](#)

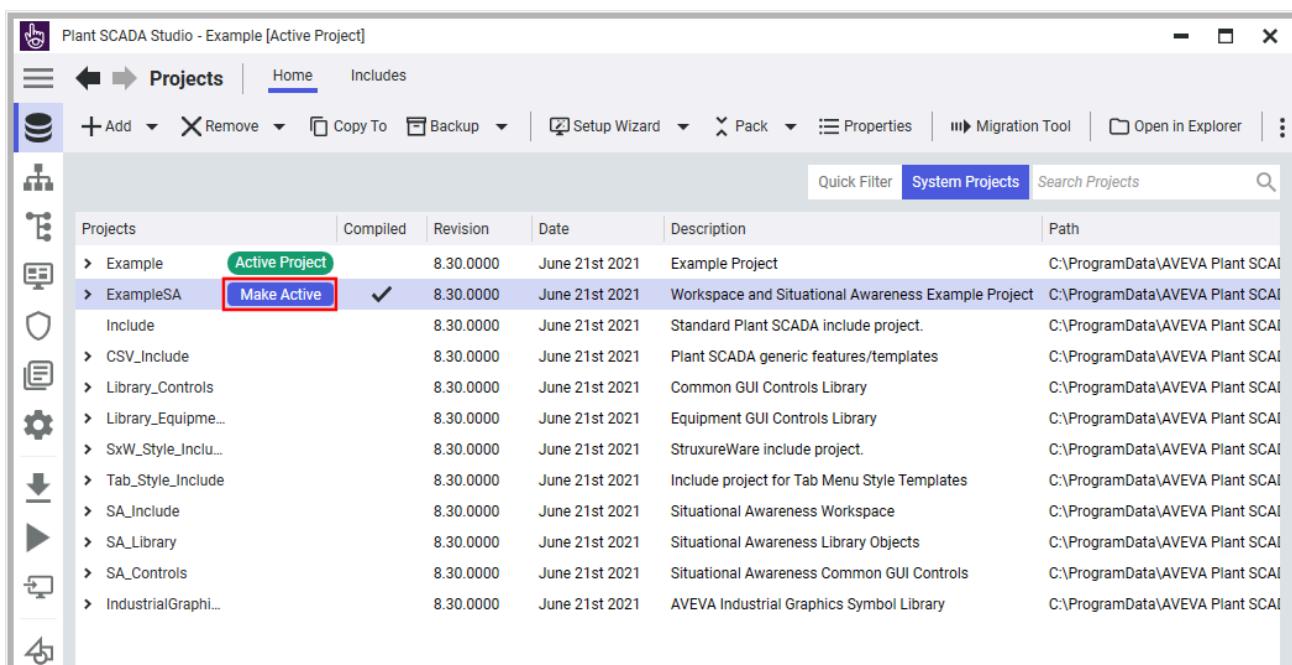
## Make Active

The Make Active functionality allows you to set the project you want to compile and run as "Active". Options on the Activity Bar, the Setup Wizard and the grid views (except the Includes view) work on the Active Project and its included projects rather than on the project selected in Projects view. You do not have to make included projects Active in order to view and edit their data; you can use the Project filter in the Grid Editor for individual projects. This eliminates the need to switch between individual projects to view the corresponding data. If there is only one project in your system, that will be the Active Project by default.

Most Command Bar operations such as **Backup** and **Copy To** in the **Project** view run on the selected project while others run on the Active Project. Tooltips on the Command Bar buttons indicate whether the associated operation will be run on the selected or the Active project.

### To make a project the Active Project:

1. In the **Project** activity, select **Home**.
2. Click to select the project you want to make active.
3. Click the **Make Active** button next to the project name. The Active Project indicator is displayed next to the project name and in the window title.



## Add an Included Project

When you "include" a project within another project, you can share content across both projects while maintaining each as a separate entity (see [Project Types](#)).

### To add an included project:

1. In the **Project** activity, select **Home**.
2. Click to select the project to which you would like to add an included project.

3. Select the **Includes** Tab.
4. Add a row to the Grid Editor.
5. Type the required information in each column, or in the Property Grid (see below for a description of the properties).
6. Click **Save**.

You can include up to 1024 projects. If the number of included projects is less than this and you receive an "out of file handles" compile error, you may need to adjust the parameter [\[CtEdit\]DbFiles](#).

## Included Project Properties

### Include Properties

Property	Description
<b>Included Project</b>	The name of the project to include in the selected project.  You can select the project you would like to include from a drop-down menu within the <b>Include Project</b> field.
<b>Comment</b>	Any useful comment.

### Project Properties

Property	Description
<b>Project</b>	The project in which the specified project is included.

## See Also

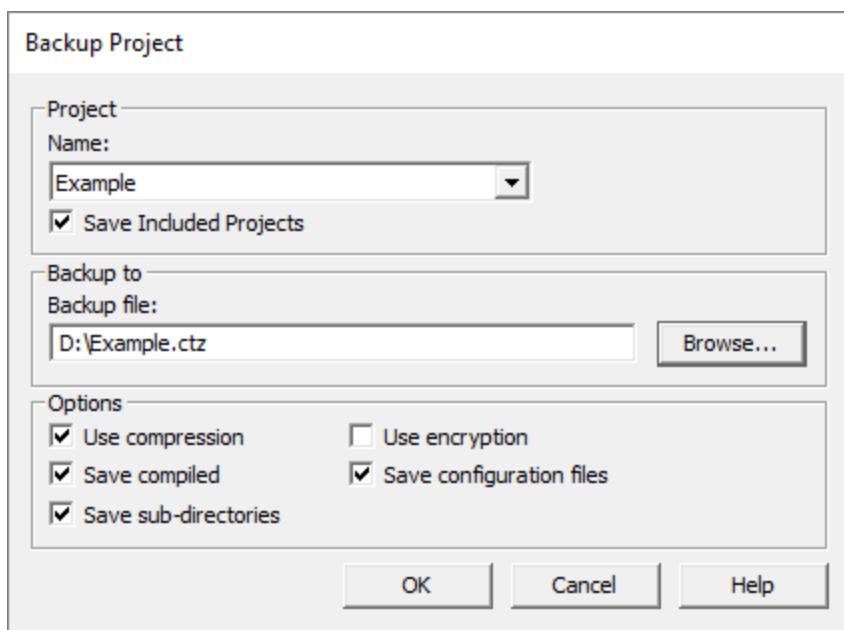
[Link to an Existing Project](#)

## Back Up a Project

Use the Plant SCADA backup functionality to archive project files using the industry standard ZIP compression algorithm. The default file extension for Plant SCADA backup files is .ctz, though any extension (including .zip) can be used. You can use any utility that supports .zip files to extract project files from a compressed Plant SCADA backup.

### To back up a project:

1. In the **Project** activity, select **Home**.
2. On the Command Bar, click **Backup**.  
The Backup Project dialog appears.



3. Complete the required fields (see below for a description of the fields).
4. Click **OK**.

### Backup Project Dialog Box

Field	Description
<b>Name</b>	Select the name of the project to back up.
<b>Save Included Projects</b>	Select to add any included projects to the backup file. The included projects are saved to a sub-directory named "_IncludeProjects".  <b>Note:</b> If you select the <b>Save Included Projects</b> option, any system projects that are installed with the product (such as "Library_Controls" and "SxW_Style_Include") will not be added to the backup file.
<b>Backup file</b>	Enter the path to the backup file location, including the file name. You can either type the path in directly or use the <b>Browse</b> button.  The backup file name defaults to <project>.ctz. If the extension is omitted then .ctz is used.
<b>Use compression</b>	Select to use data compression when you are backing up a project to save space.
<b>Save compiled</b>	By default, Plant SCADA backs up the project in

Field	Description
	<b>uncompiled</b> mode. If you select this option, Plant SCADA backs up both the <b>compiled</b> and <b>uncompiled</b> projects files, resulting in a larger backup file.
<b>Save sub-directories</b>	<p>If you select this option, Plant SCADA also backs up data in any sub-directories within the project directory. The directory structure is maintained in the backup, and you can choose to restore the sub-directories when restoring the project.</p> <p>For example, if you wish to back up your Process Analyst views, save them in a sub-directory of the project and select this option. When you restore the project, you will have the option to also restore the Process Analyst Views directory.</p>
<b>Use encryption</b>	<p>As an added security measure, you can back up your project in an encrypted format. If you select this option, a dialog box that requests a password is displayed when you click <b>OK</b> on the Backup Project dialog. It writes the project to disk in a format that encodes the password along with the protected project. The project can only be restored if the password is entered.</p>
<b>Save configuration files</b>	<p>Select this option to back up *.ini files from the Config folder. This will also backup the TimeSyncConfig.xml file used to store the time synchronization settings configured in the Time Synchronization utility.</p> <p>If you are using a custom INI file (for example 'abc.ini') and it is placed in the Config folder, it will also be backed up. If you are using a custom INI file that is stored in a sub-directory of the project, select the <b>Save sub-directories</b> option to back it up as well.</p> <p><b>Note:</b> You can define a non-default INI file for Plant SCADA by passing a parameter through to Plant SCADA Studio from the Properties dialog box that is accessible from the Shortcut tab. See <a href="#">Using an Alternative INI File</a> for further information on how to do this.</p>

---

**Note:** With the release of Plant SCADA 2023 R2, the **CSV\_Include** project is no longer supported as a system project. If you have any projects based on the CSV\_Include templates, you will need to create a separate backup of the CSV\_Include project and restore manually it in version 2023 R2 (or later). You can then include it as a user project.

---

## See Also

[Restore a Project](#)

## Run a Backup from the Command Line

You can execute the Plant SCADA backup program from the command line to back up and restore files other than Plant SCADA projects.

The backup program is called CtBack32.exe. By default, it is installed in the Plant SCADA project 'Bin' folder. The backup program archives files using a standard compression routine, producing Zip compatible files. The default extension for Plant SCADA backup files is .ctz, though any extension (including .zip) can be used. This means you can also use any Zip compatible program to extract files from a compressed Plant SCADA backup if you prefer.

---

**Note:** If you run the backup program from the command line, and you specify an INI file as a parameter, the specified INI file will be backed up instead of Citect.ini.

---

The backup program reads the citect.ini file for any parameters set using the [BACKUP] category. These settings (if any, and their defaults if not) are over-ridden by any values passed as command line options.

The table below describes the backup command line options.

Option	Description
-d<name>	Database name
-m<ext>	Include extension
-x<ext>	Exclude extension
-e (follow with a number e.g. -e1, -e0)	Encrypt with password
-p<password>	Encrypt/decrypt password
-s[+/-]	Recurse subdirectories
-u[+/-]	Save uncompiled, use -u- to save compiled
-g[+/-]	Show configure dialog
-c[+/-]	Compress files
-b<path>	Path to backup from
-r<path>	Path to restore to
-i<filename>	ini file name

Option	Description
-a	Run in auto mode <b>(Note:</b> Every necessary input needs to be in command line or INI file.)
-o	Overwrite target ctz

**Note:** If -d is not specified and the application is not running in 'auto' mode (-a), the project will be restored to the path of the project from the [CtEdit]LastDatabase ini parameter. If the ini parameter is not present then the project will be restored to the location the -r option specifies. If you restore a project with the -d option on a configuration workstation the application will look in the master.dbf for this project and restore to the path specified in the master.dbf, therefore the project must exist. If you restore a project with the -d option on a runtime only workstation the application overwrites the -d parameter with the [CtEdit]Run parameter from the ini file and restores to this path.

## Examples

- To back up "c:\data" use the following command:  
CTBACK32 -g- -bc:\data
- To restore the above data use:  
CTBACK32 -g -rc:\data
- To backup a Plant SCADA database, for example, to backup "demo" use:  
CTBACK32 -dDEMO -b -u- -c+ -d-

Ctback32 also uses the following parameters in the citect.ini file:

```
[BACKUP]
Database= ! database to backup or restore
BackupPath= ! file to backup to, for example c:\temp\example.ctz.
FilePath= ! file path, used in not a database
Password= ! encryption password
Encrypt=0/1 ! encrypt backup
Configure=0/1 ! display configure dialog
Compress=0/1 ! compress backup
Overwrite=0/1 ! overwrite
SaveCompiled=0/1 ! save compiled
Recurse=0/1 ! recurse sub directories
DeleteAll=0/1 ! delete all before restore
SaveInclude=0/1 ! add included projects to backup file
IncludedDBOn=0/1 ! display a notification after backup if include projects found
SaveIniFiles=0/1! determines whether save ini files is checked
Operation=0/1 ! 0=backup, 1=restore
RestoreInclude=0/1 ! restore included projects contained in backup file
Include= ! include list
Exclude= ! exclude list, default DBK,_CI
CompiledFiles= ! compiled files, default RDB
```

## Restore a Project

You can restore a project that has been backed up using the Restore Project functionality. This functionality allows you to:

- Overwrite a current project with a backed up version
- Restore a backed up project as a new project.

When restoring a project, you need to consider the impact it will have on the unique ID used to identify the project.

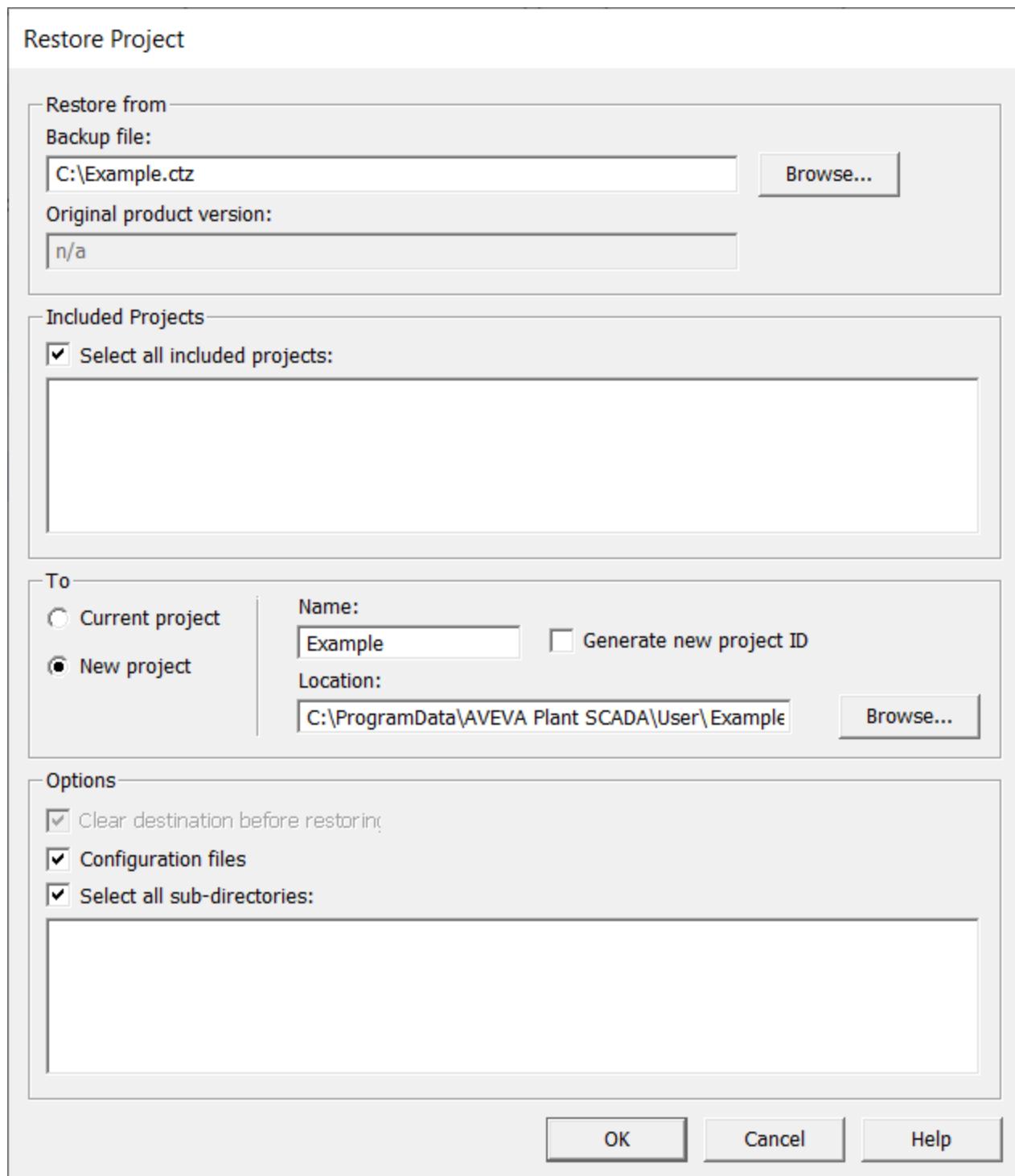
You can use the **Generate new project ID** option to create a new unique ID for the restored project. For example, you could select this option if you wanted to use the restored project as the basis for new project development.

If you want to use an archived backup to restore an earlier version of a project, you should not select this option. The unique ID of the backed up project will be preserved.

### To restore a project:

1. In the **Project** activity, select **Home**.
2. On the Command Bar, click **Backup | Restore**.

The Restore Project dialog appears.



3. Complete the required fields (see below for a description of the fields).
4. Click **OK** to initiate the restore process.

## Restore Project Dialog Box

Field	Description
<b>Backup File</b>	Name and location of the backup file for the project you would like to restore. You can use the <b>Browse</b> button to locate the file.
<b>Original product version</b>	Indicates which version of Plant SCADA was used to create the selected backup file.
<b>Select all included projects</b>	<p>Select to restore all of the included projects that were added to the backup file. This will place a check mark next to each of the included projects listed in the panel below. Clear the <b>Select all included projects</b> check box to restore the project without any of its included projects.</p> <p>Or:</p> <p>In the panel displaying the list of included projects, select the check box for each included project you would like to restore.</p> <p>If you restore an included project that already exists, the existing project will be overwritten. If you restore an included project that does not already exist, it will be added as a new project to the same parent directory as the main project.</p> <p><b>Note:</b> If the <b>Select all included projects</b> check box is disabled, it indicates that the backup file does not contain any included projects.</p>
<b>Current project</b>	Select to overwrite the project that is currently selected in the <b>Project</b> activity in Plant SCADA Studio.
<b>Generate new project ID</b>	<p>A unique ID is applied to each Plant SCADA project to provide consistent identification across a project's lifetime.</p> <p>Select this check box to generate a new unique ID for the restored project. This will not create new IDs for any included projects.</p> <p><b>Note:</b> If you use Plant SCADA 2023 to restore a project that was backed up in a previous version, a unique ID will be applied to the project even if the <b>Generate new project ID</b> option is not selected.</p>
<b>New project</b>	Select to restore a backed up project as a new project. If required, enter a new <b>Name</b> for the restored a project.
<b>Location</b>	Specify the destination for the restored project files. The path should end with a directory that reflects the

Field	Description
	project name. You can either type in the path directly, or use the <b>Browse</b> button to select a location.
<b>Clear destination before restoring</b>	<p>Select the check box to delete all existing content before the restore commences.</p> <p><b>Note:</b> Be careful when using the <b>Clear destination before restoring</b> feature, as every file and sub-directory in the destination location will be deleted before a restore commences. If you <i>accidentally</i> set the restore path to the root directory of your computer drive, the entire disk drive will be deleted.</p> <hr/> <p><b>NOTICE</b></p> <p><b>HARD DISK DRIVE ERASURE</b></p> <p>The <b>Clear destination before restoring</b> feature will delete all files and sub-directories in the restore location. Do not set the location path to the root directory of your drive (usually c:\). <b>Failure to follow these instructions can result in equipment damage.</b></p>
<b>Configuration files</b>	Select to restore backed up INI files, and the TimeSyncConfig.xml file used to store the time synchronization settings configured in the Time Synchronization utility. To make these files active, and to update the current Wizard or Time Synchronization settings with the restored settings, copy the restored files to the Config folder, replacing the currently used corresponding configuration files.
<b>Select all sub-directories</b>	<p>If the backup file includes a project's sub-directories, select this check box to restore all of the sub-directories that were added to the backup file. This will place a check mark next to each of the sub-directories listed in the panel below.</p> <p><b>Note:</b> If the <b>Select all sub-directories</b> check box is disabled, it indicates that the backup file does not contain any sub-directories.</p> <p>Or:</p> <p>In the panel displaying the list of included sub-directories, select the check box for each sub-directory you would like to restore.</p> <p>Or:</p> <p>Clear this check box to restore the project without any of its sub-directories.</p>

## See Also

[Back Up a Project](#)

[Repair A Project's Unique IDs](#)

## Link to an Existing Project

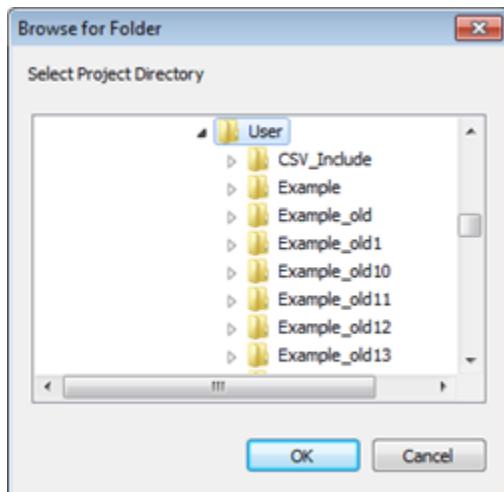
You can link to a project that already exists, even if it is located on a remote network computer. A remote project can be edited and compiled over the network. Only the selected project is linked; its includes are not linked (see [Link to Project Hierarchy](#) for more information).

**Note:** This feature is not designed to allow multiple users access to a common project from different computers. If you create a remote link to a project, you should externally manage the situation to avoid any scenario that would allow two people make changes to a project at the same time. Support for multiple users should be enabled using [Project Types](#).

### To link to an existing project:

1. In the **Project** activity, select **Home**.
2. On the Command Bar, select **Add | Add Project Link**.

The **Browse for Folder** dialog box appears. Select the required project.



3. Select the directory that contains the required project.
4. Click **OK**.

If your selection has the same name as an existing project in Plant SCADA Studio, you will be prompted to provide a new name. You will also be notified if the project has any links to an included project that cannot be located.

### To remove a project link:

1. Select the linked project in the Project activity.
2. Click **Remove | Remove Project Link**. A notification dialog will appear.
3. Select **OK** to proceed.

4. If one or more Industrial Graphics items in this project have unsaved changes, a notification dialog will appear with the list of all unsaved items.
5. Select any of the following option to proceed:

SAVE ALL	Save all the changes, close the Industrial Graphics items, and unlink the selected project from the Plant SCADA Studio.
DISCARD ALL	Discard all the changes, close the Industrial Graphics items, and unlink the selected project from the Plant SCADA Studio.
CANCEL	Cancel the procedure.

**Note:** Unlinking an existing project removes the project, but does not delete it. The project file continues to be available at its original location on the disk.

## Link to Project Hierarchy

You can link Plant SCADA Studio to a project and its includes from any location using the Link to Project Hierarchy functionality. After you have linked a project and its includes to Plant SCADA Studio, you can view these projects in the Projects activity. If one of the include projects is already linked, linking the project hierarchy will not change the existing link.

**Note:** The project to which you are linking and its includes need to be in the same folder.

### To link a project hierarchy:

1. In the **Project** activity, select **Home**.
2. On the Command Bar, select **Add | Link Project Hierarchy**. The Select Folder dialog appears.
3. Click to select the project folder for the project you want to link.
4. Click **Select Folder**. The selected project and its includes are linked.

### To unlink a project hierarchy:

1. Select the project that you want to unlink.
2. On the Command Bar, select **Remove | Unlink Project Hierarchy**. A message is displayed to confirm the action.
3. Click **OK** to proceed.
4. If one or more Industrial Graphics items in this project or its included projects have unsaved changes, a notification dialog will appear with the list of all unsaved items.
5. Select any of the following options to proceed:

SAVE ALL	Save all the changes, close the Industrial Graphics items, and unlink the selected project and its includes from the Plant SCADA Studio.
DISCARD ALL	Discard all the changes, close the Industrial Graphics

	items, and unlink the selected project and its includes from the Plant SCADA Studio.
CANCEL	Cancel the procedure.

**Note:** Unlinking a project hierarchy removes the project and its includes from the view, but does not delete them. The project files continue to be available at their original location on the disk.

## Pack a Project

When you delete project items such as equipment, alarm tags, variable tags and so on they are marked for deletion. Packing a project deletes all items marked for deletion from the various database files and re-indexes these files.

It is recommended to run the pack operation at regular intervals.

### To pack the current project:

1. In the Project activity, select **Home**.
2. On the Command Bar, click **Pack**.

To pack the current project and its included projects, on the Command Bar click **Pack | Pack with Included Projects**.

## See Also

[Repair A Project's Unique IDs](#)

## Repair a Project's Unique IDs

Unique IDs are applied to the following Plant SCADA components:

- Variable tags
- Equipment instances
- Equipment types.

This provides consistent identification across the lifetime of a project.

Each unique ID is a Globally Unique Identifier (GUID) generated by Plant SCADA using the following format:

XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

In some cases, a project may be missing unique IDs for some components. This happens in the following circumstances:

- You have used an external editing tool to make changes to a project DBF.
- You have not run the Migration Tool for a project created in an earlier version.

If you need to generate missing IDs for a project, you can run a repair process.

**To repair a project's unique IDs:**

1. In Plant SCADA Studio, go to the **Projects** activity.
2. On the command bar, open the menu on the **Pack** button.
3. From the menu, select **Repair Unique IDs**.

When the repair process is complete, a dialog will provide a tally of the changes that were made. The dialog will also include the location of a log file that captures more detailed information.

**See Also**

[Unique IDs for Variable Tags and Equipment](#)

## Copy a Project

You can copy the contents of a project to an existing project or a new project.

When you create a copy of a project, you need to consider the impact it will have on the unique ID used to identify the project.

You can use the **Generate new project ID** option to create a new unique ID for the copy you are creating. For example, you could select this option if you wanted to use the copy as the basis for new project development.

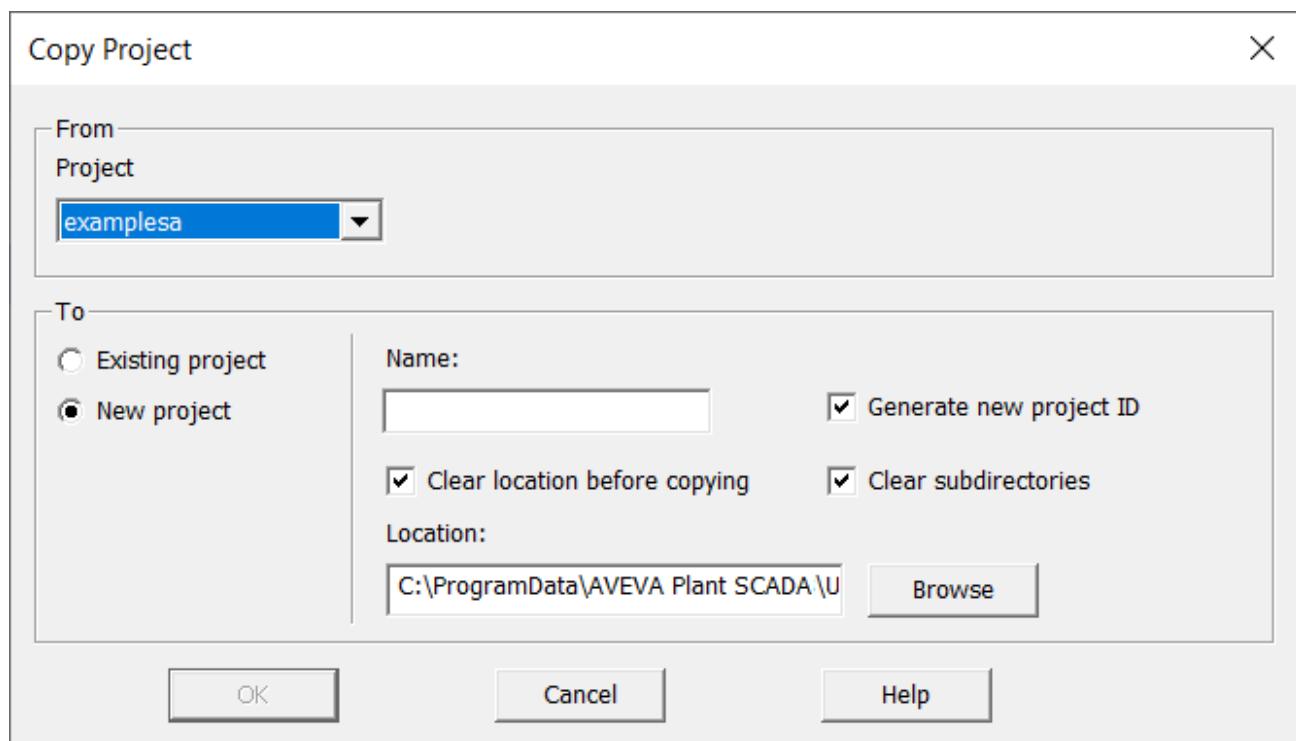
If you want to create an archived copy of a project that you eventually may use to replace the original, you should not select this option.

If you select the **[To] Existing project** option, the unique ID of the source project will replace the ID of the destination project.

**To copy the contents of a project to an existing or a new project:**

1. In the **Project** activity, select **Home**.
2. On the Command Bar, click **Copy To**.

The Copy Project dialog box appears.



3. Complete the following fields (see below for a description of the fields).
4. Click **OK**.

### Copy Project Dialog Box

Field	Description
[From] Project	The name of the project you would like to copy. The drop-down list includes all the projects that are currently available in Plant SCADA Studio. You can select a project name from this list.
[To] Existing Project	Select this option to copy the content of the source project to an existing project.
[To] New Project	Select this option to copy the content of the source project to a new a project.
[To] Name	<p>The name of the project to which the copied content will be added.</p> <p>If <b>Existing project</b> is selected, you can choose a project name from the drop-down list.</p> <p>If <b>New project</b> is selected, enter a unique name for the destination project. The name is restricted to 64 characters, and can contain any characters other than semi-colon (;) or single quote (').</p>

Field	Description
<b>[To]</b> <b>Generate new project ID</b>	<p>A unique ID is applied to each Plant SCADA project to provide consistent identification across a project's lifetime.</p> <p>Select this check box to generate a new unique ID for the copy of your project. This will not create new IDs for any included projects.</p> <p><b>Note:</b> If you are copying to an <b>Existing project</b> and you do not select this option, the unique ID of the copied project will be used to overwrite the ID of the destination project.</p>
<b>[To]</b> <b>Clear location before copying</b>	Select this check box to delete any existing content in the top level folder of the destination project directory.
<b>[To] Clear subdirectories</b>	Select this check box to delete any existing content in the destination folder's subdirectories.
<b>[To] Location</b>	<p>This field is only available if <b>New project</b> is selected. Use it to specify the directory where the copied project files will be stored.</p> <p>Use the <b>Browse</b> button to select a location.</p>

## See Also

[Repair A Project's Unique IDs](#)

## View Projects

The Project activity displays a list of all user and system projects. The name of the Active Project is displayed in the title bar of the application along with the text **[Active Project]**.

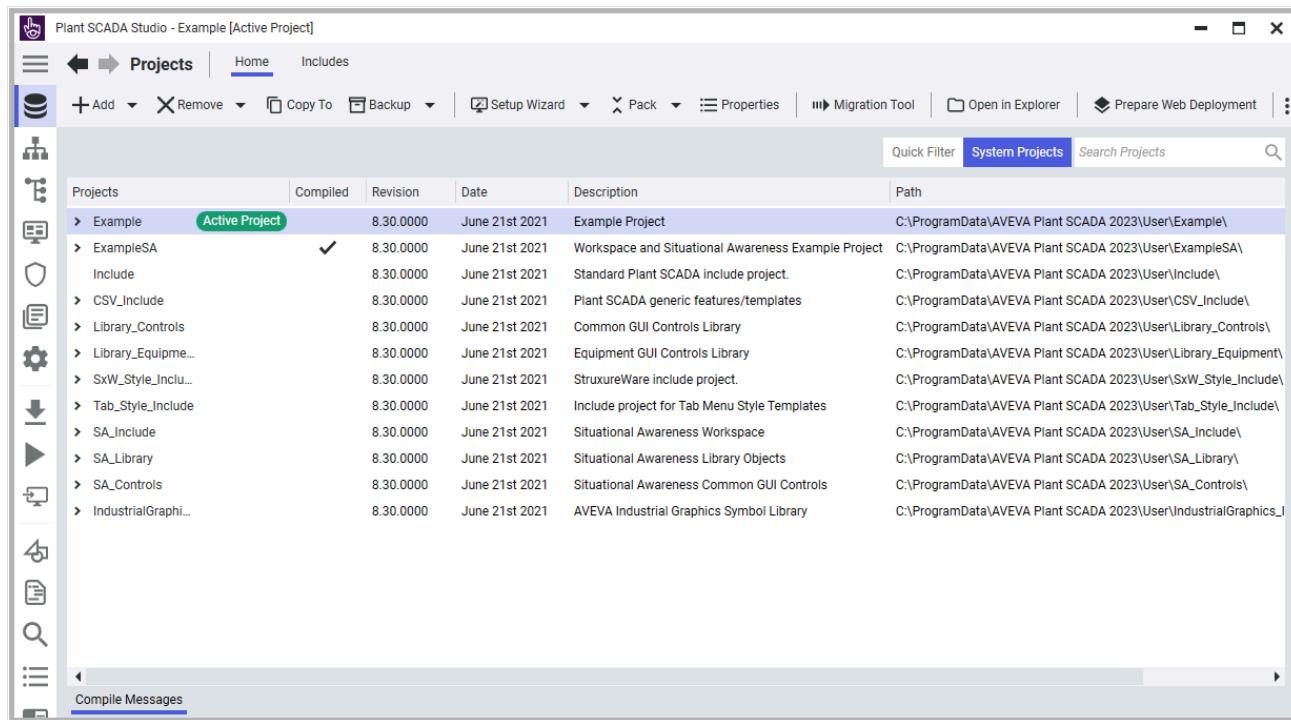
To make a different project the Active Project, click to select the project and then click the **Make Active** button on the Command Bar. Most Command Bar operations in the Project activity are run on the selected project. Only the Setup Wizard and Prepare Web Deployment operations run on the Active Project as do Activity Bar options like **Run** and **Compile**.

Use the expandable icon to the left of a project name to display any included projects.

The Project activity displays the following details for each project:

- **Project** — Name of the project.
- **Compiled** — Whether the project has been compiled. If the project has been compiled, this displays a green tick.
- **Revision** — Revision number of Plant SCADA project that you can manually assign to keep track of changes to your projects.

- **Date** — Date specified in the project Properties dialog box.
- **Description** — description specified in the project Properties dialog box.
- **Path** — Folder in which the project files are stored.



Click **Includes** to view the selected project's included projects.

Row	Project	Include Project	Comment
1	Example	SxW_Style_Include	
2	Example	Library_Equipment	

## Delete a Project

### To delete a project:

1. In the Project activity, select **Home**.
2. Click to select the project you want to delete.
3. On the Command Bar, select **Remove | Delete Project**.  
A confirmation dialog will appear.
4. Click **OK** to confirm that you would like to delete the project.

You cannot delete a [Project Types](#).

**Note:** You should consider creating a backup of a project before you delete it (see [Back Up a Project](#)).

## Filter Projects

By default, all user projects created in Plant SCADA are displayed in the **Project** activity. The name of the [Make Active](#) is displayed title bar and also next to the project name.

To view only the selected project and its included projects, click **Quick Filter**.

To view all [system projects](#), click **System Projects**.

You can also use Search to filter projects. To do so, type the search term in the **Search** box. Project names that contain the search term will be displayed. For example, if you search for "exam", the Example project and projects containing the term "exam" in their name will be displayed.

Projects	Compiled	Revision	Date	Description	Path
Example	8.30.0000	June 21st 2021	Example Project	C:\ProgramData\AVEVA Plant SCADA 2021	
SxW_Style_Include	8.30.0000	June 21st 2021	StruxureWare include project.	C:\ProgramData\AVEVA Plant SCADA 2021	
Library_Equipment	8.30.0000	June 21st 2021	Equipment GUI Controls Library	C:\ProgramData\AVEVA Plant SCADA 2021	

**Note:** Search can be used only by itself; it cannot be used in conjunction with the **Quick Filter** and **System Projects** filter buttons.

## See Also

[Make Active](#)

[Edit a Project's Properties](#)

[View a Project's Folders](#)

## Edit a Project's Properties

The Project Properties dialog box displays information about the current project across two tabs:

- **General**
- **Page Defaults**.

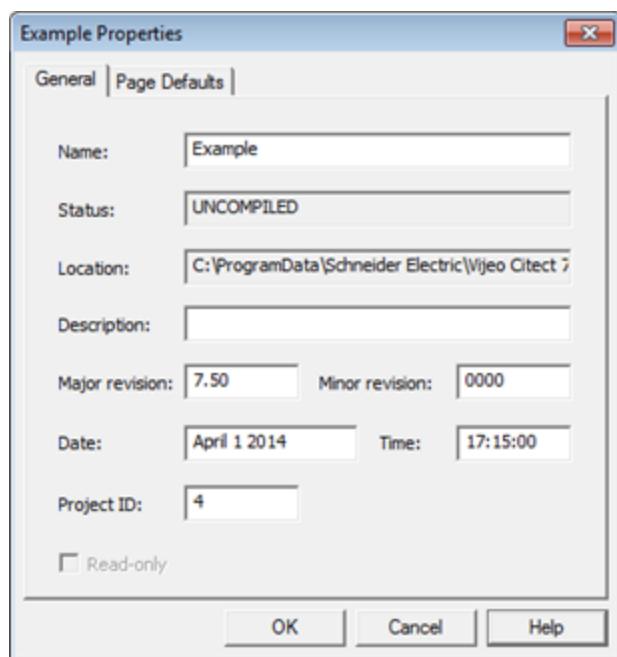
The **General** tab provides information about a project such as its name, location and current status. You can also use this tab to label revisions of a project.

The **Page Defaults** tab includes information about the appearance of the pages in a project, including the default template style and screen resolution. To edit these properties, you need to use Graphics Builder (see [Edit the Default Page Settings](#)).

**To view and edit a project's properties:**

1. In the **Project** activity, select **Home**.
2. On the Command Bar, click **Properties**.

The Project Properties dialog box appears.



3. Edit the General properties as required, or view the Page Defaults (see below for a description).
4. Click **OK**.

## General Properties

Property	Description
<b>Name</b>	The name of the project (restricted to 64 characters). It can contain any characters other than the semi-colon (;) or single quote (''). Since the project name is a unique identifier, Plant SCADA will not permit you to create, restore or rename a project with a duplicated name.
<b>Status</b>	The status of the project. This can be either <b>COMPILED</b> or <b>UNCOMPILED</b> .
<b>Location</b>	The directory path for the location of the project files (see <a href="#">View a Project's Folders</a> ). This field cannot be edited.
<b>Description</b>	A description of the project. This field is useful for providing an explanation of the project's purpose. Maximum length is 255 characters.
<b>Major Revision</b>	Plant SCADA sets this property to one (1) when the project is first created. You can use this field to track major changes to the project. You can use

Property	Description
	an incremental revision history (for example 1, 2, 3, . . . or A, B, C, . . .). Maximum length is 4 characters.
<b>Minor Revision</b>	Plant SCADA sets this property to zero (0) when the project is first created. You can use this field in conjunction with Major Revision to track your project's development. Maximum length is 4 characters.
<b>Date and Time</b>	Plant SCADA will initially set these fields to reflect when the project was created. These fields are useful when used in conjunction with the Revision fields. Maximum length is 20 characters each.
<b>Unique ID</b>	<p>A unique ID is applied to each Plant SCADA project. This provides consistent identification across the lifetime of a project.</p> <p>Each unique IDs is a Globally Unique Identifier (GUID) generated by Plant SCADA using the following format:</p> <p>XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX</p> <p>You cannot edit this field.</p> <p>If you <a href="#">Copy a Project</a> or <a href="#">Restore a Project</a> a project, you need to consider how a project's Unique ID will be impacted.</p>

## Page Defaults

**Note:** To edit these properties, you need to use Graphics Builder (see [Edit the Default Page Settings](#)).

Property	Description
<b>[Template] Style</b>	<p>The name of the project (restricted to 64 characters). It can contain any characters other than the semi-colon (;) or single quote (').</p> <p>Since the project name is a unique identifier, Plant SCADA will not permit you to create, restore or rename a project with a duplicated name.</p>
<b>[Template] Resolution</b>	The default screen resolution of the graphics pages (such as alarms pages and trend pages).

Property	Description
<b>[Template]</b> <b>Designed for showing title bar</b>	Determines whether the Windows™ title bar displays at the top of each graphics page. To display a page in full screen (without a title bar), the size of the page needs to be the same size as the display (or larger). If the page is smaller than the display, the title bar still displays, even if full screen mode is enabled.
<b>Background color</b>	The color that will display in the background of new graphics pages.

## See Also

[View a Project's Folders](#)

## View a Project's Folders

A project's configuration, data and log files are stored in different folders.

The configuration files for a project are stored in two locations:

- **Project folder** - contains the project files (database files, graphics files, and so on)
- **Config folder** - contains the configuration files for the local computer (such as citect.ini).

Project data that is associated with the runtime system is also stored in two locations:

- **Data folder** - contains runtime data
- **Logs folder** - contains runtime log files.

The following table shows the install path for each of these folders in the Windows™ Program Data directory.

Folder	Install Path
Project folder	%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\User\<current project name>
Config folder	%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Config
Data folder	%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Data
Log folder	%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Logs

**To view a project folder:**

1. In the **Project List**, select the required project.
2. On the Command Bar, click **Open in Explorer**.

A new Windows Explorer window will open and display the content of selected project folder.

**To view the Data, Configs or Log folders:**

1. Go to the System menu by clicking on the following icon in Plant SCADA Studio's top left corner.



2. Select the required folder.

A new Windows Explorer window will open and display the contents of selected folder.

**See Also**

[Create a Project Using a Starter Project](#)

[Create a Standard Project](#)

## Analyze a Project

When you use the **Project Analysis** command, a file called "ProjectSummary.xml" is created and placed in the project folder. This file presents information about the selected project and any user-created include projects.

**To create a project summary file:**

1. In the **Project** activity, select **Home**.
2. Click to select the project you would like to analyze.
3. On the Command Bar, select **Project Analysis**.

After the XML file has been created, Windows Explorer will appear with the summary file selected in the project folder.

**Note:** Project Analysis runs in the background and may take a long time depending upon the size of your project.

**See Also**

[View a Project's Folders](#)

## Project Specifications

The table below defines the specifications that are supported by Plant SCADA projects.

**Note:** Values marked with an asterisk (\*) are not restricted, they are maximum recommended values only. However, exceeding these recommendations can impact the performance of your system. For example, it may

affect the trend server (CPU loading) and the CPU loading of a client displaying the Process Analyst.

Configuration Projects	1022 *
Number of Variable Tags defined in Plant SCADA	<p>4,194,303 (but no more than 500,000* is the recommended)</p> <p>This is a system wide limitation for a running system. Tags in every project included in the project you are currently compiling and running will contribute to this figure, regardless of clustering.</p> <p>Be aware that if you are using one or more tag based drivers in your project, the figure depends on the bit length of your variables defined for the driver(s). This is due to the need for these drivers to use OIDs to identify each tag. For example, a tag based driver that uses 32-bit digitals may reduce the number of possible variables by 32 (i.e. if your project contains only 32-bit digitals, you can have a maximum of 131,072). If no bit size is defined for digitals by for the driver, it defaults to 16.</p>
Number of Included projects	1024
Simultaneously logged-in users	250 *
Number of reports	1000 *
Number of I/O Device addresses monitored by alarms	150000 *
Number of historical trends	32000 *
Number of trends displayed on the same chart	16 *
Number of alarms	65,535 per alarm type *
Number of trends displayed on the same page	16000
Number of user functions	4500 *
Number of standard built-in functions provided	1165
Number of operator commands for the system	3000 *
Number of I/O Devices connected to Plant SCADA	16383
Number of simultaneous multiple protocols	4095
Number of areas	255
Number of alarm categories	16376
Maximum simultaneous multi-tasking threads	512

DBF File size	2 GB (2,097,512 KB)
---------------	---------------------

## Run a Project Using the Setup Wizard

Before you run your project you will need to configure each computer in your Plant SCADA system. Configuration information is stored on each machine in a Citect.ini file based on your installation, database configuration and compiled project. Configuration is done with the Setup Wizard.

**Note:** The Setup Wizard runs on the Active Project and its includes.

The Setup Wizard contains a series of pages allowing configuration of the computer specific settings including:

- The role the computer has in the system network
- The project being run
- The CPU Configuration
- The Plant SCADA Events enabled for each component
- The Cicode run for each component on startup
- The cluster configuration
- The security settings applied.

The wizard uses the configuration stored in the project databases to provide information to you. Selected options are written to the Citect.ini file.

The wizard needs to be run on each computer in your system to appropriately configure Plant SCADA for each particular machine. Run after compiling your project and as the last step before running the system.

### See Also

[Run the Setup Wizard](#)

## Run the Setup Wizard

The Setup Wizard runs on the active project and its includes as a default.

### To start the Setup Wizard:

1. In the **Project** activity, select **Home**.
2. If not previously compiled, compile the project.
3. On the Command Bar, click **Setup Wizard**. The Setup Wizard is displayed.
4. Select **Express Setup** or **Custom Setup**.

The pages that are displayed depend on the configuration of the machine. They can include the following:

- [Project Configuration](#)
- [Profile Setup](#)
- [Screen Setup](#)
- [Computer Role Configuration](#)

- Network Model
- Server Password Configuration
- Server User Configuration
- Reports Configuration
- Trends Configuration
- Alarm Server Configuration \*
- CPU Configuration
- Events Configuration
- Startup Functions Configuration
- Cluster Connections Configuration \*
- Control Menu Security Configuration
- Keyboard Security Configuration \*
- Miscellaneous Security Configuration \*
- General Options Setup \*

\* Only available in **Custom Setup** mode.

## Project Configuration

The Project Setup page will vary depending on whether you are configuring a project from Plant SCADA Studio, or from a [Runtime-only Environment](#).

Select the project source to run on this Plant SCADA computer. You will have the following options:

Option	Description
<b>Deployed</b>	This option should only be selected if <a href="#">deployed</a> projects exist. If no deployed project(s) exist and you select this option, the following alert message is displayed: "Cannot use Deployed as the project source, as no projects have been deployed". Projects that have been deployed will be listed in the "Select the project to run" field. Select a project and the version of the project to run. You can use this option to roll back to previous versions of the project.
<b>Local Plant SCADA Studio</b>	Compiled projects available in the Local Plant SCADA Studio will be listed. If there is only one compiled project present, it will be selected automatically. If there are no compiled projects present, an alert message is displayed and the wizard will terminate. If this occurs, return to the Project and confirm that the necessary project is saved locally and has compiled without errors.

Option	Description
Run/Copy	No additional options are available. Refer to <a href="#">Runtime-only Environment</a> for more information.

**Note:** If you change this setting from **Deployed to Local Plant SCADA Studio**, the project that runs on the computer will no longer be synchronized with deployment updates. You need to consider the implications of changing the runtime project on the local computer.

## See Also

[Profile Setup](#)

### Runtime-only Environment

Before configuring the runtime only environment, and running the Setup Wizard, it is assumed you have transferred a project from the configuration environment, including compiled projects to your local machine. If not refer to [Back Up a Project](#) for more information.

Using the .CTZ files from your backup included projects and target project, you can restore the project. It is recommended you restore the included project before the target project. The restore project tool is available from the start menu shortcut **Runtime Configuration | Project restore**

In the Project Restore dialog :

1. In the **Backup file** field, browse or enter the name of the project to restore.
2. Under To, the option **Current project**, may be unavailable. This is due to no project having been run on the local machine before. The option **New project** will be selected to restore a backed up project as a new one.
3. In the **Name** field, enter a name of the restored project.
4. In the **Location** field, enter or browse to the location of the project to restore (restore the project under the 'User' folder).
5. Leave the remaining settings as default and click **OK**.
6. The project will be restored.

You are now ready to run the Setup Wizard available from the start menu shortcut **Runtime Configuration | Plant SCADA Computer Setup**

In the RUN path edit box enter, or use the select button to browse for, the path to the local project that you wish to the client to run. If you enter the run path set the run folder under the 'User' Folder.

If you wish to maintain a synchronized local version of an external project select the Enable RUN/COPY deployment check box. Then in the COPY path edit box enter, or use the select button to browse for, the path to the external project that you wish maintain synchronization. This feature uses the [\[CtEdit\]Run](#) and [\[CtEdit\]Copy](#) commands to maintain synchronization between the two projects.

**Note:** The Run/Copy mechanism does not transfer custom files such as meta-data, icons, images or runtime DBF files. Therefore, it is recommended deploying a project before Run/Copy to help ensure that those files at least exist. You will still need to re-deploy a project, when you want to update the custom files.

## Profile Setup

Select the INI file to be loaded on this Plant SCADA computer at runtime.

- To load the Citect.ini file at runtime, select **Configure Local Settings**.
- To use a [Profiles](#) you have created, select **Use Profiles**.

You can create a <profile>.ini file with the [Profile Wizard](#). Once created, you can use the [Computer Setup Editor](#) to configure parameters specific to that profile. To do this, select one of the profiles configured in your project from the drop-down list in the **Use Profiles** field. The selected profile will be loaded at runtime.

**Note:** If you selected the **Deployed** option on the previous page of the Wizard ([Project Configuration](#)), the **Use Profiles** field will be disabled. The profile that is loaded at runtime will be determined by your deployment configuration.

## See Also

[Screen Setup](#)

## Screen Setup

Use the Screen Setup page to select the Screen Profile that will be used. To do this:

1. From the **Screen Profile** list, select the profile you wish to use for the current workstation. The monitors defined for the selected profile are displayed.
2. For each screen, select a **Startup Page**. A list of available pages can be accessed from a drop-down menu. To just see a list of master pages (pages with their Content Type set to "Master"), select the **Show master pages only** check box.  
You can also use the parameter [\[MultiMonitors\]StartupPage<n>](#) to specify a Startup Page.
3. If your project is a [Situational Awareness](#) project, you will need to select a **Context** for each screen. The context sets the entry point of display for the screen; it could be an area of the plant or the entire site or any graphics page in your project. This can also be done through the [\[MultiMonitors\]Context<n>](#) parameter.

**Note:** Context is only supported for Situational Awareness projects.

## See Also

[Computer Role Configuration](#)

[Run the Setup Wizard](#)

## Computer Role Configuration

Use the Computer Role Setup page to specify the role of the computer running Plant SCADA. Select one of the options described below.

**Note:** To use Plant SCADA's multi-process capabilities with a Server and Control Client, networking needs to be enabled.

Option	Description
Server and Control Client	<p>This computer will be a standalone or networked I/O server and Control Client. This option is disabled if this computer has no Server components assigned to it to run. Selecting this option enables the <b>Multi-Process</b> check box.</p> <p>Select the <b>Multi-Process</b> check box to separate your client and server components into individual processes. This option can be used for distributing the components across multiple CPUs.</p> <p>If the <b>Multi-Process</b> check box is selected the <a href="#">[General]Multiprocess</a> parameter in the Citect.ini file is saved with the value 1. If not selected, the parameter is saved with the value 0.</p> <p>If you leave the <b>Multi-Process</b> check box unselected, Plant SCADA will run the client and server components in one process.</p> <p><b>Note:</b> Running Plant SCADA as a service in single process mode is not recommended. If you run Plant SCADA in this condition, the following limitations will apply:</p> <ul style="list-style-type: none"><li>• Hardware alarms originating from the server process will not appear.</li><li>• The following Cicode functions will not be executed within the server process:<ul style="list-style-type: none"><li>• AlarmSumAppend</li><li>• AlarmSumCommit</li><li>• AlarmSumDelete</li><li>• AlarmSumFind</li><li>• AlarmSumFindExact</li><li>• AlarmSumFirst</li><li>• AlarmSumGet</li><li>• AlarmSumLast</li><li>• AlarmSumNext</li><li>• AlarmSumPrev</li><li>• AlarmSumSet</li><li>• AlarmSumSplit</li><li>• AlarmSumType</li><li>• DriverInfo</li><li>• IODeviceControl</li></ul></li></ul>

Option	Description
	<ul style="list-style-type: none"> <li>• IODeviceInfo</li> <li>• SPCAlarms</li> <li>• ServerInfoEx</li> <li>• TrnAddHistory</li> <li>• TrnDelHistory</li> </ul>
Control Client	<p>This computer will only be a Control Client. This option is disabled if the computer has been assigned a server component to run.</p> <p>Selecting this option enables the <b>Full License</b> check box.</p> <p><b>Note:</b> The Full License checkbox is only available when running the Setup Wizard in Custom Mode.</p> <p>Select the full server license check box if you want this Control Client to use a full (server) license, as opposed to a client license. This sets the <a href="#">[Client]FullLicense</a> parameter in the Citect.ini file to 1 (the default value is 0).</p> <p>Generally, you would not allocate a full server license to a client as there is no difference in functionality between the two licenses. Allocate a full server license if you have insufficient client licenses available, or for some other specific need.</p>
View-only Client	<p>This computer will be a view-only client. This read only option is disabled if this computer has been assigned a Server component to run.</p>
HMI (Standalone, no networking)	<p>This computer will run as a standalone HMI setup with networking disabled. Select this option for any computers that require a dedicated HMI license from an Enterprise License Server that hosts multiple license types.</p> <p>If you are using Sentinel licensing, you can use a key that has networking disabled. If a full Sentinel key is plugged in, runtime will start in standalone mode with networking disabled.</p> <p>Select the <b>Multi-Process</b> check box to separate your client and server components into individual processes. This option can be used for distributing the components across multiple CPUs.</p> <p>If the <b>Multi-Process</b> check box is selected the</p>

Option	Description
	[General]Multiprocess parameter in the Citect.ini file is saved with the value 1. If not selected, the parameter is saved with the value 0.

Some of these options may be disabled depending on what servers have been configured to run on this computer. The Setup Wizard cross-references your computer's network identification with the network addresses configured for each server in your project configuration.

The option you select will adjust the [Client]ComputerRole parameter in the Citect.ini file.

## See Also

[Network Model](#)

[Run the Setup Wizard](#)

### Network Model

Select the network model to be applied to this Plant SCADA computer. The options include:

- Stand alone (no other SCADA computers)
- Networked (connect to other SCADA computers)

**Note:** TCP/IP address information for servers is configured within the Plant SCADA project itself. See [Add a Network Address](#) for more information.

#### Stand alone

Select **stand alone** to run all server and client components of a Plant SCADA system on a single computer. This allows you to run Plant SCADA as a small and self-contained system.

#### Networked

Select **Networked** to connect to a system of multiple Plant SCADA computers configured with different server and client components. In a **Networked** setup where hostnames are used to resolve IP addresses, specify the following:

- Select network address type from the **Address Type** drop-down list (refer to [LAN]AddressType).
- For an IPv6 **Address Type**, select the **Address Scope** from the drop-down list (refer to [LAN]AddressScope).

Refer to the following table for information regarding the status of a server with respect to the computer network configuration, the **Address Type** and the **Address Scope** selection.

Computer Network Configuration	Address Type	Address Scope	Server Status
IPv4	IPv4	-	Starts with IPv4 address.
IPv4	IPv6	Link-Local	Starts with the Link-Local IPv6 address available in the machine.
IPv4	IPv6	Global	Does not start.
IPv4	IPv6	Global and Link-Local	Starts with the Link-Local IPv6 address available in the machine.  <b>Note:</b> When you start <b>Runtime Manager</b> , the <b>User Location</b> column on the <b>Alarm Summary</b> and <b>Sequence of Events</b> pages shows the server address as "::1." It is the IPv6 loop-back address.
IPv6	IPv4	-	Starts with the IPv4 loop back address 127.0.0.1.
IPv6	IPv6	Link-Local	Starts with the Link-Local IPv6 address.
IPv6	IPv6	Global	Starts with Global IPv6 address.
IPv6	IPv6	Global and Link-Local	Starts with Link-Local / Global IPv6 address (depends on first return address).
IPv6 and IPv4	IPv4	-	Starts with IPv4 address.
IPv6 and IPv4	IPv6	Link-Local	Starts with Link-Local IPv6 address.
IPv6 and IPv4	IPv6	Global	Starts with Global IPv6 address.
IPv6 and IPv4	IPv6	Global and Link-Local	Starts with Link-Local / Global IPv6 address (depends on first return address).

When you complete the Profile Setup Wizard, the selected network model is written to the [LAN] section in the

ini file; for example:

```
...
[LAN]
TCPIP=1
...
```

## See Also

[Reports Configuration](#)  
[Run the Setup Wizard](#)

### Reports Configuration

The Reports Configuration page will only be displayed if this machine is configured as a Reports Server in the [Topology](#) activity.

**Note:** For a networked computer to be a Reports Server it needs to also be the I/O Server or needs to be able to communicate with the I/O Server on the network.

Plant SCADA has several options available for report processing:

Option	Description
Startup report	Defines the name of the report to run when Plant SCADA starts up.
Inhibit triggered reports on startup	For example, you might have a report that is triggered off the rising edge of a bit on startup. The Reports Server detects the bit come on, and runs the report. If this option is checked, the Reports Server does not run this report until it has read the I/O Devices a second time.
Run reports concurrently with primary Reports Server	Enables or disables tandem processing of reports. If this server is the standby Reports Server, it can process every report in tandem with the primary server, or it can remain idle until called.

## See Also

[Trends Configuration](#)  
[Run the Setup Wizard](#)

### Trends Configuration

The Trends Configuration page will only be displayed if this machine is configured as a Trends Server in the [Topology](#) activity.

**Note:** For a networked computer to be a Trends Server it needs to also be the I/O Server or needs to be able to

---

communicate with the I/O Server on the network.

Plant SCADA has one option available for trend processing:

Option	Description
Inhibit triggered trends on startup	You might have a trend that is triggered off the rising edge of a bit on startup. If this option is enabled, the trends server does not display the trend until it has read the I/O Devices a second time.

## See Also

[Alarm Server Configuration](#)

### Alarm Server Configuration

The Alarm Server Properties Setup page is displayed if the computer has one or more alarm servers configured that are part of a redundant pair. Correctly setting the options on this page will improve outcomes for alarm server redundancy. Enter an IP address for **Device IP 1** and **Device IP 2**.

If the Plant SCADA system is using an IPv4 network, use a standard IPv4 address format. For example, 192.1.2.34.

If the Plant SCADA system is using an IPv6 network, use a standard IPv6 address format (or any acceptable abbreviation). For example, 2001:0db8:0000:0000:0000:8a2e:0123:1234 or 2001:db8::8a2e:123:1234 (abbreviated).

**Device IP 1** and **Device IP 2** correspond to the INI parameters [\[Alarm\]IsolationDetectIP1](#) and [\[Alarm\]IsolationDetectIP2](#).

## See Also

[CPU Configuration](#)

[Run the Setup Wizard](#)

### CPU Configuration

The CPU Setup page of the Setup Wizard is used to assign client and server components to specific processors on a multi-processor computer.

The page includes a table that identifies each **Component** (and the cluster to which it belongs), its **Priority** and its **CPU** assignment.

- If the **Multi-process** option was selected on the Computer Role Setup page, you can select specific CPUs for the Client, I/O Server, Alarm Server, Trends Server and Reports Server processes.
- If the **Multi-process** option was not selected, there will only be one entry listed. It will either be “Client” or “Client and Servers”.

**To assign a CPU to a component:**

1. Select one or more components from the list (hold the **Ctrl** key down to select multiple components).
2. Click the **Modify** button.
3. Select the CPUs you want to assign to the component (you can select more than one) or select **<All CPUs>**.
4. Click **OK**.

When you complete the Setup Wizard, the CPU assignations are written to each component section in the INI file; for example:

```
...
[Alarm.Cluster1.AlarmServer1]
CPU=1
Clusters=Cluster1
...
[Trend.Cluster1.TrendServer1]
CPU=2
Clusters=Cluster1
...
```

If ALL is selected, the component sections in the INI file will be left empty and the processes will run on all CPUs.

In situations where ALL is selected, be aware that performance issues may still occur when your CPU usage is less than 100%. You can detect these issues when your CPU usage for a process is fixed at 100 divided by the number of processors.

---

**Note:** If a computer has more than 32 CPUs, the CPU Setup page in Setup Wizard will be disabled. The component sections in the INI file will be left empty and the processes will run on all CPUs.

---

## See Also

[Events Configuration](#)

[Run the Setup Wizard](#)

## Events Configuration

Events are used to trigger actions, such as a command or set of commands. For example, an operator can be notified when a process is complete, or a series of instructions can be executed when a process reaches a certain stage. Select the **Enable Events on this computer** check box if events are to be enabled on this Plant SCADA computer.

The Events Setup page lists each component's full name, including the cluster to which it belongs, alongside a list of events that can be enabled for each component. If the **Multi-process** option was not selected on the Computer Role Configuration page there will only be one entry listed, either Client or Client and Servers. If the **Multi-process** option was selected, you have the option of enabling events for each component on this computer.

---

**Note:**

- The Setup Wizard only displays named events from the selected project. If you are using events in included projects you will need to edit your ini file to add these under the [Events] section header.
- Events named 'Global' or events with no title will not appear as these are global events. These events will run on computers that have events enabled. These events will run in the client process.

**To enable an event for a component:**

1. Select the component from the list.
2. Select the events you want to enable for that component, or click **Enable All** or **Disable All**.
3. Click **Next** when finished.

When you complete the Profile Setup Wizard, the events are written to each component section in the ini file; for example:

```
...
[Alarm.Cluster1.AlarmServer1]
CPU=1
Clusters=Cluster1
Events=CSV_AlarmClient...
[Trend.Cluster1.TrendServer1]
CPU=2
Clusters=Cluster1
Events=CSV_TrendXClient,CSV_TrendXServer...
```

**See Also**

[Startup Functions Configuration](#)

[Run the Setup Wizard](#)

**Startup Functions Configuration**

The Startup Functions Setup page is used to define the Startup Cicode that is executed by each Plant SCADA process.

The Startup Functions Setup page lists each component's full name, including the cluster to which it belongs, the priorities of the components and the startup function assigned to each component. If the **Multi-process** option was not selected on the **Computer Role Configuration** page there will only be one entry listed, either Client or Client and Servers. If the Multi-process option was selected, you have the option of assigning startup functions for each component on this computer.

If the StartupCode parameter value for a process is invalid, the Plant SCADA Runtime Manager will simply ignore it on start up.

**To assign a startup function to a component:**

1. Select the component from the list. To select multiple components, hold down the Ctrl key as you select each item.
2. Click **Modify**.
3. Type the name of the Cicode function you want to call on startup for that component.
4. Click **OK**.

When you complete the Setup Wizard, the events are written to each component section in the ini file; for example:

```
...
[Alarm.Cluster1.AlarmServer1]
CPU=1
```

```
StartupCode=alarmServerStartup
...
[Trend.Cluster1.TrendServer1]
CPU=2
StartupCode=trendServerStartup
...
```

## See Also

[Cluster Connections Configuration](#)

[Run the Setup Wizard](#)

### Cluster Connections Configuration

The Cluster Connections Setup page is used to specify the clusters that each component connects to on startup. This controls what data streams a component can see in the system.

The Cluster Connections Setup page lists each component's full name, including the cluster to which it belongs, the priorities of the components and the clusters assigned to each component. If the **Multi-process** option was not selected on the Computer Role Configuration page there will only be one entry listed, either Client or Client and Servers. If the **Multi-process** option was selected, you have the option of assigning clusters for each component on this computer.

By default, every component will connect to every cluster unless otherwise modified.

If the Clusters parameter value for a process is invalid, then the Plant SCADA Runtime will simply ignore it on startup.

#### To assign a cluster to a component:

1. Select the component from the list. To select multiple components, hold down the Ctrl key as you select each item.
2. Click **Modify**.
3. Select the clusters you want the component to connect to on startup.
4. Click **OK**.

When you complete the Setup Wizard, the clusters are written to each component section in the Citect.ini file; for example:

```
...
[Alarm.Cluster1.AlarmServer1]
CPU=1
Clusters=Sydney
...
[Trend.Cluster1.TrendServer1]
CPU=2
Clusters=Sydney,Tokyo
...
```

## See Also

[Server Password Configuration](#)

[Run the Setup Wizard](#)

## Server Password Configuration

If networking is enabled, use this page to configure the server password. This password is used by computers to authenticate each other and create a trusted network between servers and, optionally, clients.

You can create a new password, or enter the password that is already used by other server computers in your Plant SCADA system.

If server processes do not have matching server passwords, they will not communicate with each other. Only trusted tran channels can handle remote requests initiated by [MsgRPC](#) or [ServerRPC](#) Cicode functions and CTAPI connections.

Configuring the password automatically sets the [\[Client\]PartOfTrustedNetwork](#) INI parameter.

---

**Note:** To configure the server password, you need to be a member of the **Configuration Users** security role. To read the server password you need to be a member of either the **Configuration Users** or the **Server Users** security role. You can be a member of these roles either directly or via an associated domain group (see [Security Roles](#)).

---

- **If a server process exists and networking has been enabled:**

The **Configure Server Password** check box is selected and unavailable.

Enter and confirm the password in the fields provided, before clicking **Next**.

If the **Password** and **Confirm Password** fields already contain an entry, it means a server password has already been configured on the local computer. If required, you can enter a new password.

- **If a client process exists and networking is enabled:**

The **Configure Server Password** check box is unchecked. To use the client computer as a CTAPI server, you need to configure the server password. Setting this password allows servers to authenticate each other and creates a trusted network between server computers.

Select **Configure Server Password** check box to enable the password fields.

Enter and confirm the password in the fields provided, before clicking **Next**.

If you change the server password while the runtime system is active, it will not disrupt any established connections as the password is only retrieved when authentication needs to occur. However, it is recommended that you apply the new password as soon as possible. To do this, you need to restart the runtime system.

If you are extending an existing system by adding computers that will run additional clusters and servers, you should determine what the existing server password is before you proceed. This will allow you to set the same server password on all existing and new servers.

## See Also

[Server User Configuration](#)

[Run the Setup Wizard](#)

## Server User Configuration

Use this page to define the user to log in for the server processes running on the machine. The user can be the

default server user, none (view-only), or a specified user. The Setup wizard will automatically configure the [\[Server\]AutoLoginMode](#) INI parameter in line with the user's settings.

Select **Specific User** to enable the Configure Server User fields. The fields will remain disabled if either the **Default Server User** option or **None** option are selected.

## See Also

[Control Menu Security Configuration](#)

[Run the Setup Wizard](#)

## Control Menu Security Configuration

The Plant SCADA window property options allow you to control an operator's access to system features.

This allows for flexibility with system security at run time.

Option	Description
Plant SCADA configuration environment on menu	Allows the operator to use the control menu (top left-hand icon) to access Plant SCADA Studio, Graphics Builder, and Cicode Editor from Plant SCADA at run time. Disabling this provides better security.
FullScreen	Allows the operator to set whether pages will be displayed in fullscreen or restored state. When checked "FullScreen" will set the ini parameter [Animator]FullScreen to 1. If "FullScreen" is set.
Show title bar	Allows the operator to set whether pages will be displayed in fullscreen mode with the title bar. The <a href="#">[Animator]FullScreen</a> ini parameter is set as follows: 0 if "Fullscreen" is unchecked; 1 if "Fullscreen" is checked and "Show title bar" is unchecked; 2 if both "Fullscreen" and "Show title bar" are checked. "Show title bar" cannot be modified if the option "Fullscreen" is unselected.
Shutdown on menu	Allows the operator to use the control menu (top-left icon) to shut down Plant SCADA at runtime. The shutdown is not password- or privilege-protected. Disabling this provides better security.
Kernel on menu	Allows the operator to use the control menu (top left icon) to display the Plant SCADA Kernel at run time. Disabling this provides better security.

## See Also

[Keyboard Security Configuration](#)

[Run the Setup Wizard](#)

### Keyboard Security Configuration

Windows has a set of standard task-swapping shortcut commands that are (optionally) supported by Plant SCADA at run time. This option allows the Alt-Space Windows command to be enabled or disabled at run time. Alt-Space provides access to the Windows control menu (even if the title bar has been disabled).

**Note:** The ability to disable Alt-Escape, Ctrl-Escape and Alt-Tab is not currently available.

## See Also

[Miscellaneous Security Configuration](#)

[Run the Setup Wizard](#)

### Miscellaneous Security Configuration

Some standard Windows features may interfere with the secure operation of your system. Use the Miscellaneous Security page to disable these features.

Option	Description
Inhibit screen saver while Plant SCADA is running	Stops the screen saver from blanking out screens that need to be visible at all times. Alternatively the screen saver password can add additional security features.
Display Cancel button at startup	Provides the ability to stop Plant SCADA from starting up automatically. Automatic startup is a potential security concern.

## See Also

[General Options Setup](#)

[Run the Setup Wizard](#)

### General Options Setup

Use the General Options Setup page to specify general options.

Option	Description
Data Directory	The directory where the Plant SCADA data files are located. The data files are the files that are generated at run time: trend files, disk PLC etc.
Backup project path	The backup directory that is used if a runtime database cannot be located (due to inoperative hardware or a file that has been moved, corrupted, or deleted).
Startup page	The Page Name of the graphics page to display when Plant SCADA starts up.
Page scan time	<p>The delay (in milliseconds) between updating a graphics page and starting the next communications cycle. The Page Scan Time sets the default for how often your graphics pages are updated. When a page is updated, relevant data (variable tags etc. represented on the graphics page) is scanned to determine if field conditions have changed. This setting is overridden by the Scan Time value specified in Page Properties (if applied).</p> <p>A value of 250 (the default value) indicates that Plant SCADA will try to update the page every 250 ms. However, if Plant SCADA cannot read the entire data from the I/O Device within 250 ms, the page is processed at a slower rate. For example, if it takes 800 ms to read the data from the relevant I/O Device, Plant SCADA processes the page every 800 ms.</p> <p>Under some conditions, you might want to slow the update of your pages to reduce the load on the I/O Servers. By reducing the page scan time, you allow more communication bandwidth to other Plant SCADA tasks or Clients. For example, you might want fast response on your main operator computers, while slowing the response time on manager computers. You can enter any value from 0 to 60000 (milliseconds).</p>
OPC Alarms and Events	<p>Click the <b>Register</b> button to register an alarm server as a runtime OPC AE server that supports the OPC AE interface specification. This will allow the alarm data to be accessed by third-party OPC client applications.</p> <p>The registration process requires administrative privileges.</p> <p><b>Note:</b> The OPC AE Server does not support DCOM for remote communications. It is recommended that you</p>

Option	Description
	run any connecting OPC client applications on the computer that is registered as the OPC AE Server.

## See Also

[Run the Setup Wizard](#)

## Finish

Select one of the following:

- **Finish** saves the setup to the INI file.
- **Cancel** quits the wizard without saving.
- **Back** navigates back to a page that requires adjusting.

## See Also

[Run the Setup Wizard](#)

# Topology

Topology comprises physical computers, clusters, server processes and I/O devices. SCADA functions such as an alarm-related event or an I/O operation run on the computers. In addition, these functions are combined into a logical entity known as a cluster and added as an alarm server or an I/O server process to computers for ease of administration.

Multiple functions from multiple clusters may run on a single machine; however, the resources available on the computer govern the feasibility of running these functions on that computer. Therefore, multiple clusters may be associated with a single computer or a single cluster may need to be configured across multiple computers.

You can configure the computers in your SCADA system and organize the processes in clusters in the Topology activity of Plant SCADA Studio. You can view, add and edit configuration information for computers, clusters, server processes and I/O devices.

The Topology activity can also be used to create and manage [Profiles](#) that can be used when deploying your project.

---

**Note:** If you created a project in a version of Plant SCADA prior to version 2016, you need to run the Migration Tool to view the project's system topology in Plant SCADA Studio.

---

## Configuring a Topology

Configuring a topology involves the following steps:

1. Add [Add a Computer](#) and [Add Network Addresses](#). Each physical computer in your system needs to be identified with a unique name and IP address.

2. Add [Add a Cluster](#) to create groups within your runtime components. Each cluster needs to be defined with a unique name.
3. Add the required **server processes**. Each process server needs to be named, and assigned to a cluster and network address.
  - [Add an Alarm Server Process](#)
  - [Add a Report Server Process](#)
  - [Add a Trend Server Process](#)
  - [Add an I/O Server Process](#)
  - [Configure an OPC DA Server](#)
4. Configure [redundancy](#) if it is required.
5. Add the [I/O Devices](#) in production your system.

## See Also

[Computers](#)

[Clusters](#)

[Server Processes](#)

[I/O Devices](#)

[Profiles](#)

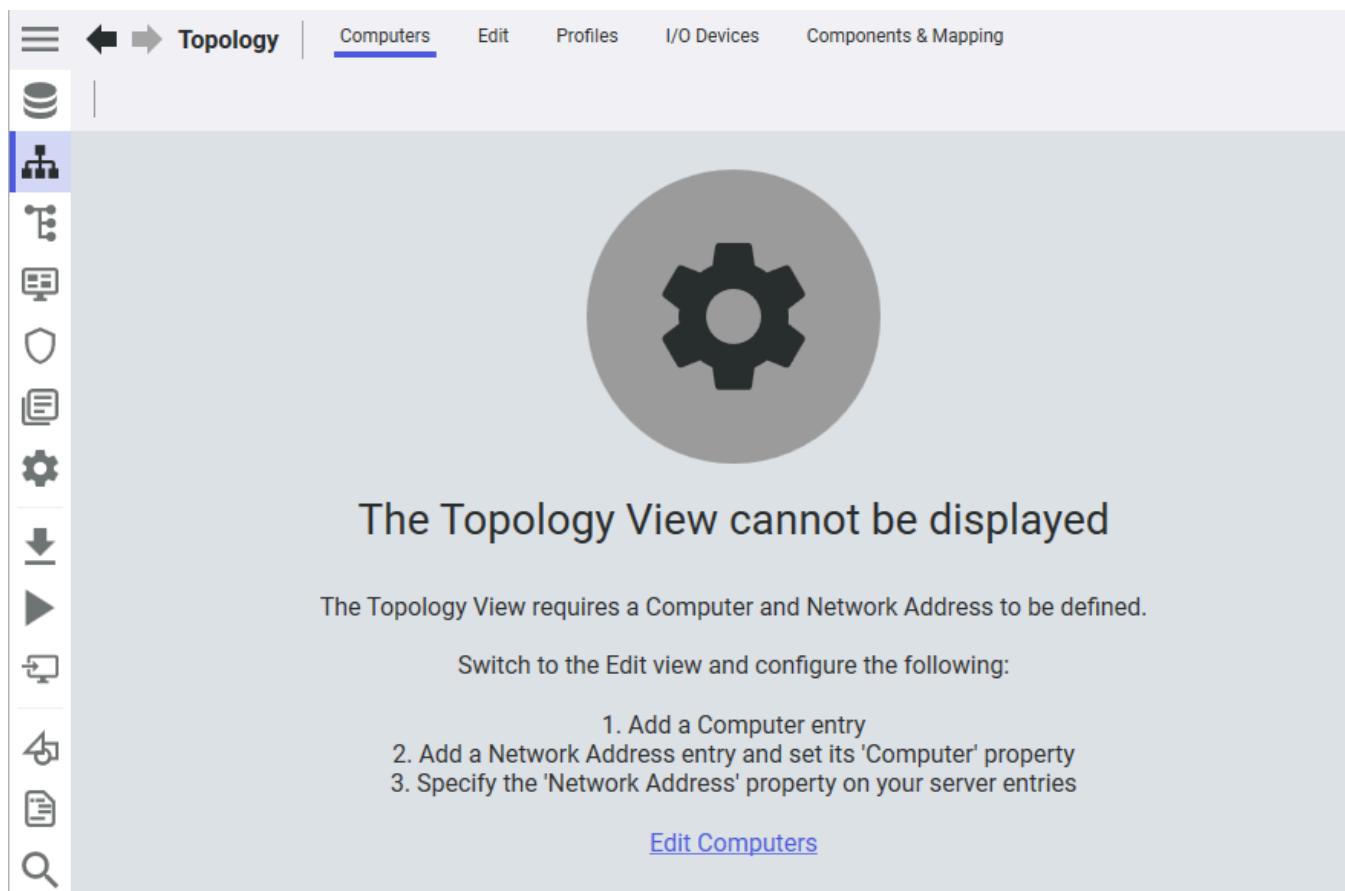
## Computers

The **Computers** view in the **Topology** activity presents the computers included in your Plant SCADA system and the server processes that run on them. Each server process need to be assigned to a computer and then run on that computer at runtime.

In the **Topology** activity you can:

- [Add a Computer](#) and configure computers
- Assign [Add Network Addresses](#) to computers
- Organize computers in [Clusters](#) for efficient management
- [View Computers](#) system topology organized by computers.

When you first launch Plant SCADA, the following information may appear in the **Computers** view.



This indicates that no computers have been configured in the current project. Follow the steps to [Add a Computer](#) (and the associated [Add Network Addresses](#)) to your topology.

When you create a project from a starter project, the computer on which the product is installed is added as the "LocalComputer" and shown in the Topology activity. It's assigned the IP address 127.0.0.1 by default.

## Add a Computer

### To add a computer:

1. In the **Topology** activity, select **Edit**.
2. On the menu below the Command Bar, select **Computers**.
3. Add a new row to the grid.
4. Type the required information in each column, or in the Property Grid (see below for a description of the properties).
5. Click **Save**.

## Computer Properties

### General Properties

Property	Description
Name	The name of the host computer on which SCADA components will be installed. The name needs to be unique.
Comment	This is a brief description of the computer.
DNS Name	<p>The Windows computer name (or localhost if you are running on a standalone computer), which is used for validating security certificates. If this is left blank, the system will use the value specified in the <b>Topology   Network Addresses   Address</b> property. If an IP address is specified for that property, a compiler error will be displayed.</p> <p>The DNS Name can only contain alphabetical characters (A-Z or a-z), numeric characters (0-9), the minus sign (-) or period (.). The name cannot start with a period.</p>

### Project Properties

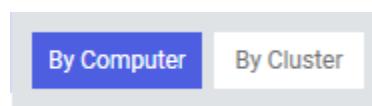
Property	Description
Project	Name of the project with which this computer is associated.

### See Also

[Add Network Addresses](#)

### View Computers

In the **Topology** activity, select **Computers**. Click **By Computer** to view your SCADA system's topology organized by the computers in the system. This view is used by default to display the topology of the selected project.



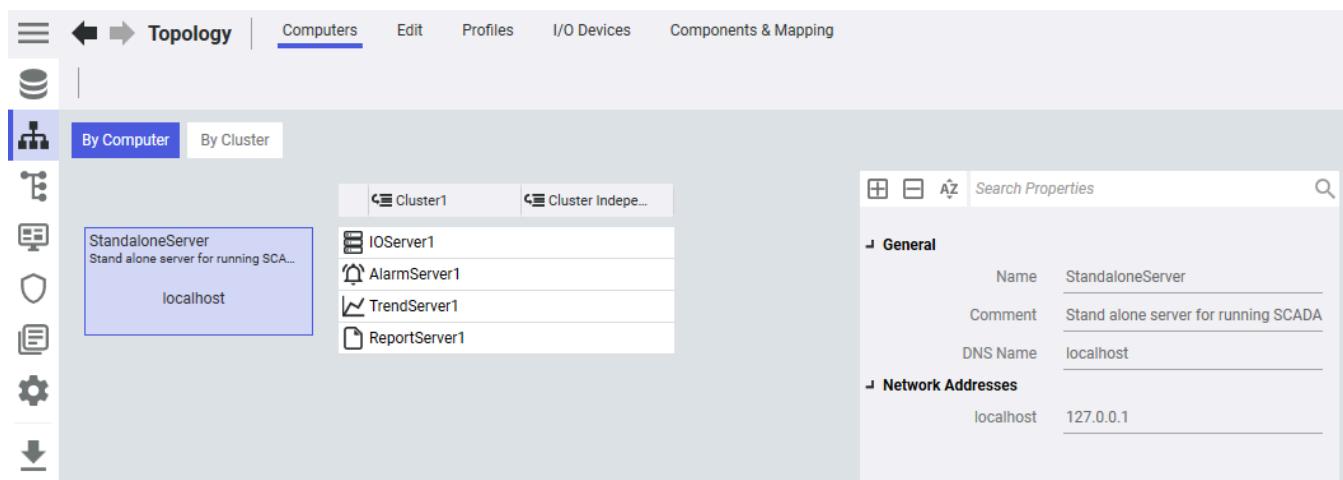
By Computer

By Cluster

---

**Note:** To access information describing the **By Cluster** view, see the topic [View Clusters](#).

In this view, computers configured in the selected project are listed with a description of each machine's clusters and the servers allocated to each cluster. The selected machine card is highlighted as shown below.



Within the machine card, the following information is displayed:

- The computer name
- A comment about the computer (if configured)
- The DNS name.

The Property Grid on the right displays the following properties for the selected computer:

- Name
- Comment
- DNS Name
- Network Addresses (assigned to this computer).

## See Also

[Add a Computer](#)

## Add Network Addresses

Network addresses are required for any computers in your Plant SCADA network that will run a server process.

**To configure a network address:**

1. In the **Topology** activity, select **Edit**.
2. On the menu below the Command Bar, select **Network Addresses**.
3. Add a new row to the grid.
4. Type the required information in each column, or in the Property Grid (see below for a description of the properties).
5. Click **Save**.

## Network Address Properties

### General Properties

Property	Description
<b>Name</b>	The name of the computer at the network address being configured. The name needs to be unique to the project and not contain spaces.
<b>Address</b>	<p>The IP address or name of the computer or network card being configured.</p> <p>In most cases, the IP address will use a standard IPv4 format. For example, 192.1.2.34.</p> <p>"Localhost" or a fully qualified domain name (the complete domain name for a computer) are not supported.</p> <p>If your Plant SCADA system is running on an IPv6 network, you will need to enter a 128 bit IPv6 address (or any acceptable abbreviation). For example, 2001:0db8:0000:0000:0000:8a2e:0123:1234 or 2001:db8::8a2e:123:1234 (abbreviated).</p> <p><b>Note:</b> A link-local IPv6 address should not have a zone index appended to the address. For example, from the IPv6 address "2001:db8::8a2e:123:1234%12" exclude the zone index "%12" and use "2001:db8::8a2e:123:1234" in the <b>Address</b> field.</p> <p>For computers with dual network cards, add a network address for each card in each computer with which you want to communicate. See <a href="#">Network Redundancy</a> for more information.</p> <p><b>Note:</b> If your computer has multiple network adapters configured in Windows that are connected to the same subnet, then we recommend you configure the address field with a computer name rather than an IP address.</p>
<b>Comment</b>	Any useful comment. This property is optional and is not used at runtime.
<b>Computer</b>	The computer with which the network address is associated.

### Project Properties

Property	Description
<b>Project</b>	Name of the project in which the network address is defined.

## Clusters

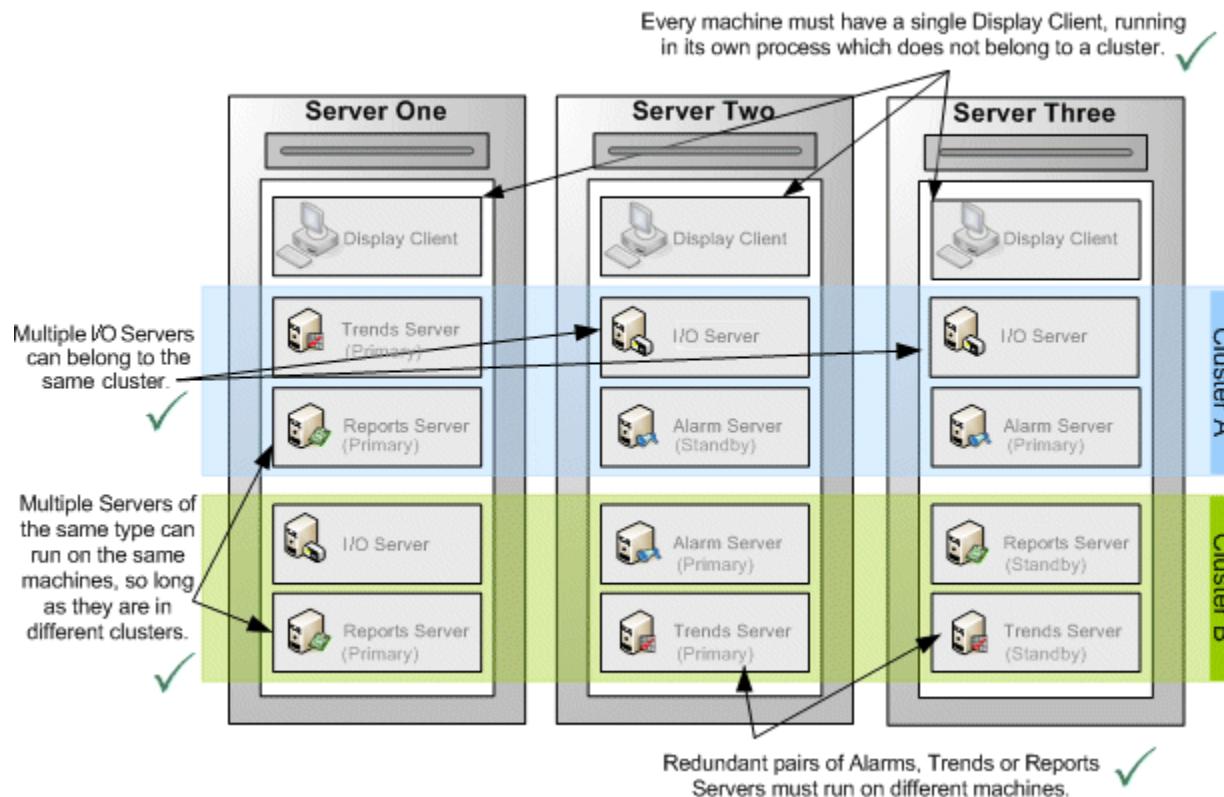
Clustering allows you to group different sets of the runtime components within a single project, allowing multiple independent systems to be monitored and controlled. Many Plant SCADA items require a cluster to be specified in order for them to work correctly.

There are countless variations in how a clustered system can be configured. An appropriate configuration will depend on the requirements for the solution to be deployed and the environment in which it is being deployed. For more information and examples, see [Typical System Scenarios](#).

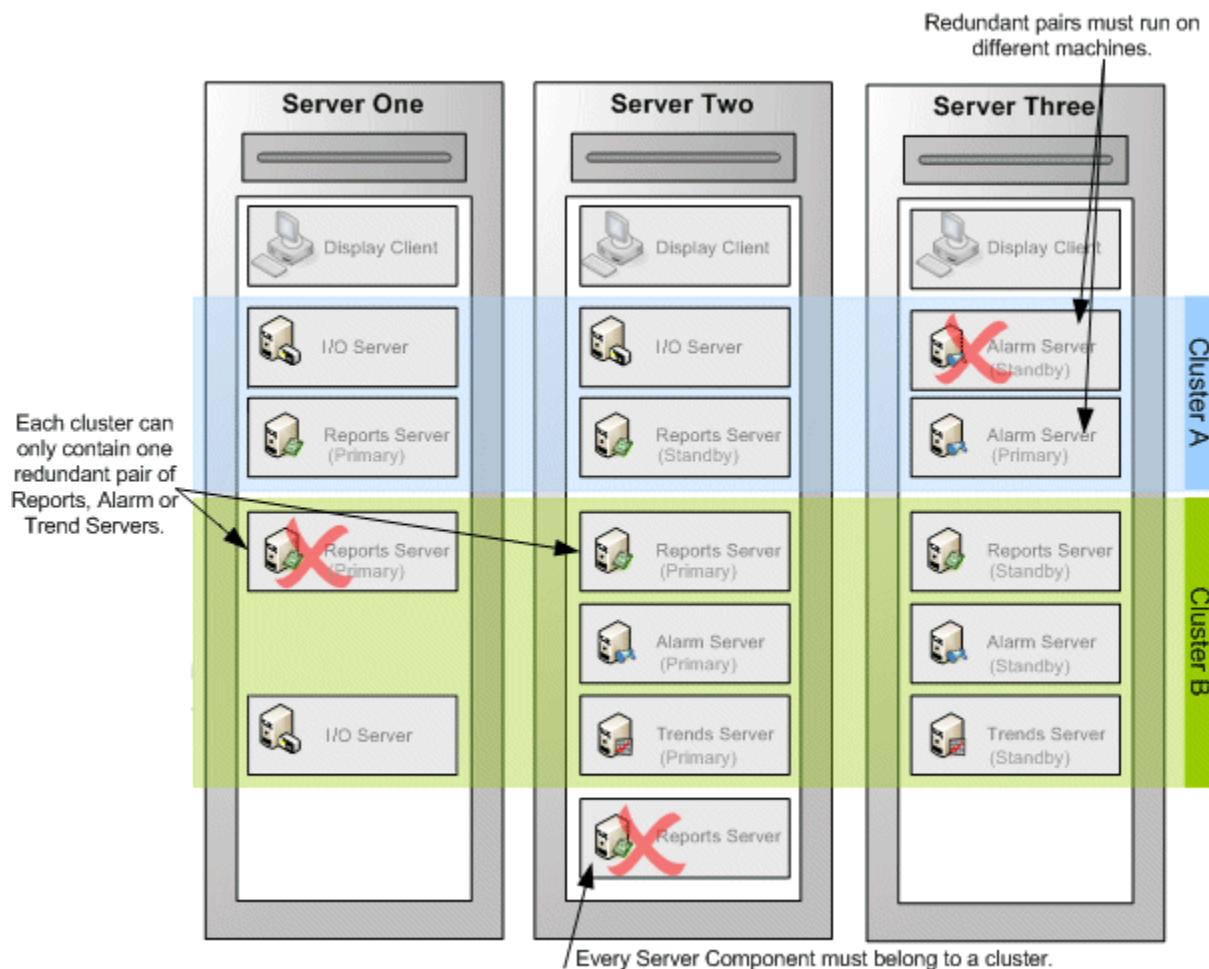
### Rules of Clustering

When configuring Plant SCADA the following clustering rules apply:

- Each cluster needs to have a unique name.
- Each server process needs to have a unique name.
- Each server process needs to belong to one cluster.
- Each cluster can contain only one pair of redundant alarm servers. They need to reside on different machines.
- Each cluster can contain only one pair of redundant reports servers. They need to reside on different machines.
- Each cluster can contain only one pair of redundant trends servers. They need to reside on different machines.
- Each cluster can contain an unlimited number of I/O servers.



The next diagram demonstrates circumstances where clustering is not configured correctly.



The Plant SCADA compiler or the [Runtime Manager](#) detects when the rules of clustering are not being observed and advises the user accordingly.

## Cluster Context

The Cluster Name can either be explicitly specified when the item is called or displayed, or an item can use the default cluster context of the calling process or page.

If your system has only one cluster, you do not need to specify a cluster. Plant SCADA will allow tags or tag references without a cluster context specified to be automatically resolved if every tag name in the project is unique.

This means if you use a unique naming convention for your tags, you will be able to configure a multi-clustered system without having to specify cluster context. For any tags that are not unique, #COM will be displayed if cluster context cannot be determined.

---

**Note:** This capability is enabled by default via the parameter [\[General\]TagDB](#).

---

## See Also

[Assigning Graphics Page Objects to a Cluster at Runtime](#)

## Add a Cluster

Clusters can be added in the Topology activity.

### To add a cluster:

1. In the **Topology** activity, select **Edit**.
2. From the drop down menu below the Command Bar, select **Clusters**.
3. Add a new row to the grid.
4. Type the required information in each column, or in the Property Grid (see below for a description of the properties).
5. Click **Save**.

## Cluster Properties

### General Properties

Property	Description
<b>Cluster Name</b>	The name of the cluster (maximum of 16 characters). The name needs to be unique to the project and not contain spaces. The fields are not case-sensitive and can contain 'A'..'Z', 'a'..'z', '0'..'9', and the underscore '_'. <b>Note:</b> When defining a cluster name, avoid using the <a href="#">Reserved Words</a> . If you use any of these, an error message will display when you compile your project.
<b>Comment</b>	Any useful comment. This property is optional and is not used at runtime.

### Project Properties

Property	Description
<b>Project</b>	The project in which the cluster is configured.

## See Also

[Clusters](#)

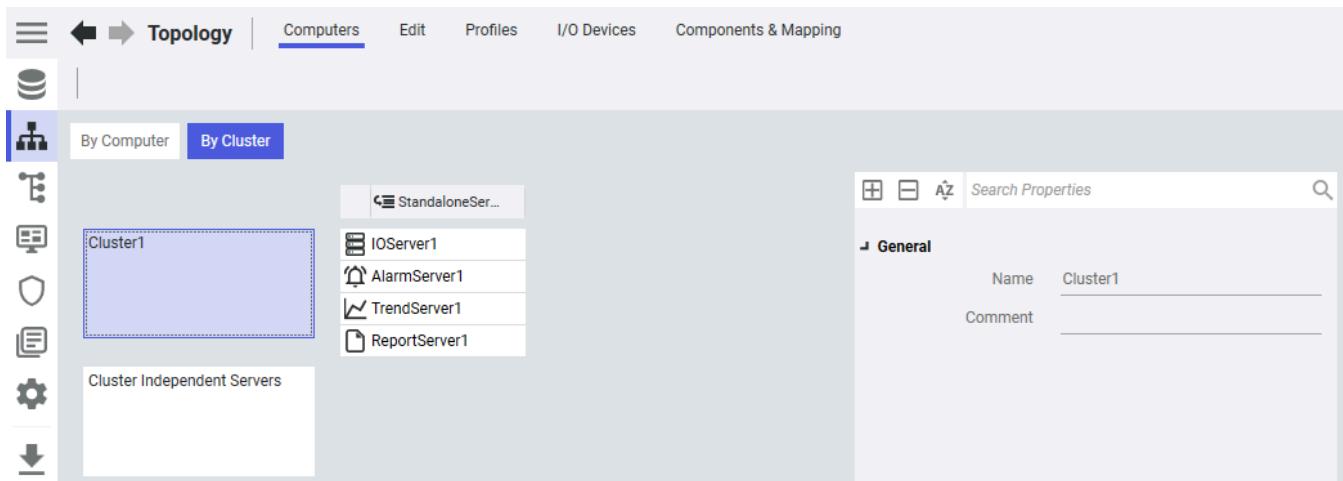
## View Clusters

In the **Topology** activity, select **Computers**. Click **By Cluster** to view your system's topology organized by clusters.

[By Computer](#)

[By Cluster](#)

The list of computers allocated to each cluster and the list of server processes (Alarm, Trend, Report, IO) installed on the computers in the cluster are displayed. The selected cluster card is displayed as shown below.



The selected cluster card displays the name of the cluster.

The Property Grid on the right displays the following information for the selected cluster:

- **Name**
- **Comment**

## See Also

[Add a Cluster](#)

## Assigning Graphics Page Objects to a Cluster at Runtime

Plant SCADA includes functionality that allows you to assign the objects on a graphics page to a specific cluster during Runtime.

If the tags on a page exist within a number of different clusters, you can use Cicode to pass a cluster name to the page as it opens. Any tags without a cluster explicitly defined will then be assigned to the specified cluster as the page is launched.

This provides a practical solution for a replicated system, where your project controls a number of identical sites or production lines. As the sites contain the same equipment, the tags representing the hardware architecture will potentially be the same for each. The ability to pass a cluster name to a page at runtime means just a single mimic page is necessary to represent multiple sites, reducing configuration time and simplifying project deployment.

**Note:** Plant SCADA will automatically resolve tags without a cluster context specified if every tag name in the project is unique. See [Clusters](#).

For example, a menu page could be created to launch the mimic page for three identical production lines, each based on the same page called "ProductionLine".

To achieve this, you would use the [PageDisplay](#) function to configure the following buttons on your menu page:

Button One	Text	<b>Production Line A</b>
	Command	PageDisplay("ProductionLine","Cluster_A")
	Comment	Display the mimic page in the context of production line A
Button Two	Text	<b>Production Line B</b>
	Command	PageDisplay("ProductionLine","Cluster_B")
	Comment	Display the mimic page in the context of production line B
Button Three	Text	<b>Production Line C</b>
	Command	PageDisplay("ProductionLine","Cluster_C")
	Comment	Display the mimic page in the context of production line C

In each case, PageDisplay would pass the name of the host cluster to the page, depending on which button is selected. As each production line shares a common architecture, any tags without a cluster explicitly defined will be assigned to the specified cluster as the page is opened.

This functionality is supported by the following Cicode functions:

- [PageDisplay](#)
- [PageGoto](#)
- [WinNew](#)
- [WinNewAt](#)
- [WinNewPinAt](#)

You can also use the [PageInfo](#) function to determine the cluster context that has been set for a page.

This functionality extends to any Cicode that may be called from a graphics page. You can write Cicode that only specifies variable tag names, allowing the cluster to be defined by the current context of the page from which it is launched.

If the Cicode is executed by the function [TaskNew](#), you have the option to specify a cluster by setting the *ClusterName* parameter.

---

**Note:** The cluster context for Cicode cannot be changed once the code is running.

---

## Server Processes

After you have configured computers, network addresses and clusters, you can add server processes and assign them to clusters and computers. Primary and standby alarms, reports, trends, and I/O server processes can be

added in Plant SCADA Studio's **Topology** activity.

## Default Ports

Each server process has a unique default port assigned to it. This default port may only be used with that type of server process. Attempting to use a default port on another type of server process will result in a compilation error message:

"Invalid port number (2080-2088,2073,3073,5482,20222 are reserved)."

The following table lists the default port numbers and their associated server process type.

Default Port	Server Process Type
2073	CtAPI  <b>Note:</b> Plant SCADA will open ports 2073 and 3073 when a CtAPI application is running. If the connection to a client is fully encrypted ("mixed mode" is not enabled), only port 3073 will be used. See <a href="#">Secure the Connection from a Remote CtAPI Application</a> .
2080	Alarm server.
2082	I/O server.
2084	Reports server.
2085	Trends server.
2087	System services process.
2088	Time synchronization port.
3073	CtAPI  <b>Note:</b> Plant SCADA will open ports 2073 and 3073 when a CtAPI application is running. If the connection to a client is fully encrypted ("mixed mode" is not enabled), only port 3073 will be used. See <a href="#">Secure the Connection from a Remote CtAPI Application</a> .
5482	Alarm server database port. This port is only used between the alarm server and its backend database on the same computer. It does not require external access.
20222	Required if Plant SCADA is configured to run as an ODBC server (see <a href="#">Using Plant SCADA as an ODBC Server</a> ).

**Note:** Ports 2074 - 2079 were reserved in earlier versions for legacy connections. These ports are no longer used.

For a list of the ports used by Plant SCADA runtime, see [Firewall Settings and Plant SCADA](#).

For details on adding and configuring server processes, refer to:

- [Add an Alarm Server Process](#)
- [Add a Report Server Process](#)
- [Add a Trend Server Process](#)
- [Add an I/O Server Process](#)
- [Configure an OPC DA Server](#)

## Add an Alarm Server Process

An alarm server process is responsible for evaluating the conditions that define an alarm (for more information, see [Alarm Server Process](#) in the section on Alarms).

Refer to the default server port numbers under [Server Processes](#).

---

**Note:** The Alarm server reuses the Alarm Properties port. As a result alarm properties are now published for configured alarm servers.

### To add an alarm server process:

1. In the **Topology** activity, select **Edit**.
2. From the drop down menu below the Command Bar, select **Alarm Server**.
3. Add a new row to the grid.
4. Type the required information in each column, or in the Property Grid (see below for a description of the properties).
5. Click **Save**.

## Alarm Server Properties

### General Properties

Property	Description
<b>Server Name</b>	The name of the server. The name needs to be unique to the project and any included projects, and not contain spaces.
<b>Mode</b>	The mode for this server, either Primary or Standby. If this property is left blank, the default value will be Primary.  The Primary and Standby servers need to run on different computers, and only one Primary and one Standby can be defined per cluster. Mode determines which server will be assigned the highest arbitration priority. The arbitration priority is used to work out which server will continue to be 'main' after a 'main-main' situation has occurred (see <a href="#">Alarm Server Side Synchronization</a> ).

Property	Description
<b>Cluster Name</b>	The name of the cluster to which this Alarm Server will belong. If there is only one cluster defined in the project, you can leave this field blank. The Alarm Server will default to the defined cluster
<b>Comment</b>	Any useful comment. This property is optional and is not used at runtime.

**Security Properties**

Property	Description
<b>Allow RPC</b>	Determines if the server process will execute remote <a href="#">MsgRPC</a> and <a href="#">ServerRPC</a> calls. Select <b>TRUE</b> or <b>FALSE</b> . If this field is left blank it will default to FALSE and the following compiler warning will be generated: "Allow RPC' permission is not defined for server (defaulting to FALSE) (W1038)".

**Communication Properties**

Property	Description
<b>Network Addresses</b>	The network address(es) of the server being configured. To specify dual network connections to the server, use a comma delimited list. See <a href="#">Network Redundancy</a> .
<b>Port</b>	The port this server will listen on. You can leave this field blank if you are running only one Alarm Server on the machine, in which case the default port number will be used.
<b>Database Port</b>	The port the server database will listen on. You can leave this field blank if you are running only one Alarm Server on the machine, in which case the default port number will be used.

**Other Properties**

Property	Description
<b>Extended Memory</b>	Determines whether the alarm server operates in Extended Memory mode (64-bit). Select TRUE to use Extended Memory Mode (see <a href="#">Alarm Server Process</a> ).
<b>Highest Priority Order</b>	Determines the order in which alarms are sorted. Select the required algorithm from the drop-down list.

Property	Description
	<p>If no algorithm is selected, the alarms will be listed according to "<b>Priority, Alarm State</b>" by default. For example:</p> <p>P1 Unacknowledged + Active  P1 Acknowledged + Active  P2 Unacknowledged + Active  P2 Unacknowledged + Inactive  P1 Shelved/disabled</p> <p>Select "<b>Alarm State, Priority</b>" to change the algorithm. The order of the example above would change to the following:</p> <p>P1 Unacknowledged + Active  P2 Unacknowledged + Active  P2 Unacknowledged + Inactive  P1 Acknowledged + Active  P1 Shelved/disabled</p> <p><b>Note:</b> Alarm servers across the system need to have the same algorithm selected. A compiler error will occur if they are different.</p> <p>If you want to create a customized sort order for alarms in a Situational Awareness project, you can use the parameter <a href="#">[Workspace]AlarmSortOrder</a>.</p>
<b>Archive Interval (Days)</b>	Schedules archiving to occur every ' number' of days. If set, archiving will occur automatically.
<b>Archive Path</b>	<p>Define where the archived data will be stored.</p> <p><b>Note:</b> The volume where data is to be saved should be labeled. This includes any removable storage devices. If this is not the case, archiving will not succeed and a message will display on the SOE page.</p>
<b>Archive Prefix</b>	Required prefix to the archive folder.

#### Project Properties

Property	Description
<b>Project</b>	The project in which the alarm server is configured.

#### See Also

[Alarm Server Side Synchronization](#)

## Add a Report Server Process

Note the default server port numbers under [Server Processes](#).

### To add a report server process:

1. In the **Topology** activity, select **Edit**.
2. On the menu below the Command Bar, select **Report Server**.
3. Add a new row to the grid.
4. Type the required information in each column, or in the Property Grid (see below for a description of the properties).
5. Click **Save**.

## Reports Server Properties

### General Properties

Property	Description
<b>Server Name</b>	The name of the server. The name needs to be unique to the project and not contain spaces.
<b>Mode</b>	The mode for this server, either Primary or Standby. If this property is left blank, the default value will be Primary. The Primary and Standby servers need to run on different computers, and only one Primary and one Standby can be defined per cluster.
<b>Cluster Name</b>	The name of the cluster to which this Reports Server will belong. If there is only one cluster defined in the project, you can leave this field blank. The Reports Server will default to the defined cluster
<b>Comment</b>	Any useful comment. This property is optional and is not used at runtime.

### Security Properties

Property	Description
<b>Allow RPC</b>	Determines if the server process will execute remote MsgRPC and <a href="#">ServerRPC</a> calls. Select <b>TRUE</b> or <b>FALSE</b> . If this field is left blank it will default to FALSE and the following compiler warning will be generated: "Allow RPC' permission is not defined for server (defaulting to FALSE) (W1038)".

### Communication Properties

Property	Description
<b>Network Addresses</b>	The network address(es) of the server being configured. To specify dual network connections to the server, use a comma delimited list. See <a href="#">Network Redundancy</a> .
<b>Port</b>	The port this server will listen on. You can leave this field blank if you are running only one Reports Server on the machine, in which case the default port number will be used.

**Project Properties**

Property	Description
<b>Project</b>	The project in which the report server is configured.

**See Also**[Add a Trend Server Process](#)

## Add a Trend Server Process

A Trend Server process controls the accumulation and logging of trend information. This information provides a current and historical view of the plant, and can be processed for display on a graphics page or printed in a report. Note the default server port numbers under [Server Processes](#).

**To add a trend server process:**

1. In the Topology activity, select **Edit**.
2. On the menu below the Command Bar, select **Trend Server**.
3. Add a new row to the grid.
4. Type the required information in each column, or in the Property Grid (see below for a description of the properties).
5. Click **Save**.

## Trend Server Properties

**General Properties**

Property	Description
<b>Server Name</b>	The name of the server. The name needs to be unique to the project and not contain spaces.
<b>Mode</b>	The mode for this server, either Primary or Standby. If this property is left blank, the default value will be

Property	Description
	Primary. The Primary and Standby servers need to run on different computers, and only one Primary and one Standby can be defined per cluster.
<b>Cluster Name</b>	The name of the cluster to which this Trends Server will belong. If there is only one cluster defined in the project, you can leave this field blank. The Trends Server will default to the defined cluster.
<b>Comment</b>	Any useful comment. This property is optional and is not used at runtime.

**Security Properties**

Property	Description
<b>Allow RPC</b>	Determines if the server process will execute remote <a href="#">MsgRPC</a> and <a href="#">ServerRPC</a> calls. Select <b>TRUE</b> or <b>FALSE</b> . If this field is left blank it will default to FALSE and the following compiler warning will be generated: "'Allow RPC' permission is not defined for server (defaulting to FALSE) (W1038)".

**Communication Properties**

Property	Description
<b>Network Addresses</b>	The network address(es) of the server being configured. To specify dual network connections to the server, use a comma delimited list. See <a href="#">Network Redundancy</a> .
<b>Port</b>	The port this server will listen on. You can leave this field blank if you are running only one Trends Server on the machine, in which case the default port number will be used.

**Project Properties**

Property	Description
<b>Project</b>	The project in which the trend server is configured.

**Add an I/O Server Process**

An I/O server process is a dedicated communications server that exchanges data between I/O devices and control clients. No data processing is performed by the I/O server (except for its local display) process.

Data is collected and passed to the control clients for display, or to another server for further processing. Data sent to an I/O device from any computer is also channeled through the I/O server. If data traffic is heavy, you can use several I/O servers to balance the load. The default server port numbers are listed in the topic [Server Processes](#).

**Note:** With the release of version 2020 R2, Plant SCADA now includes a Connectivity Service that manages tag subscriptions for the OPC UA Server and Industrial Graphics Server. This will place an additional load on your I/O servers, particularly if a subscribed tag also specifies a property or extension. The greater the number of I/O devices, the greater the processing load. If you have a large number of I/O servers on a single computer, it may impact CPU usage. To manage this, you should consider distributing your I/O servers across a number of computers.

### To add an I/O Server process:

1. In the **Topology** activity, select **Edit**.
2. On the menu below the Command Bar, select **I/O Server**.
3. Add a new row to the grid.
4. Type the required information in each column, or in the Property Grid (see below for a description of the properties).
5. Click **Save**.

## I/O Server Properties

### General Properties

Property	Description
<b>Server Name</b>	The name of the server. The name needs to be unique to the project and any included projects, and not contain spaces.
<b>Cluster Name</b>	The name of the cluster to which this I/O Server will belong. If there is only one cluster defined in the project, you can leave this field blank. The I/O Server will default to the defined cluster.
<b>Comment</b>	Any useful comment. This property is optional and is not used at runtime.

### Security Properties

Property	Description
<b>Allow RPC</b>	Determines if the server process will execute remote <a href="#">MsgRPC</a> and <a href="#">ServerRPC</a> calls. Select <b>TRUE</b> or <b>FALSE</b> . If this field is left blank it will default to FALSE and the following compiler warning will be generated: "'Allow RPC' permission is not defined for server"

Property	Description
	(defaulting to FALSE) (W1038)".

**Communication Properties**

Property	Description
<b>Network Addresses</b>	The network address(es) of the server being configured. To specify dual network connections to the server, use a comma delimited list. See <a href="#">Network Redundancy</a> .
<b>Port</b>	The port this server will listen on. You may leave this blank in which case the default port number will be used.

**Project Properties**

Property	Description
<b>Project</b>	The project in which the I/O server is configured.

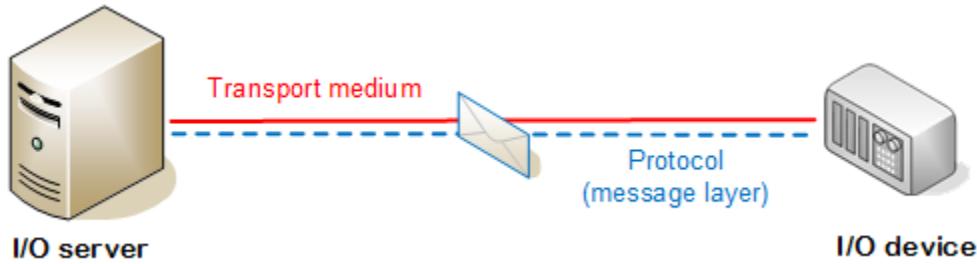
**See Also**

[I/O Devices](#)

## I/O Devices

Plant SCADA can communicate with any control or monitoring I/O device that has a communication port or data exchange interface, including PLCs (programmable logic controllers), loop controllers, bar code readers, scientific analyzers, remote terminal units (RTUs), and distributed control systems (DCS).

In a Plant SCADA system, I/O device communications typically consists of four key parts:



### [Plant SCADA I/O Server](#)

The Plant SCADA computer that directly connects to an I/O device is an **I/O server**. I/O servers are responsible for servicing the read, write and subscription requests from clients. A project can have many I/O servers, with each cluster able to include multiple I/O servers.

An I/O server keeps up-to-date information on its connected I/O devices by regularly retrieving data from each and storing it in a cache (I/O device data cache). Whenever a Plant SCADA client requires data from an I/O

device, it will use the information stored in the I/O server cache. Clients will not retrieve data directly from an I/O device.

An I/O server will read the necessary data from the I/O device to execute a requested Cicode task or process. For example, when you schedule a report, Plant SCADA reads the I/O device data that the report might need before the first line of the report starts running.

Plant SCADA performs writes to the I/O device asynchronously, allowing other operations to continue while a write is taking place.

### I/O Device

Plant SCADA is predominantly a supervisory system. It is the I/O devices that directly monitor and control automation equipment. In most I/O devices (such as PLCs) a program stored in the I/O device controls the outputs. The logic (control strategy) of this stored program and the status of the inputs determine the value of each output.

The value of each input and output is stored in a separate memory register in the I/O device. Each memory register is referenced by its address. You usually don't need to read (or write) to every register in the I/O device: Plant SCADA lets you specify which inputs and outputs you want to monitor or control.

By reading and writing to memory registers in I/O devices, Plant SCADA collects data from your plant or factory for monitoring and analysis, and provides high-level (supervisory) control of your equipment and processes.

**Note:** I/O devices such as programmable logic controllers (PLCs) usually have an internal program that controls the low-level processes within your plant. A PLC program continually scans the input registers of the PLC, and sets the output registers to values determined by the PLC program logic. While Plant SCADA can replace any PLC program, this is not recommended. PLCs are designed for high-speed response (typically 1 to 100 ms) and replacing this functionality with Plant SCADA could negatively affect your control system's performance. Only use Plant SCADA to complement your PLC program (that is, for high level control and system monitoring).

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Do not use Plant SCADA or other SCADA software as a replacement for purchased control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

### Transport Medium

The term 'transport' refers to both the physical communications medium and the low-level logic necessary to drive it. As far as Plant SCADA is concerned, it simply defines how to package a message, how to send it, and where to send it.

The available transport options are usually governed by the kind of ports available on the target I/O device. However, they are also influenced by the geography of the automation system, specific domain requirements (such as performance and redundancy), and the amount of money the owners are willing to invest.

The three main categories of available transport are:

- **Simple serial** – such as RS-232, RS-422, RS-485
- **Ethernet** – the dominant frame-based inter-network protocol
- **Proprietary** – using intermediary PC-communications hardware or software (such as OPC servers).

Of these, Ethernet is by far the transport technology of choice.

Fortunately, by using the concepts of packet encapsulation and intelligent gateway/routing devices, you are able to mix different transport layers and mediums, including high-speed wireless links or an FDDI network. For example, you could use a serial cable from Plant SCADA to a modem, a PSTN line from the modem to another modem, then a serial cable to an Ethernet gateway, and Ethernet to the I/O device.

---

**Note:** Modems are treated as a special case in Plant SCADA since they are recognized as logical devices by Windows. Refer to the specific online help on setting up modems.

---

Plant SCADA uses the ‘COMx’ driver to implement simple serial transports. Similarly, the ‘TCPIP’ driver implements the Ethernet transport. These are commonly used transport drivers, and both have a range of options and settings that can be tuned. Each proprietary board has its own specific transport driver – which is typically integrated with the protocol driver.

Similarly, the protocol-specific message could be a simple ASCII message or a complex object-based message such as DNP3. Engineers are free to assemble different combinations of I/O devices, transports, and protocols.

### Protocol

Protocol refers to the type of messages exchanged between the I/O server and I/O device.

I/O devices support at least one protocol – which governs the kind of commands and data you can exchange with the device. They vary significantly in functionality and in complexity. However, because Plant SCADA supplies the protocol drivers, the engineer does not need to know the details of the protocol.

Most modern devices support two or more protocols. This gives engineers flexibility in designing appropriate communications architectures. Occasionally the protocol is closely linked to the transport layer used – particularly in the case of proprietary protocols and communications hardware. In many cases, a specific card or module is necessary in the I/O device to support additional protocols.

### Industry Standard Protocols

The automation and associated industries have developed a number of standardized protocols for communicating with I/O devices:

- **ASCII** – for simple serial communications
- **Modbus** – a widely used simple serial protocol for automation
- **DNP 3.0** – a protocol for distributed networks such as RTUs
- **BACNet** – specifically for the building automation control industry
- **OPC** – a technology for sharing automation data at the PC level
- **IEC870-5** – communication profile for sending basic telecontrol messages
- **EIB** – European installation bus
- **Profibus** – field bus communications protocols for automation
- **SNMP** – widely used protocol for network devices.

As Plant SCADA allows you to choose whatever protocol you need for the situation, you can choose to use these protocols where supported by your I/O device. This is particularly useful when you have a mix of I/O device brands but you want to simplify and use one common protocol.

These components work in unison to expose the inputs and outputs of an I/O device to a Plant SCADA system.

- Inputs to the I/O device provide information about your plant, such as the speed of a machine, status of a conveyor, or the temperature of an oven.
- Outputs from the I/O device usually initiate tasks that control the operation of your plant, such as starting electric motors, varying their speed, or switching valves and indication lamps.

## See Also

[Configure I/O Device Communications](#)

## Configure I/O Device Communications

Setting up communications between a device and a Plant SCADA I/O server typically involves the following steps:

### 1. Define an I/O server process.

This defines the name of the Plant SCADA server with which the I/O device will communicate. See [Add an I/O Server Process](#).

### 2. Define a board.

This defines which board on your I/O server computer to use (for example, the mother board, network card, serial board or a PLC communication card). See [Add a Board](#).

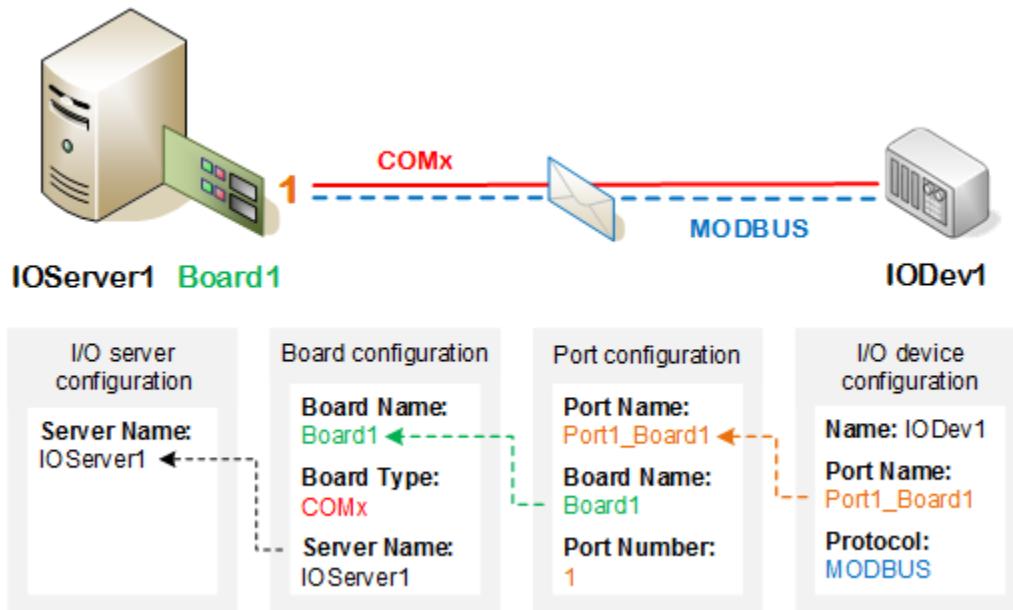
### 3. Define a port.

Often boards have multiple communication ports and you need to specify the port to use. Some equipment can have several logical (virtual) ports assigned to the one physical port. See [Add a Port](#).

### 4. Define the I/O device.

This defines the I/O device in Plant SCADA and specifies its address. The protocol is also defined at this level. See [Add an I/O Device](#).

The following diagram shows how the elements in the configuration environment map to the software components in the runtime system.



An I/O device driver is also required to implement the protocol and transport. Plant SCADA supports over 140 device drivers, enabling communication with a vast array of production devices across several communication types. These include generic drivers that support industry-standard protocols such as OPC and Modbus. Much of the functionality of the protocols and transports can be modified using the driver options and parameters. See [Working With Device Drivers](#).

A good place to start when setting up communications is with the [Using the Device Communications Wizard](#), a tool that automatically configures a default setup according to your specified requirements. Once you understand how the setup works, you can manually configure arrangements that are more complex.

**Note:** If there is no data to read or write, Plant SCADA will not communicate with an I/O device regardless of whether it is defined or not. You need to create a variable tag and use it before Plant SCADA will do a read request. For example, use an integer variable to display a number on a page.

## Add a Board

The properties of a board depend on the type of board installed in the I/O server computer. Common scenarios are:

- [Using a COM Port](#)
- [Using a Serial Board](#)
- [Using Ethernet](#)
- [Using a Proprietary Board](#).

You should confirm that the I/O server is enabled to support the necessary communications method. This may involve installing and configuring any communications hardware and/or software necessary for communications.

**Note:** The term "Board" is a legacy term from the time when ports on a computer were provided by additional boards. For example, an Ethernet card. The logical board concept is still useful even though a physical board may not exist (it may be software).

### To add a board:

- In the **Topology** activity, select **Components & Mapping**.
- On the menu below the Command Bar, select **Boards**.
- Add a row to the Grid Editor.
- Type the required information in each column, or in the Property Grid (see below for a description of the properties).  
For a description of the properties, see below.
- Click **Save**.

## Board Properties

### General Properties

Property	Description
<b>Server Name</b>	The name of the I/O server that hosts the board.
<b>Board Name</b>	A name for the board. 16 characters maximum. For example Server1_Board1. If you have more than one board in your I/O Server computer, the name of each board needs to be unique. If you have multiple I/O Servers, the board

Property	Description
	name need only be unique within each server. For example, Server1_Board2. The field is not case-sensitive.
<b>Board Type</b>	The type of board. 16 characters maximum. If you are using a serial board or your computer's COM port, enter COMx.
<b>Address</b>	The starting address of the Board. For example 0xCC00. 8 characters maximum. You need to specify the address to match the switch settings on the board when it was installed in your computer. If you are using a serial board or your computer's COM port, enter 0 as the address. <b>Note:</b> If more than one board is installed in the same computer, use a different memory address for each board.
<b>I/O Port</b>	The I/O port address of the Board. 8 characters maximum. You need to specify the address to match the switch settings on the board when it was installed in your computer. <b>Note:</b> If you are using your computer's COM port there is no need to enter the port address here. Specify the port number in the Ports form.
<b>Interrupt</b>	The interrupt number used by the Board. This is not necessary if using your computer's COM port.
<b>Special Opt</b>	Any special options supported by the board. 32 characters maximum. Please check the Hardware Arrangements Help Topic for your specific I/O Device to see if specific options are necessary.
<b>Comment</b>	Any useful comment. 48 characters maximum.

### Project Properties

Property	Description
<b>Project</b>	The project in which the board is configured.

### See Also

[Add a Port](#)

## Add a Port

The properties of a port depend on the type of board installed in the I/O server, and on the I/O device connected to the port.

### To add a port:

- In the **Topology** activity, select **Components & Mapping**.
- On the menu below the Command Bar, select **Ports**.
- Add a row to the Grid Editor.
- Type the required information in each column, or in the Property Grid (see below for a description of the properties).
- Click **Save**.

## Port Properties

### General Properties

Property	Description
<b>Server Name</b>	The name of the I/O server that hosts the board on which the port is located.
<b>Port Name</b>	A name for the port connected to your I/O Device(s). 31 characters maximum. Each port needs to have a unique name (i.e. you cannot assign the same Port Name to two ports in your system). You can use any name, for example, "Board1_Port1".  If you have more than one board in your computer, you can use the port name to identify the board, for example, "Board2_Port1".
<b>Port Number</b>	The port number that the I/O device is connected to. 4 characters maximum.  Avoid assigning the same Port Number to two ports on a board, unless connecting to a dial-up remote I/O device via a modem (see the note below). Ports on different boards can be assigned the same number.  If you are using your computer's COM port enter the port number here - the port number is defined in the Ports section of the Windows Control Panel.  <b>Note:</b> If you are connecting to a dial-up remote I/O Device (via a modem), you need to define a unique port name on the I/O server for each dial-up remote I/O device, and the port number needs to be -1 for each.

Property	Description
<b>Board Name</b>	The name you used for the board. 16 characters maximum. This is necessary to link the port to the board. For example "Server1_Board1".
<b>Baud Rate</b>	<p>The baud rate of the communication channel (between the Plant SCADA I/O Server and the I/O Device). 16 characters maximum.</p> <p><b>Note:</b> The I/O device hardware and serial board may support other baud rates. If you do choose an alternative baud rate, verify that both the I/O Device and serial board support the new baud rate.</p>
<b>Data Bits</b>	The number of data bits used in data transmission. You need to set your I/O Device to the same value or a communication link cannot be established.
<b>Stop Bits</b>	The number of stop bits used to signify the completion of the communication. You need to set your I/O Device to the same value.
<b>Parity</b>	The data parity used in data transmission.
<b>Special Opt</b>	Any special options supported by the port. 32 characters maximum. Please check the Hardware Setup Help Topic for your specific I/O Device to see if specific options are necessary.
<b>Comment</b>	Any useful comment. 48 characters maximum.

### Project Properties

Property	Description
Project	The project in which the port is configured.

## See Also

[Add a Board](#)

### Add an I/O Device

To manually add an I/O device to a project, you need to specify its properties using the **Topology** activity. The properties depend on both the I/O device and protocol.

You should verify your system meets the hardware and software requirements needed to establish communication between a device and Plant SCADA.

This may include a variety of possibilities, including:

- the installation of a proprietary communications card
- the establishment of a device server
- the installation of configuration software.

The **Driver Reference Help** provides the hardware and software requirements for each device, and includes any additional information you need to know about setting up specific devices.

**Note:** You can use the Device Communications Wizard to add an I/O device to your project. This will automatically configure many of the property settings described in this topic.

#### To add an I/O device:

- In the **Topology** activity, select **I/O Devices**.
- Add a row to the Grid Editor.
- Type the required information in each column, or in the Property Grid (see below for a description of the properties).
- Click **Save**.

## I/O Device Properties

### General Properties

Property	Description
<b>Server Name</b>	The name of the I/O server to which the I/O device is connected.
<b>Name</b>	The name for the I/O device (31 characters maximum).  The name needs to be unique in the Plant SCADA system, unless the I/O device is defined in other I/O servers (to provide redundancy). If redundancy is used, the I/O device needs to then have the same I/O device number and address for each I/O Server. use different I/O device names for your primary and standby I/O devices, otherwise I/O device Cicode functions cannot differentiate between them.
<b>Number</b>	A unique number for the I/O device (0-16383).  The number needs to be unique in the Plant SCADA system, unless the I/O device is defined in other I/O servers (to provide redundancy). If redundancy is used, the I/O device needs to then have the same I/O device number and address on each redundant I/O server. You may use the same device name, but if you want to use I/O device Cicode functions, it is easier to have different I/O device names.
<b>Address</b>	The address of the I/O device (64 characters)

Property	Description
	<p>maximum).</p> <p>The format of the address you enter in this field is determined by the type of I/O device (and protocol) used, as each has a different addressing strategy.</p>
<b>Protocol</b>	<p>The protocol used to communicate to the I/O device (16 characters maximum).</p> <p>Many I/O devices support multiple protocols, dependent on the communication method chosen.</p>
<b>Port Name</b>	<p>The port on the board to which the I/O device is connected (31 characters maximum).</p> <p>This is necessary to link the I/O device to the port. For example Board1_Port1.</p> <p><b>Note:</b> There is a limit of 255 COMx ports on a server. To avoid this limitation restricting the number of remote I/O devices you use, you can connect multiple remote I/O devices to the same port as long as communication details (telephone number, baud rate, data bits, stop bits and parity) are identical.</p>
<b>Startup Mode</b>	<p>The type of I/O device redundancy used. The options are:</p> <ul style="list-style-type: none"> <li>• Primary — Enable immediate use of this communications channel. This is the default mode if no mode is specified.</li> <li>• Standby — This channel will remain unused until the I/O device configured with the primary channel becomes inoperative.</li> <li>• StandbyWrite — This channel will remain unused until the I/O device configured with the primary channel becomes inoperative. Write requests sent to the primary channel are also sent to this channel.</li> </ul> <p>To use Standby or StandbyWrite modes, you need to also configure a primary I/O device with the same I/O device name, number and address.</p> <p><b>Note:</b> Avoid using StandbyWrite mode for scheduled I/O devices because there is the possibility that the Standby I/O device write queue will not be cleaned up. When writing to a scheduled I/O device which it is not communicating, write requests are queued up until communication resumes.</p>

Property	Description
	<p>In case of using I/O device redundancy and StandbyWrite mode, Primary and Standby I/O devices have their own queues of pending write requests. If I/O device communication is controlled automatically by the communication schedule, then both queues are flushed when the schedule dial-time occurs.</p> <p>Where manual communication is controlled by the <a href="#">IODeviceControl</a> Cicode function, only the Primary I/O device write queue is flushed and the Standby I/O device write queue will not be cleaned up.</p>
<b>Priority</b>	<p>Specifies the order standby devices are promoted in if a primary I/O device becomes inoperative during runtime (8 characters maximum).</p> <p>If there is more than one standby I/O device configured for a cluster, you can use this field to give precedence to a particular standby device. If this field is left blank, priority will be automatically allocated to the device during project compilation, based on the priority settings of other standby devices and/or the order in which the devices were configured.</p> <p>See <a href="#">I/O Device Promotion</a>.</p>
<b>Memory</b>	<p>Specifies whether the I/O device runs in Memory mode. The default value is FALSE.</p> <p>If you select TRUE, the I/O device will be created in memory and its values stored in memory at runtime. This may be useful if you are testing your system, before connecting physical I/O devices, as memory mode stops Plant SCADA from communicating with physical I/O devices. See <a href="#">Using Memory Mode</a>.</p> <p><b>Note:</b> If a device is configured with <b>Memory</b> set to TRUE, the <b>Port Name</b> and <b>Address</b> fields can be left blank as they will not be used by the compiler or the I/O server at runtime.</p>
<b>Read-Only</b>	<p>Specify whether the I/O device is read-only. The default value is FALSE.</p> <p>If you select TRUE, any attempt to write to a tag associated with the I/O device will be unsuccessful.</p>
<b>Exclusive</b>	<p>This setting controls the way in which redundant I/O device(s) configured with the same network number on different I/O servers activate communications to the physical device they represent.</p>

Property	Description
	<p>When Exclusive is set to TRUE, only one I/O device (on one I/O server) will activate communications to the physical device at a time. The best available I/O device (online with the lowest priority) at a particular time will be chosen to actively communicate. When this I/O device is no longer the best available, it will deactivate, which will in turn trigger the best available I/O device on another server to activate.</p> <p>When Exclusive is set to FALSE (default), the best available I/O device will typically be the only active I/O device; however this may not be enforced. When a different I/O device becomes the best it will activate immediately, without waiting for the other I/O device to deactivate. An I/O device that is not the best may also be forced into actively communicating with the physical device if a particular client can only communicate with the I/O server that I/O device is running on.</p> <p><b>Note:</b> The parameter <a href="#">[IOServer]PeerServerConnectTimeOut</a> has been implemented so that on startup of an I/O server, all I/O devices with Exclusive set to TRUE will not be allowed to activate before the I/O server has successfully connected to its peers. This parameter defines how long an I/O server will wait until it ceases trying to connect to peer I/O servers.</p>
Comment	Any useful comment. This field is optional and is not used at runtime.

**Note:** For the **Number**, **Protocol** and **Port** properties, the same network number is used for redundancy (as stated above), however this implies that the protocol is usually the same (though there maybe some special circumstances where devices support multiple protocols) and the Port modes are similar. i.e. a real world port reference, DISKDRV or MEMORY . If DISKDRV is being used, then redundant units (i.e. the same NUMBER) needs to be DISKDRV. This also applies for MEMORY mode.

#### External Properties

Property	Description
Linked	<p>Select TRUE or FALSE to specify whether or not the I/O device is linked to an external data source (see <a href="#">Link Tags to an External Data Source</a>).</p> <p>If you link to an external data source, Plant SCADA is updated with any changes made to the external data source when a refresh is performed (see <a href="#">Refresh the Tags for a Linked I/O Device</a>).</p>

Property	Description
	<p>If you disconnect an existing link, you can choose to make a local copy of the tags in the database or you can delete them from Plant SCADA's variable tags database altogether.</p> <p>If an I/O device is linked to an external data source the Database Type, External Database, Connection String, Tag Prefix and Live Update fields will be greyed out.</p>
<b>Database Type</b>	<p>The format of the data referenced by the external data source.</p>
<b>External Database</b>	<p>The path and file name of the external data source for the I/O device.</p> <p>If the <b>Linked</b> property is set to TRUE (and a <b>Database Type</b> has been specified), you can use the <b>Browse</b> button next to this field in the Property Grid to select and validate a database.</p> <p>Alternatively, you can enter the IP address/directory, computer name, or the URL of a data server. For example:</p> <ul style="list-style-type: none"> <li>• "C:\Work.CSV"</li> <li>• "127.0.0.1"</li> <li>• "139.2.4.41\HMI_SCADA"</li> <li>• "http://www.abicom.com.au/main/scada"</li> <li>• "\coms\data\scada".</li> </ul>
<b>Connection String</b>	<p>Enter a connection string to provide connection details for the data source. This is similar to an ODBC connection string. For example:</p> <p>UserID = XXX; Password = YYY</p> <p>or</p> <p>ServerNode=111.2.3.44; Branch=XXX</p> <p>Not every data source needs a connection string.</p>
<b>Tag Prefix</b>	<p>The prefix that will be inserted in front of the names of linked tags in Plant SCADA's variable tags database (for this I/O device only). To change the prefix, delete it first, perform a manual refresh, then add the new prefix.</p>
<b>Automatic Refresh</b>	<p>Determines whether the linked tags in Plant SCADA's variable tags database will be updated when the external data source is changed.</p>

Property	Description
	<p>This refresh will occur the first time you link to the data source, and then whenever you compile your project.</p> <p>Without an automatic refresh, you will need to perform a manual refresh to update the linked tags in Plant SCADA (see <a href="#">Refresh the Tags for a Linked I/O Device</a>).</p>
<b>Live Updating</b>	<p>Controls whether or not the linked tags in Plant SCADA and an external tag database will be synchronized if either database is changed. To enable live linking, choose Yes from the Live Update menu.</p> <p>When Live Update is enabled and the variable tag database is accessed (for example, during project compilation or when a drop-down list is populated), Plant SCADA queries the external tag database to determine if it has been modified. If so, Plant SCADA merges the changes into the local variable tag database. Conversely, any changes made to the local tag variable database will be incorporated automatically into the external tag database.</p>

### Logging Properties

Property	Description
<b>Log Write</b>	<p>Enables/disables the logging of writes to the I/O device. When enabled, writes are logged to the SYSLOG.DAT file in the <b>Logs</b> folder of the Plant SCADA <b>User and Data</b> folder selected during installation, also specified in the INI file as <a href="#">[CtEdit]Logs</a>.</p> <p>See <a href="#">Tag Functions</a> for information about <a href="#">TagWriteEventQue</a> and logging tag data.</p> <p><b>Note:</b> Logging writes to every I/O Device may slow communications as the Plant SCADA system will be writing large amounts of data to disk. However, logging of writes is useful when debugging a system.</p>
<b>Log Read</b>	<p>Enables/disables the logging of reads from the I/O device. When enabled, reads are logged in the Plant SCADA SYSLOG.DAT file.</p> <p><b>Note:</b> Logging reads to every I/O device may slow communications as the Plant SCADA system will be writing large amounts of data to disk. However, logging of reads is useful when debugging a system.</p>

### Cache Properties

Property	Description
<b>Cache</b>	<p>Enables/disables data caching. When enabled, a cache of the I/O device's memory is retained at the I/O server, thus improving communications turn-around time.</p> <p><b>Note:</b> Data caching has to be enabled for scheduled I/O devices, but disabled for memory or disk I/O devices.</p>
<b>Cache Time</b>	<p>The cache time specified in milliseconds. When caching is enabled, data that is read from a I/O device is stored temporarily in the memory of the I/O server. If another request is made (from the same or another client) for the same data within the cache time, the Plant SCADA I/O server returns the value in its memory - rather than read the I/O device a second time. Data caching results in faster overall response when the same data is needed by many clients.</p> <p>A cache time of 300 milliseconds is recommended.</p> <p>The Cache Time for a scheduled I/O device is automatically calculated. You can view a scheduled I/O device's cache time using the Kernel Page Unit command.</p>

#### Update Properties

Property	Description
<b>Background Poll</b>	<p>Specifies that tags for a particular device are continuously polled at a minimum background poll rate. The options are True or False. Set this field to True if you are operating on a slow network, with a slow device, or where tags may normally only be polled infrequently.</p> <p><b>Note:</b> This setting does not apply to drivers that use the Driver Runtime Interface (DRI), as a DRI driver controls updates to the I/O server. DRI-based devices should leave this set to FALSE.</p> <p>For a list of DRI drivers, see <a href="#">Retrieving Time-stamped Data from I/O Devices</a>.</p>
<b>Background Rate</b>	<p>When Background Poll is set to True, specifies the minimum rate by which the device will be polled. You may select any predefined value from the drop-down list, or enter your own in the format of HH:MM:SS.</p> <p>If a value is not entered, the default value of 30 seconds will be used.</p>

Property	Description
	<p><b>Note:</b> This setting does not apply to drivers that use the Driver Runtime Interface (DRI). If you are using a DRI-based device, you should not change this setting.</p> <p>For a list of DRI drivers, see <a href="#">Retrieving Time-stamped Data from I/O Devices</a>.</p>
<b>Min Update Rate</b>	<p>The DataSource will send tag update value notifications to subscription clients after a pre-defined period of time expires. You may select any predefined value from the drop-down list, or enter your own in the format of HH:MM:SS.</p> <p>If a value is not entered, the default value of 0 seconds will be used and no update value notifications will be sent.</p>
<b>Staleness Period</b>	<p>Staleness period represents total number of seconds that will elapse after the last update before extended quality of the tag element is set to "Stale". You may select any predefined value from the drop-down list, or enter your own in the format of HH:MM:SS. If a value is not entered, the default value of 0 seconds will be used and tag elements will not be set to "Stale".</p> <p><b>Note:</b> For further information on Minimum Update Rate and Staleness Period refer to <a href="#">Tag Data Persistence</a>.</p>

### Scheduling Properties

Property	Description
<b>Scheduled</b>	<p>Determines whether the I/O device is configured for scheduled communications. This is normally set using the Device Communications Wizard.</p> <p><b>Note:</b> If a schedule for a dial-up remote I/O device is not specified, the connection will be established at startup and will remain connected until shut-down.</p> <p>See <a href="#">Scheduled Communications</a>.</p>
<b>Time</b>	<p>The I/O server will attempt to communicate with the I/O device at this time, and then at intervals as defined below. This time is merely a marker for Plant SCADA. If you run up your project after this time, the I/O server will not wait until the next day to begin communicating. It will operate as if your project had been running since before the start time.</p>

Property	Description
<b>Period</b>	<p>The time between successive communication attempts. The period needs to be long relative to the driver's watchtime and the <a href="#">[Dial]WatchTime</a>. If necessary, decrease these watchtimes or increase Period to make Period more than double the watchtimes.</p> <p>Examples (based on a Synchronize at time of 10:00:00):</p> <ul style="list-style-type: none"> <li>• If you enter 12:00:00 in the Repeat every field, and start your project at 9 a.m., the I/O Server will communicate with the I/O Device at 10 a.m., then once every 12 hours after that, i.e. 10 p.m., then again at 10 a.m. of the following day, etc.</li> <li>• If you enter 12:00:00, and start your project at 4 p.m., the I/O Server will communicate with the I/O Server at 10 p.m., then again at 10 a.m. of the following day, and so on. The I/O Server will assume that communications were established at 10 a.m., so it continues as if they had been - communicating once every 12 hours after 10 a.m.</li> <li>• If you enter 3 days, and start your project at 9 a.m. on a Wednesday, the I/O Server will communicate with the I/O Device at 10 a.m., then once every 3 days after that, i.e. 10 a.m. on the following Saturday, then at 10 a.m. on the following Tuesday, etc.</li> <li>• If you enter the 6th of December in the Repeat every, and start your project during November, the I/O Server will communicate with the I/O Device at 10 a.m. on December 6, then again on December 6 of the following year, etc.</li> </ul> <p>Select On Startup for a persistent connection. To disconnect a persistent connection, you need to call the <a href="#">IODeviceControl()</a> function with type 8.</p>
<b>Connect Action</b>	Cicode to be executed once communication with the I/O device has been established (and before any read or write requests are processed).
<b>Disconnect Action</b>	Cicode to be executed once communication with the I/O device has been terminated (and after read and write requests are processed).
<b>Phone Number</b>	The telephone number that needs to be dialed to

Property	Description
	initiate contact with the I/O device. (i.e. for dial-up remote I/O Devices)

**Storage Properties**

Property	Description
<b>Persist</b>	Set to TRUE or FALSE to indicate if the persistence of the tag cache file is on or off. Default is TRUE and the data will be written to the XML file as described in "File Name" below. If one of the primary/standby I/O devices (identified by the same network number) has persistence set to FALSE, then every device has it turned off.
<b>Persist Period</b>	Indicates how often to persist the XML cache file to disk. Default is 10 minutes (00:10:00). Normally, the period used is the highest value specified for one of the primary/standby I/O devices (identified by the same network number).
<b>File Name</b>	Indicates the path to the file on disk where the XML cache will be stored. You can specify just the path or the path with the actual file name. The default is Data directory using the format:  <ClusterName>.<IOServerName><ClusterName>.<IODeviceName>.cache.  If the location cannot be created, then either the default path, default file name or both is used. Normally, the path used is the first non-empty path specified for one of the primary/standby I/O devices (identified by the same network number).  Be aware that you will need to manually configure the access control list for a new folder location so that it matches the permissions that were applied to the default location during installation. See <a href="#">Configure Directory Security for Modified Folder Locations</a> .

**Project Properties**

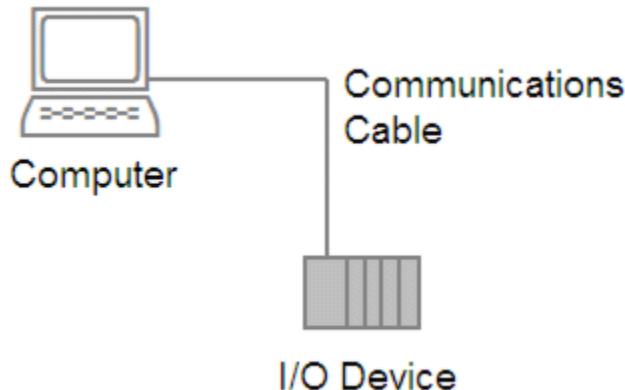
Property	Description
<b>Project</b>	The project in which the I/O device is configured.

**See Also**

[Add a Board](#)

## Using a COM Port

The simplest Plant SCADA systems use a single computer connected to the I/O device(s). You can connect an I/O device directly to a communications port with a standard RS-232 communications cable.



### To set up Plant SCADA to use your computer's COM port:

1. Verify that the **Boards** configuration has **COMx** as the **Type**, and the **Address** set to 0. The **I/O Port**, **Interrupt** and **Special Opt** can be left blank.
2. Enter the **Port Number** in the **Ports** configuration. The COM port number will usually be either 1 or 2, and is set in the Ports section of the Control Panel.

Use the **Special Opt** field to modify the behavior of the COMx driver (see [COMx Driver Special Options Reference](#) for more information).

**Note:** You only need to define the COMx board once. You can then add several ports that use the same Plant SCADA board. For example, a COM port and two serial boards could be defined as one COMx board in Plant SCADA, with multiple ports.

## See Also

[Debugging a COMx Driver](#)

[Using a Serial Board](#)

## COMx Driver Special Options Reference

**Special Options** (in the **Ports** grid) are space separated and start with the dash character (-) immediately followed by the option characters. Only a small percentage of users will need to use the following options:

Special Option Character	Description
<code>--cATS0=1</code>	Send the string 'ATS0=1<CR>' on startup to allow the modem to detect the baud rate the port is running at. Abandon transmit if DCD is low. Wait for incoming call to raise DCD.
<code>--d</code>	Data will be transmitted only when DSR is high.

Special Option Character	Description
—di	Received data is ignored when DSR is low.
—dMS	When transmitting a message the driver will wait up to 2000 ms for DSR to go high, then wait another MS milliseconds before transmitting.
—e	<p>Provides access to the output signal lines. The format is "~EIAWXYZ" where WXYZ represents one of the following options:</p> <ul style="list-style-type: none"> <li>S Simulate XOFF received</li> <li>S Simulate XON received</li> <li>RT Set RTS high</li> <li>RT Set RTS low</li> <li>D Set DTR high</li> <li>D Set DTR low</li> <li>B Set the device break line</li> <li>B Clear the device break line</li> </ul> <p>Example: "~EIADTR1" sets DTR high.</p>
—h	Data will be transmitted only when CTS is high.
—hMS	When transmitting a message the driver will wait up to 2000 ms for CTS to go high, then wait another MS milliseconds before transmitting.
—i	<p>The string sent whenever the port is initialized. The tilde (~) and '\M' characters represent special instructions:</p> <ul style="list-style-type: none"> <li>~: Delay for 500 milliseconds</li> <li>\M: Send carriage return</li> </ul> <p>Examples:</p> <ul style="list-style-type: none"> <li>~Fred: Wait 500 milliseconds and then send 'Fred'</li> <li>Fred\MMary: Send 'Fred', a carriage return, and then 'Mary'</li> </ul> <p><b>Note:</b> This option is not available for dialable devices (i.e. when the port number is -1).</p>
—nt	With some serial interfaces, line interruptions can cause the COMx read thread to shutdown. If this happens, the driver does not recover after the interruption. However, with the -nt (no terminate) option set, the thread is not shutdown, allowing the system to recover when the interruption is rectified.

Special Option Character	Description
—nts	If error messages occur when the COMx driver is starting up, it will not terminate, but will continue attempts to open the COMx port.
—r	Driver will raise DTR only when transmitting.
—ri	DTR is raised when there is enough room in the input buffer to receive incoming characters and drop DTR when there is not enough room in the input buffer.
—rPRE,POST	When transmitting a message the driver will raise DTR for PRE milliseconds, transmit message, wait for POST milliseconds then drop DTR.
—sc	Activates software flow control using XON and XOFF. XonLim: A number in bytes. This represents the level reached in the input buffer before the XON character is sent (30 bytes). XoffLim: The maximum number of bytes accepted in the input buffer before the XOFF character is sent. This is calculated by subtracting (in bytes) 100 from the size of the input buffer.
—t	Driver will raise RTS only when transmitting.
—ti	RTS is raised when there is enough room in the input buffer to receive incoming characters and drop RTS when there is not enough room in the input buffer.
—to	RTS is raised only when there are characters to transmit.
—tPRE,POST	When transmitting a message the driver will raise RTS for PRE milliseconds, transmit message, wait for POST milliseconds then drop RTS.

## See Also

[Using a COM Port](#)

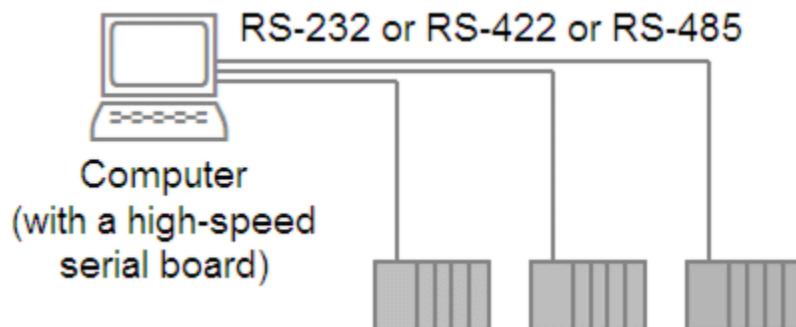
[Using a Serial Board](#)

[Debugging a COMx Driver](#)

## Using a Serial Board

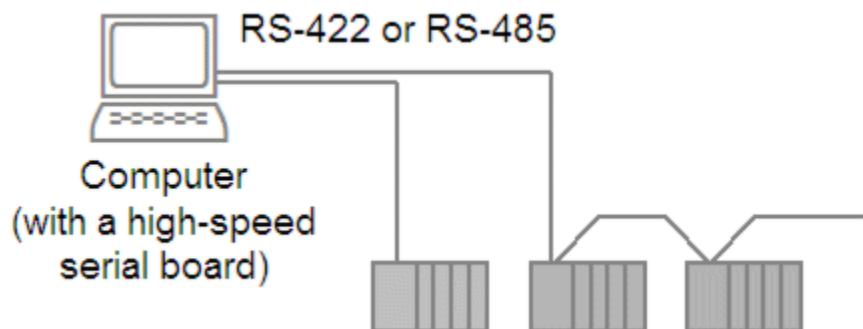
The communications port on the computer is not designed for high-speed communications and reduces system performance. Instead, install a high-speed serial board. High-speed serial boards have several ports (usually 4, 8,

or 16) to let you connect several I/O devices to your Plant SCADA system.

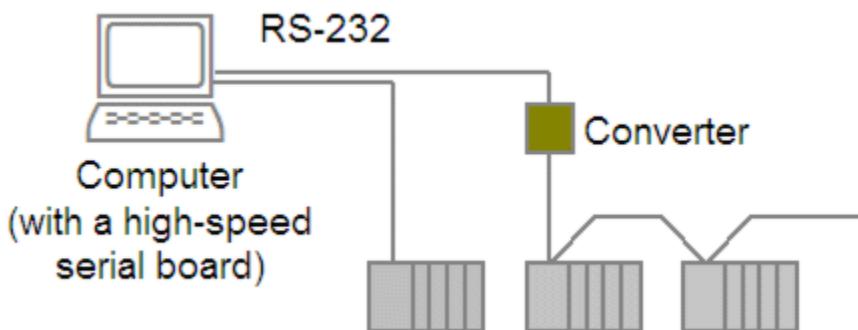


You can use identical I/O devices or I/O devices supplied by different manufacturers; Plant SCADA supports many popular I/O devices. You can connect any number of I/O devices; the only limitation is the size of your computer. High-speed serial boards are available for RS-232, RS-422, or RS-485 communication.

If you have several I/O devices from the same manufacturer and these I/O devices support multi-drop communication, you can connect them to an RS-422 or RS-485 high-speed serial board installed in your computer. (The RS-232 standard does not support multi-drop communication.)



Not every high-speed serial board supports RS-422. You can use an RS-232/RS-422 or RS-232/RS-485 converter to achieve the same arrangement.



**Note:** Using a converter can introduce handshaking/timing considerations.

#### To set up Plant SCADA to use a serial board:

1. Install the board in your computer and set it up under Windows as per the accompanying instructions. Use the latest driver from the board manufacturer.

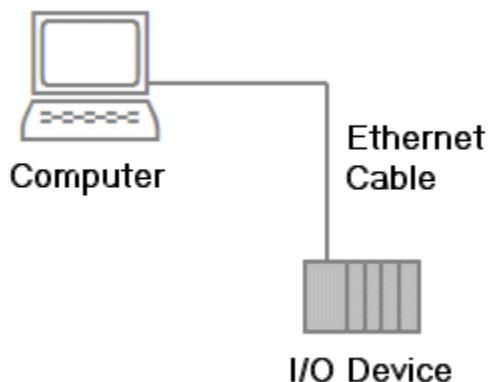
2. Check that the boards configuration has COMx as the Type, and the Address set to 0. The **I/O Port, Interrupt**, and **Special Opt** can be left blank.
3. Enter the **Port Number** in the Ports configuration. The COM port number is usually greater than 2 and set in the **Ports** section of the Control Panel. Use the **Special Options** field to modify the behavior of the COMx driver. (See [COMx Driver Special Options Reference](#) for more information).

**Note:**

- If using your computer's COM port, you don't need to install additional software.
- You only need to define the COMx board once. You can then add several ports that use the same Plant SCADA board. For example, a COM port and two serial boards could be defined as one COMx board in Plant SCADA, with multiple ports.

**See Also**[Add a Board](#)[Using a Proprietary Board](#)**Using Ethernet**

Ethernet is a popular method of communicating with a number of I/O Devices at high speed. It is certainly one of the easiest to configure. You can typically connect to Ethernet capable I/O Devices using a standard Ethernet card, or integrated Ethernet port.

**To set up Plant SCADA to use your computer's Ethernet card:**

1. Confirm that your Ethernet card is installed correctly and working under Windows.
2. Check that the **Boards** configuration has TCP/IP as the **Type**, and the **Address** set to 0. The **I/O Port, Interrupt** and **Special Opt** can be left blank.
3. In the **Ports** form, use the **Special Opt** field to enter the destination IP address using the following format:

**-la -Pn -T**

Where:

**a** = the destination IP address in standard Internet dot format. (For example 192.9.2.60)

**n** = the destination Port number. Often one physical port has several virtual ports, used for different purposes. Use this option only if you want to override the default of 2222.

**T** = forces the driver to use TCP (the default), rather than UDP (-U).

(See [TCP/IP Driver Special Options Reference](#) for more information)

Leave other fields blank. You can now define your I/O Device units.

**Note:** You only need to define the TCP/IP board once. You can then add several ports that use the same Plant SCADA board.

## See Also

[Using a Serial Board](#)

### TCP/IP Driver Special Options Reference

Special Options (in the properties for a port) are space separated and start with the dash character (—) immediately followed by the option characters. You can use the following special options for TCP/IP:

Special option character	Description
—A	Allows the TCPIP driver to be used from Cicode.
—la.b.c.d	Defines remote IP address to connect to.
—K	Sets socket SO_KEEPALIVE flag.
—Lla.b.c.d	Defines local IP address.
—LPn	Defines local PORT.
—Ma.b.c.d	Defines multicast IP address. IP address joins a multicast group listening for UDP multicast packets. For the parameter to work configure the -LP local port switch, so the driver selects the correct port on the multicast IP to listen on. This option can be set using the [TCPIP]PortName.MulticastAddress parameter. See <a href="#">Debugging a TCP/IP Driver</a> .
—Pn	Defines remote PORT to connect to.
—RC	Activates the reconnection retries on reception of FD_CLOSE event. FD_CLOSE is only received if the Keepalive option is activated. -RC can be useful where the TCP/IP driver is notified of the connection close. For a more comprehensive handling, the -k option is needed.
—T	Sets this port for TCP (stream) operation.
—U	Sets this port for UDP (datagram) operation.

Where:

**a.b.c.d** = standard IP address in dot notation using decimal numbers 0- 255. (Avoid using a leading 0 when adding an IP address).

**n** = decimal value for the port ID for the necessary service.

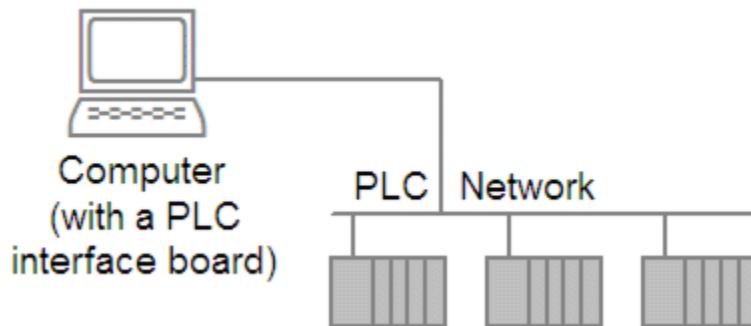
## See Also

[Using Ethernet](#)

[Using a Serial Board](#)

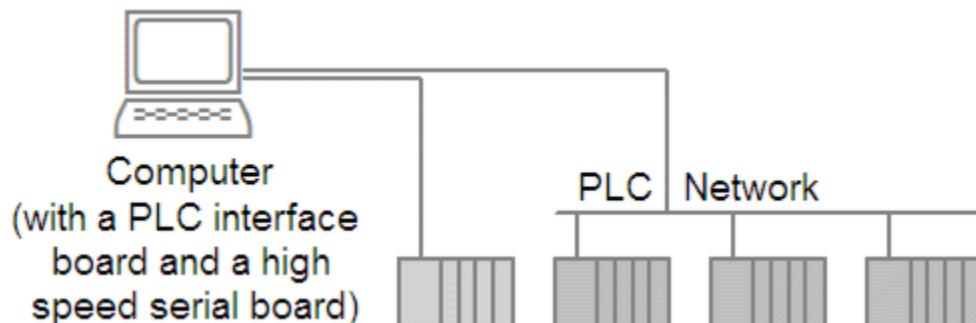
## Using a Proprietary Board

With some brands of I/O devices you can install a proprietary interface board in your computer, or intermediary software. This PLC interface board/software is supplied by the PLC manufacturer; you can connect it to a single PLC or a PLC network.



**Note:** With some PLCs, a high-speed serial board provides better performance than a PLC interface board when the system is connected to more than one PLC.

You can mix both PLC interface boards and high-speed serial boards in a single computer. You can, for example, connect a PLC network to a PLC interface board, and individual I/O Devices to a high-speed serial board.



There are many possible hardware arrangements for a Plant SCADA application. Plant SCADA is a flexible system and imposes few restraints on the type (or manufacturer) of I/O Devices that you can use, or on the way you connect them to the computer.

### To set up Plant SCADA to use a proprietary board/software:

If you are using a proprietary board/software (that is, supplied by the PLC manufacturer):

1. Install the board/software in your computer and set it up under Windows as per the accompanying

instructions. Use the latest driver from the manufacturer.

2. If possible, run diagnostics on the board before configuring Plant SCADA to check that the board works correctly.
3. Check that the **I/O Port** and **Interrupt** settings are correct.
4. Configure the **Boards** and **Ports** as instructed by the PLC-specific help.

## See Also

[Add a Board](#)

## Working With Device Drivers

The word ‘driver’ is often used when referring to communications. A driver is a component piece of software that implements a protocol or transport (or both). Drivers are modular, such that new drivers can be easily ‘plugged-in’ to existing systems.

Drivers enable Plant SCADA to communicate with the devices in your system. Every communication setup in Plant SCADA needs drivers to implement the protocol and transport. Often this means two separate drivers, but in some cases both are combined into one driver – termed a ‘board’ driver.

Plant SCADA has over 140 device drivers available, enabling communication with a vast array of production devices across several communication types. These include generic drivers that support industry-standard protocols such as OPC and Modbus.

The driver you will need to communicate with a particular device is determined by:

- the device itself
- the communication protocol the device supports
- the communication channel used to connect the device to the Plant SCADA I/O server.

To add a particular device to your Plant SCADA project, you will need to determine the driver necessary to support it. If the particular driver is not installed with your current version of Plant SCADA, you will have to obtain a driver pack and install it separately.

---

**Note:** With the release of Plant SCADA 2020 R2, a classification system for the driver portfolio was introduced to indicate the level of ongoing maintenance that customers can reasonably expect for each driver. In line with these classifications, the number of drivers included by default with the Plant SCADA installation media has been reduced to a set of ‘core’ drivers that have undergone recent updates. All other drivers continue to be available for download from the [Communication Drivers](#) page at the [AVEVA Knowledge and Support Center](#).

## See Also

[Determining Which Driver to Use](#)

[Installing a Driver Pack](#)

## Determining Which Driver to Use

To connect a device to Plant SCADA, you will initially need to determine which driver is necessary to support

communication with the I/O Server. In most cases, you can use one of the following methods:

- a native driver, engineered specifically for the device
- a generic driver, available for devices that support industry-standard protocols such as Modbus or DNP
- an OPC Server, with communication taking place via a third-party application.

Initially check if a native driver is available. To do this, go to the **Supported Devices** page in the **Driver Reference Help** and locate the device you are trying to connect to. The list is sorted alphabetically by manufacturer name.

If the device is not included on the list, consider the communication capabilities of the device to determine if it supports any industry-standard protocols, such as Modbus, DNP, COMLI or SNMP. If this is the case, you may be able to use one of the generic Plant SCADA drivers that support these standards.

Similarly, you can use OPC to connect a device to Plant SCADA, however this will require the purchase of a third party application to establish an OPC Server. If you take this approach, use an OPC Server application recommended by the manufacturer of the device you are trying to connect.

If these options do not provide an appropriate solution, you can contact Technical Support for this product.

---

**Note:** Check the **Communication Drivers** page at the [AVEVA Knowledge and Support Center](#) for information about driver updates and new driver pack releases.

---

## See Also

[Installing a Driver Pack](#)

### Installing a Driver Pack

In some cases, a required driver may not be included among those available on the Plant SCADA installer. This may occur if a driver is new, has been recently updated, or is not included with the installer due to its ongoing maintenance classification. If this is the case, you will need to obtain a Driver Pack and install it separately.

---

**Note:** Check the **Communication Drivers** page at the [AVEVA Knowledge and Support Center](#) for information about driver updates and new driver pack releases.

---

You need to install a driver pack on the computer that will connect to the target device. This computer needs to be configured as a Plant SCADA I/O server. It needs to be also be installed on any machines where the Plant SCADA project is compiled.

Before installing the driver pack, close your Plant SCADA runtime environment and Plant SCADA Studio. You need administrator permissions for the I/O server PC.

#### To install a driver pack:

1. Save the driver pack to an appropriate location on the I/O Server PC.
2. Double-click the EXE file to launch the installation.
3. Follow the instructions provided by the installation Wizard.
4. You are prompted to select the Plant SCADA installation folder. If you installed Plant SCADA in a different folder to the default, browse to that location. An alert message will alert you if you have selected the wrong folder.
5. Click **Finish** to complete the installation.

After you have installed a Driver Pack, the associated help will be automatically merged into the Driver Reference Help.

---

**Note:** If you are installing a driver pack on an older version of Plant SCADA that precedes the initial release of the driver, the supporting help may not appear automatically in the Driver Reference Help. If this is the case, you can locate the driver help in the Plant SCADA bin directory. The help file will be named <drivername>.chm.

---

## See Also

[The Driver Reference Help](#)

### The Driver Reference Help

Each driver, whether it is installed with Plant SCADA or as part of a driver pack, is supported by the **Driver Reference Help**.

#### To open the Driver Reference Help:

1. In the **Topology** activity, select **I/O Devices**.
2. On the Command Bar, click **Driver Reference Help**.

This is where you will find the specific reference information related to each driver. This information includes:

- The device(s) each driver supports
- The address used to identify and locate a device
- Any necessary device setup information
- The information necessary to configure a device in your Plant SCADA project
- The data types each driver supports
- Parameters you can use to customize a driver
- Driver-specific error messages.

You will need much of this information to establish communication with a device; some of it is more suited to advanced engineering tasks and troubleshooting.

If you familiarize yourself with the processes explained in [Configure I/O Device Communications](#), you will be instructed on how to use this information to configure devices within a Plant SCADA project.

## Using the Device Communications Wizard

You use the Device Communications Wizard to set up communications for a new I/O Device.

Based on your selections, the Device Communications Wizard provides default values and a setup tailored to your I/O device communications requirements.

#### To access the Device Communications Wizard:

1. In the **Topology** activity, select **I/O Devices**.
2. On the Command Bar, click **New Device**. The Device Communications Wizard is displayed.
3. Complete each page of the Wizard as required.

---

**Note:** Each combination of I/O device and protocol requires a unique configuration of **Boards**, **Ports**, and **I/O Device** settings. See the [The Driver Reference Help](#) for the required settings.

---

## See Also

[Device Communications Wizard - Introduction](#)

### Device Communications Wizard - Introduction

You will be asked to select an I/O Server, choose a name, and indicate the type of I/O Device(External, Memory, Disk).

From the list of available manufacturers you choose the manufacturer, model and communications method for the I/O device. If you are connecting external devices or using a proprietary board in your computer, you may be requested to nominate addresses and a communications port.

After completing your setup, the Summary Page summarizes the configuration of your I/O Device and/or internal boards. Click **Finish** to save the listed configuration, or click **Back** to change a previous selection.

## See Also

[Device Communications Wizard - Server Selection](#)

### Device Communications Wizard - Server Selection

Select **Use an existing I/O Server** (as defined in the current project), or **Create a new I/O Server**.

When you create a new I/O server, Plant SCADA automatically suggests the name **IOServer**. You can enter a different name if you want. The name you specify needs to be 16 characters or less. The fields are not case-sensitive and can contain alphanumeric characters (A-Z, a-z, 0-9). You can also use the underscore character ( \_ ).

---

**Note:** If you add a new I/O server, you will need to run the Setup Wizard on the designated computer before you attempt to run the project. This is necessary to enable the computer to function as an I/O server.

---

## See Also

[Device Communications Wizard - Device Selection](#)

### Device Communications Wizard - Device Selection

Enter the name of the new I/O device that you want to create, or accept the name **IODev** which Plant SCADA automatically suggests.

## See Also

[Device Communications Wizard - I/O Device Type](#)

## Device Communications Wizard - I/O Device Type

Select the type of I/O Device. You may choose from the following options:

- External I/O Device
- Persisted Memory I/O Device
- Disk I/O Device.

---

**Note:** Using a Persisted Memory I/O Device is recommended as an alternative to using a Disk I/O Device, as full synchronization is supported if a server becomes unavailable for a period of time. See [Using Persisted I/O Memory Mode](#) for more information.

---

## See Also

[Device Communications Wizard - I/O Device Communications Selection](#)

## Device Communications Wizard - I/O Device Communications Selection

From the list of available manufacturers, select the manufacturer, model, and communications method specific to the I/O device.

If a memory or disk I/O device has been selected, the Plant SCADA Generic Protocol is included at the top of the tree.

## See Also

[Device Communications Wizard - TCP/IP Address](#)

## Device Communications Wizard - TCP/IP Address

This page allows you to enter the IP address for an I/O device that uses TCP/IP protocol. This address is set on (or specified by) the I/O device.

In most cases, you will use a standard IPv4 address. For example, 192.1.2.34.

If you have selected a device that uses IPv6, you will need to enter a 128 bit address (or any acceptable abbreviation). For example, 2001:0db8:0000:0000:8a2e:0123:1234 or 2001:db8::8a2e:123:1234 (abbreviated).

The **Port** and **Use Protocol** (TCP or UDP) fields will be set to the default values for the I/O device. Change these fields only if necessary.

For details about addressing your specific I/O device, browse the [The Driver Reference Help](#).

## See Also

[Device Communications Wizard - I/O Device Address](#)

## Device Communications Wizard - I/O Device Address

Enter the address for the I/O device. What you enter in this field is determined by the type of I/O device (and protocol) used, as each has a different addressing strategy.

For details about addressing your specific I/O device, refer to the information in the [The Driver Reference Help](#) for your I/O device.

## See Also

[Device Communications Wizard - I/O Device Connection Schedule](#)

## Device Communications Wizard - I/O Device Connection Schedule

The Device Communications Wizard allows you to define the details of the communications schedule for your I/O device and indicate that your I/O device is remote by checking the PSTN box.

Options	Description
<b>Connect I/O Device to PSTN</b>	<p>Check this box to indicate that the I/O device is a dial-up I/O device (connected to a PSTN - Public Switched Telephone Network).</p> <p><b>Note:</b> Even if your I/O device is not connected via a modem, you need to still check this box to schedule communications (but leave the Phone number to dial and Caller ID fields blank).</p> <p>Once you have completed your I/O device setup using this Wizard, you need to go to the Ports form and change the Port number to the actual number of the COM port.</p> <p>You can choose to define the communication period in terms of <b>months, weeks, days, or hours, minutes, and seconds</b>. Alternatively, you can choose to communicate only at <b>startup</b> (persistent connection). Click a radio button to make your selection, then enter the start time and period as described below.</p>
<b>Synchronize at</b>	<p>The I/O Server will attempt to communicate with the I/O device at this time, and then at intervals as defined below. This time is merely a marker for Plant SCADA. If you run up your project after this time, the I/O Server won't wait until the next day to begin communicating. It will operate as if your project had been running since before the start time.</p>
<b>Repeat Every</b>	<p>The time between successive communication attempts. (See Examples below.)</p>

Options	Description
<b>Phone number to dial</b>	The telephone number that needs to be dialed to initiate contact with the I/O device.

**Note:** These values can also be set using the I/O devices view in the Topology activity.

## Examples

Examples are based on a **Synchronize at** time of 10:00:00:

- If you enter 12:00:00 in the **Repeat every** field, and start your project at 9 a.m., the I/O Server will communicate with the I/O device at 10 a.m., then once every 12 hours after that, i.e. 10 p.m., then again at 10 a.m. of the following day, etc.
- If you enter 12:00:00 in the **Repeat every** field, and start your project at 4 p.m., the I/O Server will communicate with the I/O Server at 10 p.m., then again at 10 a.m. of the following day, etc. It will assume that communications were established at 10 a.m., so it continue as if they had been - communicating once every 12 hours after 10 a.m.
- If you enter 3 days in the **Repeat Every**, and start your project at 9 a.m. on a Wednesday, the I/O Server will communicate with the I/O device at 10 a.m., then once every 3 days after that, i.e. 10 a.m. on the following Saturday, then at 10 a.m. on the following Tuesday, etc.
- If you enter the 6th of December in the **Repeat every**, and start your project during November, the I/O Server will communicate with the I/O device at 10 a.m. on December 6, then again on December 6 of the following year, etc.
- Select **On Startup** for a persistent connection. To disconnect a persistent connection, you need to call the [IODeviceControl\(\)](#) function with type 8.

## See Also

[Device Communications Wizard - Caller ID and Commands](#)

[Device Communications Wizard - Link to External Database](#)

### Device Communications Wizard - Caller ID and Commands

A **Caller ID** is a unique identifier which identifies a remote I/O device when it dials back to the I/O server. The caller ID can be any combination of alpha-numeric characters and/or the character '\_' (underscore).

This ID will only be used if the I/O device initiates the call to the I/O server. If the modem initiates the call, you need to set the caller ID on the modem.

**Note:** If you are multi-dropping off a single modem, use your I/O devices to issue the caller ID, not the modem. This is because using the modem to issue the ID will send the same ID no matter which I/O device the call is relevant to, which makes it difficult to identify the I/O device that triggered the call.

By using the I/O device to issue the ID, the I/O server will receive a unique caller ID for each I/O device. However, not every I/O device is capable of issuing caller IDs. If you are multi-dropping, use I/O devices that can issue caller IDs.

Option	Description
[Event Commands] On connect	Cicode to be executed once the modem is connected and the I/O Device has come online (that is, before any read or write requests are processed).
[Event Commands] On disconnect	Cicode to be executed before the connection to the I/O Device is terminated (and after read and write requests are processed).

## See Also

[Device Communications Wizard - Link to External Database](#)

### Device Communications Wizard - Link to External Database

This screen allows you to link to an external data source (see [Link an I/O Device to an External Data Source](#)).

Option	Description
Link I/O Device to an external tag database	<p>Determines whether or not you want to link the I/O device to an external data source. If you link to an external data source, Plant SCADA is updated with any changes made to the external data source when a refresh is performed.</p> <p>If you disconnect an existing link, you can choose to make a local copy of the tags in the data source or you can delete them from Plant SCADA's variable tags data source altogether.</p>
Database type	<p>The format of the data referenced by the external data source.</p>
External tag database	<p>Specify the location of the external database. This could either be a path and filename of the external data source for the I/O device, or the IP address/directory, computer name, or URL of a data server, etc. (for example "Work.CSV" or "127.0.0.1", "139.2.4.41\HMI_SCADA" or "http://www.abicom.com.au/main/scada" or "\\coms\data\scada").</p> <p>Depending on the database type selected, the browse button could:</p> <ul style="list-style-type: none"> <li>• Display a standard Windows file browse dialog to browse for the external database.</li> <li>• Display a modal dialog to browse in a tree view of</li> </ul>

Option	Description
	<p>servers on the network connected to the computer.</p> <ul style="list-style-type: none"> <li>Display the <a href="#">Unity Link Tag Import</a> dialog.</li> </ul>
<b>Connection string</b>	<p>Enter a connection string to provide connection details for the data source. This is similar to an ODBC connection string. For example:</p> <p>UserID = XXX; Password = YYY</p> <p>or</p> <p>ServerNode=111.2.3.44; Branch=XXX</p> <p>Not every data source requires a connection string.</p>
<b>Add prefix to externally linked tags</b>	<p>Check this box if you want to insert a prefix in front of the names of linked tags in your Variable.DBF.</p>
<b>Tag prefix</b>	<p>The prefix that will be inserted in front of the names of linked tags in your Variable.DBF (for this I/O device only). To change the prefix, delete it first, perform a manual refresh, then add the new prefix.</p>
<b>Automatic refresh of tags</b>	<p>Determines whether the linked tags in Plant SCADA's variable tags database will be updated when the external data source is changed (i.e. you manually change a field, etc.). This refresh will occur the first time you link to the data source, and then whenever you attempt to read the changed variables (for example you compile your project, display the variable using the Variable Tags form, or paste the tag, etc.).</p> <p>Without an automatic refresh, you will need to perform a manual refresh to update the linked tags in Plant SCADA.</p>
<b>Live Update</b>	<p>This field controls whether or not the linked tags in Plant SCADA and an external tag database will be synchronized if either database is changed. To enable live linking, choose <b>Yes</b> from the <b>Live Update</b> menu, and verify that the <b>Automatic refresh</b> check box is not selected. (Live Update and Automatic Refresh are mutually exclusive.)</p> <p>When Live Update is enabled and the Plant SCADA variable tag database is accessed (for example, during project compilation or when a drop-down list is populated), Plant SCADA queries the external tag database to determine if it has been modified. If so,</p>

Option	Description
	Plant SCADA merges the changes into the local variable tag database. Conversely, any changes made to the local tag variable database will be incorporated seamlessly into the external tag database.

## See Also

[Device Communications Wizard - Serial Device](#)

### Device Communications Wizard - Serial Device

Since your protocol is based on serial communications you need to select which port on your computer will be used for the I/O device.

The serial ports listed have been detected from your operating system registry. If you have correctly installed a proprietary serial board and the associated driver, the available port will be listed.

## See Also

[Device Communications Wizard - Summary](#)

### Device Communications Wizard - Summary

Summarizes the I/O device setup using the information you provided. The summary shows the Plant SCADA communications setup and the recommended configuration of your I/O device and/or internal boards.

## Retrieving Time-stamped Data from I/O Devices

Plant SCADA supports the retrieval of time-stamped data directly from field devices. This capability is enabled by the Driver Runtime Interface (DRI), a component that is used by some drivers, such as:

- BACNET
- DNPr
- IEC61850
- IEC870IP
- OPC
- OFSOPC
- OPCUA
- S7TCP.

**Note:** Check with Technical Support if new drivers are available that support the DRI.

The DRI allows a driver to push time-stamped data from field devices into a Plant SCADA system. This means the

following can be updated directly from devices:

- Time-stamped digital alarms (see [Add a Time Stamped Digital Alarm](#))
- Time-stamped analog alarms (see [Add a Time Stamped Analog Alarm](#))
- Double point status alarms (see [Add a Double Point Status Alarm](#))
- Event-based trends.

When these drivers start up they scan the system for time-stamped digital alarms, time-stamped analog alarms, double point status alarms and trend events. No configuration is necessary for alarms. For event-based trends, see [Configure Event Trends for a DRI Driver](#).

If the driver receives updated information for any detected tags, it will pass it directly on. Regular I/O server polling is no longer used.

---

**Note:** Non-timestamped alarms which are associated with a tag that uses the DRI mechanism will occasionally miss timestamped events. This will occur when multiple updates for the same tag occur within the period specified by [\[Alarm\]ScanTime](#).

This mechanism can work in tandem with the tag extension feature to enable access to field-generated timestamp and quality tag values (see [Tag Extensions](#)).

---

**Note:** Prior to version 7.20, time-stamped data was manually pushed into Plant SCADA using the Cicode functions `AlarmNotifyVarChange` and `TrnSetTable`. If you are upgrading a project to a version 7.20 system with a driver that uses the new DRI push mechanism, you will no longer need to use these functions.

## I/O Server Parameters

Events pushed from a DRI-supported driver can be buffered on an I/O server while an alarm or trend server is offline. The way this mechanism operates can be configured via the following parameters:

- `[IOServer]MaxEventsQueued`
- `[IOServer]MaxEventsDrop`
- `[IOServer]MaxTimeInQueueMs`

For more information, see [IOServer Parameters](#).

---

**Note:** Tags associated with devices using the DRI could be set to bad quality for reasons specific to an I/O point. You need to consider the implications this may have on your Cicode. See [Considering Tag Value Quality in Cicode](#).

## See Also

[Configure I/O Device Communications](#)

[Working With Device Drivers](#)

## Configure a Disk I/O Device

In addition to connecting to actual I/O devices, Plant SCADA supports the configuration of disk I/O devices, which exist only within your computer. The value of each variable in the disk I/O device is stored on your computer's hard disk.

---

**Note:** With the release of version 7.20, a feature called persistence could be applied to I/O devices in memory mode. Using a Persisted Memory I/O device is recommended as an alternative to using a disk I/O device, as full synchronization is supported if a server becomes unavailable for a period of time. See [Using Persisted I/O Memory Mode](#) for more information.

A disk I/O device is configured using the Device Communications Wizard. When specifying an I/O device type, select the **Disk I/O Device** option.

You are not required to configure a board or a port for a disk I/O device.

Once configured, disk I/O devices appear exactly as any other I/O device in your system, but are not connected to any field equipment. Disk I/O devices can contain any type of variable supported by Plant SCADA, and you can configure them to emulate any supported I/O device. You can also specify a generic protocol for a disk I/O device.

A disk I/O device is useful when the status of your plant needs to be restored after a planned or unplanned system shutdown. You can configure your system to continually update a disk I/O device with the subset of variables that defines the status of your plant. When you restart your system after a shutdown, Plant SCADA can restore this status immediately.

You can also use disk I/O devices for storing predefined data that needs to be recalled immediately when a process is necessary (for example, in a simple recipe system).

---

**Note:** If you create a RAM disk in the computer for the disk I/O device, you are not required to create or copy the disk file to the RAM disk. Plant SCADA automatically creates a disk file on startup. Avoid setting up your disk I/O device on a RAM disk if you want the data written to it to survive a power cycle of the servers.

If you need to edit the settings for a disk I/O device, see [Disk I/O Device Setup](#).

## See Also

[Redundant Disk I/O Devices](#)

[Disk I/O Device Error Messages](#)

## Disk I/O Device Setup

To set up communications with a device, follow the basic steps given in the I/O Device setup procedure below.

Sometimes you might need to edit the communications forms directly. They require the following specific information.

- You are not required to complete a Boards dialog box.
- You are not required to complete a Ports dialog box.
- Complete the I/O Devices dialog box as follows.

Perform the setup by entering the following information:

Text Box	Required information
<b>I/O Device name</b>	A name or your Disk I/O Device, for example: DISK_PLC. Each I/O Device needs to have a unique name in the Plant SCADA system.
<b>I/O Device number</b>	A unique number for the disk I/O Device (1-4095). Each I/O Device needs to have a unique number in the Plant SCADA system.
<b>I/O Device address</b>	<p>The path and filename of the disk file, for example: [RUN]:DSKDRV.DSK</p> <p>If you are using redundant disk I/O Devices, specify the path to both the primary file and the Standby file in the configuration of both disk I/O Devices.</p> <p>For example, if this is the primary disk I/O Device, enter:</p> <p><b>Primary_File, Standby_File</b></p> <p>If this is the Standby Disk I/O Device enter:</p> <p><b>Standby_File, Primary File</b></p> <p><b>Primary_File</b> is the name (and path) of the primary disk I/O Device file. You may use path substitution in this part of the field.</p> <p><b>Standby_File</b> is the name (and path) of the Standby Disk I/O Device file. You may use path substitution in this part of the field.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>• If the specified disk I/O device file is not found, Plant SCADA creates a new (empty) file with the specified filename.</li> <li>• To use redundant disk I/O devices, you need to use a network that supports peer-to-peer communication.</li> <li>• Disk files need to have write permission (on both primary and standby servers).</li> <li>• The frequency with which Plant SCADA writes data to the disk I/O device(s) is determined by the [DiskDrv]UpDateTime parameter.</li> </ul>
<b>I/O Device protocol</b>	<p>To use the Plant SCADA generic protocol, enter: <b>GENERIC</b></p> <p>- or -</p> <p>To select a specific protocol supported by Plant SCADA, see the I/O Devices online Help.</p>

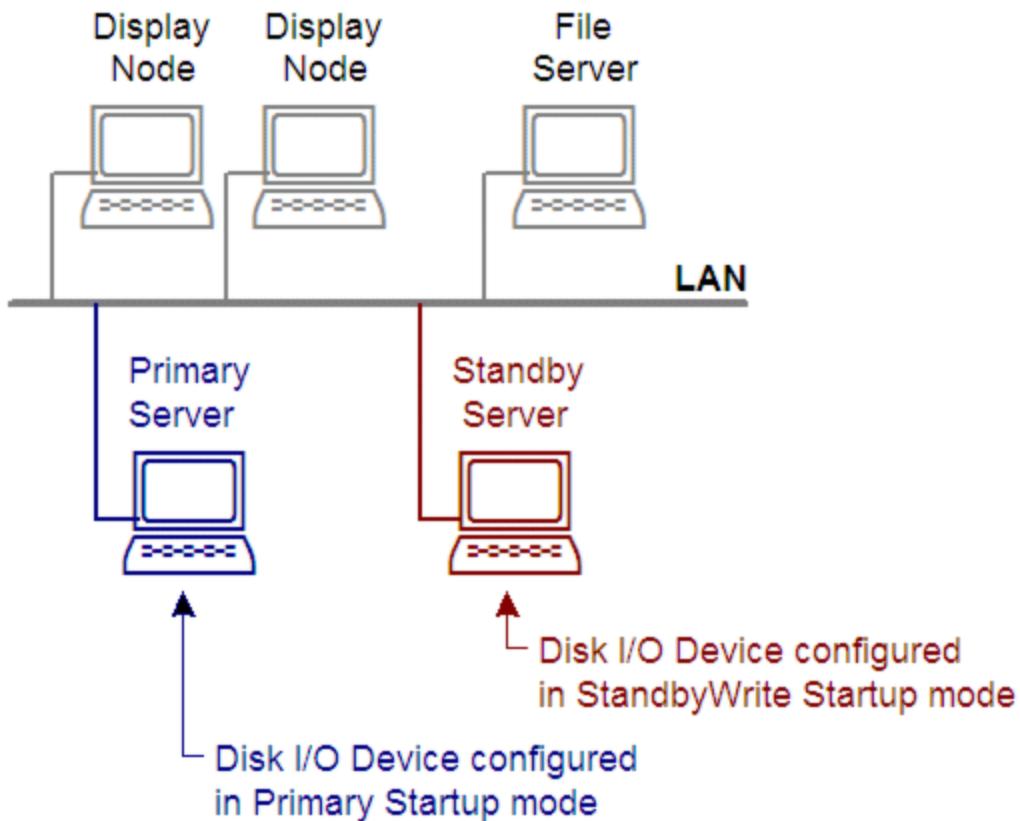
Text Box	Required information
I/O Device port name	You need to use: <b>DISKDRV</b>
I/O Device comment	Any useful comment.
I/O Device startup mode	<p>If not using redundant disk I/O Devices, leave this property blank.</p> <p>If you are using redundant disk I/O Devices, use either:</p> <p><b>Primary:</b> Enable immediate use of this disk I/O Device</p> <p><b>StandbyWrite:</b> This disk I/O Device will remain unused until the computer with the primary disk I/O device configured becomes inoperative. Every write request sent to the primary disk I/O Device is also sent to this disk I/O Device.</p> <p>To use StandbyWrite mode, you need to also configure an I/O disk device in the primary server. Both I/O Devices need to have the same I/O Device name and number.</p>
I/O Device Log Write, Log Read, Cache and Cache Time	Leave these properties blank.

## See Also

- [Redundant Disk I/O Devices](#)
- [Disk I/O Device Error Messages](#)

## Redundant Disk I/O Devices

If using a network, you can configure a redundant disk I/O device to minimize the potential for data loss (in the event the server becomes inoperative). This diagram illustrates the use of redundant disk I/O devices:



When the system is operating, Plant SCADA reads and writes runtime data to the disk I/O device configured in the primary server. It also writes runtime data to the disk I/O device configured in the standby server. (Plant SCADA maintains both disk I/O devices identically.)

If the primary server becomes inoperative, the Disk I/O device in the standby server is activated without interrupting the system. When the primary server becomes active, Plant SCADA automatically returns control to the primary server, and copies the Disk I/O Device from the standby server to the primary server. The disk I/O device in the standby server reverts to its standby role.

#### To define a redundant disk I/O device:

- Configure a new disk I/O device
- Select **StandbyWrite** for the Startup Mode.

For redundant disk I/O devices, you need to use Microsoft Networking (or another peer-to-peer network), and the hard disk of the standby server (the directory where the disk I/O device file is stored) needs to be shared. Use Windows Explorer to set the directory to shareable.

#### See Also

[Add an I/O Device](#)

[Communication Performance Considerations](#)

[Disk I/O Device Error Messages](#)

## Disk I/O Device Error Messages

The following error messages, listed in order, are specific to the Plant SCADA DISKDRV driver.

Error Codes	Message
48	Error Opening DiskFile
49	Error in DiskFile Header
50	DiskFile Header contains invalid variable
51	DiskFile out of memory error
52	Read Error in DiskFile
53	Write Error in DiskFile
54	Seek Error in DiskFile
55	Error creating queue
56	Error creating thread
57	Error creating event

## See Also

[Disk I/O Device Setup](#)

[Redundant Disk I/O Devices](#)

## Using Memory Mode

When configuring an I/O device, you have the option to set it to memory mode. This means that the I/O device will be created in memory and its values stored in memory at runtime.

Devices using memory mode are not connected to any hardware and write their values to a cache. The I/O device values can be read by many processes. The difference between a local variable and a device in Memory Mode is that an I/O device in Memory Mode will reside in the I/O Server's memory and will observe standard networking and redundancy rules of a standard I/O device.

Memory mode is useful when you are configuring a system for the first time, as you can design and test your system before connecting a physical I/O Device.

An I/O device with Memory set to TRUE or FALSE will behave differently in instances where Variable Tags share the same address.

For example:

Define a MODNET IODevice:

- 1 - INT1 (INT Tag)
- 2 - DIG1 (DIG Tag for bit0 of INT1)
- 3 - DIG2 (DIG Tag for bit1 of INT1)

At runtime the Display Client will show:

- IODevice Memory mode = FALSE, at runtime setting DIG1 = 1, DIG2 = 1, INT1 = 3, modifying DIG1 and/or DIG2 affects INT1.
- IODevice Memory mode= TRUE, at runtime setting DIG1 = 1, DIG2 = 1, INT = 0 or cached value, modifying DIG1 and/or DIG2 does not affect INT1.

In both instances the Override and Control Mode settings for each tag (INT1, DIG1, DIG2) did not affect the other tag. Setting INT1 in Override or Control Inhibit mode did not set DIG1 or DIG2 in Override or Control Inhibit mode.

As with local variables, values in an I/O device using only memory mode are not retained when you shut down. However, if you set the Persist field to TRUE in the I/O devices properties, their values will be retained. For more information on local variables, refer to Local Variables.

## See Also

[Using Persisted I/O Memory Mode](#)

### Using Persisted I/O Memory Mode

Since the release of version 7.20, a feature called persistence can be applied to I/O devices. When implemented, the tag cache for an I/O device is written to an XML file at a set period, allowing the last available data to be reloaded following a shutdown.

Persistence is enabled using the **Persist** property for an I/O devices.

You can also create a persisted memory I/O device using the Device Communications Wizard. When this type of device is selected, the wizard creates a new I/O device with the **Address** and **Port Name** properties left blank, **Memory** set to "TRUE", and **Persist** set to "TRUE". The **File Name** field is left empty to allow runtime to generate a default cache file name. This file will be located in the [DATA] directory.

### Migrating Disk I/O Devices

When applied to I/O Devices in memory mode, persistence provides an improved alternative to a disk I/O device, as synchronization is supported in scenarios where a server becomes unavailable for a period of time.

Many customers use disk I/O devices to provide system-wide global variable tags that are managed by I/O servers and are persisted to disk to maintain their latest values. Disk I/O devices take advantage of the standard I/O system redundancy features, such that, if one I/O server is unavailable, another can provide client(s) with tag values. They also perform a level of synchronization by using features such as standby write and by providing redundant paths to the persisted binary data files, so that, at startup of an I/O server, the latest value can be read into the system from the latest modified data file.

---

**Note:** Persisted memory PLCs have the ability to synchronize values between redundant devices after network connections are inoperative and regained. For this reason, it is recommended that data assigned to disk I/O devices be migrated to the new persisted memory I/O mode.

---

#### To migrate disk PLC devices to persisted I/O memory mode:

1. Perform the procedures listed in Upgrading to Plant SCADA 7.20, as is appropriate. Refer to the v7.20 documentation for more information.

2. For disk I/O devices, set the **Background Poll** property to TRUE and the **Persist** property to TRUE (the default setting). See [Add an I/O Device](#)) for more information.
3. Start the I/O server up and wait for a few minutes so that the background poll updates every tag.
4. Shutdown the I/O server.
5. On the I/O Device dialog, for disk I/O devices set Background Poll back to FALSE, set Memory to TRUE and clear out the Port Name field as it is not necessary for Memory Mode.
6. Restart the I/O server to finish the migration to persisted memory I/O mode, with your devices now containing your original values from the disk I/O device.

## See Also

[Using Memory Mode](#)

## Link an I/O Device to an External Data Source

When you add an I/O device to Plant SCADA, you can choose to link it to an external data source. This provides a link to the tag data that was saved when the I/O device was programmed.

**To link an I/O device to an external data source:**

1. In the **Topology** activity, select **I/O Devices**.
2. Scroll to the relevant I/O device.
3. Set the **Linked** property to **True** in the Grid Editor or the Property Grid.
4. Complete the remaining fields as required.
5. Click **Save**.

**To link an I/O device to an external data source using the Device Communications Wizard:**

1. In the **Topology** activity, select **I/O Devices**.
2. On the Command Bar, select **New Device**. The Device Communications Wizard is displayed.
3. Complete the wizard screens one by one, selecting the relevant I/O device, and so on.

When the **Link to External Database** screen appears, select the **Link I/O Device to an external tag database** check box and complete the remaining details.

## See Also

[Refresh the Tags for a Linked I/O Device](#)

[Import Variable Tags from an External Data Source](#)

## Refresh the Tags for a Linked I/O Device

You can manually refresh the tags for an I/O device that is linked to an external data source. This will load any changes that have occurred in the external data source into Plant SCADA Studio.

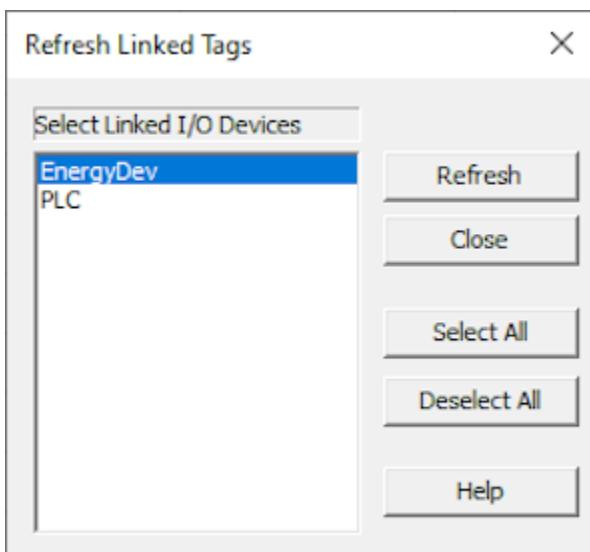
**Note:** If the **Automatic Refresh** property is set to True for a linked I/O device, a tag refresh will occur

---

automatically when you compile your project.

**To manually refresh linked tags:**

1. In the **Topology** activity, select **I/O Devices**.
2. Select **Refresh Tags** from the Command Bar. The **Refresh Linked Tags** dialog box will appear.



The **Select Linked I/O Devices** section lists every linked I/O device in your project (and included projects).

3. To refresh the tags for a device, select it and click **Refresh**.

This updates your project with the latest tag values for the selected I/O device. If you use Plant SCADA to modify any I/O device tags, your modifications will not be overwritten on refresh.

4. When you are finished, click **Close**.

## See Also

[Link an I/O Device to an External Data Source](#)

[Import Variable Tags from an External Data Source](#)

## Breaking the Link to the External Data Source

You can break the link to the external data source from the I/O Devices form or through the Device Communications Wizard.

If you break the link, you can choose to make a local copy of the tags or you can simply delete them altogether.

## See Also

[Link Tags to an External Data Source](#)

## Import Variable Tags from an External Data Source

Importing tags from an external data source allows you to program an I/O device and then import the tags

straight into Plant SCADA, where they are treated as regular tags. Plant SCADA will automatically create a variable tag records for every tag in the I/O device.

**Note:** Plant SCADA only supports the use of alphanumeric characters, the backslash and underscore characters (see Tag name syntax). Any other characters will be converted to underscores on import. This may result in Duplicate Tag error messages on compilation.

## About Importing Variable Tags

Like linking, importing tags is an I/O device specific operation: you import the tags for a particular I/O device. Unlike linking, however, imported tag records are not linked in any way to the tags in the external data source. Therefore, importing is typically a one-time operation. To update imported tags, you need to import them again.

**Note:** Tags will be imported into the project in which the selected I/O device is defined. This could be either the selected project or one of its included projects.

For most external data sources, the import process involves two steps. First you export the data from the I/O device to a format that Plant SCADA can read, then you import the database into Plant SCADA. However, tag data saved using Unity Pro can be read directly by Plant SCADA. This means that no export is necessary.

Some characteristics of the import are defined in the format files in the Config directory (with the extension .fmt). There is one format file for each database type, and each file specifies how external data is converted to Plant SCADA variable data. In general, imported tags are either added to the Plant SCADA variable database if the tag does not already exist, or updated if the tag exists. However, if a field is listed in the [EditableFields] section of the format file corresponding to the import database type:

- If the tag already exists, the Plant SCADA field for the tag will not be updated with the external value.
- If the tag does not yet exist, the Plant SCADA field for the tag will be updated with the external value.

For example, if a tag has 10 fields to be imported and five of them are listed in the [EditableFields] section of the format file, then if the tag already exists in Plant SCADA the tag will be updated with the external values for the five fields that are listed in [EditableFields]. If the tag does not already exist then the tag will be created with the external values for every 10 fields.

Fields marked as editable in this way can be changed by Plant SCADA. To avoid internal changes being lost, changes to these fields in the external database are ignored.

**Note:** If the external database supports to LiveUpdate and the **LiveUpdate Link** has been selected for the I/O device, existing Plant SCADA tags will be synchronized with the external database and the tags will be updated for the imported fields regardless of whether it is marked as an editable field. An I/O device configured as **Linked** or **Auto-refreshed Linked** will behave as in a normal import and will not update fields marked as editable.

When you import tags into Plant SCADA, you have two options for dealing with existing tags:

- Delete tags associated with that I/O device prior to the import.
- Update existing tags on import. Tags found in Plant SCADA and in the external data source are updated in Plant SCADA. Tags found in Plant SCADA but not in the external data source will remain untouched. New tags are appended.

If you import a data source that is already linked, every tag in the data source are duplicated. That is, you will have two copies of each tag: one local and one linked.

See Imported Tags for information about levels of support between Unity and Plant SCADA.

Some properties defined for the external tags will not be relevant to Plant SCADA. Also, some will not be in a

format that Plant SCADA can read. To define what information is copied to Plant SCADA's variable tags database and how this information is to be converted, you need to edit the I/O device's format file.

#### To import variable tags from an external database:

1. In the **Topology** activity, select **I/O Devices**.
2. On the Command Bar, select **Import Tags**.
3. Complete the **Import Variable Tags** dialog box.

See Import Variable Tags Properties below for a description of the information required in each field.

### Import Variable Tags Properties

Property	Description
<b>I/O Device</b>	<p><b>Note:</b> If an I/O device is linked to an external data source the Database Type, External Database, Connection String, and Tag Prefix fields will be greyed out.</p> <p>The Import Variable Tags dialog has the following fields:</p> <p>The I/O Device for which you are importing tags. Use the menu to select an I/O Device that has been defined using Plant SCADA.</p> <p><b>Note:</b> Tags will be imported into the project in which the selected I/O Device is defined. This could be either the project selected in the Explorer, or in one of its included projects.</p>
<b>Database type</b>	The format of the data referenced by the external data source.
<b>External database (128 Chars.)</b>	<p>References the source for the external database. Click the Browse button to either navigate to the file or to specify configuration options. The external database can be:</p> <ul style="list-style-type: none"><li>• An explicit path and file (for example, "C:\Data\Tags.csv")</li><li>• An IP address and node (for example, "127.0.0.1\HMI_Scada")</li><li>• A URL (for example "http://www.abicom.com.au/main/scada")</li><li>• A computer name (for example "\\\coms\data\scada").</li></ul> <p>Tag data can be read directly from a Unity SpeedLink or OPC Factory Server data sources.</p>

Property	Description
	Click the browse button to specify configuration options. See <a href="#">OPC Factory Server (OFS) Tag Import</a> or <a href="#">Unity Link Tag Import</a> .
<b>Connection string (128 Chars.)</b>	<p>Connection details for the data source. This is similar to an ODBC connection string. For example:</p> <p>UserID = XXX; Password = YYY or      ServerNode=111.2.3.44; Branch=XXX</p> <p>Only certain drivers require a connection string:</p> <ul style="list-style-type: none"> <li>• Unity SpeedLink (Static/Dynamic)</li> <li>• OPC</li> </ul> <p>Selecting one of these drivers and then browsing for the external database will automatically populate the Connection String field where necessary.</p>
<b>Add prefix to imported tags</b>	Select this box to insert a prefix in front of the names of imported tags in your <i>Variable.DBF</i> .
<b>Tag prefix</b>	The prefix (8 characters max.) that is inserted in front of the names of imported tags in the variable tags database.
<b>Delete I/O Device tags prior to import</b>	<p>Select this box to delete every one of the I/O Device's tags (from the variable tags database) before importing.</p> <p>If this check box is not selected, tags found in Plant SCADA and in the external data source are updated in Plant SCADA. Tags found in Plant SCADA but not in the external data source remain untouched. New tags are appended.</p>
<b>Purge deleted tags not found in data source</b>	Select this box to delete tags which have been removed from the external database. In other words, if a tag is still part of the I/O Device in your project, but not in the external database, it is deleted from your project.

## See Also

[OPC Factory Server \(OFS\) Tag Import Limitations](#)

## Unity Link Tag Import

The Unity Link Configuration dialog specifies the parameters used to generate alarm and trend tags. It has the following fields:

### I/O Device & Protocol Driver

The name of the destination I/O Device and its corresponding protocol. Unity SpeedLink will only import tags to an I/O Device using either the MODBUS, MODNET, MBPLUS, or UNITE.

### Unity Database Type

The import/export database driver type selected from the Import Variable Tags dialog.

### PLC Family

The family name of the PLC you are importing from. It can be either Premium or Quantum. This option is only valid for the Unity SpeedLink Static protocol.

### Unity File

Path of the Unity project file from which you want to import the tags. If the database type is Unity SpeedLink Dynamic, this is a \*.stu file. If the database type is Unity SpeedLink Static, this is a \*.xsy file.

### Unity Profile Enable

Enable this if the Unity database is restricted by user profile and a user name and password combination are needed to gain access. See the Access Security Management section of the Unity Pro online help for further information.

### Unity Username and Password

The username and password, if it is necessary, for the Unity database from which you are importing.

### DCOM Enable

Enable this if the database host is a DCOM server.

### DCOM Server Name

If enabled, the DCOM server that the client uses to hosts the database from which you are extracting tags.

### DCOM Username and Password

If enabled, the username and password for the DCOM server.

### Tag Gen Configuration Enable

Enable this to automatically generate tags from the variable Tags database. Tags are created using rules specified in the XML template file.

### Configured template is saved as

Filename of the [TagGen XML Template](#) file that specifies the rules for generating tags. Click **Configure** to select and configure the XML file. See [Additional Tag Generation Configuration](#).

### Log File Path

The path name for log files generated during the import or export process.

### Logging Level

The level of detail necessary in the import or export logs.

### Log Pool Size

Specifies the maximum number of log files for the given device.

### Validate

Click the validate button to check the Unity Link settings are correct. This button enables the OK button on the page.

### Additional Tag Generation Configuration

This dialog specifies the path of the XML template, the parameters specified in the template, and displays the template description (extracted from the desc attributes used in the template).

1. Click **Browse** to select the XML file to use.
2. Double click on a parameter in the parameters list to change its value.
3. Click **OK** to save this template into your project. The template will be copied into the project and, when validated, saved with a taggen\_import\_ioDeviceName.xml or taggen\_link\_ioDeviceName.xml filename.

**Note:** When you return to this dialog, the path name will point to the location of the template in the project, rather than the path of the original template file.

### OPC Data Access Server Tag Browser

The OPC Data Access Server Tag Browser dialog generates the strings that the OPC Data Access Server database driver requires in Plant SCADA.

The OPC Data Access Server Tag Browser dialog has the following fields:

#### Machine Name

The location of the OPC Server which can be an IP address or the network path server. Leave blank to select the local machine.

#### Database Tree Display

Displays the Servers on the network that are running the OPC Server application, and the hierarchical database node structure for each.

Select a group of tags to import from an available database. OPC data can be either a tree (hierarchical) or a flat structure. The Plant SCADA OPC driver supports access to both types of storage. If the data is in a tree structure, it can be accessed to the first branch level down from the root node. Deeper levels of branching may be supported in the future.

**Note:** The authentication values needs to be valid for read/write access, as assigned by the appropriate database administrator.

### OPC Factory Server (OFS) Tag Import

Plant SCADA can now import tags from the OPC Factory Server. For Database Type, choose the "Unity SpeedLink via OFS" option and then click the Browse button to specify the location of the database. In the OPC Data Access Server Parameters dialog that is displayed, you can browse the OFS servers installed on the PC and select tags to be imported into Plant SCADA.

There are some limitations and things to be aware of when importing tags from OFS. See [OPC Factory Server \(OFS\) Tag Import Limitations](#) for more information.

**Note:** There are two modes that can be used on the OPC Factory Server when importing tags; device mode and OFS simulator mode. If device mode is used, the time taken to import tags is increased significantly. This is because for each tag requested by Plant SCADA the OPC Factory Server needs to verify the item from the device.

---

OFS simulator mode is recommended.

**Machine Name**

The location of the OPC Factory Server which can be an IP address or a UNC path. Leave blank to select the local machine.

**Select Server and Branch to Import**

Displays the OPC Factory Servers on the network that are running the OPC Factory Server application, and the hierarchical database node structure for each.

Select a group of tags to import from an available database. OPC data can be either a tree (hierarchical) or a flat structure. The Plant SCADA OPC driver supports access to both types of storage. If the data is in a tree structure, it can be accessed to the first branch level down from the root node. Deeper levels of branching may be supported in the future.

---

**Note:** The authentication values needs to be valid for read/write access, as assigned by the appropriate database administrator.

**Tag Gen Configuration Enable**

Enable this to automatically generate tags from the variable Tags database. Tags are created using rules specified in the XML template file.

**Configured Template is Saved as**

Filename of the Tag Gen XML Template file that specifies the rules for generating tags. Click **Configure** to select and configure the XML file. See [Additional Tag Generation Configuration](#).

**Log File Path**

The path name for log files generated during the import or export process.

**Log Level**

The level of detail necessary in the import or export logs.

**Log Pool Size**

Specifies the maximum number of log files for the given device.

## OPC Factory Server (OFS) Tag Import Limitations

Please be aware of the following limitations when importing tags from an OFS database.

- For each imported tag, the Plant SCADA tag name will be the same as the name of the tag defined on the OFS except "." characters will be replaced by "\_" characters.
- Using OFS tag import to import tags from an OFS database may not succeed if Windows User Account Control (UAC) is enabled. UAC stops access to the external database. Either turn off UAC, or right click on the OFS and Unity Pro Windows shortcuts and select "Run as Administrator".
- To generate Alarm and Trend tags, a unique identifier can be set in the Custom and Comment field for Elementary Data Type (EDT) tags. For example, the string "VJA" is used to identify a tag that the user wants to use to generate a corresponding Digital Alarm Tag. Only the Comment field can be used for this feature on Derived Data Types (DDT) and Derived Function Block (DFB) tags.
- OFS built-in tags are not imported. They can be added in Plant SCADA manually.
- The length of the comment field in the Plant SCADA tag form is limited to 48 characters.
- Plant SCADA does not support DDT or DFB structure types. The OFS import will enumerate each element of this type of tag and import them as elementary tags.

- OFS tag import is only supported when the OFS links to the .STU and .XVM files.
- DFB components of type "ANY\_ARRAY\_xxxx" are not imported when the OFS links to a \*.xvm file. When a data dictionary is used, the data dictionary ignores comment and custom fields and therefore taggen cannot generate alarm and trend tags. If there are DFB components of type "ANY\_ARRAY\_xxxx", the whole DFB is not imported.

The following table shows data types supported by the tag import and TagGen when the OFS links to \*.stu or \*.xvm file.

	OFS links to *.stu file	OFS links to *.xvm file			
	<b>Tag Import</b>	<b>Tag Gen</b>		<b>Tag Import</b>	<b>Tag Gen</b>
		<b>Comment Field</b>	<b>Custom Field</b>		<b>Comment Field Only</b>
<b>Data Types</b>					
<b>Elementary Data Types (EDT)</b>					
BOOL	Yes	Yes	Yes	Yes	Yes
EBOOL	Yes	Yes	Yes	Yes	Yes
BYTE	Yes	Yes	Yes	Yes	Yes
INT	Yes	Yes	Yes	Yes	Yes
DINT	Yes	Yes	Yes	Yes	Yes
WORD	Yes	Yes	Yes	Yes	Yes
DWORD	Yes	Yes	Yes	Yes	Yes
UINT	Yes	Yes	Yes	Yes	Yes
UDINT	Yes	Yes	Yes	Yes	Yes
REAL	Yes	Yes	Yes	Yes	Yes
STRING	Yes	Yes	Yes	Yes	Yes
DATE	Yes	Yes	Yes	Yes	Yes

DT	Yes	Yes	Yes	Yes	Yes
TIME	Yes	Yes	Yes	Yes	Yes
TOD	Yes	Yes	Yes	Yes	Yes
<b>Derived Data Types (DDT)</b>					
EDT in DDT	Yes	Yes <sup>2</sup>	No	Yes	Yes <sup>3</sup>
Array in DDT	Yes	Yes <sup>2</sup>	No	Yes	No
<b>Arrays</b>					
Array of EDT 4	Yes	Yes	No	Yes	No
Array of DDT 4	No	No	No	Yes	Yes <sup>1</sup>
<b>Function Block Types (FBT)</b>					
Elementary Function Block (EFB)	Yes	Yes <sup>2</sup>	No	Yes	Yes <sup>3</sup>
Derived Function Block (DFB)	Yes	Yes <sup>2</sup>	No	Yes	Yes <sup>3</sup>
<b>Synchronization</b>					
Live Update	No	No	No	No	No
Automatic Refresh of tags	No	No	No	No	No

1. Only the BOOL type tags in the DDT can be used to generate Alarm and Trend tags when the OFS links to the \*.xvm file.
2. When OFS links to the stu file, the comment string defined in the tag instance will be taken, the comment string defined in the DDT/DFB definition will be ignored.
3. When OFS links to the xvm file, the comment string defined in the DDT/DFB definition will be taken and the comment defined in the tag instance will be ignored.
4. Array with negative index is not supported.

## TagGen XML Template

The TagGen template is an XML file that uses proprietary tags and attributes to specify the fields of input and output databases, and define filters and transformation rules that create tags from existing database fields.

The basic structure of the XML file is as follows:

```
<?xml version="1.0"?>
<template>
<param name="parameter">
</param>
<input name="variable" file="variable.dbf">
</input>
<output name="trend" file="trend.dbf" filter="">
</output>
</template>
```

This outline template specifies no parameters, takes input from the variable.dbf database and outputs the results to the trend.dbf database. The `<template>` tag is the root tag of the template document and need to be present.

Within the `<template>` tag are three sections:

- `<param>`

This is an optional section you can use to specify string constants that can be referred to in other sections of the template. For example:

```
<param name="parameters">
<string name="MyIODevice">DISK_PLCC</string>
</param>
```

In this example the variable **MyIODevice** is given the value DISK\_PLCC.

The `<param>` tag has a **name** attribute. You can use any name provided that it uniquely identifies the section within the XML file. This section name is used when referring to variables from other sections of the file. See [Referencing Variables](#).

Strings defined here are displayed in the [Additional Tag Generation Configuration](#) dialog when templates are selected. It is possible to define strings that are otherwise not used in the template to specify version or revision information in order to track template modifications.

- `<input>`

The `<input>` section specifies the name of the database and the fields that are used for processing into tags. There needs to be only one input section in the XML file. For example:

```
<input name="variable" file="variable.dbf">
<field name="name"></field>
<field name="type"></field>
<field name="unit" load="true">{parameter.IODevice}</field>
```

```
<field name="custom"></field>
<array name="taginfo">{ToProperty('{custom}', '=', ';')}</array>
<string name="alarminfo">{variable.taginfo[VJA]}</string>
<string name="trendinfo">{variable.taginfo[VJT]}</string>
</input>
```

This example identifies four fields from the variable.dbf database: **name**, **type**, **unit** and **custom**. These fields are used later in the file. The input section loads only rows where the **unit** field matches the value of the IODevice variable in the parameter section. This helps improve performance by avoiding the processing of irrelevant rows.

An array is created called taginfo that contains key value pairs based on the value of the custom field. If the custom field value was "VJA=Alarm;VJT=Trend", then an array is created containing the values taginfo[VJA] = "Alarm" and taginfo[VJT] = "Trend".

Two string variables are created based on the values of the taginfo array.

- <output>

The <output> section defines the output database, the processing to be done on the input fields and the respective output fields to be generated. There can be many output sections in the XML file. The output section can specify the same database file as the input section if necessary. For example:

```
<output name="digalm" file="digalm.dbf"
filter="'{variable.alarminfo}=VJA' AND
'{variable.type}=DIGITAL'"
<field name="tag">{variable.name}_ALARM</field>
<field name="name">{variable.name}_ALARM</field>
<field name="var_a" key="true">{variable.name}</field>
<field name="taggenlink" load="true">{parameter.IODevice}</field>
</output>
```

In this example, fields are written to the digalm.dbf database. Output processing only occurs for the current row if the input row relates to a digital alarm. That is, if the **alarminfo** variable from the input section has the value of "VJA" and the type field from the input database is "DIGITAL. The **tag** and **name** fields are given the value **name\_ALARM**.

Curly braces are usually used for substitutions to replace pre-defined texts with tag properties. To use curly brace characters as normal characters, use double curly braces which will be exceptionally converted to singular ones to express curly braces. Round brackets are not regarded as special characters when used in tags other than <calculator> tag.

### Example

```
<output name="digalm" file="digalm.dbf" filter=" ( '{variable.alarminfo1}=VJA' OR
'{variable.alarminfo2}=VJA' ) AND '{variable.type}=DIGITAL'" desc="Generate digital alarm
tags from input digital variable tags">
<field name="tag">{variable.name}_ALARM</field>
<field name="name">{variable.name}_ALARM</field>
<field name="var_a" key="true">{variable.name}</field>
<field name="taggenlink" load="true">{parameter.IODevice}</field>
<field name="comment">{{Testing1}}</field>
</output>
```

Proprietary tags used in the template are described in [XML Template Tags](#). See [Sample XML Templates](#) for more comprehensive examples.

## Referencing Variables

Variables can be used to temporarily store information. Variables can be defined either globally in the `<param>` section of the XML file or locally in an `<input>` or `<output>` section.

Variables are defined using the `<string>` tag as shown in . For example:

```
<string name="MyIODevice">DISK_PLC</string>
```

To refer to a variable, use the following syntax:

```
{SectionName.VariableName}
```

For this example:

```
{parameters.MyIODevice}
```

The **SectionName** can be omitted if you are referencing a variable defined in the same section.

Variables are evaluated sequentially in the XML file. Variables in the `<param>` section are assigned first, followed by those in the `<input>` section, followed by variables in each `<output>` section in sequence. This means that:

- Variables in the `<param>` section can be referenced by the `<input>` section and every `<output>` section in the same template.
- Variables in the `<input>` section can be referenced by every `<output>` section.
- Variables in each `<output>` section can only be referenced by subsequent `<output>` sections.

## Referencing Arrays

To reference a member of an array by member name use:

```
{SectionName.ArrayName[MemberName]}
```

To reference a member of an array by matching a pattern use:

```
{SectionName.ArrayName(PatternString)}
```

See [Pattern Matching](#) for information on pattern matching wildcards.

## XML Template Tags

The proprietary tags used in TagGen XML files are listed below.

### `<template>`

This is the root tag in the XML file. Only have one `<template>` tag in each XML file.

Can contain:

`<param>`, `<input>`, `<output>`

Attributes:

#### `desc`

Description of the template section.

### `<param>`

This optional tag defines global parameters for the XML file. Here you can specify string constants to be used in

the rest of the file.

Can contain:

<string>

Attributes:

**name**

Name of the parameter section.

**desc**

Description of the parameter section.

### <input>

This defines the source database of the XML file, the fields to be imported and any input filtering that might be necessary. There needs to be only one input section defined.

Can contain:

<field>, <string>, <array>, <calculator>

Attributes:

**name**

Name of the input section.

**file**

Input database file name.

**desc**

Description of the input section.

### <output>

This defines the output database of the XML file, the tags to be generated and any transformations necessary to generate tag data from input variable data. A template needs to have at least one output section defined.

Can contain:

<field>, <string>, <array>, <calculator>

Attributes:

**name**

Name of an output section

**file**

Output database file name

**filter**

Output section filter, syntax:

**filter="variable=FilterString"**

<FilterString> is a string expression that can combine wildcards or wildcard strings with boolean operators (AND, OR and NOT), as well as grouped boolean terms using parentheses. For example, "L\*" will match if string beginning with the letter "L"; "L\* OR H\*" will match if string beginning with "L" or "H". See [Pattern Matching](#) for a list of wildcard operators.

**desc**

Description of the output section.

**<field>**

This tag specifies the database fields that will be processed in <input> and <output> sections.

When used in the input section it specifies the named fields for each record loaded from the input database. It is read only.

When used in an output section it specifies the named fields to write to for each generated tag record in the output database.

Attributes:

**name**

Case insensitive name of a database field.

**key**

Valid for output sections only, this boolean flag indicates whether this is a key field. This is useful for special handling in the output section. For example, you can synchronize between a new record set and an old record set by matching the key fields.

**load**

Valid for both input and output sections, this boolean flag indicates whether this is a load filter field. Load filter fields are used to filter records when loading data from the input database and output database.

**desc**

Description of the field element.

---

**Note:** Key and Load string expressions can contain a regular string, a variable reference, wild card operators, or a mixture of these. See [Pattern Matching](#) for a list of wildcard operators.

---

**<string>**

This tag defines a string constant that can be used in <param>, <input> and <output> sections.

Attributes:

**name**

Name of a variable string.

**desc**

Description of the string element.

---

**Note:** For Unity SpeedLink, the string "IODevice", when defined in the <param> section, refers to the project's IO Device. Do not assign it another value.

---

**<array>**

This tag defines a dynamic array of strings in <input> and <output> sections.

The set of strings depends on the fields of the record being processed and those strings that are generated by the ToProperty() and Split() commands. See [Built-In Functions](#) for more information.

Attributes:

**name**

Name of a variable string array.

**desc**

Description of the array element.

## <calculator>

This tag defines an expression variable in <input> and <output> sections.

The content of this tag is a mathematical expression used to specify a simple operation for data transformation.

The operands in the expression can be only numbers or variables with numeric values. The expression can use parentheses to change the precedence of the evaluation.

For example, the following calculator in an <input> section counts input records by incrementing its current value.

```
<calculator name="c1" initial="0">{c1} + 1</calculator>
```

Attributes:

**name**

Name of a calculator.

**initial**

Initial value of the calculator, zero by default.

**desc**

Description of the calculator element.

## Pattern Matching

The following wildcards can be used in strings.

Character	Description
%	Matches any string in the input data stream
?	Matches any single character in the data stream
%d	Matches any decimal integer (nnn... where n is 0-9) in the data stream
%e	Matches any octal number (0onnn... where n is 0-7) in the data stream
%h	Matches any hexadecimal number (0xnnn... where n is 0-9, A-F or a-f) in the data stream
%s	Matches any string in the data stream (same as *)
{	Begin a token string. The characters between { and } in the Input Pattern (including regular and special characters) represent a "token string". The characters in the data stream that match this token string are a token, and can be referenced by the Output Data String, and written directly to the output database as a group
}	End of a token string

Character	Description
\	Treat the following character as a literal. For example, if a literal '*' character was expected in the input data stream, you would use \* to denote this. If a literal backslash \ is expected, use \\.

## Built-In Functions

There are three built-in functions that can be used to manipulate strings in the XML template. The syntax to call a built-in function as follows:

```
{<FunctionName>(<arg1>, <arg2>,...)}
```

The three built-in functions are:

**SubString(string, 'start', 'count')**

This function extracts a substring from the given source string starting at character **start** and ending after **count** characters.

```
<string name="AddrWord">{SubString('{'tagprefix'}','3','2')}</string>
```

**Split(string, delimiter)**

This function splits the source string into parts separated by the delimiter and stores them in a zero based string array. For example:

```
<array name="tagid">
{split('VJA;VJT', ';')}
</array>
```

produces tagid[0] = VJA, tagid[1] = VJT.

**ToProperty(string, separator, delimiter)**

This function splits the source string parts separated by the delimiter, and then further into key value pairs separated by the separator. For example:

```
<array name="taginfo">
{ToProperty('VJA=Alarm;VJT=Trend', '=', ';')}
</array>
```

This produces taginfo[VJA] = Alarm, taginfo[VJT] = Trend.

## Sample XML Templates

A sample template is provided to create analog or digital alarm tags and trend tags. The template shown below specifies the rules for generating Plant SCADA Alarm and Trend tags from a Unity SpeedLink device/database, for the import and/or synchronization of variable tags.

---

**Note:** You will need to modify this template to suit your particular requirements. Refer to [TagGen XML Template](#) for more information.

For each Unity Database variable tag with the text "VJA" in the custom field imported in to Plant SCADA of type "DIGITAL", this file will generate a linked Digital Alarm Tag within Plant SCADA (digalm.dbf) with the name <variablename>\_ALARM.

For each Unity Database variable tag with the text "VJA" in the custom field imported in to Plant SCADA not of

type "DIGITAL" and not of type "STRING", this file will generate a linked Analogue Alarm Tag within Plant SCADA (anaalm.dbf) with the name <variablename>\_ALARM.

For each Unity Database variable tag with the text "VJT" in the custom field imported in to Plant SCADA, this file will generate a linked Trend Tag within Plant SCADA (trend.dbf) with the name <variablename>\_TREND. This example file does not enter values for any of the other Trend tag properties.

```
<?xml version="1.0"?>
<template desc="Default SpeedLink Unity Pro TagGen template">

<param name="parameter">
<string name="IODevice" desc="SpeedLink necessary parameter, it will be set to the given I/O Device name by the system."></string>
</param>

<input name="variable" file="variable.dbf" desc="Load variable tags for the specified I/O Device">
<field name="name"></field>
<field name="type"></field>
<field name="unit" load="true">{parameter.IODriver}</field>
<field name="custom"></field>

<array name="taginfo">{ToProperty('{custom}', '=', ';')}</array>

<string name="alarminfo">{variable.taginfo[VJA]}</string>
<string name="trendinfo">{variable.taginfo[VJT]}</string>
</input>

<output name="digalm" file="digalm.dbf" filter="'{variable.alarminfo}=VJA' AND '{variable.type}=DIGITAL'" desc="Generate digital alarm tags from input digital variable tags">
<field name="tag">{variable.name}_ALARM</field>
<field name="name">{variable.name}_ALARM</field>
<field name="var_a" key="true">{variable.name}</field>
<field name="taggenlink" load="true">{parameter.IODriver}</field>
</output>

<output name="anaalm" file="anaalm.dbf" filter="'{variable.alarminfo}=VJA' AND '{variable.type}=NOT DIGITAL' AND '{variable.type}=NOT STRING'" desc="Generate analog alarms from input analog variable tags">
<field name="tag">{variable.name}_ALARM</field>
<field name="name">{variable.name}_ALARM</field>
<field name="var" key="true">{variable.name}</field>
<field name="taggenlink" load="true">{parameter.IODriver}</field>
</output>

<output name="trend" file="trend.dbf" filter="'{variable.trendinfo}=VJT'" desc="Generate trend tags from input variable tags">
<field name="name">{variable.name}_TREND</field>
<field name="expr" key="true">{variable.name}</field>
<field name="taggenlink" load="true">{parameter.IODriver}</field>
</output>

</template>
```

### Simple Wildcard Example

The following example shows how an empty string variable called "wildcard" can be used as a wildcard pattern

to load a set of data from a variable database. The string variable is substituted for a "%" wildcard character when the xml file is processed. The example below loads data where the names of variable tags contain the word "Tank".

```
<input name="variable" file="variable.dbf">
<string name="wildcard"></string>

<output name="anaalm" file="anaalm.dbf" filter="'{variable.alarminfo}=VJA' AND
'{variable.type}=INT'>
<field name="tag">{variable.name}_ALARM</field>
<field name="name">{variable.name}_ALARM</field>
<field name="var" key="true">{variable.name}</field>
<field name="taggenlink" load="true">{parameter.IODevice}</field>
<calculator name="highhh" >({variable.eng_full}-{variable.eng_zero})/
{parameter.Divisor} </calculator>
<field name="highhigh">{highhh}</field>
</output>
```

```
</template>
```

## Export Variable Tags to an External Data Source

The export feature allows you to export I/O device data to an external data source, specifying the destination and format of your choice (for example RSLOGIX driver). This file might then be imported into a third-party I/O device programming package database. Alternatively, you might use it as a backup.

**Note:** Exporting using the OPC driver is not supported.

The file to which you are exporting needs to already exist; otherwise the export will not work. You can choose to delete its contents before exporting, or you can leave it and create duplicate tags.

**Note:** The export tags feature is not yet supported by every database type. If you have existing links to any external data source, the linked tags will also be exported. As the structure of each type of external data source differs, some tag data might not be exported. This is determined by the format file for the I/O Device.

See [Exported Tags](#) for information about levels of support between Unity and Plant SCADA.

**Note:** Tag names greater than 32 Chars cannot be exported to Unity.

### To export variable tags to an external database:

1. In the **Topology** activity, select **I/O Devices**.
2. On the Command Bar, select **Export Tags**.
3. Complete the **Export Variable Tags** dialog box.

See the Export Variable Tags Properties below for a description of the information required in each field.

## Export Variable Tags Properties

Property	Description
<b>I/O Device</b>	The I/O device for which you are exporting tags. Use the menu to select an I/O device that has been defined using Plant SCADA.
<b>Database type</b>	The format of the data referenced by the external data source.
<b>External database</b> (128 Chars.)	A reference (128 characters) to the external data source to which your variable tags will be exported. This can be: <ul style="list-style-type: none"><li>• An explicit path and file (for example, "C:\Data\Tags.csv")</li><li>• An IP address and node (for example, "127.0.0.1\HMI_Scada")</li><li>• A URL (for example "http://www.abicom.com.au/main/scada")</li><li>• A computer name (for example "\coms\data\"</li></ul>

Property	Description
	<p>scada").</p> <p><b>Note:</b> This data source needs to exist before the export can be performed.</p>
<b>Connection string (128 Chars.)</b>	<p>Enter a connection string to provide connection details for the data source. This is similar to an ODBC connection string. For example:</p> <p>UserID = XXX; Password = YYY</p> <p>Not every data source requires a connection string.</p>
<b>Remove prefix from tags</b>	<p>Select this box to remove a known prefix from the front of the exported tag names.</p>
<b>Tag prefix</b>	<p>The prefix (8 characters max.) to be removed from exported tag names.</p>
<b>Delete existing tags</b>	<p>Select this box to delete any tags in the external database before exporting.</p> <p><b>Note:</b> For the Unity driver, existing tags are not deleted, even if you select this check box.</p>

## See Also

[External Data Source](#)

[Format File](#)

## External Data Source

When setting up an import, export, or link, you need to provide a data source and the format of the data.

Your data source can be entered as:

- An explicit path and file (for example, "C:\Data\Tags.csv")
- An IP address and node (for example, "127.0.0.1\HMI\_Scada")
- A URL (for example, "http://www.abicom.com.au/main/scada")
- A computer name (for example, "\coms\data\scada").

The database type field specifies the format of the external data source. When Plant SCADA attempts to read from this data source, it will use the mechanism specified by the database type. The supported database types are:

- OPC (OPC1 is not supported)
- CSV (comma-separated values)

- Concept Ver 2.1 ASCII file.

### To configure the external data source as a file

The example uses a CSV file, in this case an RSLOGIX database driver

In the Import/Export or Links dialog box, enter details as follows:

- **External database** C:\Data\Tags.csv
- **Database type** RSLOGIX Driver
- **Connection string** (leave blank)

### To configure the external data source using a specialized driver

---

**Note:** This example uses the supplied OPC driver.

---

In the Import/Export or Links dialog, enter details as follows:

- **External database:** The name of the OPC server process, for example, FactorySoft.InProc
- **Database type:** OPC
- **Connection string:** The parameters are **ServerNode** or **Branch**, though both are optional. Their use depends on the location of the OPC server and the scope of the necessary data. **ServerNode** can be an IP address or the network path to the server. For example:
  - ServerNode=127.0.0.1
  - ServerNode=\Server
  - ServerNode=www.server.comFor **Branch**, OPC data can be either a tree (hierarchical) or a flat structure. The OPC driver supports access to both types of storage. If the data is in a tree structure, it can be accessed to the first branch level down from the root node, by entering the name of the branch. For example:
  - Branch=device1Deeper levels of branching might be supported in the future.

## See Also

[Format File](#)

### Format File

The format file defines the import/export/linking rules. The file maps columns from the external data source format to the internal Plant SCADA database format. In other words, it determines what information is imported/exported/linked and how this information is modified during the operation.

The format file also provides the information to allow Plant SCADA to use the correct driver for accessing the external data source.

Some format files are provided with Plant SCADA; however, sometimes you might need to write or modify a format file using an editor such as Microsoft Notepad.

Following is an example of how format file rules work. (For more information, see [Format File Layout](#).)

1. Column 3 in the external data source needs to be copied into the "Name" column in the Plant SCADA's tag

database. (Plant SCADA names are restricted to alphanumeric characters (a to z, A to Z, and 0 to 9) and the underscore character "\_", but this is not typically a consideration in the format file; Plant SCADA does the conversion automatically). However, if Column 3 in the external data source happens to be blank, there is no need to copy this record across (it needs to be rejected).

2. Column 1 in the external data source needs to be copied straight into the "Addr" column in Plant SCADA's tag database, because they both mean exactly the same thing.
3. Columns 4, 5, 6, and 7 in the external data source need to be copied into the "Comment" column in Variable.DBF. (It is not uncommon for external data sources to split the comments across several fields). The fields need to be copied in that order, so that if the data in Column 4 in the external data source is "**Loop**", Column 5 is "**1**", Column 6 is "**Process**", and Column 7 is "**variable**", these fields are copied across in order, so that the "Comment" column in Variable.DBF reads "**Loop 1 Process variable**". This process is called 'concatenation'. (For the "Comment" column, Plant SCADA automatically adds a space between each field from the external data source.)
4. The data in Column 1 in the external data source determines what Plant SCADA needs to write in the "Type" column. However the data cannot be copied across directly, because it would not make sense to Plant SCADA. Instead, it needs to go through a conversion (or filtering) process. This conversion needs its own set of rules, such as:
  5. If Column 1 in the external data source is "BT%d:%d.%d" (where %d means "any decimal number"), Plant SCADA needs to write the string "DIGITAL" in the "Type" column.
  6. If Column 1 in the external data source is "F%d:%d/%d" (where %d means "any decimal number"), Plant SCADA needs to write the string "DIGITAL" in the "Type" column.
  7. If Column 1 in the external data source is "O:%e" (where %e means "any octal number"; that is, every digit from 0 to 7), Plant SCADA needs to leave the "Type" column blank. It still accepts the record (provided other columns pass any filtering tests) but it does not write anything in the "Type" column. The assumption is that Plant SCADA currently does not have (or does not need) a suitable corresponding type.
  8. If Column 1 in the external data source is "PD%d:%d.%d", Plant SCADA needs to write the string "REAL" in the "Type" column.
  9. If Column 1 in the external data source is "ST%d:%d", Plant SCADA needs to write the string "STRING" in the "Type" column.
  10. If there are no rules covering the contents of Column 1 in the external data source, Plant SCADA needs to reject the whole record and not copy it into the Plant SCADA database.

## See Also

[Format File Layout](#)

## Format File Layout

The format file is divided into sections:

- [\[General\] Section](#)
- [\[Columns\] Section](#)
- [\[ImportFilterMap\] and \[ExportFilterMap\] Sections](#).

Each section consists of a section header (the section name enclosed in square brackets (for example

"[my\_section]")) on a line by itself. This is followed by the body of the section, typically single line statements of the form:

```
"something = something_else -> something_else_again"
```

Any white space (or none at all) is acceptable around the "=" and the "->", but the whole statement needs to be on one line. Most statements within a format file follow this pattern, but in many cases there might be no "->" (the converter), or there might just be a converter without anything following it.

The following sections are necessary in every format file:

```
[General]  
[Columns]  
[ImportFilterMap]  
[ExportFilterMap]
```

Other sections might be necessary depending on the complexity of the conversion between Plant SCADA and the external data source. This is determined by the contents of [ImportFilterMap] and [ExportFilterMap].

Comments can also be added within or between sections. To do this, place a semicolon ";" as the first character on the line. The rest of the line is then considered a comment, and is ignored by Plant SCADA. For example:

```
; I am putting the [General] section here  
[General]
```

## [General] Section

The General section consists of 4 lines:

```
[General]  
Name=name  
Description= description  
DriverName= driver name  
DriverInst="a special string"
```

The **name** and the **description** are not currently used by Plant SCADA. Plant SCADA uses the **driver name** to load the correct driver for accessing the external data source. This driver might be one that is part of the Plant SCADA installation, or it might be a customized driver (including a driver that you have written yourself), for accessing a particular data source (which could be a protocol, type of hardware, server or file type). The driver needs to be an OLE DB-compliant driver.

The **special string** allows extra information to be passed to the driver. It is added to the connection string (in Plant SCADA Studio). So the connection string can be used for information that is likely to change often, and this **special string** can be used for more persistent information (such as the comma "," delimiter for a .CSV file). The main use of this string is as a delimiter for an input file. To specify that a comma "," is used by an input file as the delimiter, the following syntax would be used:

```
DriverInst="delimiter=,"
```

## [Columns] Section

The Columns section defines the format of the columns in the external data source. It is structured as follows:

```
[Columns]  
External column name 1 = column width -> data type  
External column name 2 = column width -> data type..  
External column name n = column width -> data type
```

The only restriction that Plant SCADA places on the data for **External Column name n** is that it needs to be unique within the section. For convenience, you can use the names that the external data source uses (such as "Description", "PLC\_id", "Iotype") or you can just make up names like "Column1", "Column2", etc. The order in which these entries appear needs to be the same as the order of the fields in the external data source. The names used for the External Data Source columns in the [ImportFilterMap] and [ExportFilterMap] sections needs to come from this list.

**Column width** is the number of characters in the field, and **data type** is the type of data for that column. Currently the only acceptable data type is "STRING".

### [ImportFilterMap] and [ExportFilterMap] Sections

The [ImportFilterMap] and [ExportFilterMap] sections have identical syntax and functionality, except that the [ImportFilterMap] describes how to convert data from the External data source on import, while the [ExportFilterMap] describes the opposite conversion.

These are the most complex sections in the format file. (The rest of the text will just focus on the [ImportFilterMap], as the [ExportFilterMap] follows basically the same logic.)

The [ImportFilterMap] is structured as follows:

```
[ImportFilterMap]
Import Rule 1 = External column name 1 -> Citect Column i
Import Rule 2 = External column name m -> Citect Column j..
Import Rule nn = External column name n -> Citect Column k
```

The values in **Import Rule nn** can be any name strings, but they needs to be unique within the section.

Therefore, for convenience, you might want to use names like "ImportRule1", "ImportRule2", "Mapping1", "Filter1" etc., or you might want something that is descriptive of the conversion involved, such as "Description\_to\_comment".

The name used for **External column name n** needs to be identical to a name that appears in **External column name n** in the [Columns] section above.

The name used for **Citect Column k** needs to be the same as one of the columns in the Plant SCADA internal tags database, such as "Name", "Type", "Addr", "Comment" etc.

Thus the values for the external column and the Plant SCADA column provide information on how to transfer data from the external column to the Plant SCADA column during import.

For example:

```
ImportRule1 = Description -> Comment
```

This indicates that there is a relationship between the data in the "Description" field in the external data source and the data that needs to go into the "Comment" field in the Plant SCADA database.

The name that you use for **Import Rule nn** might be the same as the name of another optional section in the format file: here, the extra section provides Plant SCADA with more information. In the simplest case, if there is no section with that name in the format file, the rule simply states that the data in **External column name n** is to be copied directly into **Citect Column k** without modification or filtering.

So if the "Description" column in the external database contains "Truck Position 1" and the above entry appears in the [ImportFilterMap] section, and there is no section called [ImportRule1], then after the import, the "Comment" column in the Plant SCADA database will contain the string "Truck Position 1".

## Concatenation

To concatenate fields from the external database into one field in the Plant SCADA database, add separate entries to the [ImportFilterMap] section. Each section needs to contain the name of a relevant external column and the name of the destination column in Plant SCADA. The entries need to appear in the order in which the fields are to be concatenated.

So, if the external data source has a field called "IOdev" containing the value "M", and another field called "IOaddr" containing the value "61", and you want to join them together so that the value "M61" is imported into the Plant SCADA "Addr" field, this is how it would be done:

```
[ImportFilterMap]  
Addr1= IOdev -> Addr  
Addr2= IOaddr -> Addr
```

Here, you need to verify that there are no sections in the format file called [Addr1] or [Addr2], unless you need some filtering or conversion.

## See Also

[Field Conversion](#)

## Field Conversion

To modify mapped data or to apply filtering (to reject certain records), create a new section using the name of the relevant line from the mapping section. For example, if you have the following mapping section:

```
[ImportFilterMap]  
Test1_to_type = Test1 -> Type
```

and you want to convert the data from the Test1 column before importing it, somewhere in the file you need to have a section called [Test1\_to\_type] followed by the necessary conversion rule.

---

**Note:** The name needs to be from the import mapping section, not from the export mapping section. If you use a name from the export mapping section, the conversion applies to the export, not the import.

The basic format of this conversion/filtering section is as follows:

```
[Relevant mapping name]  
Filtering Rule 1 = External Pattern 1 -> Citect String 1  
..  
Filtering Rule m = External Pattern m  
..  
Filtering Rule n = External Pattern n -> Citect String n
```

The name used for **Filtering Rule n** has no intrinsic significance to Plant SCADA (except that it uses it as a key to locate the entry). The only restriction is that it needs to be unique within the section, so you can use whatever is convenient.

The value in **External Pattern n** is a combination of characters which Plant SCADA will look for in the external data source column. This pattern can be any combination of the following:

Character in format file	Matches what string in external data source
<specific text>	<specific text>
*	Any string.
?	Any single character.
%d	Any decimal integer (nnn. . . where n is 0-9).
%e	Any octal number (0nnnn. . . where n is 0-7).
%h	Any hexadecimal number (0xnnnn. . . where n is 0-9, A-F or a-f).
%s	Any string.
{	Begin a "token string". Any characters enclosed by { } in the Input Pattern (including regular and special characters) represent a token string. The characters in the data stream that match a token string are referenced by the Output Data String and written directly to the output database as a group.
}	End of a token string.
\	Treat the following character as a literal. For example, if a literal * character was expected in the input data stream, you would use \* to denote this. If a literal backslash \ is expected, use \\.

Any other characters need to literally match the same character.

If External Pattern n is found in the external column, Citect String n is written to the relevant column in Plant SCADA (as per the mapping).

In addition, two special characters can appear in the output data string:

Character in output string	Meaning
\$	The pattern \$n (where n is any integer) is replaced in the output data stream by the nth "token"; a token is a matching sequence of characters enclosed by { } in the input pattern. (An alert message will result if \$ not followed by a token number.)
!REJECT!	This sequence needs to appear by itself in the output data pattern. The whole record is rejected. As the record had already been matched to the input pattern, no further rules are checked.
\	Treat the following character as a literal. This would be used if a literal \$ sign was necessary (use \\$) or if

Character in output string	Meaning
	another digit immediately follows. For example, if the string "3August2001" needs to immediately follow the token, use "\$1\3August2001".
\ (at end of line)	Insert a literal space ' ' character at the end of the output line. Without this provision, the system could not distinguish between the end of the input line (which is likely to be followed by characters, such as spaces, that Windows will ignore) and a space being necessary at the end of the output line.

Other characters are written literally to the output database.

Plant SCADA works through each filtering rule in the section, looking for a match. If a rule does not match, the next one is tried, then the next, and so on, until a match is found. If no match is found, the whole record is rejected; none of the data from any field is copied to Plant SCADA.

For example, to convert the string "FLOAT" in the external data source to "DIGITAL" in Plant SCADA, you could use the following entry:

```
[ImportFilterMap]
Test1_to_type = Test1 -> Type
..
[Test1_to_type]
Rule1 = FLOAT -> DIGITAL
..
```

For a more complex example, let us assume that the external data source has a column called "Tag" which is equivalent to the "Name" field in Plant SCADA. Let us also assume that the external database has no direct equivalent of Plant SCADA's "Type" field, yet Plant SCADA needs this field to be filled in. We need to use the "Tag" field to decide what goes into the "Type" field of the Plant SCADA database.

If the "Tag" column in the external data source has the value "I:060/07", we have determined that we write the string "DIGITAL" into Plant SCADA's "Type" field. In fact, if that field has "I:" followed by any octal value, followed by a slash "/", followed by any octal value, we want the string "DIGITAL" to appear in our "Type" field. How do we express this in the format file?

Firstly, there are two sets of relationships to consider, one connecting the "Tag" field in the external data source to the "Name" field in Plant SCADA, and the other connecting it to the "Type" field in Plant SCADA. So we need two "mappings" (entries) in the [ImportFilterMap] section:

```
[ImportFilterMap]
Tag_to_Name = Tag -> Name
Tag_to_Type = Tag -> Type
..
```

As we want the data in the "Tag" field to be copied directly into the "Name" field, we do this by not having a [Tag\_to\_Name] section anywhere in the format file.

But because we are not copying directly from the "Tag" field to the "Type" field, but are just using the data to decide what goes into the "Type" field, we need a [Tag\_to\_Type] section.

Recall the desired outcome: If the "tag" field has "I:" followed by any octal value, followed by a slash "/", followed by any octal value, we want the string "DIGITAL" to appear in our "Type" field.

We express this in the format file as follows:

```
[Tag_to_Type]
Rule1 = I:%e/%e -> DIGITAL
..
```

This will match "I:060/07" or "I:0453/02343445602" (and cause the string "DIGITAL" to be written to Plant SCADA's Type field), but will not match "I:060/98" or "I:054".

To give a few examples of how the wild-card characters (%s, \* and ?) might be used, the pattern "HE%sLD" or "HE\*LD" in the format file would match "HELLO WORLD" or "HE IS VERY BOLD" in the external data source. The pattern "HE??????LD" would match "HELLO WORLD" but not "HE IS BALD", as each question mark "?" needs to match exactly one character.

Plant SCADA will also handle multiple wildcard patterns, such as "%s/%s:%s".

For an example more useful than "Hello World", imagine that we need to copy the data straight across without modification, but we want to verify that no blank fields are copied across. The pattern "?%s" or "?\*" will match any string that has at least one character, but will not match a blank.

Sometimes only part of the input stream is necessary in the output, or the input might need to be split up into different output columns. In these situations "tokens" are useful.

In this example of an export situation, the "Addr" field in the Plant SCADA database needs to be split among two fields in the external database: the "IOdev" (whose value is always "D" or "M"); and "IOaddr" (whose value is a decimal integer of no more than 3 digits). Values in the "Addr" field of the Plant SCADA database are strings such as "D62", "M546", etc.

This situation could be solved by concatenation, i.e. using one mapping to write to the IOdev field, and three other separate mappings to copy each digit separately into the IOaddr field of the external database. But this would be complex and in some situations would not work.

It is better to use a token to address the situation:

```
[ExportFilterMap]
..
Addr2IOdev = Addr -> IOdev
Addr2IOaddr = Addr -> IOaddr
..
[Addr2IOdev]
D = D* -> D
M = M* -> M
AnythingElse = * ->
..
[Addr2IOaddr]
Rule = ?{%-d} -> $1
```

In the [Addr2IOaddr] section, the {%-d} is the token string, and as it is the first (and only) token appearing in the rule, \$1 is used to reference it on the output stream side. So if the "Addr" field of the Plant SCADA database contains "D483", "D" is written to the "IOdev" field of the external data sink, and "483" (the token) to the "IOaddr" field.

Here is another example illustrating the use of multiple tokens. Suppose we need to: convert period characters (.) to colons (:); remove the first two characters (which are blank); and remove any unnecessary characters from the data we are expecting; that is, convert "..BJ6452.78....." to "BJ6542:78". This can be achieved by using the following rule:

```
Rule = ??{*%-d}.{%-d}* -> $1:$2
```

At this point, we introduce another feature of the format file. If you use the following rule:

```
[Relevant mapping name]  
Filtering Rule = External pattern
```

i.e., without "-> Citect String" included, Plant SCADA interprets this as "check that the string matches the External Pattern, if it does, copy it across unchanged".

If this rule is used:

```
Filtering Rule = External pattern ->
```

i.e., without "Citect String", it would mean: "If the string pattern matches then accept the record but copy a NULL string to the Plant SCADA database."

Using the above example again, we can add the restriction that any records with no data (i.e. a blank or NULL string) in the Tag field of the external data source will not be imported into Plant SCADA. We would add a [Tag\_to\_Name] section, and would have just one rule: that we accept everything except for a blank.

```
[Tag_to_Name]  
RejectBlanks = ?*  
..
```

Recall that Plant SCADA checks the pattern in each filter rule sequentially until a pattern that matches the string is found in the external data source. With this in mind, a huge range of conversions and filterings are possible by ordering the rules correctly and, in some cases, by making use of concatenation.

For instance, if certain string types need to be converted but others need to be copied unmodified to Plant SCADA, you could have a section with a set of rules at the top, followed by a final rule to let everything else through unmodified.

```
[Tag_to_Name]  
Rule n = ..... -> .....,  
..  
LetEverythingElseThru = %s
```

A single %s or \*, without anything else, matches anything and everything, including blanks.

For an example of how to reject a particular string or pattern, let's suppose we want to reject any tags starting with "DFILE"(another real-life example). We would simply use the following:

```
[Tag_to_Name]  
Rule1 = DFILE* -> !REJECT!  
..  
LetEverythingElseThru = %s
```

Clearly, it is pointless having the !REJECT! rule not followed by other rules concerning patterns that you do want to accept, as anything that does not match an input pattern is rejected. The logic behind the order that the rules appear can become particularly important when using a !REJECT! rule. You would typically have any reject rule(s) as the first rule(s) in the mapping. There would not be any point in putting a !REJECT! rule as the last rule in the mapping.

!REJECT! rules can also be useful where some text file generated by another system contains some sort of header lines that are not wanted, but the rest of the data is necessary.

## See Also

[Recognizing Format Files](#)

## Recognizing Format Files

Your format file needs to be saved to the config folder of the Plant SCADA User and Data folder selected during installation.

For Plant SCADA to import, export, or link, it needs to know the name of the format file and the name of the driver that will access the external data source. It also needs the text of the string that will appear in the **Database type** field of the Import or Export dialog box. This information is stored in another file, called tagdrv.ini, in the same directory.

The format of tagdrv.ini is simple and based on the odbc.ini format. When it is installed it already has the necessary information for the format files and drivers that come shipped with Plant SCADA. You just need to copy the same layout for your new format file, and the driver that you are using.

The tagdrv.ini file has several sections. The first section is the [External data sources] section, with the following general layout:

```
[External data sources]
```

Section name 1 = the name you want to appear in the import/export/ link menu for entry 1

Section name 2 = the name you want to appear in the import/export/ link menu for entry 2..

Section name nn = the name you want to appear in the import/export menu for entry nn

Each entry in this section refers to a combination of format file and driver necessary for a particular import, export, or link operation.

The text on the left of the "=" sign needs to refer to the name of another section which needs to appear in tagdrv.ini.

The text on the right of the "=" sign is exactly what will appear in the menu under "**Database type**" for importing, exporting or refreshing variable tags.

The other sections each refer to a type of import or export described in the [External data sources] section, and give details about the format file and driver pair. The general layout of these sections is as follows:

```
[Section name matching an entry in [External data sources] ]
```

driverid = Driver ID

datastring = The name of the format file

Currently the Driver ID is always CiTrans.

So if we assume that the version of Plant SCADA you are installing contains five format files, there would be five sections in tagdrv.ini, as shown in the following example:

```
; This file contains the driver name, driver prog id, and format file mappings
; The format file needs to reside in the BIN directory
```

```
[External data sources]
```

CSV = CSV Driver

Concept ver 2.1 Ascii = Concept Ver 2.1 ASCII file

OPC = OPC

UnityProDynamic = Unity SpeedLink Dynamic

UnityProStatic = Unity SpeedLink Static

```
[CSV]
```

driverid = CiTrans

datastring = csv.fmt

```
[Concept ver 2.1 Ascii]
```

```
driverid = CiTrans
datastring = concept ver 2_1(fmt

[OPC]
driverid = CiTrans
datastring = OPC(fmt

[UnityProDynamic]
driverid = CiTrans
datastring = UnityProDynamic(fmt
LiveUpdate = TRUE

[UnityProStatic]
driverid = CiTrans
datastring = UnityProStatic(fmt
LiveUpdate = FALSE
```

This adds 5 entries to the database type drop down menu: CSV, ASCII, OPC, Unity Pro Dynamic and Unity Pro Static.

The 5 entries in the menu match the strings on the right of the "=" sign in the [External data sources] section.

If you add another format file, you will also need to add a matching entry to tagdriv.ini. For example, if you add a new format file for a Simatic data source, you need to add a line similar to this to the [External data sources] section:

```
SIMATIC = Siemens SIMATIC Driver
```

You need to also add the following section to the bottom of the file:

```
[SIMATIC]
driverid = CiTrans
datastring = SIMATIC(fmt
```

Save the file, restart Plant SCADA Studio, and "Siemens SIMATIC Driver" appears in the menu.

Selecting this entry causes the format file in the datastring entry under the [SIMATIC] section to be used for the import, export, or link; that is, simatic(fmt.

## Communicating with Remote Devices via Modems

A dial-up remote I/O device is one which is connected to Plant SCADA through a PSTN (Public Switched Telephone Network), and is accessible through pre-selected and pre-configured modems. Once connected, Plant SCADA can write to, and read from, dial-up remote I/O devices just as it does with any other local or remote I/O device.

Communications can be:

- **On request:** initiated by Plant SCADA using IODeviceControl() or by the remote I/O device (for instance, to report an alarm condition).
- **Periodic:** for instance, to transfer the logged events for a period.
- **Persistent:** for instance, to monitor and control the water level at a remote dam.

The only limiting factor would be an inability to connect a modem to an I/O device due to incompatible communications capabilities. Many I/O devices have fixed settings and can only communicate at a pre-set rate determined by the manufacturer. If a modem cannot match these settings, communication cannot be established.

To make communications setup easier, you can connect dial-up remote I/O devices with identical communications to the same modem and port. Where I/O devices are connected to the same modem, Plant SCADA can communicate with each I/O device one after the other using the same phone connection, rather than hanging up and re-dialing. This reduces the number of necessary telephone calls and increases the speed and efficiency of communications.

## See Also

- [Modems at the I/O Server](#)
- [Modems at the I/O Device](#)
- [Configure a Modem](#)
- [I/O Device Constraints for Multi-dropping](#)
- [Configuring Multidrop Remote I/O Devices](#)
- [I/O Server Redundancy for Dial-up Remote I/O Devices](#)
- [Troubleshooting Dial-up Remote I/O Device Communications](#)

## Modems at the I/O Server

To decide how many modems to use at the I/O server end, decide what function each modem will perform. A single modem can do any one of the following functions:

<b>Dial-out</b>	Makes calls to remote I/O devices in response to a Plant SCADA request; for example, scheduled, event-based, operator request, and so on. Also returns calls from remote I/O devices.
<b>Dial-in</b>	Only receives calls from remote I/O devices, identifies the caller, then hangs up immediately so it can receive other calls. Plant SCADA then returns the call using a dial-back or dial-out modem.
<b>Dial-back</b>	Only returns calls from remote I/O devices.
<b>Dial-in and dial-out</b>	Receives calls from remote I/O devices and makes scheduled calls to remote I/O devices.
<b>Dial-in and dial-back</b>	Receives and returns calls from remote I/O devices.

Your modem setup depends on your system requirements. When making your decision, consider the following guidelines:

- If you need to communicate with multiple remote I/O devices at once, you will need a separate modem for each I/O device. Otherwise you'll have to wait as I/O devices are contacted one after the other, and incoming calls may be held up.

- If you have scheduled requests to I/O devices and you also need to urgently return calls from remote I/O devices that dial in, you will need at least one modem for each of these functions. For example, if you have a large number of remote I/O devices and you require fast responses by Plant SCADA, provide an additional modem at the I/O server. This would reduce the chances of it being engaged when an I/O device dials in (say, with an urgent alarm).
- In a big system with many remote I/O devices or a system where calls from remote I/O devices can be time sensitive, it's a good idea to dedicate at least one modem to dial-back. This will give you quick responses to dial-In calls (from remote I/O devices). It also means your dial-out schedules won't be disrupted (if you use the same modem for returning calls and scheduled calls, the scheduled calls are forced to wait until the dial-back call is complete).

## See Also

[Modems at the I/O Device](#)

[Example Configurations for Modems at the I/O Server](#)

### Modems at the I/O Device

You can have multiple I/O Devices connected to a single modem if they have the same communication requirements (phone number, baud rate, data bits, stop bits, and parity). A separate port and modem needs to be used for each remote I/O Device with different communication requirements.

When deciding how many modems to use, consider the following:

- Once an I/O Device has been contacted, the connection can be retained for other I/O Devices in the group. If the I/O Server needs to request data from another I/O Device with the same communication details, it will wait until the current request has been completed, then it will use the same connection to make the second request. You can configure your modem to initiate telephone calls (call Plant SCADA), and/or receive telephone calls. This is done independently of Plant SCADA.

---

**Note:** Wherever your modem is, you need to verify that its **Data bits**, **Parity**, **Stop bits**, and **Serial Port Speed** settings are compatible with the remote I/O Device as defined by the device's manufacturer or your communications link will not work.

---

## See Also

[Modems at the I/O Server](#)

### Configure a Modem

Each modem connected to a Plant SCADA I/O server computer needs to be configured within Windows™ and then in Plant SCADA.

#### To set up a modem in Windows:

1. Open Windows **Control Panel | Phone and Modem**.
2. Select the **Modems** tab, and click **Add** to launch the Install New Modem wizard.
3. Check the box labeled **Don't detect my modem; I will select it from a list**, then click **Next**.

4. Select **Standard Modem Types** in the list of manufacturers.

**Note:** Do not select a brand name modem from the Manufacturers list, even if the name of the modem you're installing is included in the list. Do not click **Have Disk**.

5. Select the **Standard xxxx bps modem** rate from the list of models to exactly match the bit per second rate of the I/O device that is going to be communicating via this modem. Check the device to determine the device communication rate. If you are still unsure, select the 9600 bps model. This can be changed later if necessary.
6. Do not click **Have Disk**. Click **Next**.
7. Select the **COMx Port** that the modem is connected to. Click **Next**.
8. Click **Finish**. Windows displays the modem in the list of modems on the Modems Properties form.
9. No option was given for the selection and setting of the Data bits, Parity, or Stop bits information. The Modems wizard automatically defaults to 8-none-1 for Standard Modem types. To change these settings to match the Data bits, Parity, Stop bits requirements of the remote I/O device, select a modem in the list, then click the **Properties** button.
10. Click the **Advanced** tab and click **Change Default Preferences**.
11. Click the **Advanced** tab at the next dialog to gain access to the Data bits, Parity, Stop bits settings for the modem.
12. Change the Data bits, Parity, and Stop bits settings using the drop-down options, so that they exactly match those being used by the remote I/O device and its remote modem. Don't change any advanced settings. (The default is Hardware flow control.)
13. Click **OK**. If a modem of the same rate is installed to the same port as an existing modem, Windows asks for confirmation that you want to install the same thing more than once. Click **Yes** to install a duplicate copy of the modem.
14. Preconfigure the modem(s) to be used at the remote dial-up I/O device(s). These will be used to test modem configuration settings in the next step.
15. With Plant SCADA not running, confirm that the local and remote modems will properly communicate with each other by using a terminal communications program such HyperTerminal or PhoneDialer (both supplied with Windows).

Once the modem(s) are set up and tested with proven communications in Windows, they can then be set up in Plant SCADA.

#### To add a modem to Plant SCADA:

1. In the **Topology** activity, select **Components & Mappings**.
  2. On the menu below the Command Bar, select **Modems**.
  3. Add a row to the Grid Editor.
  4. Type the required information in each column, or in the fields in the Property Grid (see below for a description of the fields).
- For a description of the properties, see below.
5. Click **Save**.

**Note:** Plant SCADA allows you to set up a maximum of 256 modems on the I/O server for communication with remote dial-up I/O Devices. Before it can successfully establish communication, any targeted remote I/O devices need to also be properly configured within Plant SCADA on the I/O server.

## Modem Properties

### General Properties

Property	Description
<b>Server Name</b>	The name of the I/O server to which the modem is attached (maximum 16 characters)
<b>Modem Name</b>	The name of the modem you are configuring (as it appears in the Windows™ Control Panel   Phone and Modems). 64 characters maximum.
<b>Use this modem to make outgoing calls</b>	Determines whether this modem will be used to initiate calls from the I/O server to a dial-up remote I/O device. (Dial-Out)  This may include calls that are scheduled, event driven, or in response to I/O devices that dial in.
<b>Use this modem to answer incoming calls</b>	Determines whether this modem will be used to receive calls from a dial-up remote I/O device. (Dial-In)
<b>Use this modem to call back I/O devices</b>	Determines whether this modem will be used to initiate calls from the I/O server to a dial-up remote I/O device in response to a call received from the I/O device. (Dial-Back)
<b>Comment</b>	Any useful comment.

### Project Properties

Property	Description
<b>Project</b>	The project in which the modem is configured.

## I/O Device Constraints for Multi-dropping

If you are multi-dropping off a single modem, use your I/O Devices to issue the caller ID, not the modem. This is because using the modem to issue the ID will send the same ID no matter which I/O Device the call is relevant to. This makes it difficult to identify the I/O Device that triggered the call.

By using the I/O Device to issue the ID, the I/O Server will receive a unique caller ID for each I/O Device. However, not every I/O Device is capable of issuing caller IDs. If multi-dropping, use I/O Devices that can issue caller IDs.

### To configure dial-up remote I/O Devices for communication with Plant SCADA:

1. Run the Device Communications Wizard.
2. Complete the wizard, selecting the relevant I/O Server, then the I/O Device, creating each new instance when necessary.
3. On the Scheduling page of the wizard, select the **Connect I/O Device to PSTN** check box.

4. Select an appropriate schedule for Plant SCADA to communicate with the remote I/O Device. (For a persistent connection - whenever Plant SCADA is running - select **On Startup**.) For example (all based on a **Synchronize at** time of 10:00:00):
  - If you enter **12:00:00** in the **Repeat every** field, and start your project at 9 a.m., the I/O Server will communicate with the I/O Device at 10 a.m., then once every 12 hours after that; that is, 10 p.m., then again at 10 a.m. of the following day, and so on.
  - If you enter 12:00:00 and start your project at 4 p.m., the I/O Server will communicate with the I/O Server at 10 p.m., then again at 10 a.m., of the following day, and so on. Plant SCADA will assume that communications were established at 10 a.m., so will continue as if they had been, communicating once every 12 hours after 10 a.m.
  - If you enter 3 days and start your project at 9 a.m. on a Wednesday, the I/O Server will communicate with the I/O Device at 10 a.m., then once every 3 days after that; that is, 10 a.m. on the following Saturday, then at 10 a.m. on the following Tuesday, and so on.
  - If you enter the 6<sup>th</sup> of December in the **Repeat every** field and start your project during November, the I/O Server will communicate with the I/O Device at 10 a.m. on December 6, then again on December 6 of the following year, and so on.
5. Select **On Startup** for a persistent connection. To disconnect a persistent connection, call the [IODeviceControl](#) function with type 8.
6. Type in the phone number necessary for Plant SCADA to dial the remote modem attached to the remote I/O Device. Include any pre-dial numbers necessary to obtain a connection to an outside PSTN line (dial tone) before dialing (for example, **0** (zero)) - if appropriate.
7. On the next wizard page, if the device is configured to dial-in to Plant SCADA, create a unique identifying caller name for the remote I/O Device so that it can be identified by Plant SCADA.
8. Follow the instructions on the next page of the wizard and click **Finish**.

## See Also

[Configuring Multidrop Remote I/O Devices](#)

## Configuring Multidrop Remote I/O Devices

Multidropping remote I/O devices from the same remote modem enables Plant SCADA to communicate with each I/O device one after the other, using the same phone connection, rather than hanging up and re-dialing. Although you can configure multidrop remote I/O devices using the Device Communications Wizard, we recommend that you do it manually. The wizard would create a new port for each I/O device. This would mean you couldn't have any more than 255 I/O devices.

1. Run the Device Communications Wizard to configure the first device.
2. Configure every other I/O device manually in the Topology activity (see "Add an I/O Device" on page ).
3. To increase the efficiency and capacity of your system you can allocate the same port name to I/O devices with the same communication settings.

**Note:** If you are multi-dropping and you want to be able to dial in to the I/O server, use your I/O devices to issue the caller ID, not the modem. This is because using the modem to issue the ID will send the same ID no matter which I/O device the call is relevant to. This makes it difficult to identify the I/O device that triggered the call.

By using the I/O device to issue the ID, the I/O server will receive a unique caller ID for each I/O device. However, not every I/O device is capable of issuing caller IDs. If multi-dropping, use I/O devices that can issue caller IDs.

### To set up a modem connected to your dial-up remote I/O devices:

You can connect multiple I/O devices to the same modem. This means Plant SCADA can communicate with these I/O Devices one after the other using the same phone connection, rather than hanging up and re-dialing. This will reduce the number of telephone calls and increase the speed and efficiency of communications.

1. Connect the modem to a PC with a telephony program installed (for example HyperTerminal or PhoneDialer). This is where you will configure the modem to answer calls from Plant SCADA and/or initiate calls.
2. If the modem is necessary to make calls to Plant SCADA, configure it to initiate the phone call to a pre-determined Plant SCADA I/O server dial-In type modem (following manufacturer instructions).

Depending on your hardware, either the modem or an intelligent PLC can be responsible for initiating calls to Plant SCADA and identifying the caller. Whichever is responsible needs to have a caller ID set. The caller ID can be any combination of alpha-numeric characters and/or the character '\_' (underscore).

Some modems have dip-switch settings, and some have initiation strings which can include auto-dial-up numbers that are stored within the modem's non-volatile memory. Consult the manual provided with the modem for exact details.

You can use either the Device Communications Wizard or the I/O devices form to set the caller ID for an I/O device.

If multi-dropping off a single modem, use your I/O devices to issue the caller ID, not the modem. This is because using the modem to issue the ID will send the same ID no matter which I/O device the call is relevant to, making it difficult to identify which I/O device triggered the call.

By using the I/O device to issue the ID, the I/O server will receive a unique caller ID for each I/O device. However, not every I/O device is capable of issuing caller IDs. If you are multi-dropping, use I/O devices that can issue caller IDs.

3. Set the modem's **Data bits**, **Parity**, **Stop bits**, and **Serial-Rate** to match manufacturer specifications for communication with the I/O devices.

Some modems do not allow you to manipulate their communications settings via methods such as extended AT commands or dip switches. If this is the case, the only way of setting the necessary values is to communicate with the modem using the values (for example, via HyperTerminal). Once this is done, the modem remembers the last values used to communicate with its serial port.

4. Connect the modem to the I/O devices.

To configure a modem at the I/O server you need to set it up in Windows and then set it up in Plant SCADA. See [Configure a Modem](#).

If every one of your I/O devices are the same, you only have to do this once for each modem. However, if your I/O devices talk using different communication specifications (data bits, parity, stop bits, and serial-rate), your modem has to be able to talk using each of these details as well. To set this up, you have to create a modem in Windows and Plant SCADA for each specification. See [Example Configurations for Modems at the I/O Server](#).

### Example Configurations for Modems at the I/O Server

The examples below demonstrate how to set up the modems at your I/O Server to accommodate different combinations of I/O Devices.

#### Example 1

All your remote I/O Devices have the same communication requirements (data bits, stop bits, parity, and baud rate) - 19200 8 E 1.

You don't expect any important calls from your I/O Devices, or you only have a few remote I/O Devices. This means you can use a single modem at the I/O Server end. This modem would be set up to answer and return incoming calls and make scheduled and other Plant SCADA initiated calls.

To configure your modem, define it in Windows. Assuming that the logical modem is called 'Standard Modem', configure it as follows:

Port	Modem Name	Max. Speed	Data Bits	Parity	Stop Bits
COM1	Standard Modem	19200	8	E	1

You would then configure it in Plant SCADA as a dial-out modem and dial-in modem:

Modem Name	Dial-out	Dial-in	Dial-back
Standard Modem	TRUE	TRUE	FALSE

### Example 2

In this example, your I/O Devices use a total of two different communication specifications - 9600 7 O 1 and 19200 8 E 1.

You don't expect important calls from I/O Devices or you have only a few I/O Devices. This means you can get by with a single modem at the I/O Server end. This modem has to receive and return calls from I/O Devices as well as initiate calls (dial out) to I/O Devices.

To configure your modem, you need to first define it in Windows (through the Windows Control Panel).

Remember, you're not just defining the physical modem here. You have to define a separate Windows (virtual) modem for each communication specification.

So far, this gives you two virtual modems - one for 9600 7 O 1 and one for 19200 8 E 1. However, Windows won't let you define both of these modems as dial-in. It only lets you define one dial-in modem per port. If you choose the first, it won't be able to receive calls with the second, and vice versa.

This means you have to set up a separate virtual modem that can answer calls no matter which communication specification is used. This modem would be set with a generic communication specification of 9600 8 N 1.

So in Windows, you'll end up with three logical modems (two for Dial-Out and one for Dial-In). Assuming that the logical modems are called 'Standard Modem' to 'Standard Modem #3', you would configure them as follows:

Port	Modem Name	Max. Speed	Data Bits	Parity	Stop Bits
COM1	Standard Modem	9600	7	O	1
COM1	Standard Modem #2	19200	8	E	1
COM1	Standard Modem #3	9600	8	N	1

You would then configure the modems in Plant SCADA as follows.

Modem Name	Dial-out	Dial-in	Dial-back
Standard Modem	TRUE	FALSE	FALSE
Standard Modem #2	TRUE	FALSE	FALSE
Standard Modem #3	FALSE	TRUE	FALSE

**Example 3**

In this example, there are five different communications frameworks - 9600 7 O 1, 19200 8 E 1, 4800 8 N 1, 9600 8 N 1, and 19200 8 N 1.

If you expect important calls from I/O Devices or you have many I/O Devices, you would set up three modems at the I/O Server end:

- One on COM3 dedicated to receiving calls from **9600 7 O 1** I/O Devices.
- One on COM2 for dialing out to **4800 8 N 1**, **9600 8 N 1**, and **19200 8 N 1** I/O Devices.
- One on COM1 for dialing out to **9600 7 O 1** and **19200 8 E 1** I/O Devices.

The two dial-out modems would return calls as well as initiate calls in response to scheduled requests, and so on.

To configure your modems, you need to first define them in Windows (through the Windows Control Panel).

Remember, you're not just defining the physical modem here. You have to define a separate Windows (virtual) modem for each communication framework.

Assuming that the logical modems are called 'Standard Modem' to 'Standard Modem #6', you would configure them as follows:

Port	Modem Name	Max. Speed	Data Bits	Parity	Stop Bits
COM1	Standard Modem	9600	7	O	1
COM1	Standard Modem #2	19200	8	E	1
COM2	Standard Modem #3	4800	8	N	1
COM2	Standard Modem #4	9600	8	N	1
COM2	Standard Modem #5	19200	8	N	1
COM3	Standard Modem #6	9600	7	O	1

You would then configure the modems in Plant SCADA as follows:

Modem Name	Dial-out	Dial-in	Dial-back
Standard Modem	TRUE	FALSE	FALSE
Standard Modem #2	TRUE	FALSE	FALSE
Standard Modem #3	TRUE	FALSE	FALSE
Standard Modem #4	TRUE	FALSE	FALSE
Standard Modem #5	TRUE	FALSE	FALSE
Standard Modem #6	FALSE	TRUE	FALSE

**Example 4**

In this example, your I/O Devices use three different communication frameworks: 9600 7 O 1, 19200 8 E 1, and 9600 8 N 1. However, in this example, you are expecting important calls from I/O Devices, so you need a modem dedicated to returning calls.

Here you need to configure your modems like this:

- One modem on COM1 to dial remote I/O Devices (for scheduled calls, and so on).
- One modem on COM2 to receive calls from remote I/O Devices.
- One dedicated modem on COM3 to return these calls.

To configure your modems, first define them in Windows (through the Windows Control Panel). Remember, you're not just defining the physical modem here: you need to define a separate Windows (virtual) modem for each communication framework. This means you have to configure:

- Three logical modems on the port to which the physical dial-out modem is attached.
- One logical modem on the port to which the physical dial-in modem is attached.
- Three logical modems on the port to which the physical dial-back modem is attached.

Assuming that the necessary total of seven logical modems are called 'Standard Modem' through to 'Standard Modem #7', configure these modems as follows:

Port	Modem Name	Max. Speed	Data Bits	Parity	Stop Bits
COM1	Standard Modem	9600	7	O	1
COM1	Standard Modem #2	19200	8	E	1
COM1	Standard Modem #3	9600	8	N	1
COM2	Standard Modem #4	9600	8	N	1
COM3	Standard	9600	7	O	1

Port	Modem Name	Max. Speed	Data Bits	Parity	Stop Bits
	Modem #5				
COM3	Standard Modem #6	19200	8	E	1
COM3	Standard Modem #7	9600	8	N	1

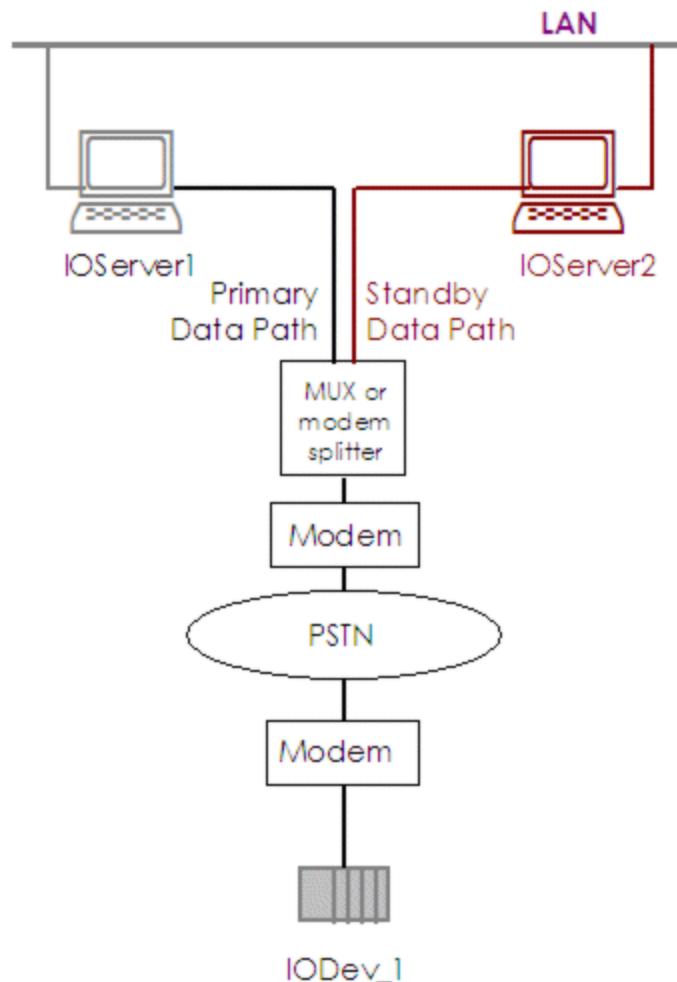
You would then configure the modems in Plant SCADA as follows:

Modem Name	Dial-out	Dial-in	Dial-back
Standard Modem	TRUE	FALSE	FALSE
Standard Modem #2	TRUE	FALSE	FALSE
Standard Modem #3	TRUE	FALSE	FALSE
Standard Modem #4	FALSE	TRUE	FALSE
Standard Modem #5	FALSE	FALSE	TRUE
Standard Modem #6	FALSE	FALSE	TRUE
Standard Modem #7	FALSE	FALSE	TRUE

## I/O Server Redundancy for Dial-up Remote I/O Devices

You can change the number of rings the I/O Server will wait before answering the call (using the **[Dial]RingCount** parameter). If you are using redundant I/O Servers, the primary I/O Server will be called by default. However, by setting **[Dial]RingCount** to a different value on each of the I/O Servers, you can make the standby I/O Server answer the call if the primary does not.

Consider the following setup:



If you set the ring count to 3 on IOServer1 and 4 on IOServer2, IODev\_1 will attempt to call IOServer1. If IOServer1 has not answered the call after 3 rings, IOServer2 will answer it.

## See Also

[Troubleshooting Dial-up Remote I/O Device Communications](#)

### Troubleshooting Dial-up Remote I/O Device Communications

The challenges most often encountered when using a dial-up remote I/O Device involve communications speed, parity, and control signals from the connected equipment. If changing the speed and parity does not resolve a communications interruption, evaluate the modem's answering codes and command echoing.

Following is a list of settings that might be helpful in resolving dial-up communications interruptions. (Since not every modem supports the same commands in the same way, this is only a guide. Consult the modem manual for exact details.)

#### On the modem at the PC end

```
ATV1 //Enables long-form (verbose) result codes
ATQ0 //Result codes are sent on the RS-232 connection
ATE0 //Commands sent from the computer are not echoed back to the RS-232 connection
```

```
AT&C1 //DCD will follow carrier on the line
AT&K0 //Handshaking OFF
ATW0 //Upon connection, only DTE speed is reported
AT%C0 //Compression OFF
AT&D0 //DTR always on
```

If the modem at the PC end is configured so that calls are automatically answered even when your Plant SCADA project is off, data being reported from the I/O devices may be lost. Therefore, it is recommended that you turn off the PC modem's auto-answer feature before taking your project offline. To do this, set the following parameter to zero:

```
ATS0 = 0 // Auto answer OFF
```

Be aware, however, this will also impact applications that might use the modem other than Plant SCADA, as the modem cannot answer a call while Plant SCADA is not driving its functionality.

#### On the modem at the I/O Device end

```
ATV0 //Enables short-form result codes
ATQ1 //No result codes are sent on the RS-232 connection
ATE0 //Commands that are sent from the computer are not echoed
back to the RS-232 connection
AT&C1 //DCD will follow carrier on the line
AT&K0 //Handshaking OFF
ATW0 //Upon connection, only DTE speed is reported
AT%C0 //Compression OFF
AT&D0 //DTR always on
ATS0 //Set to greater than 0 (sets the number of rings necessary before the modem answers
an incoming call).
```

### Alternative (backward compatibility) Method of Persistent Connection

If you are setting up your modem to dial a dial-up remote I/O Device, follow the procedures described in [Communicating with Remote Devices via Modems](#). This method is available for backward compatibility.

## Scheduled Communications

Plant SCADA allows you to schedule communications with your I/O Devices (regardless of the type of connection: modem, radio link, etc.). For example, if you have multiple I/O Devices on a single network or line, you can schedule reads so that the important I/O Devices are read more often than less important I/O Devices. Alternatively, a water supplier with radio link connections to dam level monitors might schedule hourly level reads from Plant SCADA in order to conserve band-width.

### See Also

- [Specifying a Schedule](#)
- [Writing to the Scheduled I/O Device](#)
- [Reading from the scheduled I/O Device](#)

### Specifying a Schedule

To configure scheduled communications with an I/O Device, you need to flag it as a "Scheduled" device. This is

done using the **Device Communications Wizard**. If your I/O Device is not capable of scheduled communications, the scheduling options will not be presented by the wizard.

---

**Note:** Scheduled communications will not work for drivers that are designed to handle Report-By-Exception protocols. For communicating with your I/O Device outside of schedule use the IODeviceControl function.

---

### To schedule communications with an I/O Device:

1. Navigate to the Topology activity | I/O Devices.
2. Click **New Device** on the Command Bar. The
3. Double-click the **Express I/O Device Setup** icon in the Communications folder of the current project.
4. Follow the instructions to work through the screens, selecting the relevant I/O Device, and so on. When the scheduling screen displays, check the **Connect I/O Device to PSTN** box, even if your I/O Device is not connected via a modem (but leave the Phone number to dial and Caller ID fields blank).
5. Fill out the schedule fields to achieve the desired schedule. For example (all based on a **Synchronize at time** of 10:00:00).

If you enter 12:00:00 in the **Repeat every** field, and start your project at 9 a.m., the I/O Server will communicate with the I/O Device at 10 a.m., then once every 12 hours after that, i.e. 10 p.m., then again at 10 a.m. of the following day, etc.

- If you enter 12:00:00, and start your project at 4 p.m., the I/O Server will communicate with the I/O Server at 10 p.m., then again at 10 a.m. of the following day, etc. i.e. it will assume that communications were established at 10 a.m., so it continues as if they had been, communicating once every 12 hours after 10 a.m.
  - If you enter 3 days, and start your project at 9 a.m. on a Wednesday, the I/O Server will communicate with the I/O Device at 10 a.m., then once every 3 days after that, i.e. 10 a.m. on the following Saturday, then at 10 a.m. on the following Tuesday, etc.
  - If you enter the 6th of December in the **Repeat every**, and start your project during November, the I/O Server will communicate with the I/O Device at 10 a.m. on December 6, then again on December 6 of the following year, and so on.
6. Select **On Startup** for a persistent connection. To disconnect a persistent connection, you need to call the IODeviceControl() function with type 8.
  7. If your I/O Device is not connected via a modem, you need to go to the Ports settings and change the Port number to the actual number of the COM port.

### See Also

[Writing to the Scheduled I/O Device](#)

### Writing to the Scheduled I/O Device

Whenever an I/O Device is actively communicating (as per its schedule), you can write to it directly. However, if you try to write to it when it is not communicating, your write request will be queued until it is. For example, you might decide to schedule one write per hour. If someone at a Control Client changes a tag's value during that hour, that change will not be written to the I/O Device until the hour has expired.

As write requests are not written to the I/O Device until it is communicating, confirm that pending writes have been completed in full before shutting down.

**Note:** Don't control hardware using a scheduled I/O Device, as the exact state of the hardware may not be known. Although you can read the state from the cache, it may have changed since the cache was created.

## ⚠️ WARNING

### UNINTENDED EQUIPMENT OPERATION

Do not use a scheduled I/O device to control hardware in a system managed by Plant SCADA.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## See Also

[Reading from the scheduled I/O Device](#)

### Reading from the scheduled I/O Device

When the I/O Server initiates communication with the I/O Device, it immediately writes any queued write requests, then reads the I/O Device's tags. These values are then stored in a cache so that you can still access them between communications.

**Note:** Because the I/O Server reads tags on initiation of communication, the license point count is increased, even though some of the tags read may not actually be used.

### Keeping Data Up-to-Date during Prolonged connections

Normally, communication is terminated as soon as every read and write request is complete. Sometimes, however, you may want to prolong the communication (for example by calling the IODeviceControl() function with Type 7).

In this situation (if Read-Through Caching is disabled), when client computers request device data from the I/O Server, it retrieves the data from its cache, not from the I/O Device. This occurs even though the I/O Server maintains a connection to the device.

To retrieve fresh data from the I/O Device, you can force a periodic read using Cicode, or change the cache timeout by setting the IODeviceControl() function to Type 11.

For example:

```
INT hTask;
// Initiate communications and read tags.
// Sleep time will depend on how fast your
// modems connect.
FUNCTION
DialDevice(STRING sDevice)
    INT bConnected = 0;
    INT nRetry = 5;
    hTask = TaskHnd("");
    IODeviceControl(sDevice, 7, 0);
    Sleep(20);
    WHILE bConnected <> 1 AND nRetry > 0 DO
        bConnected = IODeviceInfo(sDevice, 18);
        nRetry = nRetry - 1;
        Sleep(10);
```

```
END
IF bConnected = 1 THEN
    WHILE TRUE DO
        Sleep(2);
        IODeviceControl(sDevice, 16, 0);
    END
END
// Kill the read task and terminate the connection.
FUNCTION
HangupDevice(STRING sDevice)
    TaskKill(hTask);
    IODeviceControl(sDevice, 8, 0);
END
```

You can also force the I/O Server to read data directly from an I/O Device by enabling Read-Through Caching. With [\[Dial\]ReadThroughCache](#) set, while the I/O Server is connected to a device it will supply data to requesting clients directly from the device. The cache is not updated during this time, but is refreshed with the most recent device data just before the server disconnects.

**Note:** If using modems, you might need to adjust or deactivate the inactivity timer in your modems to stop them from disconnecting while no data is being read. The inactivity timer is controlled by the S30 register. If your modem doesn't support this register, please consult your modem's manual.

### Avoiding Unnecessary Multiple Reads of I/O Device Data

To avoid unnecessary reads of an I/O Device, you can use data caching to temporarily store data read from the device in the memory of the I/O Server. This means that if the I/O Server receives more than one request for device data in a short time period, instead of contacting the I/O Device a second time and reading identical data, it can retrieve the data from the cache.

## Communication Performance Considerations

Many external factors influence the performance of control and monitoring systems. The capabilities of the computer, the I/O Device(s), and the communication pathway between them are obvious factors. The faster they can transfer data, the faster your system operates. (Plant SCADA always attempts to maintain a data transfer rate as fast as the I/O Device hardware can support.) The data transfer rate is hardware dependent: once you have installed the hardware, your ability to influence this characteristic is limited.

However, there is one area where you can directly affect the performance of your runtime system: the arrangement of data registers in the I/O Device(s).

## See Also

- [Caching Data](#)
- [Grouping Registers](#)
- [Remapping Variables in an I/O Device](#)

## Caching Data

On large networked systems with many clients, you can improve communications turn-around time by using memory caching.

When caching is enabled, data that is read from an I/O Device is stored temporarily in the memory of the I/O Server. If another request is made (from the same or another client) for the same data within the cache time, the Plant SCADA I/O Server returns the value in its memory, rather than rereading the I/O Device. Data caching results in faster overall response when the same data is necessary by many clients.

A cache time of 300 milliseconds is recommended. Avoid using long cache times (in excess of 1000 milliseconds), as this may negatively impact the timely delivery of information to the system.

**Note:** Do not use data caching for [Configure a Disk I/O Device](#) as they implement a cache themselves.

---

## Grouping Registers

When you configure a Plant SCADA system, you need to define each variable (register address) that Plant SCADA will read when your system is running. When your runtime system is operating, Plant SCADA calculates the most efficient method of reading registers. Plant SCADA optimizes communication based on the type of I/O Device and the register addresses.

When Plant SCADA requests data from an I/O Device, the value of the register is not returned immediately; an overhead is incurred. This overhead (associated with protocol headers, checksum, device latency, and so on) depends on the brand of I/O Device, and is usually several times greater than the time necessary to read a single register. It is therefore inefficient to read registers individually, and Plant SCADA usually reads a contiguous block of registers. Because the overhead is only incurred once (when the initial request is made), the overhead is shared across every register in the block, increasing the overall efficiency of the data transfer.

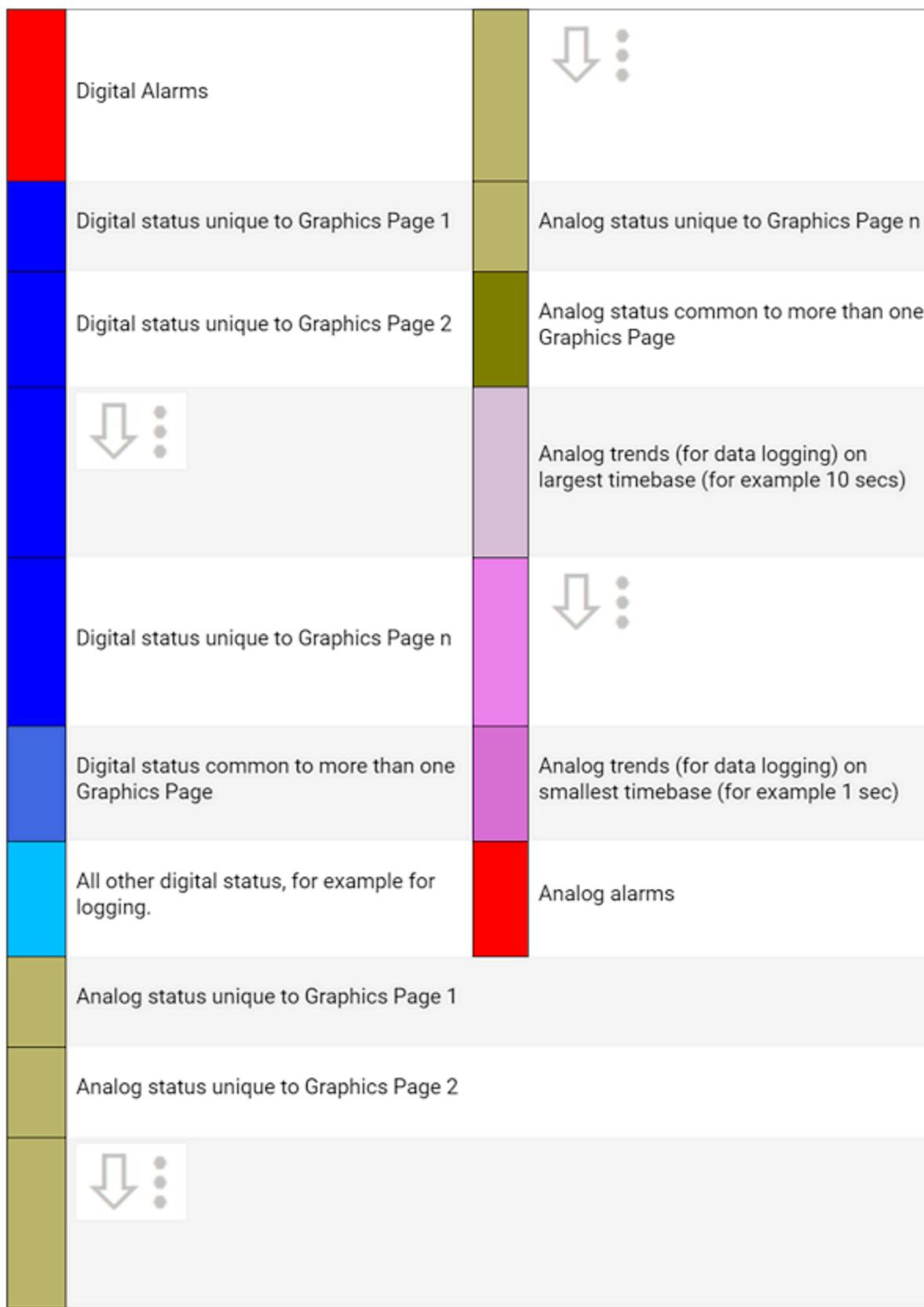
However, reading a block of registers where only a small percentage of the block is actually used is also inefficient. If the registers that your Plant SCADA system will read are scattered throughout the memory of your I/O Device, excessive communication will be necessary. Plant SCADA needs to either read many contiguous blocks (and discard the unused registers), or read registers individually, degrading system performance. You can avoid this by grouping the registers that Plant SCADA will read.

Plant SCADA continually reads registers associated with alarms. (If an alarm condition occurs, Plant SCADA can display the alarm immediately.) therefore group registers that indicate alarm conditions.

Registers associated with status displays (objects, trends, and so on) are only read as they are necessary (that is, when the associated graphics page is displayed) and are appropriately grouped according to the pages on which they are displayed.

Registers used for data logging are read at a frequency that you define. They are grouped according to the frequency at which they are read.

The following table shows an ideal register grouping for a Plant SCADA system:



While memory constraints and the existing PLC program might impose limitations, grouping registers into discrete blocks, even if they are not consecutive blocks, will improve system performance.

When designing your system, allow several spare registers at the end of each block for future enhancements.

## See Also

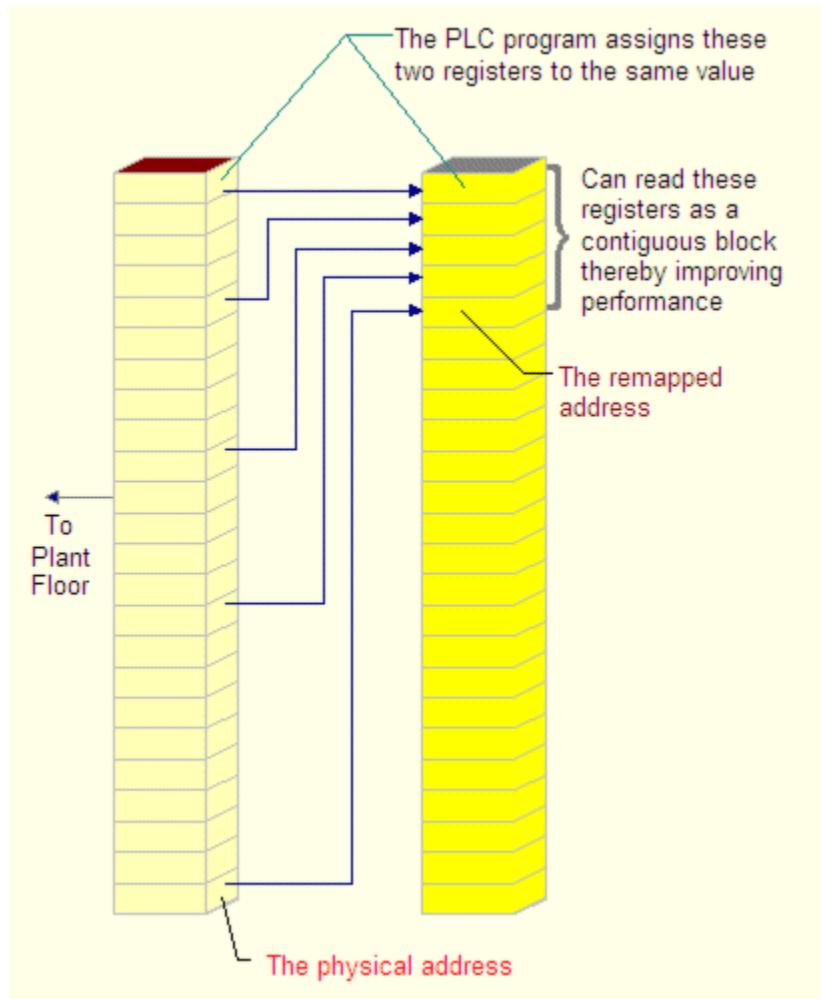
[Remapping Variables in an I/O Device](#)

### Remapping Variables in an I/O Device

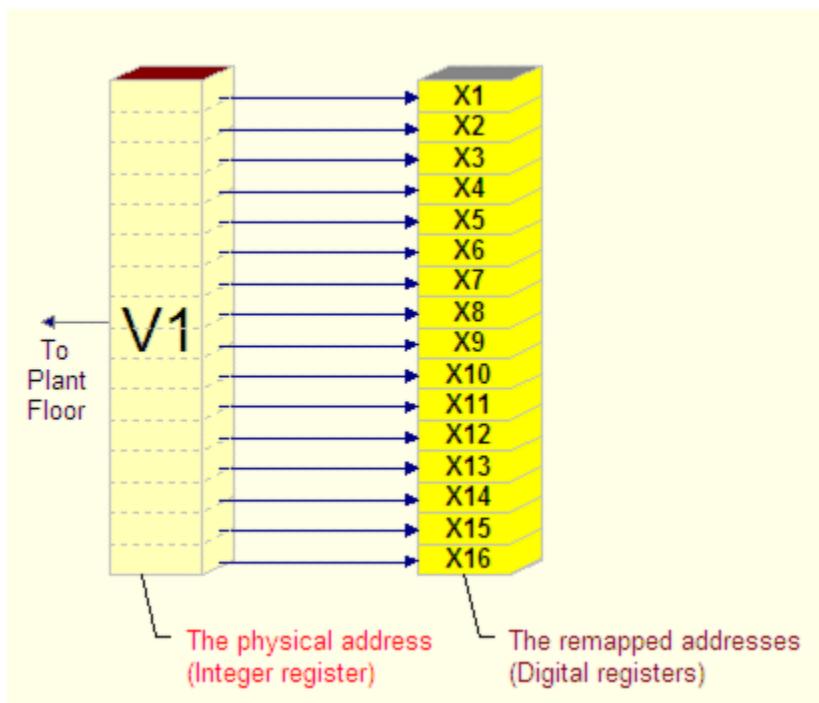
Some PLCs allow you to remap (or copy) an I/O device variable to another register address. Plant SCADA allows you to remap to:

- Group registers more efficiently to increase performance.
- Allow Plant SCADA to interpret a variable type (for example, an analog variable) as a different variable type (for example, a digital variable). For example, you can create additional digital addresses if an I/O device has run out of digital addresses.

To remap a variable in your PLC, you need to design (or modify) the logic in the PLC to associate both addresses. Plant SCADA can then read or write the variable to and from the remapped address instead of the physical address.



You can also reassign one type of variable (for example, an integer) to another type of variable (for example, a digital variable).



To remap in Plant SCADA, first create the variables in your project as you would normally. Then you can set up the remapping, specifying that any variable with an address in the desired range will be remapped. The I/O server will redirect the addresses at runtime as per the remapping instruction.

---

**Note:** Not every PLC and/or Plant SCADA driver supports remapping. It is not recommended unless necessary. Contact Technical Support if you need to evaluate whether the PLC and/or driver support remapping.

---

#### To remap a variable in Plant SCADA:

1. In the **Topology** activity, select **Components & Mappings**.
2. On the menu below the Command Bar, select **Remapping**.
3. Add a row to the Grid Editor.
4. Type the required information in each column, or in the fields in the Property Grid.
5. For a description of the properties, see below.
6. Click **Save**.

---

**Note:** To determine if the device supports remapped reads or writes, see the I/O Device Data Type help.

---

## Remapping Properties

### General Properties

Property	Description
<b>Plant SCADA variable</b>	The first remapped (Plant SCADA) variable defined in the variable tags database (using the Tags dialog box); for example: Motor_1_Run.

Property	Description
	(79 characters maximum). Alternatively, use the direct <Unit Name> <Address>  format (using values specific to your I/O device); for example: IODev X1 . The address entered here is remapped. At runtime the I/O server will read/write data through the physical address instead.
<b>Length</b>	The number of remapped variables. Plant SCADA reads enough physical variables to remap this number of variables (10 characters maximum). The length needs to be less than the maximum request length of the protocol. The protocol overview displays the maximum request length of the protocol.
<b>Physical Variable</b>	The first physical variable in the PLC, for example: ReMapIntV7. (79 characters maximum). This variable does not need to be defined in the variable tags database. You can use the <Unit Name> <Address>  format (using values specific to your I/O Device). For example, IODev V7 .
<b>Remap Read</b>	Determines whether to perform the remapping for reads. Set to TRUE or FALSE.  FALSE - Read normal (not remapped) variables. The actual address of the Plant SCADA variables will be read directly from the I/O Device, instead of through the Physical Variable. (Use this mode if your I/O Device does not support remap reads.)  TRUE - Read remapped variables (through the physical variable).
<b>Remap Write</b>	Determines whether to perform the remapping for writes. Set to TRUE or FALSE.  FALSE - Write to normal (not remapped) variables. The actual address of the Plant SCADA variables will be written directly in the I/O Device, instead of through the physical variable. (You need to use this mode if your I/O Device does not support remap writes.)  TRUE - Write to remapped variables (through the physical variable).
<b>Comment</b>	Any useful comment.

## Project Properties

Property	Description
Project	The project in which the remapping is configured.

## Advanced Driver Information

This section provides advanced information about drivers.

- [Variable \(Digital\) Limitations](#)
- [Validating Distributed Project Data for Tag-based Drivers](#)
- [Write Delay Effects](#)

### Variable (Digital) Limitations

Devices often have memory areas that are of a designated data type, like Byte, Integer, or Word. Some protocols do not support the reading and writing of data in these memory areas using a different data type. This situation is most common in the case of reading and writing of individual bits within the data types like Bytes, Integers, and Words.

In this case, reading individual bits within these larger data types is done by reading the designated data type and getting the Plant SCADA driver to sub-divide it into individual bits. Writing to bits within the larger data types is more complicated, as writing to one bit within the larger data type will at the same time overwrite the other bits within that same data type. To prevent overwriting existing bits when writing a new bit value, a 'read-modify-write' scenario can be used to write to a bit within the larger data type. Using this approach, the Plant SCADA driver will read the larger data type, modify the appropriate bit within the larger data type, and then write the larger data type back to the device.

While the 'read-modify-write' approach is necessary to avoid overwriting existing bits in the registers of larger data types, it can create an issue if the device being written to is also configured or programmed to modify these same registers. For example, if a PLC device modifies the registers of one of its larger data types after the Plant SCADA driver has read these same registers, but before Plant SCADA has written the modified value, the changes made by the PLC will be overwritten. This outcome can be avoided if Plant SCADA and any devices using these data types are configured so that only one or the other has write access at any given time.

This 'read-modify-write' method has a serious operational concern: if the device modifies the larger data type after the Plant SCADA driver has read it, but before Plant SCADA has written the new value, any changes made by the device are overwritten. This issue could be serious in a control system, and it is recommended that the device and Plant SCADA be configured so that only one of these systems writes to the data types of this kind.



### UNINTENDED EQUIPMENT OPERATION

When the read-modify-write method will be used to alter a data type's bit values, configure your system so that Plant SCADA and the host device do not have simultaneous write access to the affected memory ranges.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Consider the following example:

- The initial state of a PLC register is 0x02h.
- The Plant SCADA driver reads the value of this register (effectively making a copy) in preparation for a change to bit 3.
- However, before the driver writes its change back to the PLC, the PLC code changes the value of bits 0 and 4 of this register to 0x13h.
- The Plant SCADA driver then changes bit 3 of its copy of the register to 0x0Ah. When it writes to the PLC, it overwrites the PLC's copy of the whole register (not just the changed bit). Because the PLC code modified bits 0 and 4 in the interval between Plant SCADA's read and write, these changes are overwritten.

## Validating Distributed Project Data for Tag-based Drivers

Plant SCADA uses numeric index values to uniquely identify the variable tags in a project. They are used as a reference point when requesting data from the I/O Server for a tag-based driver.

These index values are automatically generated when a project is compiled. Circumstances may arise where a distributed project has index values that represent different tag addresses on different computers. For this reason, Plant SCADA has a number of automatic checks in place that validate a project's tag index values and flag any discrepancies.

An initial security check takes place on client machines at a unit level, allowing a tag index mismatch to be isolated to a particular client before any requests are sent to the I/O Server. This confirms that the unit address, the unit type, the raw data type and the tag address match for index values across the client and server machines. Any discrepancies found are flagged by a hardware alarm on the client machine.

Each page is also checked to confirm that it was compiled against the current version of the variables database. There is also a check performed whenever a tag-based driver loads the variable database to test whether it matches the current tag addresses. The parameter TagAddressNoCase allows you to adjust the case-sensitivity of these checks.

In addition, Plant SCADA will also check if a project is currently running on the local machine when a compile is attempted, as this is one of the circumstances that may lead to mismatched index values.

If the project uses a tag-based driver and is currently in runtime, Plant SCADA will stop the compiler and generate an error in the error database noting that Citect32.exe was still running. The .ini parameter [General]CitectRunningCheck allows you to override this feature, however it is recommended that you leave it enabled so that your tag index values are assigned as intended.

## Write Delay Effects

Plant SCADA performs writes to the I/O Device asynchronously (that is, when you write to the I/O Device, the write takes time to get to the I/O Device, during which Plant SCADA continues to perform other operations). If the other Cicode assumes that the write has completed immediately, you might encounter some side effects such as inconsistencies in the value of a variable tag across two Cicode threads.

If you have the following Cicode:

```
PLC_VAR = 1234;  
Prompt("Variable is " + PLC_VAR : #####);
```

the first line is a write to the PLC.

When Plant SCADA executes the first line, it generates a request to the I/O Server to write the value 1234 into the PLC variable PLC\_VAR. Plant SCADA then executes the next line of Cicode before the PLC write is completed. Plant SCADA does this so that the Cicode is not stopped while waiting for a slow I/O Device. As the write to the

PLC has not completed, you might think that the next line of Cicode will display the last value of PLC\_VAR and not the value 1234. However, this Cicode will display the correct value (1234) because whenever Plant SCADA writes to the PLC, it first updates its local copy of the variable: any following Cicode will get the correct value.

Sometimes this solution will not work as Plant SCADA might keep multiple copies of an I/O Device variable, and only update the one associated with the current Cicode. The other variables will contain the old value of the I/O Device variable until they are refreshed (with a read from the I/O Device). There is a separate data area for each display page, Cicode file, for alarms, trends and reports. If you write to an I/O Device variable from a page keyboard command, the copy of the I/O Device variable associated with that page will be updated; however, the copy associated with other pages and the Cicode functions is not updated until the next read (as determined by the [\[Page\]ScanTime](#) parameter). If you call a Cicode function that assumes the write has completed, it will get the last value.

The workaround is to write to the I/O Device variable in the Cicode function. Every Cicode function shares the same I/O Device variables, so the writes will operate as expected.

#### FUNCTION

```
TestFunc(INT nValue)
PLC_VAR = nValue;
END
```

Another side effect of the delays inherent in asynchronous writes to I/O devices is that you might assume that Plant SCADA has successfully written to the I/O Device, when in fact the write operation might not yet have been completed or even attempted. In such cases, it is still possible that a hardware- or configuration-based error will prevent the success of the write operation. While such an unsuccessful write operation will generate an error, your Cicode cannot be notified or use the IsError() function because the Cicode has continued to execute past the initial I/O device write command. Therefore, when it is important to check the success of a write operation before allowing code execution to continue, you might insert the function TagRead() after the write and then verify the value of the variable. TagRead() forces Cicode to re-read the I/O Device variable so that you can check the new value. TagRead() is a blocking function. It blocks the calling Cicode task until the operation is complete.

```
PLC_VAR = 1234;
sTestStr = TagRead("PLC_VAR");
IF sTestStr <> 1234 THEN
Prompt("Write not completed");
END
```

Here the data will be read from the physical PLC, not from the I/O Server cache, as the I/O Server will invalidate any cached data associated with a PLC write. This will allow you to test for a completed write. Please be aware that other Cicode tasks running at the same time will not be waiting on the TagRead, so they might see the old or new value, depending on if they are using the same copy of the PLC variable. You can also stop Plant SCADA from writing to the local copy of the variable by using the function CodeSetMode(0, 0).

## Troubleshooting Device Communications

Debugging device communications involves the following processes:

- Gathering system information about current communication status
- Analyzing the available information to identify problems

The information needed to facilitate this is available from the following resources:

- Hardware alarms

- Driver errors
- Log files (system, trace, debug and driver logs)

This information can be analyzed directly to identify problems, or you can use the Plant SCADA Kernel to perform low-level diagnostic and debugging operations.

There are also procedures you can follow to debug low-level communications protocols, such as COMx and TCP/IP.

## See Also

[Gathering Communications Information](#)

[Debugging a COMx Driver](#)

[Debugging a TCP/IP Driver](#)

[Debugging a Protocol Driver Using Serial Communications](#)

## Gathering Communications Information

If you cannot establish communication with a device, check the following resources for evidence of a problem:

- Hardware alarms
- Plant SCADA log files
- Driver errors
- Runtime Kernel.

## Hardware Alarms

When an error occurs in a Plant SCADA operation, a hardware alarm is generated. Hardware alarms are typically displayed on a dedicated page within a project, allowing an operator to monitor the status of a system's infrastructure.

Hardware alarms will indicate if break in communications has occurred, if Cicode can't execute, if a server becomes inoperative, and so on. If you are having problems communicating with a device, initially check the hardware alarms page for evidence of an error.

See *Hardware Alarms* in the topic [Alarms](#).

## Plant SCADA log files

Plant SCADA supports a number of log files that track operational status during runtime.

The primary log file, syslog.dat, contains a log of system information; from low-level driver traffic and Kernel messages, to user-defined messages.

The **Log Read** and **Log Write** fields in the I/O Devices Properties dialog determine if logging is enabled for each I/O device.

Driver logs are also available. These relate to the operation of a particular driver and are named accordingly. For example, the ABCLX driver is logged in 'ABCLX.dat'. However some driver logs may be in differing locations, refer to the help for a particular driver to confirm the location.

For a description of the available log files and where they are stored, see [Log Files](#).

## Driver errors

There are two types of driver errors that are logged to syslog.dat:

- Generic
- Driver-specific

Generic errors are common to many drivers, and are mapped to general hardware errors.

Driver-specific errors are unique to a particular driver. A driver will convert a specific error into a generic error so that it can be recognized by the hardware alarm system. However, if further investigation is necessary, you can refer to a description of the original error in the [The Driver Reference Help](#). This includes a description of the errors associated with each driver.

See [Error Messages](#).

## Runtime Kernel

The Runtime Kernel is a gateway into the internal workings of Plant SCADA at runtime, and is provided for diagnostics and debugging purposes.

The Kernel can display several different diagnostic windows each providing an active view of the Plant SCADA runtime system.

The Runtime Kernel Driver window, launched via the Page Driver Kernel command, displays information about each driver in the Plant SCADA system. The statistics it presents include read requests, physical reads, digital reads per second, register reads, cache reads, error count, timeouts, and so on.

See [The Kernel](#).

## See Also

[Troubleshooting Device Communications](#)

## Debugging a COMx Driver

A COMx driver is the board driver typically used for serial communications. COMx driver events are logged in the following file:

`syslog.IOServer.<Cluster name>.<IO Server name>.dat`

This file is located in the directory specified by the [\[CtEdit\]Logs](#) parameter in the citect.ini file.

The COMx driver can log different combinations of trace levels. This is achieved by setting the following Citect.ini parameters:

[COMx] Parameter	Description	Allowable Values	Default Value
DebugLevel	The trace level used for the log file.	Any combination of the following, separated by a pipe character ( ):	-

		ERROR WARN TRACE DEBUG ALL  For example, ERROR   TRACE.	
DebugCategory	The categories used for the log.	The COMx driver supports one category: PROT = protocol level debug	-

The log file size and log file rollover process is controlled by the [\[Debug\]SysLogSize](#) and [\[Debug\]SysLogArchive](#) ini parameters.

When the log file reaches a specified size (100000 kilobytes by default), it may be rolled over into a timestamped file when [\[Debug\]SysLogArchive](#) is set to 1, otherwise it will be overwritten.

**Note:** The more detailed the logging, the larger the amount of log data that will be generated. This could create a large, cumbersome log file, or cause multiple wraparounds that write over data. Also when the number of pending messages to process exceeds the value set by the [\[Kernel\]ErrorBuffers](#) parameter, these messages are no longer logged. Messages not logged can be monitored in the General Kernel Window under the item "Lost Errors". It is recommended to use the following DebugLevel and DebugCategory settings:

`[COMx]DebugLevel=ERROR|WARN|TRACE`  
`[COMx]DebugCategory=PROT`

## Debugging a TCP/IP Driver

A TCPIP driver is the board driver typically used for serial communications. It might be over Ethernet, Token Ring or Arcnet. This driver communicates between Plant SCADA and Winsock, so it uses the normal networking functionality of Windows.

TCP/IP driver events are logged in the following file:

`syslog.IOServer.<Cluster name>.<IO Server name>.dat`

This file is located in the directory specified by the [\[CtEdit\]Logs](#) parameter in the `citect.ini` file.

The TCPIP driver can log different combinations of trace levels. This is achieved by setting the following `Citect.ini` parameters:

[TCPIP] Parameter	Description	Allowable Values	Default Value
DebugLevel	The trace level used for the log file.	Any combination of the following, separated by a pipe character ( ):  ERROR WARN TRACE DEBUG ALL	-

		For example, ERROR   TRACE.	
DebugCategory	The categories used for the log.	The TCPIP driver supports one category: PROT = protocol level debug	-
PortName. MulticastAddress	This parameter allows you to implement the TCPIP driver option "-Maa.bb.cc.dd" in the Citect.ini file.  This may be necessary if you have run out of characters in the Special Options field of the Ports settings, which is limited to 16 characters.	You need to specify the following in the Citect.ini file:  [TCPIP] PortName. MulticastAdress = aaa.bbb.ccc.ddd  Where:  PortName = the name of the port as entered in the Ports settings aaa.bbb.ccc.ddd = the multicast Class D IP address  See <a href="#">TCP/IP Driver Special Options Reference</a> .	
NoBufferSleepTime	Sleep time in milliseconds (ms) to wait AFTER getting a WSAENOBUFS (no system buffers message) before reading another buffer. There is no need to adjust this.	-	0
Timeout	Defines the timeout period between connection retries.		1000 (ms)

The log file size and log file rollover process is controlled by the [\[Debug\]SysLogSize](#) and [\[Debug\]SysLogArchive](#) ini parameters.

When the log file reaches a specified size (100000 kilobytes by default), it may be rolled over into a timestamped file when [\[Debug\]SysLogArchive](#) is set to 1, otherwise it will be overwritten.

**Note:** The more detailed the logging, the larger the amount of log data that will be generated. This could create a large, cumbersome log file, or cause multiple wraparounds that write over data. Also when the number of pending messages to process exceeds the value set by the [\[Kernel\]ErrorBuffers](#) parameter, these messages are no longer logged. Messages not logged can be monitored in the General Kernel Window under the item "Lost Errors". It is recommended to use the following DebugLevel and DebugCategory settings:

[\[TCPIP\]DebugLevel=ERROR|WARN|TRACE](#)

---

[TCPIP]DebugCategory=PROT

---

## Debugging a Protocol Driver Using Serial Communications

To debug a protocol driver that uses serial communications, do the following:

- Keep using your Simple As Possible Project (SAPP).
- Set the "DebugStr=\* all" for your protocol.
- Backup and delete both the syslog.dat and syslog.bak files. The system will recreate a fresh version of this file the next time Plant SCADA is started.
- Start the project. From the information you can see in the maximized Main window of the kernel be able to see if Plant SCADA is sending requests to your I/O device to initialize communications with it.

If there are no requests being sent then your software is improperly configured, and check that there were no errors on Startup of Plant SCADA. If there were errors on startup look them up in the Online Help. Also check that your computer is an I/O Server (and that it matches the one in your project). To do this run the **Setup Wizard**, and configure the computer for a standalone configuration.

If there are requests being sent but no reply, then Plant SCADA is trying to communicate. When Plant SCADA is sending requests but getting no reply, these are the most common causes:

- **The request Plant SCADA is sending is not getting to the I/O device** - Check the Address field in the I/O Devices form, and verify that it is correct. If the I/O device is one that needs a unique identifier (such as a node address), or you need some type of routing path, then verify that it is correct.

Check that you have the same parameters in the Ports form that the I/O device is using. If you have 8 data bits and the I/O device uses 7 data bits, communications will not work.

Check that your cable is OK. The easiest way to do this is to create a new project and use the Loopback protocol. You can use this to verify the **Tx** and **Rx** lines' integrity by placing a jumper on these lines. Initially test this with a jumper between pins 2 and 3 on your PC. Then plug in your cable and test again with the jumper between the **Tx** and **Rx** lines. Keep moving the jumper until it is at the end of your communications bus. You can find more information about the Loopback protocol via the AVEVA Knowledge and Support Center. Search for "Loopback".

Even if the Loopback protocol shows no errors, your cable might still be responsible for the loss of communications. Plant SCADA usually places a far higher constant load on serial communications than programming software does, this usually means that Plant SCADA will require much more stringent handshaking than the programming software. So it is possible that the cable you use to program your I/O device works fine for programming, but not for Plant SCADA.

Another major cause of improper cable connections is 9-pin to 25-pin converters. Many of these converters are made specifically for serial mice. These typically only use the **Tx**, **Rx** and **Ground** signals. If you use one of these converters they do not support any handshaking and will most likely not work for your Protocol.

If the above checks OK, use the parameters for COMx (as mentioned above) to create log files. Examine these log files and verify that what Plant SCADA thinks it is sending is actually what it is sending. The log files produced by using these parameters get their information from a lower level than Plant SCADA and show you exactly what is going through the COMx driver.

- **The Response from the I/O device is not getting to Plant SCADA** - This is unlikely and usually caused by cabling that is damaged, has insufficient bandwidth, or is connected improperly.. Check your cabling as above. Also, check that you are specifying everything you need within Plant SCADA. Many protocols require

Plant SCADA to send a unique identifier in its request packet. If this identifier is incorrect then the response cannot get back to Plant SCADA.

- **The I/O device does not understand the Request** - Every Plant SCADA protocol can check if an I/O device is running. Typically the protocol attempts to read data from the I/O device, usually a status register or other register that is there. However, many pseudo-standard protocols, such as Modbus, do not conform to the exact specification for that protocol. Many protocols supplied with Plant SCADA have some extra parameters to allow you to choose the specific initialization request from Plant SCADA. These can be found in either the Online Help or the AVEVA Knowledge and Support Center. Check with the manufacturer of your I/O device to confirm that it will respond to the request that Plant SCADA sends. If you are unsure of the request being sent for initialization, use DebugStr=\* to get the actual variable address that Plant SCADA is asking for in its initialization.

Check the protocol you are using. Plant SCADA may have many different protocols for communicating to an I/O device. PLCs such as the AB PLC5 can use different serial protocols, depending on the method you are trying to use. Make sure you are using the correct one. If you are unsure, try the other possible protocols.

- **The I/O device is not functioning properly** - There is usually some sort of software from the I/O device manufacturer that can be used to diagnose any problems with the I/O device.

## See Also

[Debugging Proprietary Board Drivers](#)

## Debugging Proprietary Board Drivers

Proprietary board drivers (such as the Allen Bradley KTX card, Modicon SA85 card, Siemens TIWAY card, and so on) have their own low-level drivers. Each driver has debugging parameters to make it easier to debug device behavior. See the driver documentation for details.

The debugging process is exactly the same as with a serial connection:

- Keep using your Simple As Possible Project (SAPP).
- Set any debugging parameters for the protocol and board drivers.
- Start Plant SCADA with clean log files.
- Find any errors and then look them up in the manufacturer's documentation.

## File Formats

The **Write** file will adopt the following format:

LINE 1	WRITE Debug file Started PORTNAME Debug Level 1 DOW MONTH DOM HH:MM:SS.sss
LINE 2	HH:MM:SS.sss
LINE 3	Out W In X nBytes Y Status Z
LINES 4...	AA BB CC ..

Where:

W & X =	Values of buffer pointers
Y =	Number of bytes written
Z =	Return status of the WriteFile
AA BB CC =	Values in hex of each byte written

**Example:**

```
WRITE Debug file Started PORT1_BOARD1 Mon Dec 15 16:07:.998
16:07:09.810
Out 0 In 8 nBytes 8 iStatus 997
0e 02 00 00 10 00 00
16:07:10.802
Out 8 In 16 nBytes 8 iStatus 997
0e 02 00 00 00 10 00 00
..
..
```

The **Read** file will adopt the following format:

LINE 1	READ Debug file Started PORTNAME Debug Level 1 DOW MONTH DOM HH:MM:SS.sss
LINE 2	HH:MM:SS.sss
LINE 3	Out W InRx X nBytes Y Status Z
LINES 4...	AA BB CC ..

Where:

W & X =	Values of buffer pointers
Y =	Number of bytes read
Z =	Number of characters remaining in the buffer
AA BB CC =	Values in hex of each byte written

**Example**

```
READ Debug file Started PORT1_BOARD1 Mon Dec 15 16:07:07.998
16:07:09.830
Out 0 In 0 nBytes 8 iStatus 0
0e 02 00 00 00 10 00 00
16:07:10.822
Out 0 In 8 nBytes 8 iStatus 0
0e 02 00 00 00 10 00 00
```

The **Status** file will adopt the following format:

LINE 1	STATUS Debug file Started PORTNAME Debug Level 1 DOW MONTH DOM HH:MM:SS.sss
LINE 2	HH:MM:SS.sss

LINE 3	modemStatus X
--------	---------------

**Example**

```
STATUS Debug file Started PORT_1 Debug Level 1 Wed Nov 05
15:28:55.310
15:29:55.950
modemStatus 34
..
```

More parameters might be added later, so check the COMx information and the AVEVA Knowledge and Support Center regularly.

**See Also**

[Debugging a TCP/IP Driver](#)

**Serial Port Loop-Back Test**

You can use the serial port loop-back test to test your serial hardware configuration. This test can be used with any COM port, whether it is local, or on a multi-port serial board (such as a Digiboard). The test can be performed internally or externally with loop-back cable attached.

**See Also**

[Test Setup](#)

[Serial Port Loop-back Cable](#)

**Test Setup**

1. Do not configure any other protocols. Temporarily delete other boards and units while performing this test.
2. Configure a unit for each port to be tested. Make the I/O Devices form look as follows:
  - **Name:** <unique name for the I/O Device>
  - **Number:** <unique network number for the I/O Device>
  - **Address:** NA
  - **Protocol:** LOOPBACK
  - **Port Name:** <the "Port Name" in the Ports form>

3. The following Citect.ini options are supported:

- [LOOPBACK]

LoopBack - Set this to 1 if internal loop-back is to be performed (make sure this is deleted after running the test). When set to 0, a loop-back connector which ties pins 2 and 3 together is necessary at each port.

**Note:** The COMx driver does not support internal loop-back. The external loop-back is the default mode.

**Size** - Sets the maximum frame size. The length of each frame transmitted is random between 1 to 'Size'-1. The default size is 512.

4. Start up Plant SCADA. Each port will transmit a frame of random length. This process is repeated when the

entire frame is received.

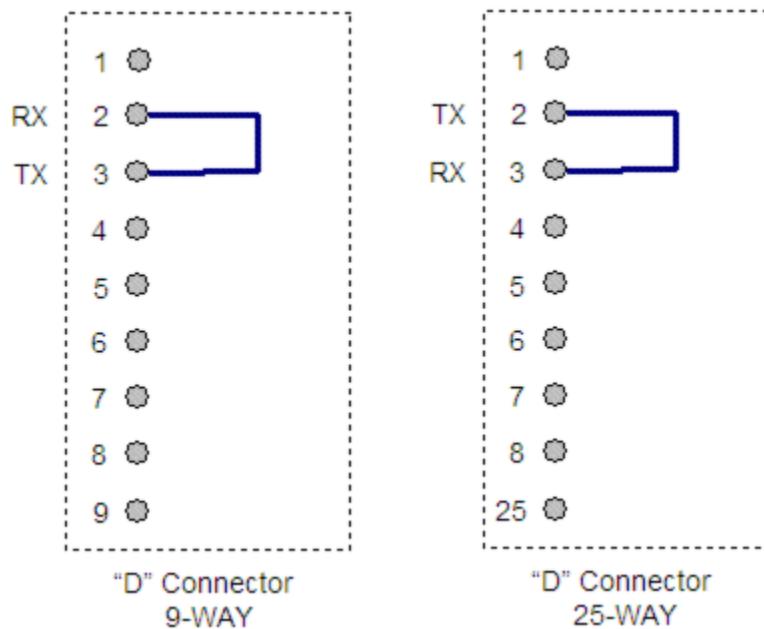
5. Open the kernel, type "page driver" and press **Enter**. Type **V** to set the display mode to 'verbose'. The following statistics appear:

- Number of characters transmitted.
- Number of frames transmitted.
- Number of characters received.
- Elapsed time in milliseconds.
- Characters received per second.
- Start time in milliseconds.
- Total number of errors.
- Error code of last detected error.

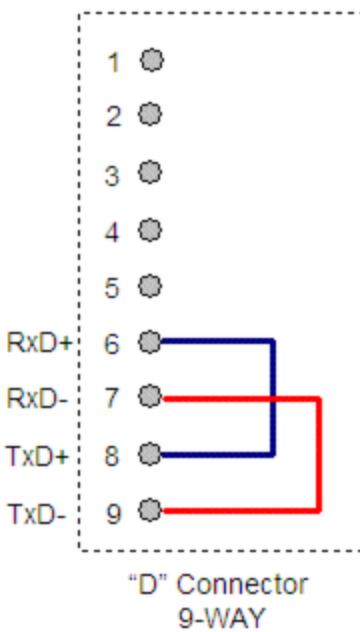
**Note:** The total number of errors should be 0. If the number of errors reported is not zero, your serial hardware is not operating properly.

### Serial Port Loop-back Cable

The diagram below shows the loop-back connections to use with RS-232.



The diagram below shows the loop-back connections to use with RS-422 or RS-485.



## Error Messages

Plant SCADA has two kinds of protocol driver errors:

- generic
- driver-specific

Generic errors are hardware errors 0-31, and are common to every protocol.

Drivers have their own specific errors, which can be unique and therefore cannot be recognized by the hardware alarm system. The drivers convert their specific errors into generic errors that can be identified by the I/O Server.

For example, when a driver becomes inoperative, there is often both a driver-specific error and a corresponding generic error.

---

**Note:** For reference information related to the implementation of device drivers, including driver alert messages, please refer to the [The Driver Reference Help](#).

---

## See Also

- [Protocol Generic Errors](#)
- [Protocol-Specific Errors](#)
- [Standard Driver Errors](#)

### Protocol Generic Errors

Plant SCADA has two kinds of protocol driver errors: generic and [Protocol-Specific Errors](#). Generic errors are hardware errors 0-31, and are common to every protocol.

Protocol drivers also have their own specific errors, which can be unique and therefore cannot be recognized by

the hardware alarm system. The drivers convert their specific errors into generic errors that can be identified by the I/O Server. For example, when a driver has a fault, there is often both a protocol-specific error and a corresponding generic error.

## Generic Errors

The table below describes the generic protocol errors.

Error number	Error title	Description
1	Address is out of range	A request was made to a device address that does not exist. For example, you tried to read register number 4000 when there are only 200 registers in the I/O device. Check the Variable Tags database to find the variable in error.
2	Command canceled	The server canceled the command while it was being processed by the driver. The driver may have taken too long to process the command. If a driver does not respond during the specified time limit, Plant SCADA cancels the command. The time limit is the product of the timeout period and the number of times to retry a command after each timeout. You can increase these values in the Timeout and Retry parameters for the protocol. also check the WatchTime parameter for the frequency with which the driver checks the link to the I/O device. Check also for communication errors.
3	Unknown data type	A request was made that specified a data type not supported by the protocol. This error will not occur during normal operation. Restart the computer to reset every driver and hardware. If the problem persists, contact Technical Support for this product. If you have written your own protocol driver, this error is caused by a mismatch in the compiler specification and the driver's database.

Error number	Error title	Description
4	Unknown data format	A write request contains invalid data, for example you tried to write to a floating-point address with an invalid floating-point number. Check the Plant SCADA database.
5	Command is unknown	The server sent a command that the driver did not recognize. This error will not occur during normal operation. Try re-booting the computer to reset drivers and hardware. If the problem persists, contact Technical Support of this product.
6	Response bad or garbled	A problem exists with the communication channel, causing errors in the transmitted data. Inspect the setup for the communication channel hardware. For example, there may be a mismatch in parity, baud rate, stop bits, or data bits between the transmitter and receiver. Check that the setup of the I/O device and the field data in the Plant SCADA Ports and Boards forms are the same.
7	I/O device not responding	An I/O device is not responding to read or write requests. The driver sent a command to the I/O device and the I/O device did not respond within the timeout period. This is usually the first indication of loss of communications. Check that the I/O device is correctly connected to the server and is switched on. This error can also occur if the timeout period is too short. Try increasing the timeout period in the Timeout parameter for the protocol. You could also increase the delay time between receiving a response and sending the next command, by increasing the Delay parameter.
8	General error	Plant SCADA has established

Error number	Error title	Description
		<p>communications with the I/O device; however, the I/O device has detected an error in the protocol. This error could be caused by a fault in the communications link, or an error in the ladder logic (in the I/O device).</p> <p>Solution:</p> <ol style="list-style-type: none"> <li>1. Check that the I/O device is operating correctly.</li> <li>2. Check the communication cable is connected correctly (at both ends).</li> <li>3. Use the Device Communications Wizard to check that the configuration of the I/O device (in particular, the Address and Special Options fields) matches the recommended settings and the settings on the I/O device.</li> <li>4. If you are using serial communications, use the Device Communications Wizard to check that the configuration of the Port (in particular the Baud Rate, Data Bits, Stop Bits, and Parity) matches the recommended settings and the settings on the I/O device.</li> <li>5. Display the hardware alarm page, and note the protocol error that is displayed.</li> <li>6. Use the documentation that was supplied with your I/O device, network, and communication board to locate the error.</li> <li>7. Check the ladder logic in the I/O device for errors.</li> <li>8. Run the Setup Wizard.</li> <li>9. Re-compile the project and start the Plant SCADA runtime.</li> </ol>
9	Write location is protected	A write operation was attempted to a location that is protected against unauthorized modification. Change

Error number	Error title	Description
		the access rights to this location to permit a write operation.
10	Hardware error	A problem exists with either the communication channel, server, or I/O device hardware. Examine hardware components. The command or data request has not been processed. The server's operation may no longer operate normally.
11	I/O device warning	The communication link between the server and the I/O device is functioning correctly, however the I/O device has some alert condition active, for example the I/O device is in program mode. Check that the I/O device is in the correct mode.
12	I/O device off-line, cannot talk	<p>The I/O device is in off-line mode, preventing any external communication.</p> <p><b>Solution:</b></p> <ol style="list-style-type: none"> <li>1. Check that the I/O device is operating correctly.</li> <li>2. Check the communication cable for breakage.</li> <li>3. Check the communication cable is connected correctly (at both ends).</li> <li>4. If you are using serial communications, check that the communication cable matches the diagram in the help system.</li> <li>5. Use the Device Communications Wizard to check that the configuration of the I/O device (in particular, the Address and Special Options fields) matches the recommended settings and the settings on the I/O device.</li> <li>6. If you are using serial communications, use the Device Communications Wizard to check</li> </ol>

Error number	Error title	Description
		<p>that the configuration of the port (in particular the baud rate, data bits, stop bits, and parity) matches the recommended settings and the settings on the I/O device.</p> <p>7. Run the Setup Wizard.</p> <p>8. Check the Citect.ini file for the following:</p> <p>[IOSERVER]</p> <p>Server=1</p> <p>Name=&lt;name&gt;</p> <p>where:</p> <p>&lt;name&gt; is the name of the server configured in the Plant SCADA project. (Use Custom Setup to check the server name.)</p> <p>9. Re-compile the project and start the Plant SCADA runtime system.</p> <p><b>Note :</b> If you have standby I/O devices configured, this error will cause any standby I/O devices to become active. The command or data request current when the I/O device went off-line has not finished.</p>
13	Driver software error	An internal software error has occurred in the driver. This error should not occur during normal operation. Try re-booting the computer to reset drivers and hardware. If the problem persists, contact Technical Support of this product.
14	User access violation	An attempt has been made by an unauthorized user to access information. Check the user's access rights.
15	Out of memory - FATAL	The server is out of memory and cannot continue execution. Minimize buffer and queue allocation or expand memory in the

Error number	Error title	Description
		server computer. The command or data request has not been processed.
16	No buffers, cannot continue	There are no communication buffers available to be allocated, or the computer is out of memory. The performance of the server may be reduced, however it can continue to run. Increase the memory.
17	Low buffer warning	This error may occasionally occur in periods of high transient loading, with no ill effects. If this error occurs frequently, increase the number of communication buffers.
18	Too many commands to driver	Too many commands have been sent to the driver.
19	Driver is not responding	The server is not receiving any response from the driver. This error should not occur during normal operation. Try re-booting the computer to reset the drivers and hardware. If the problem persists, contact Technical Support of this product.
20	Too many channels opened	Each driver can only support several communication channels. You have exceeded the limit. This error may occur if you abnormally terminate from the server and then restart it. Try re-booting the computer to reset drivers and hardware. If the problem persists, contact Technical Support of this product. The command or data request has not finished.
21	Channel off-line, cannot talk	A communication channel is currently off-line, disabling communication. Either the server cannot initialize the communication channel or the channel went off-line while running. Check the

Error number	Error title	Description
		channel hardware for errors. When this error occurs, I/O devices connected to this channel are considered off-line, and standby I/O devices become active. The command or data request has not finished.
22	Channel not yet opened	The server has attempted to communicate with a channel that is not open. Try re-booting the computer to reset drivers and hardware. If the problem persists, contact Technical Support for this product.. The command or data request has not finished.
23	Channel not yet initialized	The server is attempting to communicate with a channel that has not been initialized. This error should not occur during normal operation. Try re-booting the computer to reset drivers and hardware. If the problem persists, contact Technical support for this product. The command or data request has not finished.
24	Too many I/O devices per channel	A channel has too many I/O devices attached to it. This error should not occur during normal operation. The command or data request has not finished. Try re-booting the computer to reset drivers and hardware. If the problem persists, contact Contact Technical Support for this product..
25	Data not yet valid	The data requested is still being processed and will be returned in due course. This error only occurs with drivers that need to establish complex communication to retrieve data from the I/O device. Ignore this alert.
26	Could not cancel command	The server tried to cancel a

Error number	Error title	Description
		command, but the driver could not find the command. This error should not occur during normal operation. Try re-booting the computer to reset drivers and hardware. If the problem persists, contact Contact Technical Support for this product..
27	Stand-by I/O device activated	Communication has been switched from the primary to the standby I/O device(s). The server returns this message when a "hot" changeover has occurred. Rectify the error in the primary I/O device(s).
28	Message overrun	A response was longer than the response buffer. If this error occurs on serial communication drivers, garbled characters may be received. Check the communication link and the baud rate of the driver.
29	Bad user parameters	There is a configuration error, for example invalid special options have been set.
30	Stand-by I/O device error	There is an error in a standby I/O device. Rectify error in the standby I/O device.
31	Request Timeout from I/O Server	One or more requests sent to the I/O Server have not finished in the timeout period. Either the I/O Server is off line or the I/O Server is taking too long to finish the requests. Check the PLC communication link, PLC timeouts, PLC retries, and network communication. This error can occur even if you have no network, i.e. if the I/O Server is the same computer as the Control Client. If the error persists, increase the [LAN] TimeOut parameter. The default timeout is 8000

Error number	Error title	Description
		milliseconds.
32	Cannot talk to remote unit	The remote I/O device is not connected.
274	Invalid argument passed	An invalid argument has been passed to a Cicode function. This is a general error message and is generated when arguments passed to a function are out of range or are invalid. Check the value of arguments being passed to the function. If arguments are input directly from the operator, check that the correct arguments are being passed to the function.
281	No server could be found	The specified Plant SCADA server cannot be found. Either the server is not running or there is some communication problem with the network. Check that the network is set up correctly, and you are using the same Server Name on both the client and server.
418	No server of type on cluster	There is no server of the necessary type configured on the server.
448	Record size mismatch	An RDB file contains records with the wrong size.
451	Server previous reload busy	Unable to start a reload using the ServerReload Cicode function as a reload is already in progress for the specified server.
452	Invalid RDB version	An RDB file was compiled using an incompatible version of the software.
454	Cicode library timestamp differs	The timestamp of the Cicode library in memory is different from the timestamp of the Cicode library on disk. The Cicode libraries are potentially different.
519	Remote Cicode call Interrupted	Cicode call that triggers an RPC

Error number	Error title	Description
		remote call is interrupted before the expected result is returned.
520	Alarm category out of range	A category number is out of its valid range (from 0 to 16376 inclusively).
521	Data browse record is deleted	A record was deleted during a reload of ART server.
522	Trend archive property mismatch	A trend record's archive properties have changed during start-up or reload.
523	Alarm priority out of range	A category priority is out of its valid range (from 0 to 256 inclusively).

## Generic Driver Errors

The following errors are generic to every Plant SCADA driver. A driver error needs to be mapped to a generic error before Plant SCADA can interpret it.

Error	Description
GENERIC_ADDRESS_RANGE_ERROR (0x0001   SEVERITY_ERROR)	A request was made to a device address that does not exist. For example, an attempt was made to read register number 4000 when there are only 200 registers in the device.
GENERIC_CMD_CANCELED (0x0002   SEVERITY_ERROR)	The server canceled the command while the driver was processing it. This can happen if the driver takes too long to process the command. Check the timeout and retries for the driver.
GENERIC_INVALID_DATA_TYPE (0x0003   SEVERITY_ERROR)	A request was made specifying a data type not supported by the protocol. This error will not occur during normal operation.
GENERIC_INVALID_DATA_FORMAT (0x0004   SEVERITY_ERROR)	A request contains invalid data; for example, writing to a floating-point address with an invalid floating-point number. Check the Plant SCADA database.
GENERIC_INVALID_COMMAND (0x0005   SEVERITY_ERROR)	The server sent a command to the driver that it did not recognize. This error will not occur during normal operation.
GENERIC_INVALID_RESPONSE	The communication channel is not performing normally, and is causing errors in the transmitted data.

<b>Error</b>	<b>Description</b>
(0x0006   SEVERITY_ERROR)	
GENERIC_UNIT_TIMEOUT (0x0007   SEVERITY_ERROR)	A device is not responding to read or write requests. The driver sent a command to the device and the device did not respond within the timeout period.
GENERIC_GENERAL_ERROR (0x0008   SEVERITY_ERROR)	Unmapped driver specific errors are normally reported as a general error. Refer to the protocol-specific errors listed with the protocol you are using.
GENERIC_WRITE_PROTECT (0x0009   SEVERITY_ERROR)	A write operation was attempted to a location that is protected against unauthorized modification. Change the access rights to this location to permit a write operation.
GENERIC_HARDWARE_ERROR (0x000A   SEVERITY_UNRECOVERABLE)	The communication channel, server, or device hardware is not performing normally. Examine hardware components. The server's operation needs to also be examined for proper operation.
GENERIC_UNIT_WARNING (0x000B   SEVERITY_WARNING)	The communication link between the server and the device is functioning correctly; however, the device is experiencing an error or is in a non-operational state, for example, the device is in program mode.
GENERIC_UNIT_OFFLINE (0x000C   SEVERITY_SEVERE)	The device is in offline mode, preventing any external communication. This error will cause any stand-by units to become active. Plant SCADA will attempt to re-initialize the unit.
GENERIC_SOFTWARE_ERROR (0x000D   SEVERITY_SEVERE)	An internal software error has occurred in the driver. This error should not occur during normal operation.
GENERIC_ACCESS_VIOLATION (0x000E   SEVERITY_ERROR)	An attempt has been made by an unauthorized user to access information. Check the user's access rights.
GENERIC_NO_MEMORY (0x000F   SEVERITY_UNRECOVERABLE)	The server or driver has run out of memory and cannot continue execution. Minimize buffer and queue allocation or expand the server computer's memory (physical or virtual memory).
GENERIC_NO_BUFFERS (0x0010   SEVERITY_ERROR)	There are no communication buffers left to allocate. The performance of the server may be reduced; however, it can still continue to run. Increase the number of communication buffers.
GENERIC_LOW_BUFFERS (0x0011   SEVERITY_WARNING)	This error may occur in periods of high transient loading with no ill effects. If this error occurs

Error	Description
	frequently, increase the number of communication buffers.
GENERIC_TOO_MANY_COMMANDS (0x0012   SEVERITY_WARNING)	Too many commands have been sent to the driver.
GENERIC_DRIVER_TIMEOUT (0x0013   SEVERITY_ERROR)	The server is not receiving any response from the driver. This error should not occur during normal operation.
GENERIC_NO_MORE_CHANNELS (0x0014   SEVERITY_SEVERE)	Each driver can only support a fixed number of communication channels. You have exceeded the limit. The command or data request has not been completed.
GENERIC_CHANNEL_OFFLINE (0x0015   SEVERITY_SEVERE)	A communication channel is currently offline, disabling communication. The server cannot initialize the communication channel or the channel went offline while running. Every device (units)connected using this channel will be considered to be offline so this will cause any stand-by devices to become active. Plant SCADA will attempt to re-initialize the channel.
GENERIC_BAD_CHANNEL (0x0016   SEVERITY_SEVERE)	The server has attempted to communicate using a channel that is not open.
GENERIC_CHANNEL_NOT_INIT (0x0017   SEVERITY_SEVERE)	The server is attempting to communicate with a channel that has not been initialized. This error should not occur during normal operation. The command or data request has not been completed. If the condition persists, contact support.
GENERIC_TOO_MANY_UNITS (0x0018   SEVERITY_SEVERE)	A channel has too many devices attached to it. This error should not occur during normal operation.
GENERIC_INVALID_DATA (0x0019   SEVERITY_ERROR)	The data requested is not in a valid format or expected type.
GENERIC_CANNOT_CANCEL (0x001A   SEVERITY_WARNING)	The server tried to cancel a command, but the driver could not find the command. This error should not occur during normal operation.
GENERIC_STANDBY_ACTIVE (0x001B   SEVERITY_WARNING)	Communication has been switched from the primary to the stand-by unit(s). The server returns this message when a hot changeover has occurred.
GENERIC_MSG_OVERRUN	A response was longer than the response buffer. If this

Error	Description
(0x001C   SEVERITY_ERROR)	error occurs on serial communication drivers, garbled characters may be received. Check the communication link and the baud rate of the driver.
GENERIC_BAD_PARAMETER (0x001D   SEVERITY_ERROR)	There is a configuration error, for example invalid special options have been set.
GENERIC_STANDBY_ERROR (0x001E   SEVERITY_WARNING)	There is an error in a stand-by unit.
GENERIC_NO_RESPONSE (0x001F   SEVERITY_ERROR)	There is no response from the communications server.
GENERIC_UNIT_REMOTE (0x0020   SEVERITY_ERROR)	Cannot talk with remote unit (for example dial-up I/O Devices). Only used for scheduled I/O Devices.
GENERIC_GENERAL_WARNING (0x0024   SEVERITY_WARNING)	The driver is performing the action requested, but needs to notify of a potential issue. For example, some drivers may use this to alert you to stale data.

### Protocol-Specific Errors

Though each protocol may have multiple unique errors, the first 34 protocol-specific errors are standard for every protocol. Every protocol-specific error is also reported under error numbers 1 to 31 above. Although these errors have their own error number (also given in hexadecimal), it is only used as a notation.

**Note:** Errors that are protocol-specific are listed in the Protocol-Specific Errors help topic for each driver. Refer to the documentation that was supplied with your I/O Device if you cannot locate an error description.

Error Number	Error Title	Description
1 (0x01)	Cannot process received characters fast enough	Cannot process received characters fast enough. Lower the baud rate or use a faster computer. If the error persists, contact Technical Support for this product.
2 (0x02)	Parity error	The received message has a parity error. Check that the correct baud rate, parity, stop bits, and data bits are specified in the Plant SCADA Ports form. This error may be caused by a disconnected cable to the I/O Device or by excessive noise on the communication link.
3 (0x03)	Break detected in receive line	A break has been detected in the

Error Number	Error Title	Description
		receive line. This error may be caused by a disconnected cable to an I/O Device or by excessive noise on the communication link.
4 (0x04)	Framing error	The wrong baud rate may have been specified. Check that the correct baud rate is specified in the Plant SCADA Ports form.
5 (0x05)	Message too long	The message received from the I/O Device is too long. This error may be caused by a disconnected cable to an I/O Device or by excessive noise on the communication link. Contact Technical Support for this product. if the error continues.
6 (0x06)	Invalid checksum	The checksum in the received message does not match the calculated value. Check that the correct baud rate, parity, stop bits, and data bits are specified in the Plant SCADA Ports form. This error may be caused by a disconnected cable to the I/O Device or by excessive noise on the communication link. You can also try increasing the number of retries in the Retry parameter for the protocol.
7 (0x07)	Start of text missing	A start of text (STX) character is not present in the received message. Check that the correct baud rate, parity, stop bits, and data bits are specified in the Plant SCADA Ports form. This error may be caused by a disconnected cable to the I/O Device or by excessive noise on the communication link.
8 (0x08)	End of text missing	An end of text (ETX) character is not present in the received message. Check that the correct baud rate, parity, stop bits, and data bits are specified in the Plant

Error Number	Error Title	Description
		SCADA Ports form. This error may be caused by a disconnected cable to the I/O Device or by excessive noise on the communication link.
10 (0x0A)	Cannot transmit message	Plant SCADA cannot transmit the message. This error may be caused by a disconnected cable to an I/O Device or by excessive noise on the communication link.
11 (0x0B)	Cannot reset serial driver	An error has occurred with the serial (COMXI, PCXI, or COMx) driver. Try re-booting the computer to reset drivers and hardware.
12 (0x0C)	Length of request inconsistent	The length of a request is not consistent with the driver's requirements.
15 (0x0F)	Command from server invalid	The command from the server is invalid. Contact Technical Support for this product.
16 (0x10)	Cannot allocate timer resource for driver	Driver timer resources cannot be allocated. Contact Technical Support for this product.
17 (0x11)	Too many channels specified for driver	Too many channels have been specified for the device. Contact Technical Support for this product.
18 (0x12)	Channel number from server not opened	The channel number from the server is not open. Contact Technical Support for this product.
19 (0x13)	Command cannot be cancelled	A driver command cannot be cancelled. Contact Technical Support for this product.
20 (0x14)	Channel not on-line	The channel is not on-line. This error can occur if timeouts are occurring, and may be caused by a disconnected cable to an I/O Device or by excessive noise on the communication link.
21 (0x15)	Timeout error	No response was received from the

Error Number	Error Title	Description
		I/O Device within the specified timeout period. This error may be caused by a disconnected cable to the I/O Device or by excessive noise on the communication link. You can try increasing the number of retries in the Retry parameter for the protocol.
22 (0x16)	I/O Device number from server not active or out of range	The I/O Device number from the server is not active or is out of range. Contact Technical Support for this product.
23 (0x17)	I/O Device not on-line	Check that the I/O Device Address specified in the Plant SCADA I/O Devices form is the same as that configured on the I/O Device.
24 (0x18)	Data type from server unknown to driver	The data type from the server is unknown to the driver. Contact Technical Support for this product.
25 (0x19)	I/O Device type from server unknown to driver	The I/O Device type from the server is unknown to the driver. Contact Technical Support for this product.
26 (0x1A)	Too many I/O Devices specified for channel	Too many I/O Devices have been specified for the channel. Contact Technical Support for this product.
27 (0x1B)	Too many commands issued to driver	Too many commands have been issued to the driver. Contact Technical Support for this product.
28 (0x1C)	Data read invalid	The data read is not valid. Contact Technical Support for this product.
29 (0x1D)	Command is cancelled	A driver command has been cancelled. Contact Technical Support for this product.
30 (0x1E)	Address invalid or out of range	The address you tried to access has an invalid data type or is out of range. Check that you are using data types and ranges of addresses that are valid for the I/O Device.

Error Number	Error Title	Description
31 (0x1F)	Data length from server incorrect	The data length from the server is wrong. Contact Technical Support for this product.
32 (0x20)	Cannot read data from device	Plant SCADA cannot read the data from the I/O Device. Contact Technical Support for this product.
33 (0x21)	Device specified does not exist	The device specified does not exist. Contact Technical Support for this product.
34 (0x22)	Device specified does not support interrupt	The I/O Device specified does not support interrupt handling. You have specified an interrupt, either on the Boards form or by setting the PollTime parameter to 0, for a hardware device that does not support interrupts. Check the interrupt set for the board and set the PollTime parameter for the protocol.

### Standard Driver Errors

The following errors are low-level errors which are generic to every driver. These errors are mapped to Generic errors so that Plant SCADA can recognize them. Most drivers also have a set of driver specific errors in addition to these errors.

Error	Description
0 (0x00000000) NO_ERROR	No error condition exists.
1 (0x00000001) DRIVER_CHAR_OVERRUN	Transmitted characters could not be received fast enough. This error mapped to Generic Error GENERIC_INVALID_RESPONSE.
2 (0x00000002) DRIVER_CHAR_PARITY	Parity error in received characters. This error mapped to Generic Error GENERIC_INVALID_RESPONSE.
3 (0x00000003) DRIVER_CHAR_BREAK	A break was detected in the receive line. This error mapped to Generic Error GENERIC_INVALID_RESPONSE.
4 (0x00000004) DRIVER_CHAR_FRAMING	Framing error. Check the baud rate. This error mapped to Generic Error GENERIC_INVALID_RESPONSE.

Error	Description
5 (0x00000005) DRIVER_MSG_OVERRUN	The message received from the device was too long. This error mapped to Generic Error GENERIC_INVALID_RESPONSE.
6 (0x00000006) DRIVER_BAD_CRC	Checksum in received message does not match the calculated value. Error mapped to Generic Error GENERIC_INVALID_RESPONSE.
7 (0x00000007) DRIVER_NO_STX	Start of text character not present. Error is mapped to Generic Error GENERIC_INVALID_RESPONSE.
8 (0x00000008) DRIVER_NO_ETX	End of text character not present. Error is mapped to Generic Error GENERIC_INVALID_RESPONSE.
9 (0x00000009) DRIVER_NOT_INIT	Driver has not been initialized. This error is mapped to Generic Error GENERIC_UNIT_OFFLINE.
10 (0x0000000A) DRIVER_BAD_TRANSMIT	Cannot transmit message. This error is mapped to Generic Error GENERIC_UNIT_OFFLINE.
11 (0X0000000B) DRIVER_CANNOT_RESET	Cannot reset serial driver. This error is mapped to Generic Error GENERIC_CHANNEL_OFFLINE.
12 (0X0000000C) DRIVER_BAD_LENGTH	Response length is incorrect. This error is mapped to Generic Error GENERIC_GENERAL_ERROR.
13 (0X0000000D) DRIVER_MSG_UNDERRUN	Message length too short. This error is mapped to Generic Error GENERIC_INVALID_RESPONSE.
15 (0X0000000F) DRIVER_INVALID_COMMAND	The command from the server is invalid. This error is mapped to Generic Error GENERIC_INVALID_COMMAND.
16 (0X00000010) DRIVER_NO_TIMER	Cannot allocate timer resource for the driver. This error is mapped to Generic Error GENERIC_HARDWARE_ERROR.
17 (0x00000011) DRIVER_NO_MORE_CHANNELS	Too many channels specified for device. This error is mapped to Generic Error GENERIC_NO_MORE_CHANNELS.
18 (0x00000012) DRIVER_BAD_CHANNEL	The channel number from the server is not opened. This error is mapped to Generic Error GENERIC_BAD_CHANNEL.
19 (0x00000013) DRIVER_CANNOT_CANCEL	Command cannot be cancelled. This error is mapped to Generic Error GENERIC_CANNOT_CANCEL.

Error	Description
20 (0x00000014) DRIVER_CHANNEL_OFFLINE	The channel is not online. This error is mapped to Generic Error GENERIC_CHANNEL_OFFLINE.
21 (0x00000015) DRIVER_TIMEOUT	No response have been received within the user configure time. This error is mapped to Generic Error GENERIC_UNIT_TIMEOUT.
22 (0x00000016) DRIVER_BAD_UNIT	The unit number from the server is not active or is out of range. This error is mapped to Generic Error GENERIC_UNIT_OFFLINE.
23 (0x00000017) DRIVER_UNIT_OFFLINE	The unit is not online. This error is mapped to Generic Error GENERIC_UNIT_OFFLINE.
24 (0x00000018) DRIVER_BAD_DATA_TYPE	The data type from the server is unknown to the driver. This error is mapped to Generic Error GENERIC_INVALID_DATA_TYPE.
25 (0x00000019) DRIVER_BAD_UNIT_TYPE	The unit type from the server is unknown to the driver. This error is mapped to Generic Error GENERIC_INVALID_DATA_TYPE.
26 (0x0000001A) DRIVER_TOO_MANY_UNITS	Too many units specified for channel. This error is mapped to Generic Error GENERIC_TOO_MANY_UNITS.
27 (0x0000001B) DRIVER_TOO_MANY_COMMANDS	Too many commands have been issued to the driver. This error code can also occur if you are running a restricted version of a driver (i.e. one that will run for a limited time) for every issued read and write. This error is mapped to Generic Error GENERIC_TOO_MANY_COMMANDS.
29 (0x0000001D) DRIVER_CMD_CANCELED	Command is cancelled. This error is mapped to Generic Error GENERIC_COMMAND_CANCELLED.
30 (0x0000001E) DRIVER_ADDRESS_RANGE_ERROR	The address/length is out of range. This error is mapped to Generic Error GENERIC_ADDRESS_RANGE_ERROR.
31 (0x0000001F) DRIVER_DATA_LENGTH_ERROR	The data length from the server is wrong. This error is mapped to Generic Error GENERIC_INVALID_RESPONSE.
32 (0x00000020) DRIVER_BAD_DATA	Cannot read the data from the device. This error is mapped to Generic Error GENERIC_INVALID_DATA.
33 (0x00000021)	Device specified does not exists. This error is mapped

Error	Description
DRIVER_DEVICE_NOT_EXIST	to Generic Error GENERIC_HARDWARE_ERROR.
34 (0x00000022) DRIVER_DEVICE_NO_INTERRUPT	Device specified does not support interrupt. This error is mapped to Generic Error GENERIC_HARDWARE_ERROR.
35 (0x00000023) DRIVER_BAD_SPECIAL	Invalid special options in port database. This error is mapped to Generic Error GENERIC_BAD_PARAMETER.
36 (0x00000024) DRIVER_CANNOT_WRITE	Cannot write to variable. This error is mapped to Generic Error GENERIC_GENERAL_ERROR.
37 (0x00000025) DRIVER_NO_MEMORY	The driver has run out of memory and cannot continue execution. Minimize buffer and queue allocation or expand the computer's memory (physical or virtual memory). This error is mapped to Generic Error GENERIC_NO_MEMORY.

## Profiles



### UNINTENDED EQUIPMENT OPERATION

- Do not under any circumstances change or remove any of the undocumented <profile\_name>.ini parameters.
- Before deleting sections of the <profile\_name>.ini file, confirm that no undocumented parameters will be deleted.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

A profile stores runtime parameter settings that are specific to a particular type of computer within a Plant SCADA system (for example, a client or server). These settings are stored in a "<profile\_name>.ini" file that can be distributed to any computer where a project is deployed.

Profiles are created in the **Topology** activity using the [Profile Wizard](#).

The Plant SCADA Example Project includes three profiles that you can select from the [Profile Setup](#) page in the Setup Wizard. Once you have selected the profile you would like, you can compile and run the Example project.

You can use the **Deployment** activity to distribute Profile.ini files to the computers in a Plant SCADA system (see [Deployment Computers](#)).

To use an INI file you create with the Profile Wizard, either select it locally on a computer using the Computer Setup Wizard, or use the deployment feature.

### See Also

[Add a Profile](#)

## Add a Profile

### To add a Profiles:

1. In the **Topology** activity, select **Profiles**
2. On the Command Bar, click the **Add/Edit Profile** button. This launches the [Profile Wizard](#).

## See Also

[Run the Profile Wizard](#)

## Profile Wizard

The Profile Wizard is similar to the [Setup Wizard](#) in that it allows you to configure parameters for a specific computer. Configuration information is stored in an INI file which can be deployed to the for the specific computer for which it was configured.

The Profile Wizard contains a series of pages allowing configuration of computer-specific settings including:

- The role the computer has in the system network
- The project being run
- The CPU Configuration
- The Plant SCADA Events enabled for each component
- The Cicode run for each component on startup
- The cluster configuration
- The security settings applied.

Parameter values collected from the user through the Wizard interface are written to the <ProfileName>.ini file.

---

**Note:** To use a profile, either select it locally on a computer using the Computer Setup Wizard, or use the deployment feature (see Deployment Computers). The values in the profile.ini will take precedence over the values set in the local Citect.ini. However, values that are set for the parameters [\[CtEdit\]Run](#), [\[CtEdit\]Bin](#), [\[CtEdit\]Data](#), [\[CtEdit\]Logs](#), [\[CtEdit\]Config](#) will be taken from the local citect.ini and not from the profile.ini.

---

See [Run the Profile Wizard](#) for more information.

## Run the Profile Wizard

### To run the Profile Wizard:

1. Select a project in the Project activity. If not previously compiled, compile the project.
2. In the **Topology** activity, select **Profiles**.
3. On the Command Bar, click **Add/Edit Profile**. The Profile Setup Wizard is displayed.
4. Select **Express Setup** or **Custom Setup**.

The pages that are displayed depend on the configuration of the computer. They can include:

- [Profile Setup](#)
- [Add a New Profile](#)
- [Select a Profile](#)
- [Select a Node](#)
- [Select a Screen Profile](#)
- [Configure the Computer Role](#)
- [Configure the Network Model](#)
- [Configure Reports](#)
- [Configure Trends](#)
- [Configure Alarms](#)
- [Configure CPU](#)
- [Configure Events](#)
- [Configure Startup Functions](#)
- [Configure Cluster Connections \\*](#)
- [Configure the Server Password](#)
- [Configure Server User](#)
- [Configure Control Menu Security](#)
- [Configure Keyboard Security \\*](#)
- [Configure Miscellaneous Security \\*](#)
- [Configure General Options \\*](#)

\* Only available in Custom Setup mode.

Each screen of the wizard is described in the sections that follow.

## See Also

[Profile Wizard](#)

### Profile Setup

In the Profile Setup screen you can select the name of an existing profile to edit from the drop-down list, or choose the option to add a new profile.

- [Select a Profile](#)
- [Add a New Profile](#)

### Add a New Profile

#### To Add a New Profile:

1. Select the project that will host the Profile ini.
2. Select the **Add a New Profile** option.

3. Click **Next**.
4. In the **Profile Name** field enter the name of the profile.

---

**Note:** Profile names with spaces and the following characters <>:\\"/\?\*. are invalid and will generate a compile error message.

---

5. In the Comment field, add a description for the profile.
6. Click Next.

## See Also

[Select a Node](#)

### Select a Profile

Use this option to edit an existing Profile ini file belonging to the selected project.

The **Project Name** will display the name of the currently selected project.

1. In the **Profile Name** field, select the <profile>.ini file you want to edit.
2. Click **Next**.

## See Also

[Select a Node](#)

### Select a Node

Select whether the machine (the profile.ini belongs to) will be a client or server.

1. If Client , select the option **Configure Client Profile** before clicking **Next**.
2. If Server, select the option **Configure Server Profile**.
  - Enter the IP address of the machine you are configuring.
3. Click **Next**.

## See Also

[Select a Screen Profile](#)

### Select a Screen Profile

Use the Screen Setup page to select the Screen Profile that will be used. To do this:

1. From the **Screen Profile** list, select the profile you wish to use for the current workstation. The monitors defined for the selected profile are displayed.
2. For each screen, select a **Startup Page**. A list of available pages can be accessed from a drop-down menu. To just see a list of master pages (pages with their Content Type set to "Master"), select the **Show master pages**

only check box.

You can also use the parameter [\[MultiMonitors\]StartupPage<n>](#) to specify a Startup Page.

3. If your project is a [Situational Awareness](#) project, you will need to select a **Context** for each screen. The context sets the entry point of display for the screen; it could be an area of the plant or the entire site or any graphics page in your project. This can also be done through the [\[MultiMonitors\]Context<n>](#) parameter.

**Note:** Context is only supported for Situational Awareness projects.

## See Also

[Configure the Computer Role](#)

### Configure the Computer Role

Use the Computer Role Setup page to specify the role of the computer running Plant SCADA. Select one of the options described below.

**Note:** To use Plant SCADA's multi-process capabilities with a Server and Control Client, networking needs to be enabled.

Option	Description
Server and Control Client	<p>This computer will be a standalone or networked I/O server and Control Client. This option is disabled if this computer has no Server components assigned to it to run. Selecting this option enables the <b>Multi-Process</b> check box.</p> <p>Select the <b>Multi-Process</b> check box to separate your client and server components into individual processes. This option can be used for distributing the components across multiple CPUs.</p> <p>If the <b>Multi-Process</b> check box is selected the <a href="#">[General]Multiprocess</a> parameter in the Citect.ini file is saved with the value 1. If not selected, the parameter is saved with the value 0.</p> <p>If you leave the <b>Multi-Process</b> check box unselected, Plant SCADA will run the client and server components in one process.</p> <p><b>Note:</b> Running Plant SCADA as a service in single process mode is not recommended. If you run Plant SCADA in this condition, the following limitations will apply:</p> <ul style="list-style-type: none"><li>• Hardware alarms originating from the server process will not appear.</li><li>• The following Cicode functions will not be</li></ul>

Option	Description
	<p>executed within the server process:</p> <ul style="list-style-type: none"><li>• AlarmSumAppend</li><li>• AlarmSumCommit</li><li>• AlarmSumDelete</li><li>• AlarmSumFind</li><li>• AlarmSumFindExact</li><li>• AlarmSumFirst</li><li>• AlarmSumGet</li><li>• AlarmSumLast</li><li>• AlarmSumNext</li><li>• AlarmSumPrev</li><li>• AlarmSumSet</li><li>• AlarmSumSplit</li><li>• AlarmSumType</li><li>• DriverInfo</li><li>• IODeviceControl</li><li>• IODeviceInfo</li><li>• SPCAlarms</li><li>• ServerInfoEx</li><li>• TrnAddHistory</li><li>• TrnDelHistory</li></ul>
Control Client	<p>This computer will only be a Control Client. This option is disabled if the computer has been assigned a server component to run.</p> <p>Selecting this option enables the <b>Full License</b> check box.</p> <p><b>Note:</b> The Full License checkbox is only available when running the Setup Wizard in Custom Mode.</p> <p>Select the full server license check box if you want this Control Client to use a full (server) license, as opposed to a client license. This sets the <a href="#">[Client]FullLicense</a> parameter in the Citect.ini file to 1 (the default value is 0).</p> <p>Generally, you would not allocate a full server license to a client as there is no difference in functionality between the two licenses. Allocate a full server license if you have insufficient client licenses available, or for</p>

Option	Description
	some other specific need.
View-only Client	This computer will be a view-only client. This read only option is disabled if this computer has been assigned a Server component to run.
HMI (Standalone, no networking)	<p>This computer will run as a standalone HMI setup with networking disabled. Select this option for any computers that require a dedicated HMI license from an Enterprise License Server that hosts multiple license types.</p> <p>If you are using Sentinel licensing, you can use a key that has networking disabled. If a full Sentinel key is plugged in, runtime will start in standalone mode with networking disabled.</p> <p>Select the <b>Multi-Process</b> check box to separate your client and server components into individual processes. This option can be used for distributing the components across multiple CPUs.</p> <p>If the <b>Multi-Process</b> check box is selected the <a href="#">[General]Multiprocess</a> parameter in the Citect.ini file is saved with the value 1. If not selected, the parameter is saved with the value 0.</p>

Some of these options may be disabled depending on what servers have been configured to run on this computer. The Setup Wizard cross-references your computer's network identification with the network addresses configured for each server in your project configuration.

The option you select will adjust the [\[Client\]ComputerRole](#) parameter in the Citect.ini file.

## See Also

[Configure the Network Model](#)

### Configure the Network Model

Select the network model to be applied to this Plant SCADA computer. The options include:

- Stand alone (no other SCADA computers)
- Networked (connect to other SCADA computers)

---

**Note:** TCP/IP address information for servers is configured within the Plant SCADA project itself. See [Add a Network Address](#) for more information.

---

## Stand alone

Select **stand alone** to run all server and client components of a Plant SCADA system on a single computer. This allows you to run Plant SCADA as a small and self-contained system.

## Networked

Select **Networked** to connect to a system of multiple Plant SCADA computers configured with different server and client components. In a **Networked** setup where hostnames are used to resolve IP addresses, specify the following:

- Select network address type from the **Address Type** drop-down list (refer to [LAN]AddressType).
- For an IPv6 **Address Type**, select the **Address Scope** from the drop-down list (refer to [LAN]AddressScope).

Refer to the following table for information regarding the status of a server with respect to the computer network configuration, the **Address Type** and the **Address Scope** selection.

Computer Network Configuration	Address Type	Address Scope	Server Status
IPv4	IPv4	-	Starts with IPv4 address.
IPv4	IPv6	Link-Local	Starts with the Link-Local IPv6 address available in the machine.
IPv4	IPv6	Global	Does not start.
IPv4	IPv6	Global and Link-Local	Starts with the Link-Local IPv6 address available in the machine. <b>Note:</b> When you start <b>Runtime Manager</b> , the <b>User Location</b> column on the <b>Alarm Summary</b> and <b>Sequence of Events</b> pages shows the server address as "::1." It is the IPv6 loop-back address.
IPv6	IPv4	-	Starts with the IPv4 loop back address 127.0.0.1.
IPv6	IPv6	Link-Local	Starts with the Link-Local IPv6 address.
IPv6	IPv6	Global	Starts with Global IPv6 address.

Computer Network Configuration	Address Type	Address Scope	Server Status
IPv6	IPv6	Global and Link-Local	Starts with Link-Local / Global IPv6 address (depends on first return address).
IPv6 and IPv4	IPv4	-	Starts with IPv4 address.
IPv6 and IPv4	IPv6	Link-Local	Starts with Link-Local IPv6 address.
IPv6 and IPv4	IPv6	Global	Starts with Global IPv6 address.
IPv6 and IPv4	IPv6	Global and Link-Local	Starts with Link-Local / Global IPv6 address (depends on first return address).

When you complete the Profile Setup Wizard, the selected network model is written to the [LAN] section in the ini file; for example:

```
...
[LAN]
TCPIP=1
...
```

## See Also

[Configure Reports](#)

### Configure Reports

The Reports Configuration page will only be displayed if this machine is configured as a Reports Server in the [Topology](#) activity.

---

**Note:** For a networked computer to be a Reports Server it needs to also be the I/O Server or needs to be able to communicate with the I/O Server on the network.

Plant SCADA has several options available for report processing:

Option	Description
Startup report	Defines the name of the report to run when Plant SCADA starts up.
Inhibit triggered reports on startup	For example, you might have a report that is triggered off the rising edge of a bit on startup. The Reports Server detects the bit come on, and runs the report. If this option is checked, the Reports Server does not

Option	Description
	run this report until it has read the I/O Devices a second time.
Run reports concurrently with primary Reports Server	Enables or disables tandem processing of reports. If this server is the standby Reports Server, it can process every report in tandem with the primary server, or it can remain idle until called.

## See Also

[Configure Trends](#)

### Configure Trends

The Trends Configuration page will only be displayed if this machine is configured as a Trends Server in the [Topology](#) activity.

---

**Note:** For a networked computer to be a Trends Server it needs to also be the I/O Server or needs to be able to communicate with the I/O Server on the network.

Plant SCADA has one option available for trend processing:

Option	Description
Inhibit triggered trends on startup	You might have a trend that is triggered off the rising edge of a bit on startup. If this option is enabled, the trends server does not display the trend until it has read the I/O Devices a second time.

## See Also

[Configure Alarms](#)

### Configure Alarms

The Alarm Server Properties Setup page is displayed if the computer has one or more alarm servers configured that are part of a redundant pair. Correctly setting the options on this page will improve outcomes for alarm server redundancy. Enter an IP address for **Device IP 1** and **Device IP 2**.

If the Plant SCADA system is using an IPv4 network, use a standard IPv4 address format. For example, 192.1.2.34.

If the Plant SCADA system is using an IPv6 network, use a standard IPv6 address format (or any acceptable abbreviation). For example, 2001:0db8:0000:0000:8a2e:0123:1234 or 2001:db8::8a2e:123:1234 (abbreviated).

**Device IP 1** and **Device IP 2** correspond to the INI parameters [\[Alarm\]IsolationDetectIP1](#) and [\[Alarm\]IsolationDetectIP2](#).

## See Also

[Alarms Server Redundancy](#)

[Configure CPU](#)

### Configure CPU

The CPU Setup page of the Setup Wizard is used to assign client and server components to specific processors on a multi-processor computer.

The page includes a table that identifies each **Component** (and the cluster to which it belongs), its **Priority** and its **CPU** assignment.

- If the **Multi-process** option was selected on the Computer Role Setup page, you can select specific CPUs for the Client, I/O Server, Alarm Server, Trends Server and Reports Server processes.
- If the **Multi-process** option was not selected, there will only be one entry listed. It will either be “Client” or “Client and Servers”.

#### To assign a CPU to a component:

1. Select one or more components from the list (hold the **Ctrl** key down to select multiple components).
2. Click the **Modify** button.
3. Select the CPUs you want to assign to the component (you can select more than one) or select **<All CPUs>**.
4. Click **OK**.

When you complete the Setup Wizard, the CPU assignments are written to each component section in the INI file; for example:

```
...
[Alarm.Cluster1.AlarmServer1]
CPU=1
Clusters=Cluster1
...
[Trend.Cluster1.TrendServer1]
CPU=2
Clusters=Cluster1
...
```

If ALL is selected, the component sections in the INI file will be left empty and the processes will run on all CPUs.

In situations where ALL is selected, be aware that performance issues may still occur when your CPU usage is less than 100%. You can detect these issues when your CPU usage for a process is fixed at 100 divided by the number of processors.

---

**Note:** If a computer has more than 32 CPUs, the CPU Setup page in Setup Wizard will be disabled. The component sections in the INI file will be left empty and the processes will run on all CPUs.

---

## See Also

[Configure Events](#)

## Configure Events

Events are used to trigger actions, such as a command or set of commands. For example, an operator can be notified when a process is complete, or a series of instructions can be executed when a process reaches a certain stage. Select the **Enable Events on this computer** check box if events are to be enabled on this Plant SCADA computer.

The Events Setup page lists each component's full name, including the cluster to which it belongs, alongside a list of events that can be enabled for each component. If the **Multi-process** option was not selected on the Computer Role Configuration page there will only be one entry listed, either Client or Client and Servers. If the **Multi-process** option was selected, you have the option of enabling events for each component on this computer.

---

### Note:

- The Setup Wizard only displays named events from the selected project. If you are using events in included projects you will need to edit your ini file to add these under the [Events] section header.
  - Events named 'Global' or events with no title will not appear as these are global events. These events will run on computers that have events enabled. These events will run in the client process.
- 

### To enable an event for a component:

1. Select the component from the list.
2. Select the events you want to enable for that component, or click **Enable All** or **Disable All**.
3. Click **Next** when finished.

When you complete the Profile Setup Wizard, the events are written to each component section in the ini file; for example:

```
...
[Alarm.Cluster1.AlarmServer1]
CPU=1
Clusters=Cluster1
Events=CSV_AlarmClient...
[Trend.Cluster1.TrendServer1]
CPU=2
Clusters=Cluster1
Events=CSV_TrendXClient,CSV_TrendXServer...
```

## See Also

[Configure Startup Functions](#)

## Configure Startup Functions

The Startup Functions Setup page is used to define the Startup Cicode that is executed by each Plant SCADA process.

The Startup Functions Setup page lists each component's full name, including the cluster to which it belongs, the priorities of the components and the startup function assigned to each component. If the **Multi-process** option was not selected on the **Computer Role Configuration** page there will only be one entry listed, either Client or Client and Servers. If the Multi-process option was selected, you have the option of assigning startup functions for each component on this computer.

If the StartupCode parameter value for a process is invalid, the Plant SCADA Runtime Manager will simply ignore it on start up.

#### To assign a startup function to a component:

1. Select the component from the list. To select multiple components, hold down the Ctrl key as you select each item.
2. Click **Modify**.
3. Type the name of the Cicode function you want to call on startup for that component.
4. Click **OK**.

When you complete the Setup Wizard, the events are written to each component section in the ini file; for example:

```
...
[Alarm.Cluster1.AlarmServer1]
CPU=1
StartupCode=alarmServerStartup
...
[Trend.Cluster1.TrendServer1]
CPU=2
StartupCode=trendServerStartup
...
```

#### See Also

[Configure Cluster Connections](#)

#### Configure Cluster Connections

The Cluster Connections Setup page is used to specify the clusters that each component connects to on startup. This controls what data streams a component can see in the system.

The Cluster Connections Setup page lists each component's full name, including the cluster to which it belongs, the priorities of the components and the clusters assigned to each component. If the **Multi-process** option was not selected on the Computer Role Configuration page there will only be one entry listed, either Client or Client and Servers. If the **Multi-process** option was selected, you have the option of assigning clusters for each component on this computer.

By default, every component will connect to every cluster unless otherwise modified.

If the Clusters parameter value for a process is invalid, then the Plant SCADA Runtime will simply ignore it on startup.

#### To assign a cluster to a component:

1. Select the component from the list. To select multiple components, hold down the Ctrl key as you select each item.
2. Click **Modify**.
3. Select the clusters you want the component to connect to on startup.
4. Click **OK**.

When you complete the Setup Wizard, the clusters are written to each component section in the Citect.ini file; for example:

```
...
[Alarm.Cluster1.AlarmServer1]
CPU=1
Clusters=Sydney
...
[Trend.Cluster1.TrendServer1]
CPU=2
Clusters=Sydney,Tokyo
...
```

## See Also

[Clusters](#)

[Configure the Server Password](#)

### Configure the Server Password

If networking is enabled, use this page to configure the server password. This password is used by computers to authenticate each other and create a trusted network between servers and, optionally, clients.

You can create a new password, or enter the password that is already used by other server computers in your Plant SCADA system.

If server processes do not have matching server passwords, they will not communicate with each other. Only trusted tran channels can handle remote requests initiated by [MsgRPC](#) or [ServerRPC](#) Cicode functions and CTAPI connections.

Configuring the password automatically sets the [\[Client\]PartOfTrustedNetwork](#) INI parameter.

---

**Note:** To configure the server password, you need to be a member of the **Configuration Users** security role. To read the server password you need to be a member of either the **Configuration Users** or the **Server Users** security role. You can be a member of these roles either directly or via an associated domain group (see [Security Roles](#)).

---

- **If a server process exists and networking has been enabled:**

The **Configure Server Password** check box is selected and unavailable.

Enter and confirm the password in the fields provided, before clicking **Next**.

If the **Password** and **Confirm Password** fields already contain an entry, it means a server password has already been configured on the local computer. If required, you can enter a new password.

- **If a client process exists and networking is enabled:**

The **Configure Server Password** check box is unchecked. To use the client computer as a CTAPI server, you need to configure the server password. Setting this password allows servers to authenticate each other and creates a trusted network between server computers.

Select **Configure Server Password** check box to enable the password fields.

Enter and confirm the password in the fields provided, before clicking **Next**.

If you change the server password while the runtime system is active, it will not disrupt any established connections as the password is only retrieved when authentication needs to occur. However, it is recommended

that you apply the new password as soon as possible. To do this, you need to restart the runtime system.

If you are extending an existing system by adding computers that will run additional clusters and servers, you should determine what the existing server password is before you proceed. This will allow you to set the same server password on all existing and new servers.

## See Also

[Configure Server User](#)

### Configure Server User

Use this page to define the user to log in for the server processes running on the machine. The user can be the default server user, none (view-only), or a specified user. The Setup wizard will automatically configure the [\[Server\]AutoLoginMode](#) INI parameter in line with the user's settings.

Select **Specific User** to enable the Configure Server User fields. The fields will remain disabled if either the **Default Server User** option or **None** option are selected.

## See Also

[Configure Control Menu Security](#)

### Configure Control Menu Security

The Plant SCADA window property options allow you to control an operator's access to system features. This allows for flexibility with system security at run time.

Option	Description
Plant SCADA configuration environment on menu	Allows the operator to use the control menu (top left-hand icon) to access Plant SCADA Studio, Graphics Builder, and Cicode Editor from Plant SCADA at run time. Disabling this provides better security.
FullScreen	Allows the operator to set whether pages will be displayed in fullscreen or restored state. When checked "FullScreen" will set the ini parameter <a href="#">[Animator]FullScreen</a> to 1. If "FullScreen" is set.
Show title bar	Allows the operator to set whether pages will be displayed in fullscreen mode with the title bar. The <a href="#">[Animator]FullScreen</a> ini parameter is set as follows: 0 if "Fullscreen" is unchecked; 1 if "Fullscreen" is checked and "Show title bar" is unchecked; 2 if both "Fullscreen" and "Show title bar" are

Option	Description
	checked. "Show title bar" cannot be modified if the option "Fullscreen" is unselected.
Shutdown on menu	Allows the operator to use the control menu (top-left icon) to shut down Plant SCADA at runtime. The shutdown is not password- or privilege-protected. Disabling this provides better security.
Kernel on menu	Allows the operator to use the control menu (top left icon) to display the Plant SCADA Kernel at run time. Disabling this provides better security.

## See Also

[Configure Keyboard Security](#)

### Configure Keyboard Security

Windows has a set of standard task-swapping shortcut commands that are (optionally) supported by Plant SCADA at run time. This option allows the Alt-Space Windows command to be enabled or disabled at run time. Alt-Space provides access to the Windows control menu (even if the title bar has been disabled).

**Note:** The ability to disable Alt-Escape, Ctrl-Escape and Alt-Tab is not currently available.

## See Also

[Configure Miscellaneous Security](#)

### Configure Miscellaneous Security

Some standard Windows features may interfere with the secure operation of your system. Use the Miscellaneous Security page to disable these features.

Option	Description
Inhibit screen saver while Plant SCADA is running	Stops the screen saver from blanking out screens that need to be visible at all times. Alternatively the screen saver password can add additional security features.
Display Cancel button at startup	Provides the ability to stop Plant SCADA from starting up automatically. Automatic startup is a potential security concern.

## See Also

[Configure General Options](#)

### Configure General Options

Use the General Options Setup page to specify general options.

Option	Description
Data Directory	The directory where the Plant SCADA data files are located. The data files are the files that are generated at run time: trend files, disk PLC etc.
Backup project path	The backup directory that is used if a runtime database cannot be located (due to inoperative hardware or a file that has been moved, corrupted, or deleted).
Startup page	The Page Name of the graphics page to display when Plant SCADA starts up.
Page scan time	<p>The delay (in milliseconds) between updating a graphics page and starting the next communications cycle. The Page Scan Time sets the default for how often your graphics pages are updated. When a page is updated, relevant data (variable tags etc. represented on the graphics page) is scanned to determine if field conditions have changed. This setting is overridden by the Scan Time value specified in Page Properties (if applied).</p> <p>A value of 250 (the default value) indicates that Plant SCADA will try to update the page every 250 ms. However, if Plant SCADA cannot read the entire data from the I/O Device within 250 ms, the page is processed at a slower rate. For example, if it takes 800 ms to read the data from the relevant I/O Device, Plant SCADA processes the page every 800 ms.</p> <p>Under some conditions, you might want to slow the update of your pages to reduce the load on the I/O Servers. By reducing the page scan time, you allow more communication bandwidth to other Plant SCADA tasks or Clients. For example, you might want fast response on your main operator computers, while slowing the response time on manager computers. You can enter any value from 0 to 60000 (milliseconds).</p>
OPC Alarms and Events	Click the <b>Register</b> button to register an alarm server as

Option	Description
	<p>a runtime OPC AE server that supports the OPC AE interface specification. This will allow the alarm data to be accessed by third-party OPC client applications. The registration process requires administrative privileges.</p> <p><b>Note:</b> The OPC AE Server does not support DCOM for remote communications. It is recommended that you run any connecting OPC client applications on the computer that is registered as the OPC AE Server.</p>

## Finish

Select one of the following:

- **Finish** saves the setup to the INI file.
- **Cancel** quits the wizard without saving.
- **Back** navigates back to a page that requires adjusting.

## Redundancy

Redundancy in Plant SCADA can be defined at many different levels. When building redundancy for your system, you need to consider the degree of protection necessary to meet your requirements. This will involve the following:

- Deciding how important each processes is
- Determining the overall likelihood of system downtime, and the processes and equipment that will take the longest to restore to service
- Deciding which components require redundant operation
- Considering design and maintenance requirements.

The section covers the following redundancy concepts:

- [I/O Server Redundancy](#)
- [I/O Device Promotion](#)
- [Redundancy and Persistence](#)
- [Data Path Redundancy](#)
- [Network Redundancy](#)
- [Reports and Trends Server Redundancy](#)
- [Alarms Server Redundancy](#)

**Note:** Using the Setup Wizard will allow you to define the level of redundancy you require by defining the function of each computer (See [Run a Project using the Setup Wizard](#)).

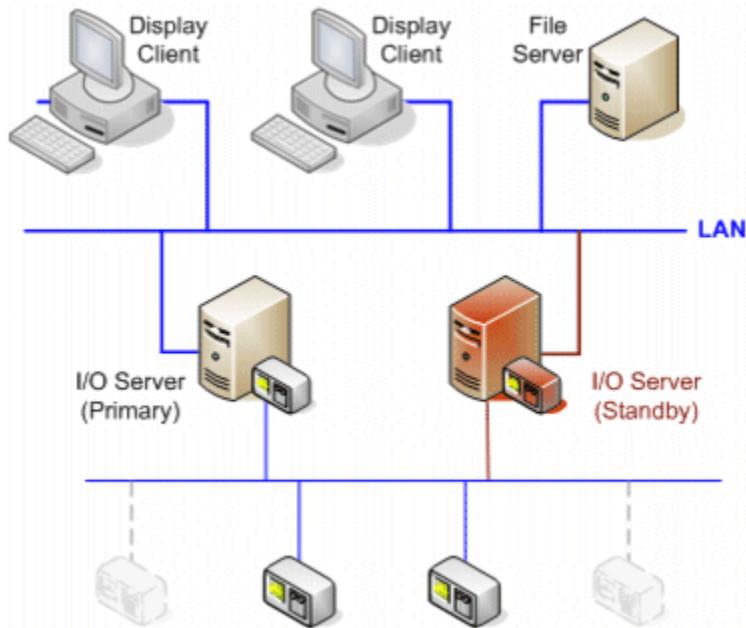
## See Also

- [Reports and Trends Server Redundancy](#)
- [File Server Redundancy](#)
- [Redundancy of Standalone Systems](#)

## I/O Server Redundancy

Systems with a single I/O Server can be interrupted by the inoperability of a single device or process. If the primary server isn't operating normally, control and monitoring of the system is lost. By introducing a second I/O Server and dedicating it to communicating with the same I/O Devices, the ability of a single device to influence the system as a whole is minimized. You now have a **primary** and **standby** I/O Server in your system where the standby I/O Server will assume operations in case the primary I/O Server becomes inoperative.

The diagram below illustrates the introduction of a standby I/O Server into the existing system. When the system is in operation, both I/O Servers are identically maintained.



---

**Note:** Although both I/O servers are identical, recognize that the standby server is not duplicating the primary server's functions. If it were, the load on the PLC portion of the network would be double and would significantly reduce performance. Therefore, only the primary server communicates with the PLCs at any given time.

Redundancy is provided as follows:

- When the system is in operation and the primary I/O Server becomes inoperative, or if you wish to perform some maintenance and take it offline, every client will be reverted to the standby server with minimal or no interruption to the system.
- When the primary server is brought back online, the system returns control of the I/O Devices back to the primary server. This is done by copying the disk image from the standby server to the primary, allowing the clients to reconnect to it, and thereby resuming control of the system.

When the system is running, you can also use redundant I/O Servers to split the processing load. This would

result in higher performance as every I/O Server would be running in parallel when servicing the I/O Devices.

## See Also

[I/O Device Promotion](#)

### I/O Device Promotion

I/O device promotion refers to when a system event forces a standby I/O device to assume a primary role during Runtime.

If there is more than one standby device available within a cluster, the order in which they are promoted will depend on how your project has been configured.

If necessary, you can manually set the order your standby devices are promoted in via the I/O device's **Priority** property (see [Add an I/O Device](#)). If priorities are not set, the compiler automatically allocates a priority value to each standby device based on the following rules:

- User-configured priority settings take precedence
- Any standby devices without a priority setting will follow those that do, their priority being allocated in the order they were configured
- If no standby devices have a priority setting, they will be assigned priority according to the order they were configured.

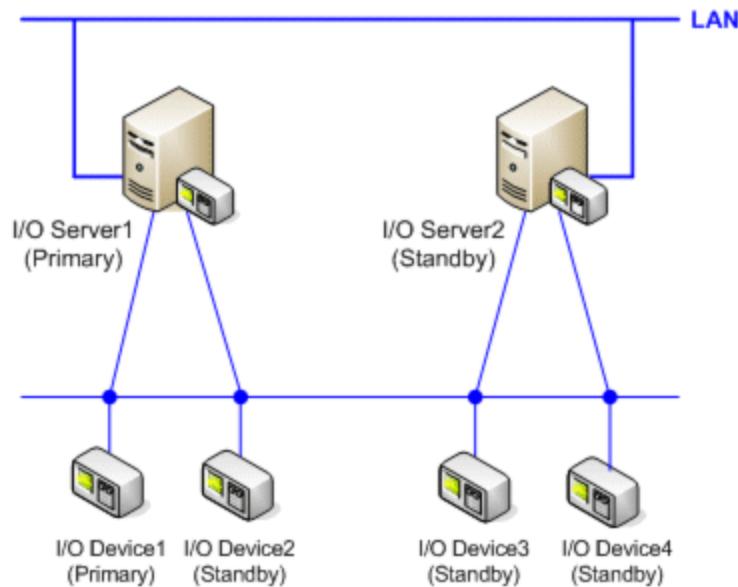
These rules apply regardless of the connections between the I/O devices and I/O Servers configured in a cluster.

The standby device priorities are confirmed and/or allocated during project compilation. Notification is provided for each allocation issued by the compiler. Compile error messages will result from the following circumstances:

- If a primary device has a priority setting other than the default value of 1
- If standby devices have duplicated priority values.

## Example

The following diagram shows four I/O devices connected to two I/O Servers, with just one primary device configured.



If no priorities were set for the standby devices, the compiler would allocate the following:

- I/O Device1 is allocated **Priority 1** by default (no compiler alert)
- I/O Device2 is allocated **Priority 2** (compiler alert generated)
- I/O Device3 is allocated **Priority 3** (compiler alert generated)
- I/O Device4 is allocated **Priority 4** (compiler alert generated)

This presumes the devices were configured in numerical order.

If I/O Device3 has been manually set to priority 2, the compiler would allocate the following:

- I/O Device1 is allocated **Priority 1** by default (no compiler alert)
- I/O Device2 is allocated **Priority 3** (compiler alert generated)
- I/O Device3 is allocated **Priority 2** (no compiler alert)
- I/O Device4 is allocated **Priority 4** (compiler alert generated)

In this case, the setting for I/O Device3 has taken precedence. The remaining standby devices are set for promotion based on the order they were configured.

**Note:** The automated priorities work on an n+1 equation; if I/O Device3 has been set to Priority 5, the remaining devices would have been allocated priority 6 and 7.

## See Also

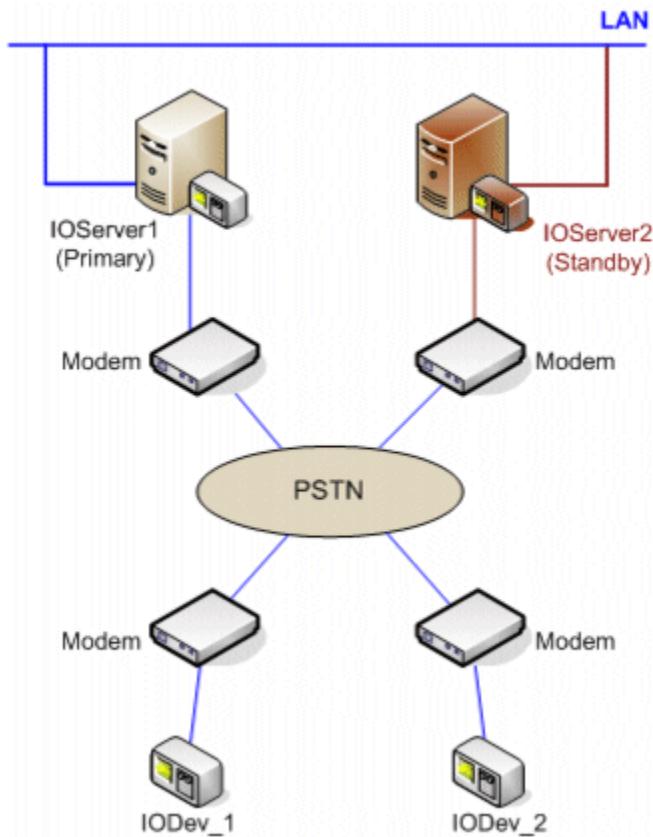
[Redundancy and Persistence](#)

[Data Path Redundancy](#)

## Redundancy and Persistence

If you are using server redundancy, persistence caches (I/O server cache) keep standby servers updated with the most recently read device data. A persistence cache, or I/O server cache, is created for each cached I/O device.

The following diagram introduces the concept of a persistence cache.



The diagram shows that there are two I/O servers, namely IOServer1 (primary) and IOServer2 (standby). Each connects to the public switched telephone network (PSTN) via a modem, which is in turn connects to the I/O devices, also over a modem. Persistence caches work as follows:

1. Every IODevices->Cache Time period, data from an I/O Device is stored temporarily in the memory of the I/O server (I/O server cache).
2. For every [IOServer]SavePeriod, IOServer1 stores its in-memory cache to disk.
3. The cache is saved in persistence caches -one for each cached device.
4. IOServer1 broadcasts to other I/O servers the UNC path of the persistence caches (set with [IOServer]SaveNetwork).
5. From these persistence caches, IOServer2 updates its in-memory cache for its I/O devices.
6. Depending on the value of the I/O server parameter of '[IOServer]SavePeriod' (determines how often the persistence cache is saved to the hard disk in seconds), IOServer1 stores its in-memory cache to the hard disk every x amount of seconds.

---

**Note:** You can define an I/O Device on an I/O server using the Device Communications Wizard, or by adding a device in the Topology activity.

You are not limited to just one Standby Server, since the UNC path name set in [IOServer]SaveNetwork is broadcast to I/O servers. Each I/O server updates its cache from the persistence caches only for the I/O devices defined on that server. It is then possible, therefore, set up several I/O servers which update their in-memory caches with the most recently read data.

For example, we set the [IOServer]SaveFile and [IOServer]SaveNetwork parameters as follows:

On IOServer1	On IOServer2
[IOServer]	[IOServer]
SaveFile=C:\Data\IOServer1.dat	SaveFile=C:\Data\IOServer2.dat
SaveNetwork=\\IOServer1\Data\IOServer1.dat	SaveNetwork=\\IOServer2\Data\IOServer2.dat

IOServer1 would broadcast the following UNC path of the persistence cache to other I/O servers: '\\IOServer1\ Data\IOServer1.dat'.

IOServer2 would then use the persistence caches to update its in-memory cache with the device data most recently read by IOServer1.

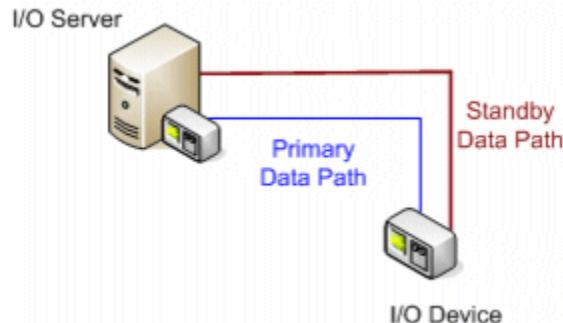
## See Also

[Data Path Redundancy](#)

## Data Path Redundancy

Data path redundancy is another form of redundancy involving defining data paths between the I/O server and the connected I/O devices. By providing a second (parallel) data path, you improve the chances that if one data path to the I/O device is disconnected, the other can be used.

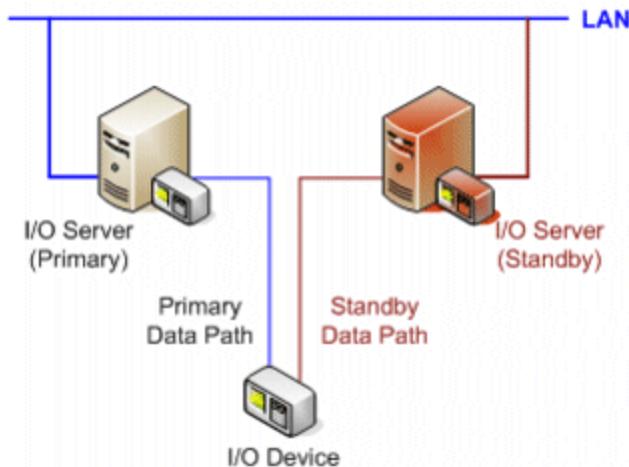
Many PLCs have the facility to allow you to install a parallel data path from the I/O server to the I/O device.



The diagram above shows that an additional data path (running in parallel) has been defined. The redundancy is provided as follows:

- When you start your runtime system, Plant SCADA connects to the I/O device using the **primary data path**.
- If communications with the I/O device is lost at any time (for example if the communications cable is disconnected), Plant SCADA will **switch to the standby data path** with minimal or no interruption to the system.
- Plant SCADA reconnects through the primary data path when it is returned in to service.

On a larger system (such as one running on a network), you can also use data path redundancy to maintain device communications with multiple I/O server redundancy, as shown in the following diagram.



The redundancy is provided as follows:

- By using a redundant data path from the I/O device (one path to each I/O server), you can maintain I/O device communication.
- If communications with either the primary I/O server or standby I/O server be disconnected, the I/O device is still accessible.

## See Also

[Multiple Device Redundancy \(Standby Data Paths\)](#)

### Multiple Device Redundancy (Standby Data Paths)

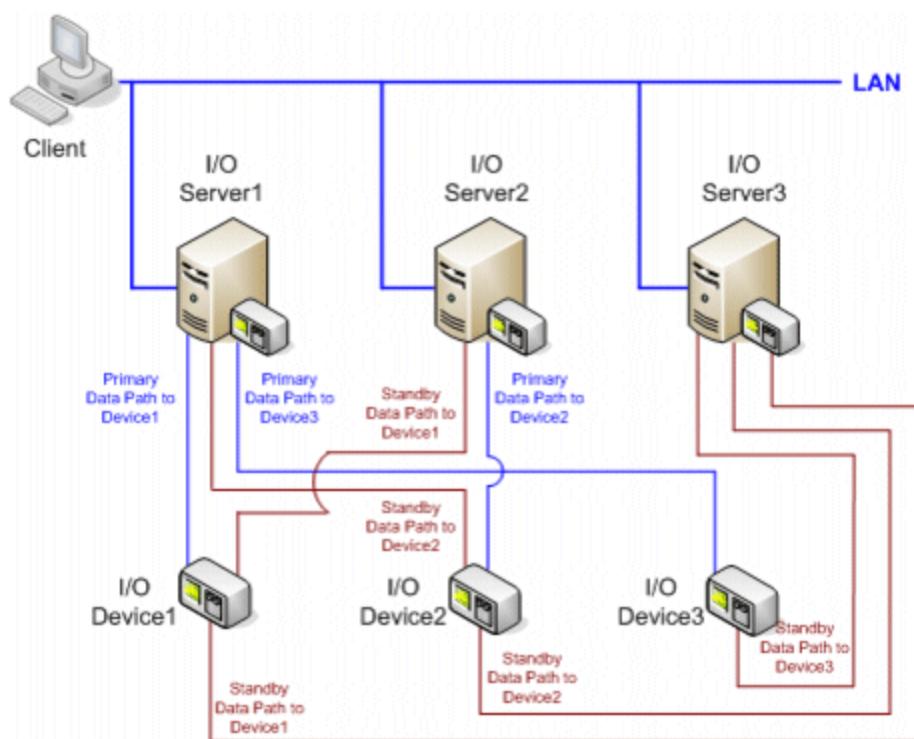
If your I/O devices support peer-to-peer communication, you can add another level of redundancy to your system by duplicating the I/O devices.

---

**Note:** Although I/O servers are not assigned the Primary or Standby role based on the I/O devices to which they are connected, it is common practice in redundant I/O systems to connect the Primary I/O devices to the Primary I/O server and the Standby I/O devices to the Standby I/O server. One I/O server can connect to a mixture of Primary and Standby I/O devices. The I/O server can support any number of Standby Data Paths.

---

The following diagram demonstrates multiple device redundancy and Standby Data Paths:



In this scenario, we have three I/O servers connected to three I/O devices in the following manner:

I/O Server	I/O Devices Connected
IOServer1	I/O Device1 (Primary) I/O Device2 (Standby) I/O Device3 (Primary)
IOServer2	I/O Device1 (Standby) I/O Device2 (Primary)
IOServer3	I/O Device1 (Standby) I/O Device2 (Standby) I/O Device3 (Standby)

The following is known:

- Plant SCADA clients communicate with every configured I/O server at the same time (on startup, the clients try to connect to each configured I/O server. If they cannot establish communications with an I/O server, a hardware error message is generated).
- When every device is running, Plant SCADA processes the I/O on the primary I/O devices as this reduces the I/O load on the I/O device (and PLC network), which can help improve performance.
- The client creates network sessions to the three I/O servers.
- The client then sends requests for I/O Device1 and I/O Device3 to I/O Server1, and requests for I/O Device 2 to I/O Server2.

Redundancy is provided as follows:

- If I/O Device1 become inoperative on I/O Server1, the client will send requests for I/O Device1 to I/O Server2

through the standby data path. It continues to send requests for I/O Device3 to I/O Server1.

- If I/O Device also becomes inoperative on I/O Server2, the client sends requests to I/O Server3.
- If the connection between I/O Device1 and I/O Server1 be re-established, the client resumes sending requests for I/O Device1 to I/O Server1.

The Standby I/O devices will be activated strictly in the order in which they are first created in the project. This can be viewed by looking in the Units.dbf file in the project directory.

When you connect primary and standby I/O devices to your I/O servers, share the primary I/O devices between your I/O servers to balance the loading. Be aware, however, that this might not apply for every protocol because the loading could be dependent on the PLC network and not the I/O server CPU. In this case, more than one active I/O server on the same PLC network can impact the PLC network and therefore, slow the total response.

## See Also

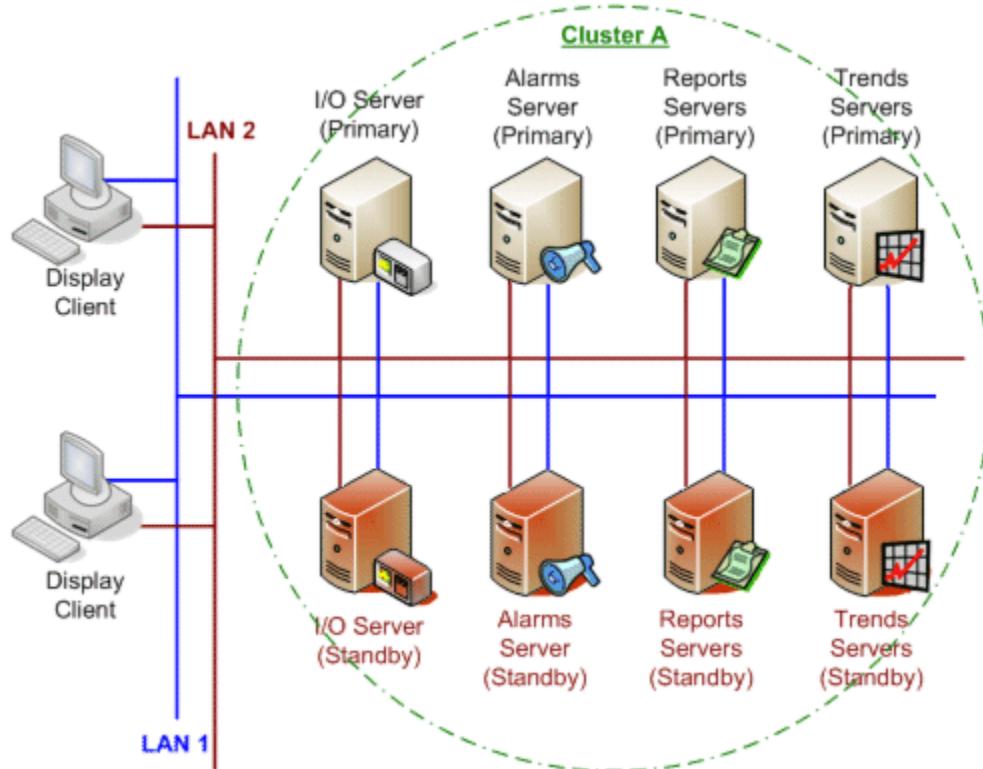
[Alarms Server Redundancy](#)

[Reports and Trends Server Redundancy](#)

## Network Redundancy

You can use the dual NIC (or multiple network interface) capabilities of each client or server, enabling you to specify a complete and unique network connection from a client to a server.

Consider the following diagram:



The above example shows two levels of redundancy:

- network redundancy
- server redundancy

## Network Redundancy

**Note:** Currently Plant SCADA does not support dynamic port binding.

Each of the system components in the cluster are connected to two networks (LAN 1 and LAN 2). This is achieved by using the dual end points of each server. This provides LAN redundancy as follows:

- If LAN 1 is suddenly inoperative, each component in the cluster can easily maintain connection by using LAN 2.
- In turn, if LAN 2 becomes inoperative, LAN 1 remains in operation.

**Note:** For this type of redundancy to work, each system component needs to have two NICs to support two parallel LANs. On every computer, each NIC needs to use a separate IP network or subnet.

## Server Redundancy

For information about server redundancy, see [Reports and Trends Server Redundancy](#) and [Alarms Server Redundancy](#).

## See Also

[Configuring Network Redundancy](#)

### Configuring Network Redundancy

To connect to machines using dual network connections for redundancy you need to first define network addresses for each network interface and then specify which network addresses to use for each server.

#### To configure network redundancy:

1. In the **Topology** activity, choose **Edit | Network Addresses**.
2. Using the Grid Editor or the Property Grid, define the network address for each network interface card. See [Add Network Addresses](#).
3. In the **Topology** activity, choose **Edit** and then the server type you want to connect to the network.
4. In Network Addresses field, type the dual addresses for the server separated by a comma. For example, "AlarmPrimaryLAN1,AlarmPrimaryLAN2".

**Note:** If you are using address forwarding to enable server redirection, you need to consider how this may be impacted by network redundancy. For more information see [Server Redirection Using Address Forwarding](#).

## Using hostnames with dual network interface cards

Plant SCADA allows you to use a hostname when defining network addresses. This can simplify the configuration of a redundant network, as static IP addresses are not required. It also enables network redundancy for an OPC UA Server or an Industrial Graphics Server.

However, if you have used hostnames to define the network addresses for dual network interface cards in a computer, you will need to confirm some additional network configuration settings. This is required when one of the cards is used for the SCADA network, and the other for the PLC network.

You need to confirm a DNS setting for the communications adapter used by the PLC network.

In Windows **Network and Sharing Center** (accessible via Control Panel), select **Change adapter settings** and view the properties for the PLC network connection.

Locate TCP/IPv4 or TCP/IPv6 in the list of used items, and select **Properties** again. Open the **Advanced** properties, then go to the **DNS** settings.

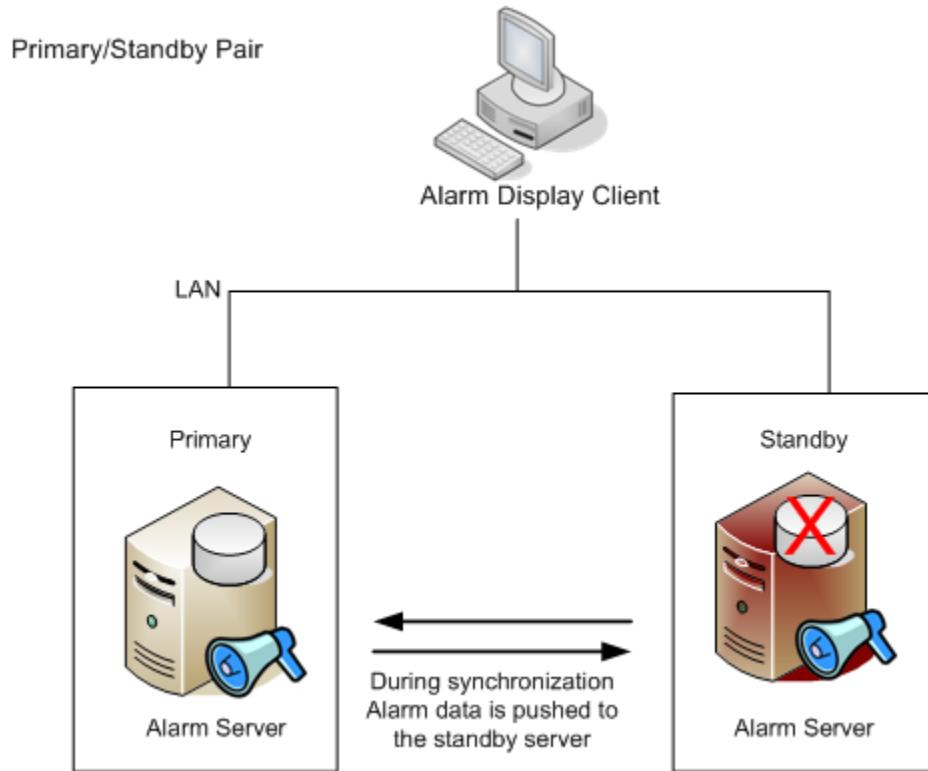
Make sure **Register this connection's addresses in DNS** is NOT selected.

## See Also

[Network Redundancy](#)

## Alarms Server Redundancy

To support alarm server redundancy you can configure a main/standby pair. To keep with the terms used in previous versions of Plant SCADA the main server is called the primary server.



With a primary/standby pair only one server (the primary) updates the database. However, the display client is able to communicate with both the primary and standby servers.

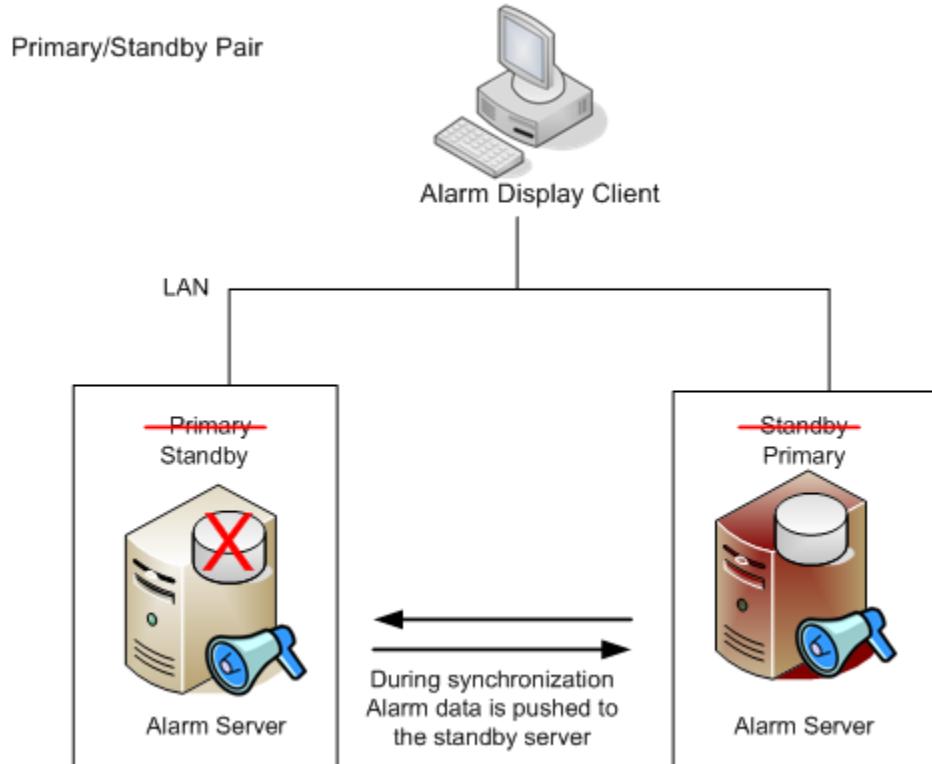
On starting up, the display client will attempt to connect to the 'main' (primary) server. By default the alarm server which starts first will become 'main'.

If you want a display client to connect to a particular server on startup, use the parameter [\[Alarm\]Priority](#). This parameter assigns a higher priority to either the primary or standby server. This means when the client starts, it

will communicate with the server assigned the highest priority, even if the lower-priority server started first (and is therefore the main server).

If the primary server becomes inoperative (including loss of communication with the I/O server), the standby server takes over the duties of the primary server. It will process alarm data and update the database. When the inoperative server resumes operations, it establishes communication with the other alarm server (which is now the main) and become the standby server.

**Note:** In cases where both the primary and standby server become isolated loss of data may occur.



During synchronization, the primary server updates the standby server. This confirms that the standby server contains data that accurately represents the data on the primary server (see [Alarm Server Side Synchronization](#)).

**Note:** Before starting your system for the first time (i.e. the database is empty), the primary and standby server should be connected. If not connected when the system starts, the primary and standby servers may create distinct IDs for the same objects.

## See Also

[Reports Server Redundancy](#)

### Alarm Server Side Synchronization

Synchronization occurs when the primary server establishes or re-establishes a connection with the standby server. Synchronization takes place on startup and then periodically during runtime while connection exists between the Primary and stand by servers. During synchronization the Primary server (or main) updates the data on the standby sever as required with alarm action and event data.

Each time the alarm primary server connects to the standby server, the synchronization process will take place. This will allow primary server to update the standby server with the most recent copy of the primary server's

data.

**Note:** Synchronization occurs only if the primary and standby alarm servers are running and connected.

To configure synchronization between the primary and standby alarm server refer to the [Alarm Parameters](#).

## Main-Main Scenario

If the standby alarm server loses its connection with the primary alarm server, a 'main-main' situation may occur, where both servers communicate and process data from the I/O server. On reconnection the data on both servers will need to be synchronized. With duty mode enabled by default, the servers determine what data to merge. In most instances the data with the latest timestamp is kept while the other is discarded.

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

In a 'main-main' situation do not make any configuration changes to your system until the primary and standby server is reconnected. Configuration changes made to the system during a 'main-main' situation may cause the system to become inoperative.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

#### **Deciding which server will be 'Main' after a 'Main-Main' scenario**

When the connection between the servers is restored, the servers arbitrate to determine which server continues as main and which server becomes the standby.

In Plant SCADA, when configuring the alarm server you select which server is primary and which is standby. By default, the server configured as primary is given the highest priority, thus when the servers arbitrate to determine which becomes main, the primary server will revert to being the main server.

## Configuring Isolation Parameters

It is likely that when a redundant pair of alarm servers cannot communicate with each other, it is due to one of the servers becoming isolated from the network. The alarm server that has become isolated is also likely to be isolated from the physical devices from which it obtains data, so it will not be obtaining alarm events from the field during the period of isolation.

The parameters [\[Alarm\]IsolationDetectIP1](#) and [\[Alarm\]IsolationDetectIP2](#) can be configured so that the system knows which alarm server became isolated. The isolated alarm server can then become the standby when the connection between the alarm servers is restored. This will help avoid the loss of alarm event data during the period of an outage, as the alarm server that was not isolated becomes the main server and synchronizes the events that occurred to the standby.

## Duty Mode

Duty mode enables a standby server to transfer data to a primary server following a 'main-main' situation.

When the connection between the servers is restored, the servers arbitrate to determine which server switches to standby.

In Plant SCADA duty mode is enabled by default, and helps prevent the data on the server that is deemed to be the standby (after the arbitration process) from not being accessible. In Duty mode the timestamps of the data

on the servers are compared. The data with the recent timestamp is kept and is merged into the data on the primary server and the remainder of the data is discarded.

When the primary server receives the more recent data, from the standby server, it merges the data with its own data and then sends an update to the standby server. This allows for primary and standby server to operate with synchronized data sets.

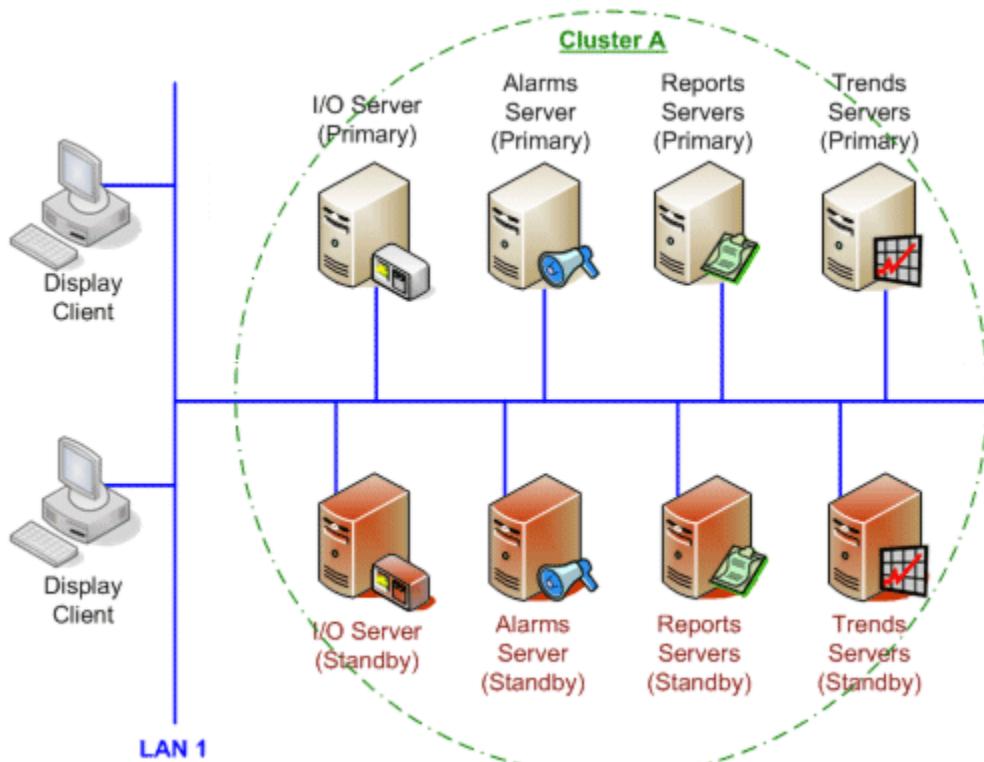
## See Also

[Add an Alarm Server Process](#)

## Reports and Trends Server Redundancy

Redundancy of Reports, Trends, and I/O servers is achieved by adding standby servers, within a defined cluster, to provide redundant system components. In addition you can also utilize the dual end point (or multiple network interfaces) capabilities of each component, effectively enabling you to specify a complete and unique network connection from a client to a server. See [Network Redundancy](#).

Consider the following diagram:



Server redundancy of each component in the diagram is achieved by providing a corresponding standby server on the same network (LAN 1). Redundancy is provided as follows:

- If any of the servers become inoperative or communications become inoperative, their standby counterpart assumes operation.
- For I/O Server redundancy, when the inoperative I/O Server comes back online, it resumes control based on the individual I/O device primary or standby priority configuration..

- For Trend and Report servers, when the inoperative primary server returns online, the client will remain with the standby unless the primary has a higher priority.
- For Reports, and Trends Server redundancy, clients connect to either the Reports, or Trends primary server or standby server. On startup, clients try to establish a connection with the primary server. If a connection with the primary server cannot be established, they will try to establish a connection with the standby server. If the primary server becomes available, any clients connected to the standby server remain connected to the standby server unless the primary has a higher priority. If the standby server becomes inoperative the client will revert to the primary server. The priority is set using the connectivity parameters described below.

Alarm Server Redundancy uses a main/standby model. Refer to [Alarms Server Redundancy](#) for more information.

## Connectivity Parameters

The client sub system runs in every SCADA client or server process. It connects to all servers within all clusters that are enabled, allowing any process to send data requests to any other server, either through internal systems or Cicode function calls.

Two Citect.ini parameters allow you to configure how the client sub system will manage its connections to redundant pairs of Reports or Trends servers.

These parameters are [Type.ClusterName.ServerName]Priority and [Type.ClusterName.ServerName]DisableConnection, where Type is the relevant server type (Report or Trend). The default setting for these parameters provides behavior as described above, in that if the client sub system is redirected to a standby server it will retain a connection to that server, providing that it is operable, even when the primary server is restored. This is also the behavior that will occur if these parameters are not added for a given server type.

The connectivity parameters can be set at the system level using the Parameters form of the project, or specifically for each computer in the local Citect.ini file.

For a detailed explanation of these parameters see:

- [\[Report.<ClusterName>.<ServerName>\]Priority](#)
- [\[Trend.<ClusterName>.<ServerName>\]Priority](#)
- [\[Report.<ClusterName>.<ServerName>\]DisableConnection](#)
- [\[Trend.<ClusterName>.<ServerName>\]DisableConnection](#)

## See Also

[Alarms Server Redundancy](#)

[Reports Server Redundancy](#)

[Trends Server Redundancy](#)

[File Server Redundancy](#)

## Reports Server Redundancy

It is possible to configure two Reports Servers in a project. That is, a primary Reports Server and a standby Reports Server.

When both Reports Servers are in operation, the scheduled reports only run on the primary Reports Server. If the primary Reports Server becomes inoperative, the scheduled reports run on the standby Reports Server (you can also configure the standby Reports Server so that it also runs the scheduled reports in parallel with the primary Reports Server). Please be aware that no report data is transferred between the primary and standby Reports Servers (Plant SCADA does not synchronize the report data because reports can write their data to any type of device).

## See Also

[Trends Server Redundancy](#)

### Trends Server Redundancy

It is possible to configure two Trends Servers in a project. That is, a primary Trends Server and a standby Trends Server.

When both Trends Servers are in operation, trends are processed on both servers in parallel, and written to disk (each server needs to write to its own disk or its own private area on the file server).

When a Trends Server starts up, it tries to establish a connection to the other Trends Server. If it can establish a connection, it will transfer the trend data from the last time it was shutdown until the current time (this minimizes the chance that trend data will be lost).

## See Also

[File Server Redundancy](#)

### File Server Redundancy

Plant SCADA allows for redundancy of the file server. The [\[CtEdit\]Backup](#) parameter specifies a backup project path. If Plant SCADA cannot find a file in the Run directory (i.e. as specified by the [\[CtEdit\]Run](#) parameter), it will look in the backup path. If the file is found in the backup path, Plant SCADA will assume that the run path has become unavailable for some reason (for example, the file server has become inoperative). It will then look for relevant files in the backup before changing over. When Plant SCADA changes over to the backup path, it will call event number 11 and generate the hardware error: "File server failed, to Standby".

File server redundancy will only operate correctly if the redirector (or shell) on the computer can respond appropriately if the file server becomes inoperative.

---

**Note:** Only Plant SCADA switches to a backup path. Any other applications that are using files on the file server will become inoperative when the file server becomes inoperative. This may cause the computer to wait for long periods for the file server (or to itself become inoperative). This includes Windows itself, so install Windows on a local drive.

---

To enable file server redundancy, set the [\[CTEDIT\]Backup](#) parameter to a backup database path. For example, if your primary path is **F:\Plant SCADA\USER\DB**, set the backup path to another file server or a local drive, such as: "C:\ProgramData\AVEVA Plant SCADA <VersionNumber>\User".

You will want to verify that the project in the Backup path is the same as the one in the Run directory - each time you compile the project in the run directory copy it into the backup directory.

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Before placing your Plant SCADA system into service, confirm that the standby file server has an identical copy of the current project, and that the [CTEDIT]Backup parameter is correctly set.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**See Also**

[\[CtEdit\]Copy](#)

[\[CtEdit\]Run](#)

## Redundancy of Standalone Systems

If you are using Plant SCADA as a standalone system, in the single process model, you can still achieve redundancy by implementing your standby systems on another CPU, provided you have a multi-CPU system. You may also wish to implement this for load-balancing purposes.

**See Also**

[Run a Project Using the Setup Wizard](#)

## System Model

In the **System Model** activity, you can configure and view equipment, variable tags, alarms, trend tags, SPC tags and accumulators.

The following topics are covered in this section:

- [Equipment](#)
- [Variable Tags](#)
- [Alarms](#)
- [Trends](#)
- [Accumulators](#)
- [Statistical Process Control \(SPC\)](#)

Each of these sections provides conceptual information followed by the set of tasks that you can perform.

## Equipment

In Plant SCADA, the term "equipment" is used to describe an object-oriented architecture within a project that can be used to reference the machinery or processes being monitored. Equipment definitions create logical groupings that allow you to organize your project using a hierarchical design. Each definition brings together the

base SCADA properties into a single entity that can be easily replicated within your project.

For example, if a pump is represented in your project as a piece of equipment, it will bring together the tags, events, alarms, permissions, communications, scheduling and scripting associated with the pump. You can associate the pump with a particular geographical area, or include it as part a functional process.

**Note:** To be able to define equipment, you need to have a report server process defined (see [Add a Report Server Process](#)).

This section of the help explains the concepts that define the way equipment is configured in a Plant SCADA project. These include:

- [Equipment Hierarchy](#) — the object model that is created by the equipment definitions in a project
- [Equipment Types](#) — the templates you can use to add multiple instances of a particular piece of equipment to your project
- [Equipment Instances](#) — the representation of a physical piece of equipment in your project that can be based on a particular equipment type
- [Equipment States](#) — operational states for a piece of equipment that you can use for scheduling
- [Equipment References](#) - the link added between a piece of equipment, and/or items belonging to other equipment within your project. The equipment reference can also be to equipment, and/or items that is outside the equipment hierarchy.

This section also explains the different approaches you can take when defining equipment within a project.

- If you are creating a new project (or adding a new section to an existing project), use the Equipment Editor to build an equipment hierarchy and automatically generate tags from that hierarchy (see [Equipment Editor](#)).

For specific, task-based information that describes how to configure equipment in a Plant SCADA project using the Equipment Editor, see:

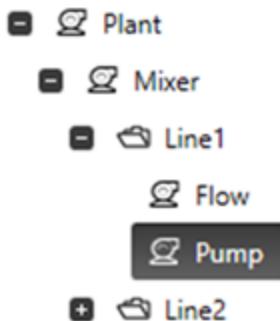
- [Configure Equipment Types Using Equipment Editor](#)
- [Configure Equipment Instances Using Equipment Editor](#).
- If you would like to add an equipment hierarchy to an existing project, you can associate your existing variable tags, trends and alarms with equipment in Plant SCADA Studio (see [Define an Equipment Association for a Tag](#)).

For specific, task-based information that describes how to configure equipment in Plant SCADA Studio, see [Configure Equipment in Plant SCADA Studio](#).

## Equipment Hierarchy

The term "equipment hierarchy" refers to the object model that is created by the equipment definitions in a project. When you define an equipment instance, its location in the hierarchy is reflected in the equipment name with a period (.) used to indicate levels in the hierarchy.

For example, the equipment name "Plant.Mixer.Line1.Pump" refers to the selected item in the following equipment hierarchy:



This type of architecture allows you to logically reference each piece of equipment based on its location in the object model, which could represent a geographical location or a functional process. Each piece of equipment becomes a parent to the items that are associated with it, including variables, trends, alarms, accumulators, and so on.

At runtime, the object model hierarchy can be used to search and display information that is aligned to the process operational model.

## Example

If your project uses Plant SCADA's built-in templates (accessible via a starter project), the equipment hierarchy is integrated into the project pages. For example, the equipment hierarchy displays in a panel on pages where a list of variable tags or alarms is displayed, allowing you to browse, sort and filter the list.

The example below shows the equipment hierarchy for the Plant SCADA Example project displayed on the active alarms page.

Active alarms		
Acknowledge page   Filter...   Auto-fit columns		
(Alarms Filtered)	Date	Time
+ Building (2)	3/06/2014	11:15:40 AM
+ Loops (3)	27/05/2014	03:56:26 PM
- Plant (19)	27/05/2014	03:55:33 PM
+ Bottler (8)	8/05/2014	03:07:59 PM
+ Mixer (11) <input checked="" type="checkbox"/>	8/05/2014	02:22:44 PM
SteelMill (1)	8/05/2014	01:50:00 PM
CoilCar	8/05/2014	01:28:02 PM
Shearer	8/05/2014	12:55:00 PM
Uncoiler	29/04/2014	01:10:00 PM
Welder (1)	29/04/2014	01:00:00 PM
	29/04/2014	12:30:00 PM

The integrated object model has been used to filter the alarms by placing a check mark next to the Mixer branch. This means the list only includes the 11 active alarms associated with the Mixer.

## See Also

[Equipment Types](#)

[Equipment Instances](#)

## Equipment Types

In a Plant SCADA project, the term "equipment type" refers to a template that you can use to associate multiple instances of a particular piece equipment in your project.

For example, if you have a motor that is deployed in numerous locations across a production facility, you could create an equipment type to act as a template for the motor. This will bring together many of the configuration settings required to add each instance of the motor to your project, such as the associated variable tags, operational states, cluster associations, and so on.

To then add one of the motors to your project, you simply create a new instance of the equipment type. This offers the following benefits:

- Minimal configuration is required to deploy multiple equipment instances
- Configuration is consistent across multiple equipment instances
- Equipment instances are automatically integrated into the project's equipment hierarchy
- An equipment type maintains an association with each equipment instance that is created from it, allowing any changes to be easily applied across an entire project.

An equipment type is made up of the following:

- **Items** - used to represent attributes of the physical piece of equipment, which can include controls, actions, conditions, settings, and so on. Each item is made up of a set of elements, which can include variable tags, trends or alarms (see [Items](#))
- **Configuration Parameters** - used to define specific values associated with the piece of equipment (see [Add Configuration Parameters to an Equipment Type](#))
- **States** - used to define operational states for the associated piece of equipment that you can use for scheduling (see [Equipment States](#))
- **Linked Genie** - specifies a genie that you can use to represent the piece of equipment on your graphics pages (see [Link a Genie to an Equipment Type](#)).

To create equipment types, you use the Equipment Editor (see [Equipment Editor](#)).

Alternatively, you can create your own XML template and add it to your project as an equipment type (see [Import Equipment Using XML Templates](#)).

---

**Note:** Equipment types are primarily designed to assist with the creation of new tags. If you would like to define an equipment hierarchy for existing tags, you should use the process described in the topic [Define an Equipment Association for a Tag](#).

## See Also

[Equipment Instances](#)

## Items

The term "item" is used to describe a particular attribute of a physical piece of equipment that is represented in an equipment definition.

Items can be used to reflect different aspects of a piece of equipment, such as an associated action, condition or setting. They group the tags associated with equipment in a way that allows you to reference the particular attribute the item represents.

[Equipment Types](#) are defined as a set of items. Each item includes a number of elements that directly link to the physical device, such as variable tags, trends or alarms.

## Example

The Plant SCADA Example project includes an equipment type named "Motor". It includes the following items:

- Running
- Stopped
- Auto
- OutOfService
- Alert
- IntLock
- Start
- Stop
- IntBypass
- Reset
- FailStart
- FailStop

This list demonstrates the different types of attributes you can represent with an item, including operational states (Running, Stopped), actions (Start, Stop, Reset), alarm conditions (FailStart, FailStop) and settings (IntBypass).

In each case, the item is made up of a set of elements (tags, trends and alarms) that enable interaction with the physical device.

For example, FailStart includes a variable tag definition that is associated with the physical motor, and a digital alarm tag that indicates when an attempt to start the motor has been unsuccessful.

---

**Note:** When defining an item name, avoid using [Reserved Words](#). If you use any of these, an error message will display when you compile your project.

---

## See Also

[Equipment Instances](#)

## Equipment Instances

The term "equipment instance" refers to an equipment record in a Plant SCADA project.

When you add an equipment instance to your project, you specify a location for it within the project's equipment hierarchy. This is reflected in the complete equipment name (where a period (.) can be used to indicate levels in the hierarchy). This location can then be used to refer to the variable tags, alarms and trends that are associated with the equipment instance.

An equipment instance contains a set of properties that may be used in the generation of tags, alarms and trends for that instance. These properties can also be used at runtime in conjunction with the equipment browse functions.

Every equipment instance includes a predefined set of properties that are common to all equipment instances (see [Define Equipment in Plant SCADA Studio](#)).

If an equipment instance has been created using an equipment type as a template, it will include additional properties that reflect the items and configuration parameters specified in the associated equipment type. Each instance maintains an association with the equipment type from which it was created, allowing equipment type changes to be instantiated across all related instances.

To create equipment instances from an equipment type, you use the Equipment Editor (see [Equipment Types](#)).

---

**Note:** Equipment does not need to be associated with an equipment type. This scenario is typical in projects where equipment associations have been configured for existing variable tags. See [Define an Equipment Association for a Tag](#).

---

## See Also

[Equipment Types](#)

[Items](#)

## Equipment States

States are defined in an equipment type to identify operational conditions for a piece of equipment that you can use for scheduling.

Each state includes a set of fields that reflect the associated Equipment State properties. These fields identify the entry action that is performed when a state transition occurs, and any other settings that may apply while the state is active such as delays and repeat actions.

You can create states for an equipment type using Equipment Editor (see [Add a State to an Equipment Type](#)).

Alternatively, you can manually define equipment states in Plant SCADA Studio (see [Define Equipment States in Plant SCADA Studio](#)).

## See Also

[Equipment Types](#)

## Equipment References

Equipment references define an association between two pieces of equipment or items on equipment. They create relationships in your equipment definitions that extend beyond those that occur intrinsically within the equipment hierarchy.

Using equipment reference browse functions, you can track, group and display information for equipment and its referenced equipment in one interface at runtime.

For example, you could create a pop up for a pump that monitors the following:

- The flow of the pump.
- The pump's power and energy usage.
- The pump's running status.

When an anomaly is identified on the pump, the operator may need to look at other pieces of equipment to determine the cause. This could include the following:

- The low level alarm on the sump from which the pump is pumping.
- The level of the tank that the pump is filling.
- The position of the inlet and drain valves.

You can add multiple equipment references from the pump to these associated pieces of equipment. For example, you could create an equipment reference from the pump to the alarm on the source sump. At runtime, when the operator is viewing the information for this pump, they will be able to view the status of the low-level alarm on the sump as well.

Equipment references are defined in the **Equipment** view of the **System Model** activity. See [Define Equipment References](#) for more information.

You can use equipment references to configure interlocks in a Situational Awareness Project. See [Interlocks](#) for more information.

## See Also

- [Configure Interlocks](#)
- [EquipRefBrowseOpen](#)
- [AlarmCountEquipment](#)

## Equipment Editor

Equipment Editor is a tool designed to generate tags in a Plant SCADA project that adhere to an equipment hierarchy.

### To open the Equipment Editor:

1. In the **System Model** activity, select **Equipment**.
2. On the Command Bar, click **Equipment Editor**.

The Equipment Editor interface includes two different views:

- Equipment Types
- Equipment.

These two views are accessible via the tabs along the left edge of the Equipment Editor window, or via the **View** menu.

The View menu also includes a command to **Refresh** the Equipment Editor interface.

You can work with equipment in three easy steps:

1. Create **equipment types** (see [Configure Equipment Types Using Equipment Editor](#)).

The items you include will define the tags, alarms and trends required to support each associated equipment instance.

2. Create **equipment instances** (see [Configure Equipment Instances Using Equipment Editor](#)).

The location of an instance in the equipment hierarchy and its field values provide the information required to create the tags, alarms, trends and states defined in the associated equipment type.

3. Run an **equipment update** (see [Equipment Updates](#)).

This generates the related configuration records required to support each equipment instance.

The Equipment Editor also supports the following functionality:

- [Equipment Property Referencing](#) — syntax you can use to pass property values from an equipment instance to an equipment type
- [Configuration Parameters](#) — named placeholders that allow you to identify and associate values with a piece of equipment.

You can also use the Equipment Editor to [Link a Genie to an Equipment Type](#).

## See Also

[Equipment](#)

## Equipment Updates

An equipment update generates the configuration records associated with the equipment instances defined in the Equipment Editor. The records that are generated for each equipment instance are specified by the items and states defined in the associated equipment type.

---

**Note:** When you perform an equipment update, any numeric values in your project need to be in a format that matches the current system locale setting. For example, a decimal separator needs to be a comma (,) on systems using French, Russian or Turkish. For this reason, you are not able to run an equipment update for the Example project on some non-English locales.

By default, Plant SCADA will perform an incremental equipment update that only includes equipment and equipment types that have been modified since the last update occurred.

If you disable incremental updates, Plant SCADA will regenerate every tag associated with the equipment hierarchy each time an update is run.

### To disable incremental equipment updates:

1. In Plant SCADA Studio, select **Options** from the Activity Bar. The Options dialog will appear.
2. Locate Incremental equipment update on the Options dialog and clear its check box.
3. Click **OK**.

---

**Note:** If you deselect the Incremental equipment update program option, an equipment update will take more time to complete. The time required to complete an update may increase significantly if a project includes a large equipment hierarchy.

---

### See Also

[Update Equipment in Equipment Editor](#)

### Equipment Property Referencing

Equipment property referencing allows you to pass equipment property values from an equipment instance to an equipment type. This information is then used to generate the tags, alarms, trends and states associated with the items defined in the equipment type.

- To refer to a particular equipment property you use the following format:

{equipment.<equipment\_property>}

Where <equipment\_property> is the name of the property that is being referenced.

Be aware that any spaces and non-alphanumeric characters are removed from the field name. For example, the "I/O Device" field is referenced with "{equipment.iodevice}".

- To refer to ungrouped [Configuration Parameters](#), use the following format:

{equipment.param\_list[parameter\_name]}

Where <parameter\_name> is the name of the parameter that is being referenced.

- To refer to a grouped configuration parameter, use the following format:

{equipment.parameter.group\_name[parameter\_name]}

Equipment property referencing is used in the Equipment Editor (see [Configure Equipment Types Using Equipment Editor](#)).

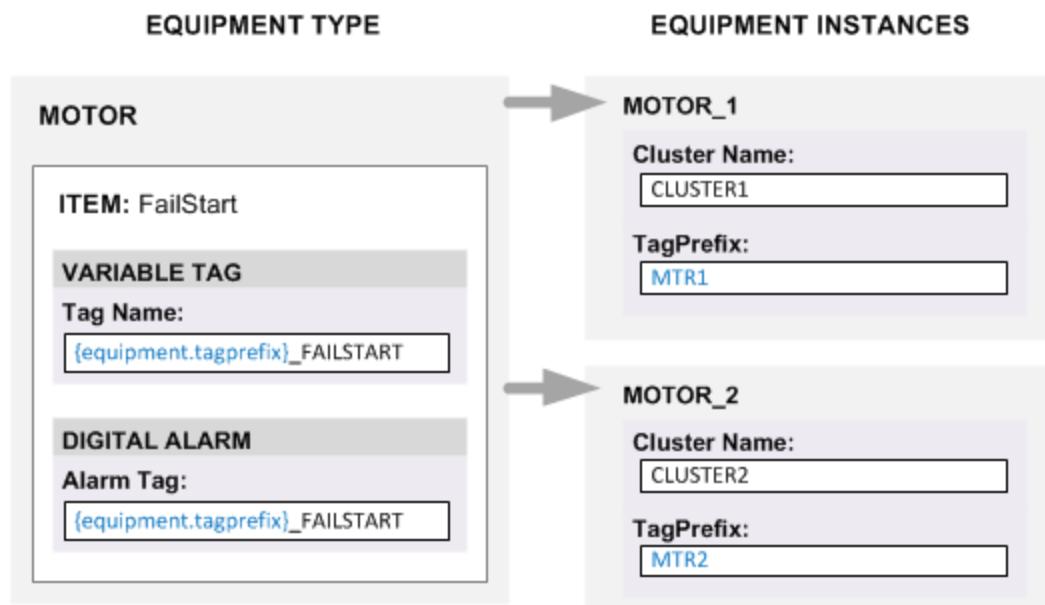
---

**Note:** If you delete a configuration parameter (either directly, or by deleting its parent parameter group), any fields that reference the parameter will also be cleared.

---

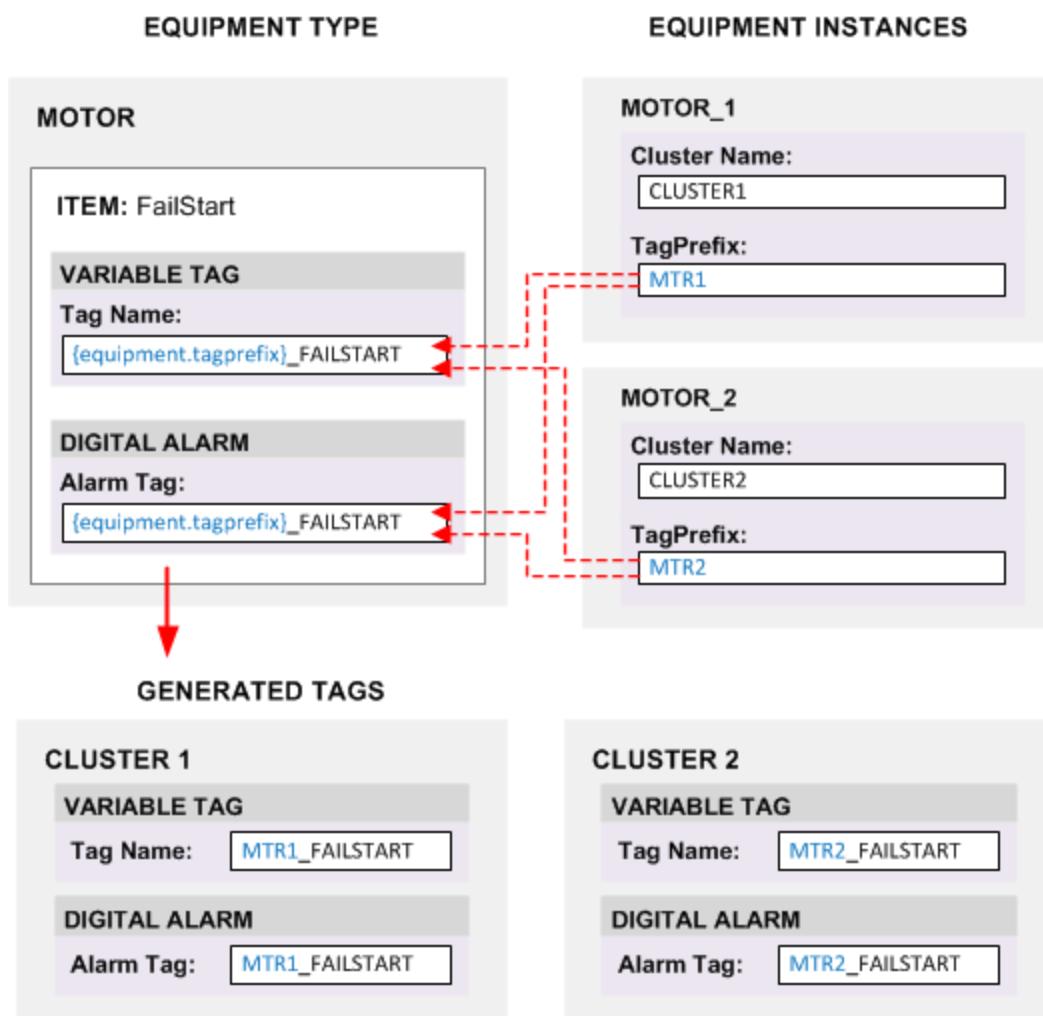
### Example

The following diagram shows two equipment instances ("Motor\_1" and "Motor\_2") based on the "Motor" equipment type. In each instance, the values defined for the **Cluster Name** and **TagPrefix** properties are displayed.



The equipment type includes an item named "FailStart" that has a variable tag and a digital alarm defined as elements. In both cases, the equipment property reference "{equipment.tagprefix}\_FAILSTART" is used to define the name of the associated tags.

In the two equipment instances, the Tag Prefix property has been set to "MTR1" and "MTR2" respectively. When an equipment update occurs, these values are applied to equipment type, causing the tags displayed below to be generated.



These tags are automatically added to the project and linked to the associated equipment instance. Each equipment instance maintains an association with the equipment type from which it was created, allowing changes to be applied across your project.

## See Also

[Edit the Field Values for an Equipment Instance](#)

## Configuration Parameters

Configuration parameters can be added to an equipment type to represent specific information about a physical piece of equipment, such as a set point or delay setting.

When you add a configuration parameter to an equipment type, it creates a named placeholder that is included as a field in each associated equipment instance. Parameters can be grouped together to provide context or added individually as ungrouped parameters.

Configuration parameters support [Equipment Property Referencing](#). You can also specify a default value that is used as the initial value for a parameter when a new instance is created.

To create configuration parameters for an equipment type, use the Equipment Editor (see [Add Configuration](#)

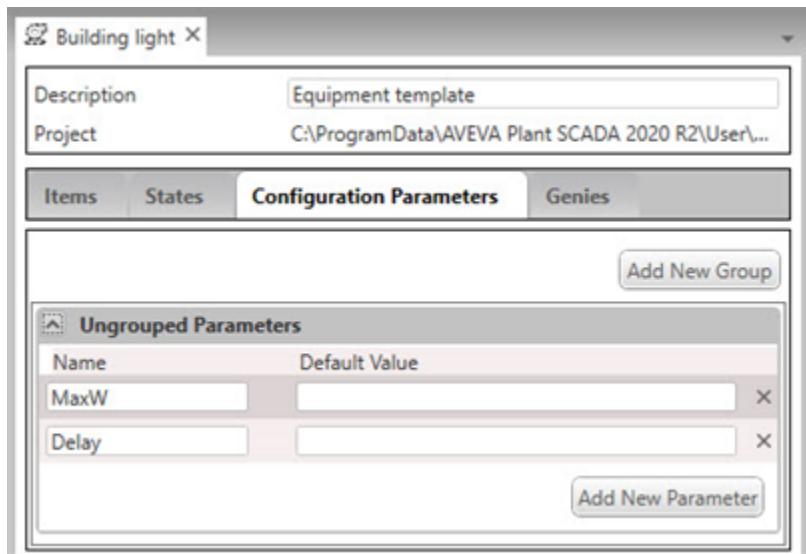
Parameters to an Equipment Type).

**Note:** In Plant SCADA 2023, configuration parameters replaced "custom parameters" as they were restricted by a limitation of 256 characters. Configuration parameters provide the same functionality as custom parameters, but they are stored in a dedicated configuration parameters database. This change required any existing equipment types to be migrated to a new format. If Equipment Editor detects any version 1.0 equipment types, a dialog will appear asking if you want to migrate them. If this occurs, see [Migrate Custom Parameters to the Configuration Parameters Database](#).

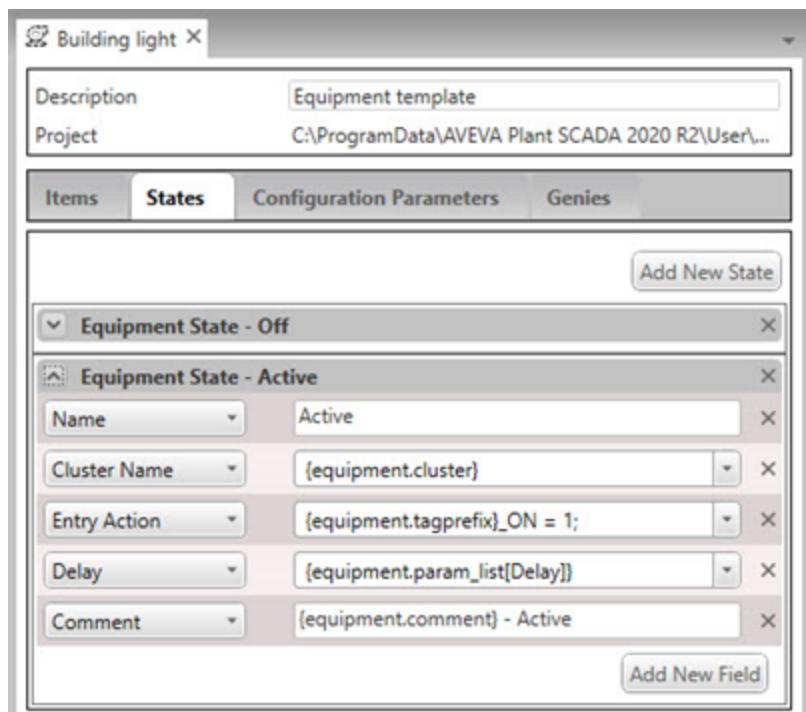
## Example

The Plant SCADA Example project includes an equipment type named "Building Light" that has two configuration parameters defined:

- MaxW
- Delay



An equipment state called "Active" is also configured for the equipment type. It includes a field called **Delay** that references the Delay parameter.



Observe that the equipment property reference in the **Delay** field uses the following syntax:

{equipment.param\_list[Delay]}

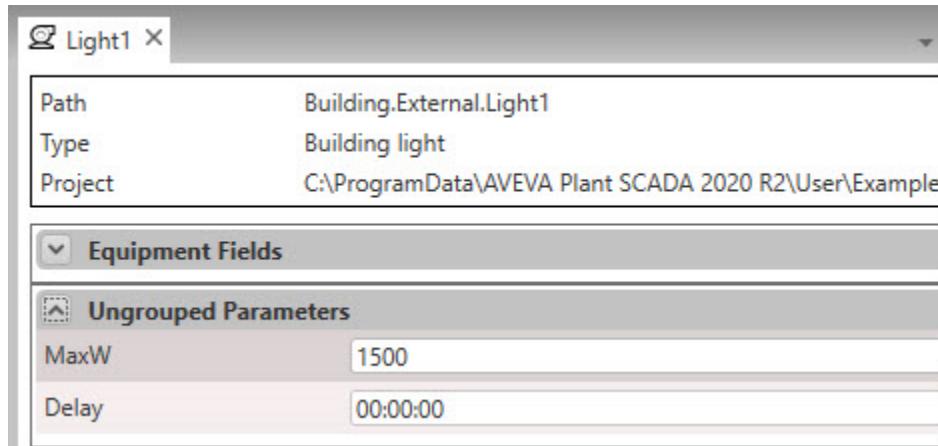
**Note:** The syntax for a grouped parameter will be:

{equipment.parameter.group\_name[parameter\_name]}

See [Equipment Property Referencing](#).

When the value is set for the Delay parameter in an associated equipment instance, it is applied to the Delay property when the equipment state is generated.

Consider the following equipment instance (defined as "Building.External.Light1").



This will result in the creation of the following equipment state (with the Delay property set to "00:00:00").

The screenshot shows the 'Equipment' tab of the 'System Model' editor. The table lists equipment states across seven rows. Row 6 is highlighted with a red border, indicating it is selected. The columns are: Row, Name, Cluster Name, Equipment, Entry Action, and Delay.

Row	Name	Cluster Name	Equipment	Entry Action	Delay
1	Off	Cluster1	Building		
2	Active	Cluster1	Building		
3	Off	Cluster1	Building.External		
4	Active	Cluster1	Building.External		
5	Off	Cluster1	Building.External.Light1	BLD_EXT_L1_ON = 0	
6	Active	Cluster1	Building.External.Light1	BLD_EXT_L1_ON = 1; 00:00:00	
7	Off	Cluster1	Building.External.Light2	BLD_EXT_L2_ON = 0	

You can also view the configuration parameters in a project directly in Plant SCADA Studio (see [Define Equipment Configuration Parameters in Plant SCADA Studio](#)).

## Configure Equipment Types Using Equipment Editor

To configure equipment types in Equipment Editor, you use the **Equipment Types** view. When displayed, it allows you to perform the following configuration tasks:

- Add an Equipment Type
- Open an Equipment Type
- Duplicate an Equipment Type
- Copy an Equipment Type to a User-created Project
- Enable Editing for an Equipment Type in a System Project
- Edit an Equipment Type
- Delete an Equipment Type
- Add an Item to an Equipment Type
- Add an Accumulator to an Equipment Type
- Add an Element to an Item
- Edit the Fields for an Element or Accumulator
- Add a State to an Equipment Type
- Add Configuration Parameters to an Equipment Type
- Link a Genie to an Equipment Type

## See Also

[Update Equipment in Plant SCADA Studio](#)

## Add an Equipment Type

To add an equipment type to a project, you use **Equipment Types** view in [Equipment Editor](#).

**Note:** You can not add an equipment type to a system project (such as a template project or library project). You can only add equipment types to user-created projects.

### To add an equipment type:

1. Open Equipment Editor.
2. Select the **Equipment Types** tab.

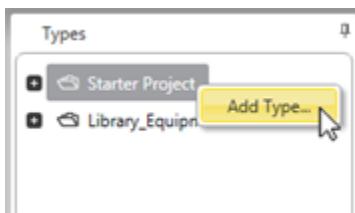
Or:

Select **Equipment Types** from **View** menu.

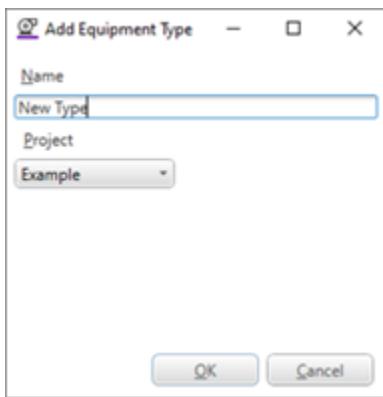
3. In the **Types** list, locate the project to which you would like to add a new equipment type.

The **Types** list includes the project that is currently selected in Plant SCADA Studio and its included projects. If the project you are after is not displayed in the **Types** list, check that the correct project is selected in Plant SCADA Studio. You can then select **Refresh** from the Equipment Editor's **View** menu to reload the **Types** list.

4. Right-click on the project and select **Add Type**.



The Add Equipment Type dialog will display.



5. Enter a **Name** for the equipment type.

**Note:** The name you give to an equipment type at this point cannot be changed once it has been saved. Equipment types cannot be renamed as this impacts the equipment associations defined within the Equipment Editor.

6. Confirm the **Project** to which you would like to add the equipment type. If required, you can select a user-created include project from the drop-down list.
7. Click **OK**.

The dialog will close and the **Types** list will display the new equipment type. The new type will also open for editing.

## See Also

- [Add an Item to an Equipment Type](#)
- [Add an Accumulator to an Equipment Type](#)
- [Add a State to an Equipment Type](#)
- [Add Configuration Parameters to an Equipment Type](#)
- [Link a Genie to an Equipment Type](#)

## Open an Equipment Type

When you open an equipment type, it displays on a tab in the Equipment Editor.

### To open an equipment type:

1. Open Equipment Editor and select the **Equipment Types** tab.
2. Double-click on the required equipment type in the **Types** list.

Or:

Right-click on the equipment type and select **Open** from the menu that appears.

Or:

Select the equipment type and press the **Enter** key.

The equipment type will display on a new tab.

3. Add, delete or modify the items, states and parameters as required. An asterisk will appear in the tab to indicate there are unsaved changes.
4. To save the changes select **Save** or **Save All** from the **File** menu, or use the **Ctrl+S** keyboard shortcut.

**Note:** An equipment type cannot be modified if any associated instances are currently open with unsaved changes.

## See Also

- [Equipment Types](#)
- [Add an Item to an Equipment Type](#)
- [Add an Accumulator to an Equipment Type](#)
- [Add an Element to an Item](#)
- [Edit the Fields for an Element or Accumulator](#)
- [Add a State to an Equipment Type](#)
- [Add Configuration Parameters to an Equipment Type](#)
- [Link a Genie to an Equipment Type](#)

## Duplicate an Equipment Type

When you duplicate an equipment type, the new instance will have "\_n" appended to its name (where "n" is the next integer required to create a unique name).

**Note:** An equipment type cannot be duplicated if any associated instances are currently open with unsaved changes.

#### To duplicate an equipment type:

- Right-click the required equipment type in the Types list and select **Duplicate** from the menu that appears.

The duplicated type will appear in the Types list in the same project branch as the original (with the appended name). The name is highlighted, allowing you to enter a new name if required.

**Note:** The name you give to an equipment type at this point cannot be changed once it has been entered. Equipment types cannot be renamed as this impacts the equipment associations defined within the Equipment Editor.

#### See Also

[Open an Equipment Type](#)

[Copy an Equipment Type to a User-created Project](#)

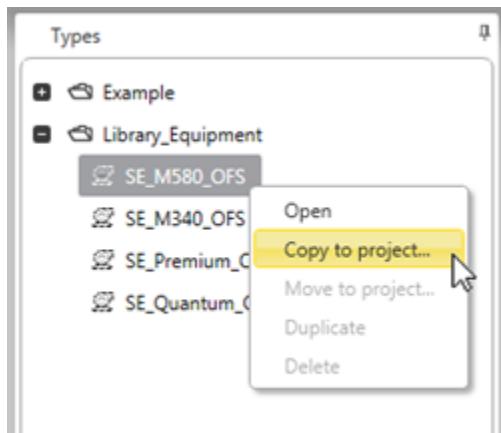
[Delete an Equipment Type](#)

#### Copy an Equipment Type to a User-created Project

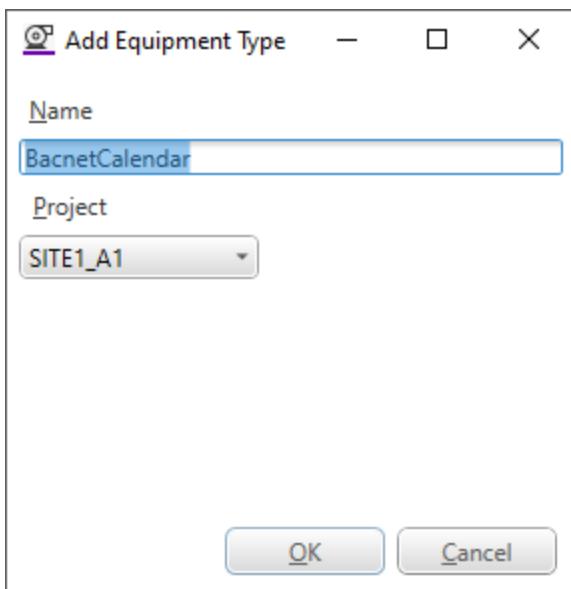
Equipment Types that belong to a system project (such as template or library project) are read-only and cannot be modified. However, you can copy an equipment type to a non-system project and then modify it.

#### To copy an equipment type to a user-created project:

- Open Equipment Editor and select the **Equipment Types** tab.
- In the **Types** list, locate the system project that contains the equipment type you would like to copy.
- Within the project branch, right-click on the required equipment type and select **Copy to project...** from the menu that appears.



- The **Add Equipment Type** dialog will open.



5. If required, enter a new **Name** for the equipment type.  
If a new name is not specified, "*\_n*" will be automatically appended to the existing name (where "*n*" is the next integer required to create a unique name).
6. Select the destination **Project** from the list of available user-created projects.
7. Click **OK**.

The copied equipment type will appear in the **Types** list within the destination project.

## See Also

[Move an Equipment Type to an Included Project](#)

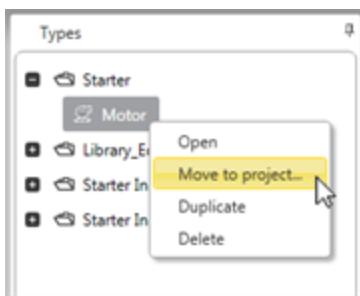
### Move an Equipment Type to an Included Project

You can move an equipment type from its current project to any non-system included projects.

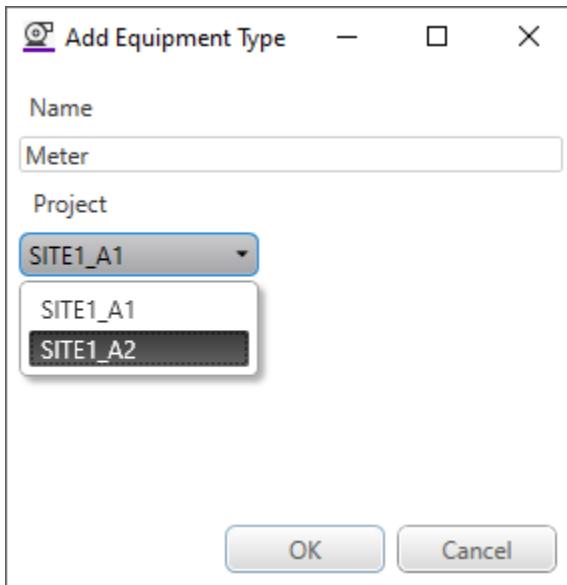
**Note:** An equipment type cannot be moved if any associated instances are currently open with unsaved changes.

#### To move an equipment type to a user-created include project:

1. Open Equipment Editor and select the **Equipment Types** tab.
2. In the **Types** list, locate the project that contains the equipment type you would like to move.
3. Within the project branch, right-click on the required equipment type and select **Move to project** from the menu that appears.



4. The **Add Equipment Type** dialog will open.



5. If required, enter a new **Name** for the equipment type.

If a new name is not specified, "\_n" will be automatically appended to the existing name (where "n" is the next integer required to create a unique name).

6. Select the destination **Project** from the list of user-created included projects.  
7. Click **OK**.

The equipment type will appear in the **Types** list within the selected destination project.

## See Also

[Equipment Types](#)

[Copy an Equipment Type to a User-created Project](#)

## Enable Editing for an Equipment Type in a System Project

Equipment Types that belong to a system project are read-only by default. To make changes to a system project equipment type, you need to make it writable.

**To enable editing for an equipment type in a system project:**

1. Open Equipment Editor and select the **Equipment Types** tab.

2. In the **Types** list, locate the system project that contains the equipment type you would like to edit.
3. Within the project branch, right-click on the required equipment type and select **Enable editing** from the menu that appears.

You will now be able to make changes to the equipment type.

**Note:** You can also copy an equipment type to a non-system project and then modify it (see [Copy an Equipment Type to a User-created Project](#)).

## See Also

[Move an Equipment Type to an Included Project](#)

## Edit Equipment Types

To create and edit **Equipment Types** in Equipment Editor, you need to display the **Equipment Types** view.

**Note:** An equipment type cannot be modified if any associated instances are currently open with unsaved changes.

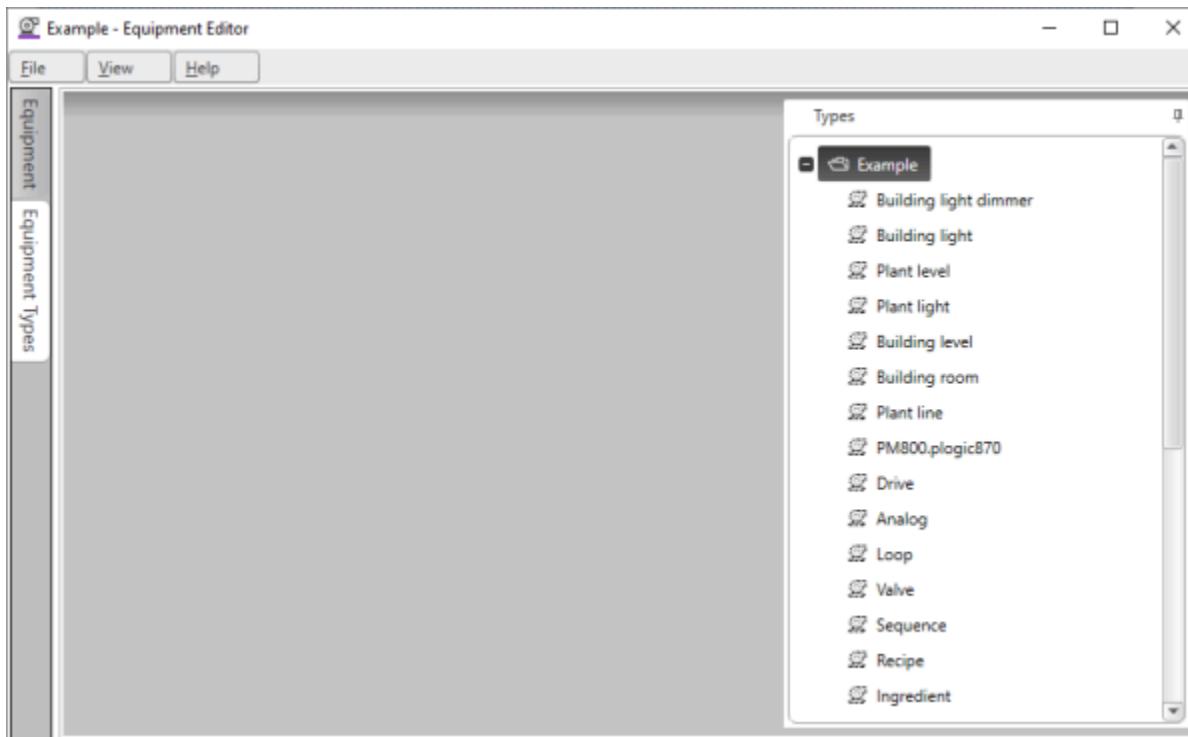
### To edit an Equipment Type:

- Select the **Equipment Types** tab on the left edge of the Equipment Editor window.

Or:

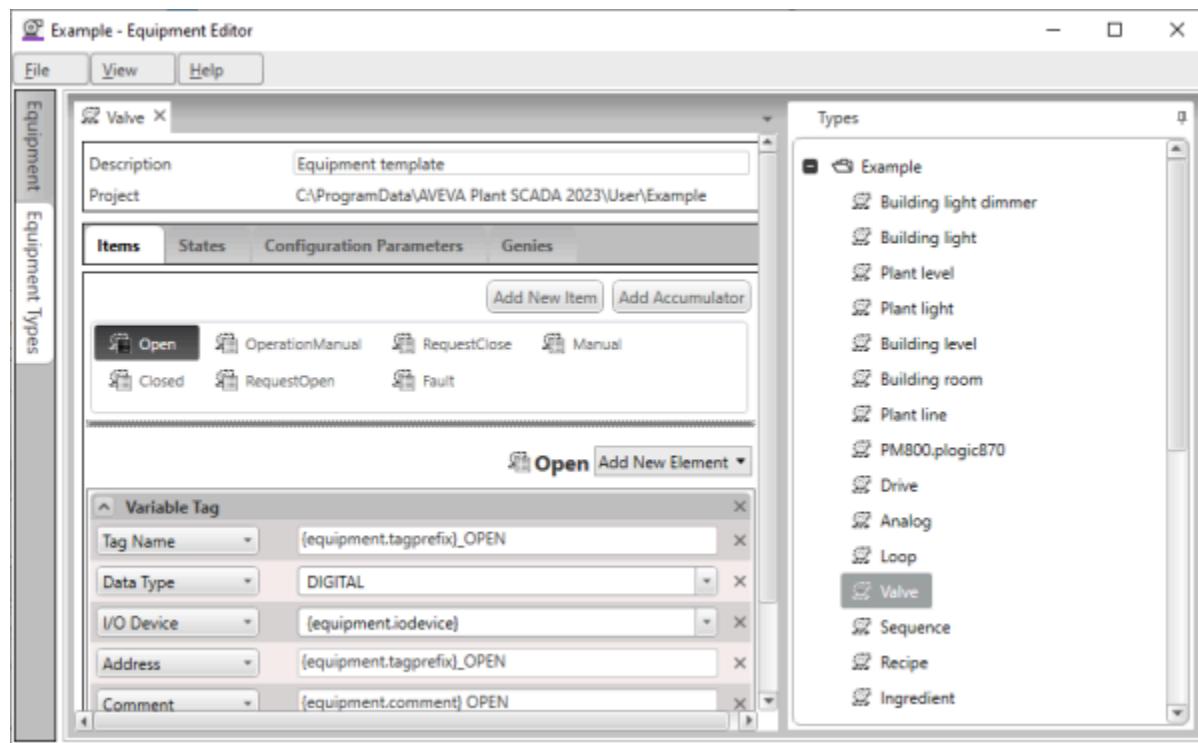
Select **Equipment Types** from the **View** menu.

The **Types** panel lists the existing equipment types in the current project and its included projects.



When you open an equipment type, it displays in the main work area of the Equipment Editor. If two or more

equipment types are open at the same time, each will appear on its own tab.



**Note:** Equipment types that belong to system projects (such as a template or library project) are read-only and cannot be modified.

## See Also

[Configure Equipment Types Using Equipment Editor](#)

## Delete an Equipment Type

Before you can delete an equipment type, you need to save your changes for any associated instances that are currently open.

### To delete an equipment type:

1. Right-click the required equipment type in the Types list and select **Delete** from the menu that appears.  
Or:  
Select the type and press the **Delete** key.  
A confirmation dialog will appear.
2. Click **Yes** to continue with the deletion, or **No** to cancel.

**Note:** If you delete an equipment type, any tags that were generated for the associated equipment instances will also be deleted from your project the next time an equipment update is run. The associated equipment instances themselves are not deleted, however, the Type field for each instance will be updated to 'No type'.

Before you delete an equipment type, it is recommended that you backup your project so that the equipment type can be restored if required.

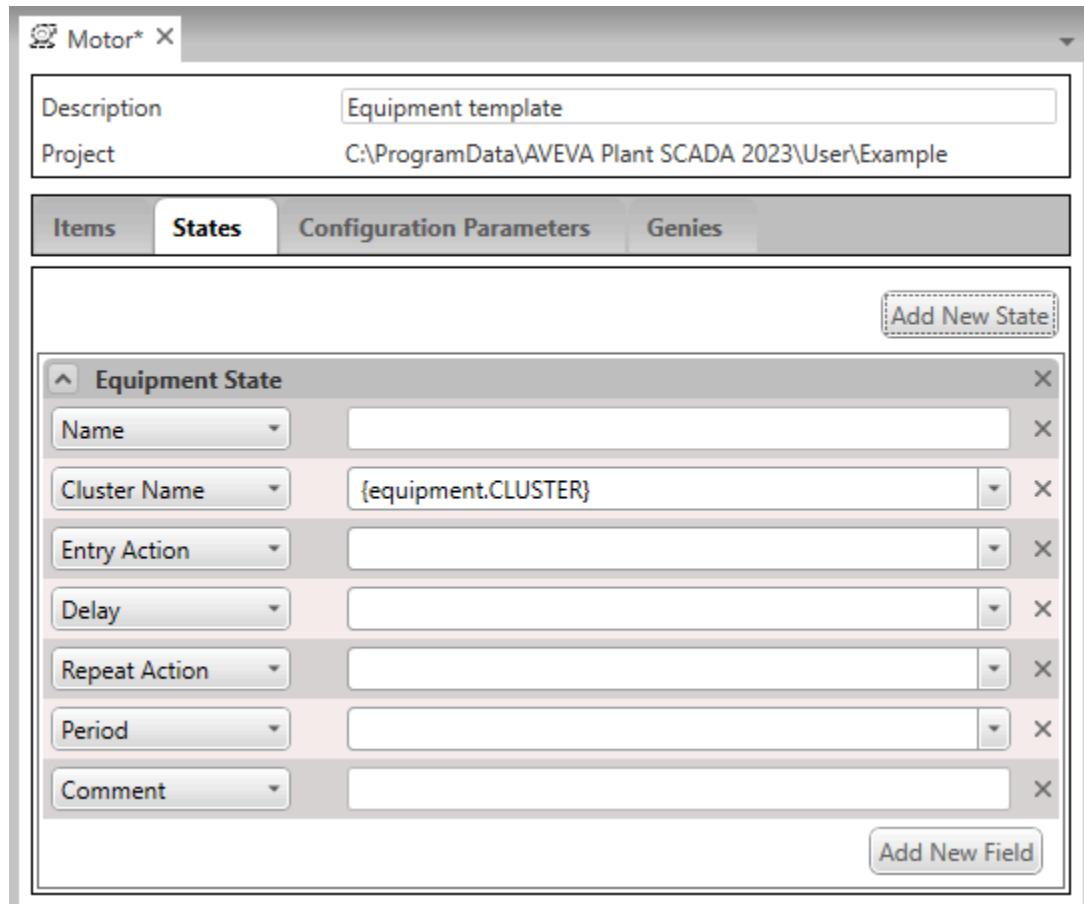
## Add a State to an Equipment Type

You can define multiple [Equipment States](#) for an equipment type to assist with scheduling.

### To add a state to an equipment type:

1. Open Equipment Editor and select the **Equipment Types** tab.
2. Open the equipment type to which you would like to add a state.
3. Select the **States** tab, and click on the **Add New State** button.

A new Equipment State will appear with a set of associated fields displayed.



4. Enter a **Name** for the state.

The name needs to be unique among the states defined for the equipment type. If this is not the case, the Name field will display a red border. This indicates that you need to enter a different name.

5. Enter the required values for the fields. You can use [Equipment Property Referencing](#) if required. In some cases, a menu of suggested options will appear when you select the field.
6. If required, you can add an additional fields by clicking on the **Add New Field** button. The label on a new field is derived from the next available property for the state. (For information on state properties, see [Define Equipment States in Plant SCADA Studio](#).)

To associate a field with a property other than the one indicated by the current label, click on the label and select the required property from the menu that appears.

To remove a field that is not required, click on the close button to the right of the field (x).

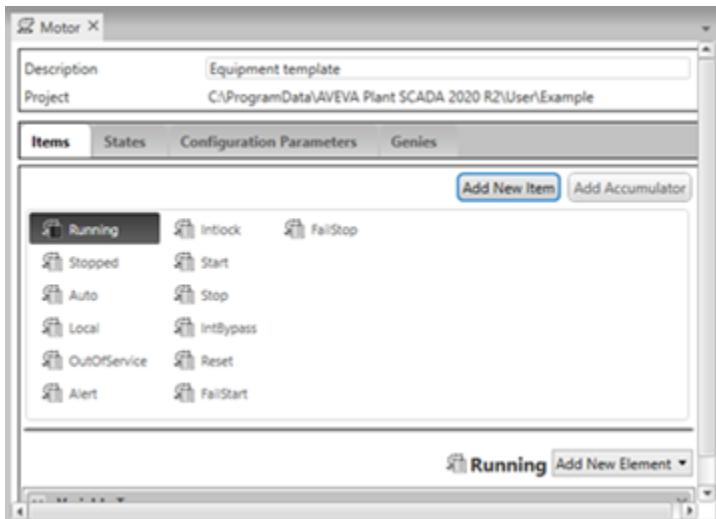
- To save your changes, select **Save** from the **File** menu, or use the **CTRL+S** keyboard shortcut.

## Add an Item to an Equipment Type

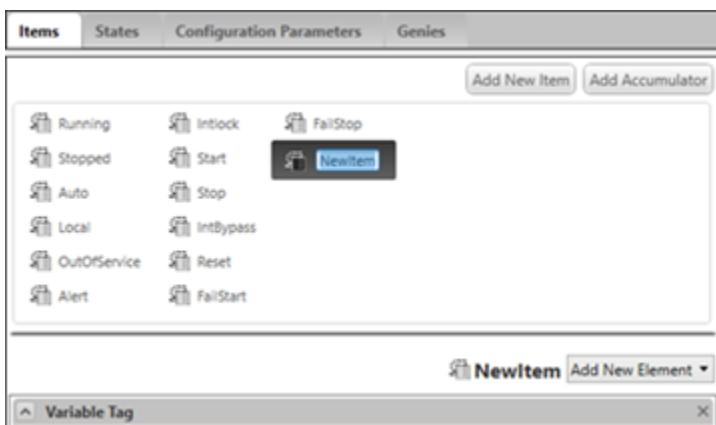
**Items** bring together a set of elements that connect an equipment type to the physical piece of equipment it represents. You can use an item to reflect a particular aspect of a piece of equipment, such as an associated action, condition or settings.

### To add an item to an equipment type:

- Open Equipment Editor and select the **Equipment Types** tab.
- Open the equipment type to which you would like to add an item (see [Open an Equipment Type](#)).
- Select the **Items** tab.
- Click on the **Add New Item** button.



The new item will appear in the list of items with the name "NewItem" highlighted, allowing you to enter a name. Each item requires a unique name.



#### Note:

- When defining an item name, avoid using [Reserved Words](#). If you use any of these, an error message will display when you compile your project.
- You cannot use "me" as an item name, as it is reserved for use with Industrial Graphics.

By default, the new item will include a variable tag as an element.

5. To save your changes, select **Save** from the **File** menu, or use the **Ctrl+S** keyboard shortcut.

## See Also

[Add an Element to an Item](#)

[Add an Accumulator to an Equipment Type](#)

### Add an Element to an Item

Elements represent the configured components of a Plant SCADA project that link an equipment type to the physical device it represents. They are grouped together as [Items](#).

An item can comprise the following elements:

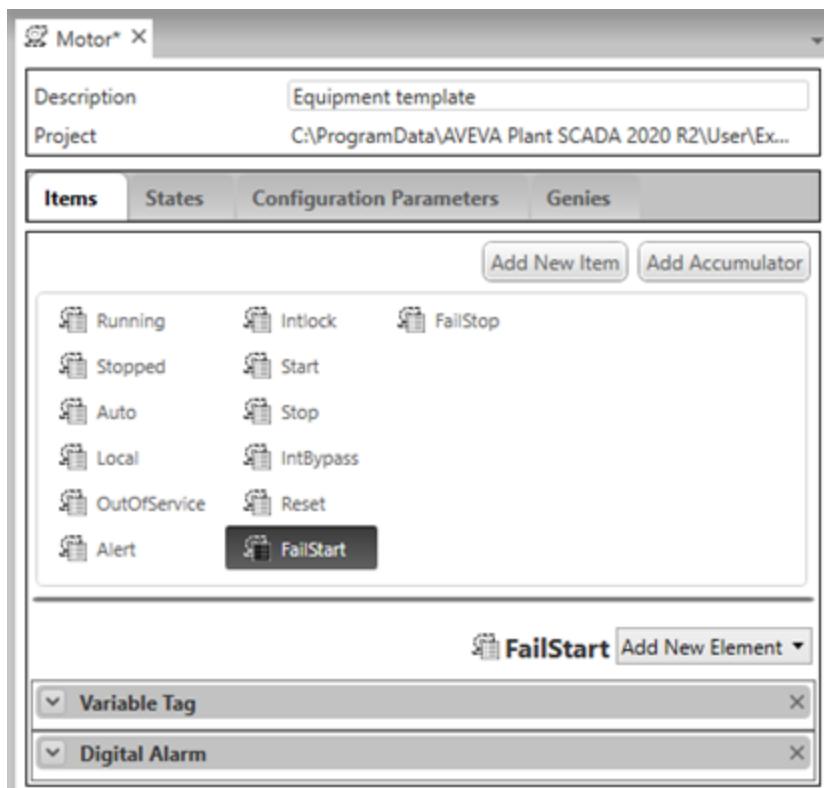
- one variable tag
- one alarm
- one trend tag or SPC tag.

The **Add New Element** menu enforces this limitation.

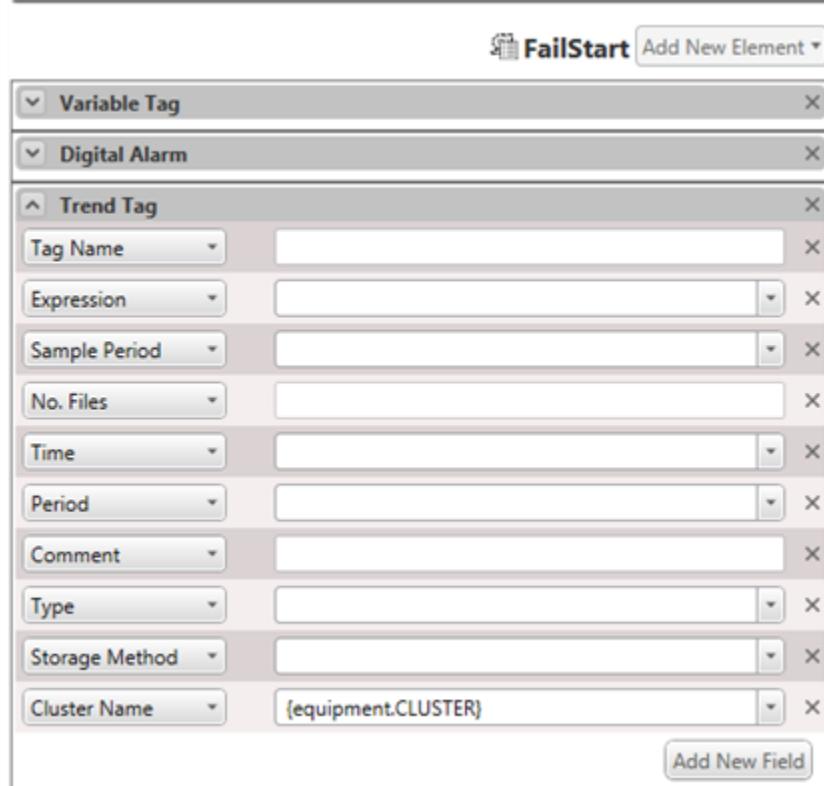
#### To add an element to an item:

1. Open Equipment Editor and select the **Equipment Types** tab.
2. Open the required equipment type (see [Open an Equipment Type](#)).
3. Click on the **Items** tab, and select the item to which you would like to add an element.

The elements that are currently configured for the selected item will appear as a list. In the example below, the **FailStart** item has two elements configured (a variable tag and a digital alarm).



4. Click on the **Add New Element** button and select a type from the list that appears. The new element will appear with its associated fields displayed.



Observe that the **Add New Element** list does not include elements that have already been added to the

item. If this means you need to delete an existing element, select the close button to the right of the element title bar.

5. To save your changes, select **Save** from the **File** menu, or use the **Ctrl+S** keyboard shortcut.

## See Also

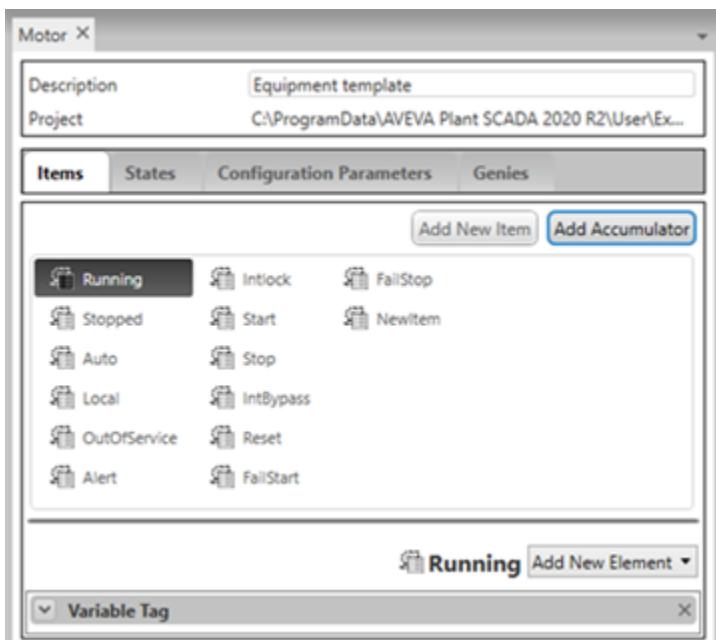
[Edit the Fields for an Element or Accumulator](#)

## Add an Accumulator to an Equipment Type

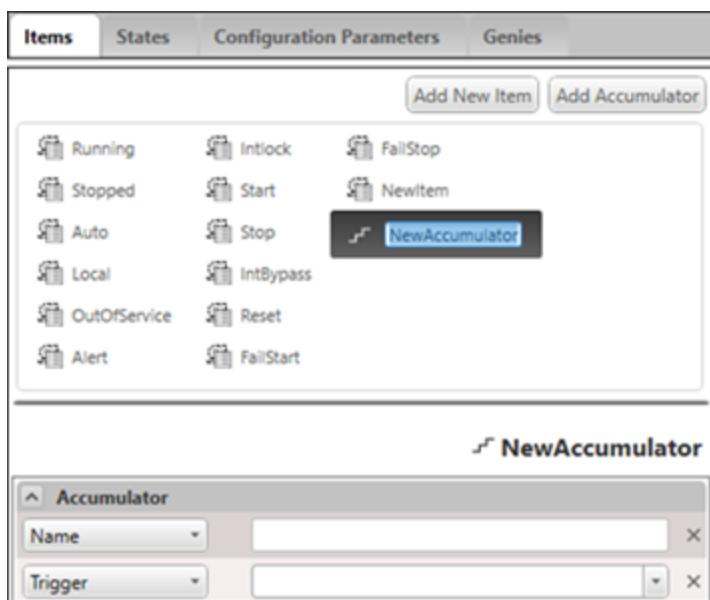
You can associate an accumulator with an equipment type.

### To add an accumulator to equipment type:

1. Open Equipment Editor and select the **Equipment Types** tab.
2. Open the equipment type to which you would like to add an accumulator (see [Open an Equipment Type](#)).
3. Select the **Items** tab.
4. Click on the **Add Accumulator** button.



A new accumulator will appear in the list of items with the name "NewAccumulator" highlighted, allowing you to enter a unique name.



5. To save your changes, select **Save** from the **File** menu, or use the **CTRL+S** keyboard shortcut.

You can add multiple accumulators to an equipment type, however each will require a unique name.

The fields that display for an accumulator are edited in the same way as those associated with an element. For more information, see [Edit the Fields for an Element or Accumulator](#).

## See Also

[Add an Item to an Equipment Type](#)

[Accumulators](#)

## Edit the Fields for an Element or Accumulator

The elements and accumulators associated with an equipment type include a set of fields that you can display and edit in Equipment Editor. These fields reflect the properties of the tag type associated with an element, or the accumulator properties.

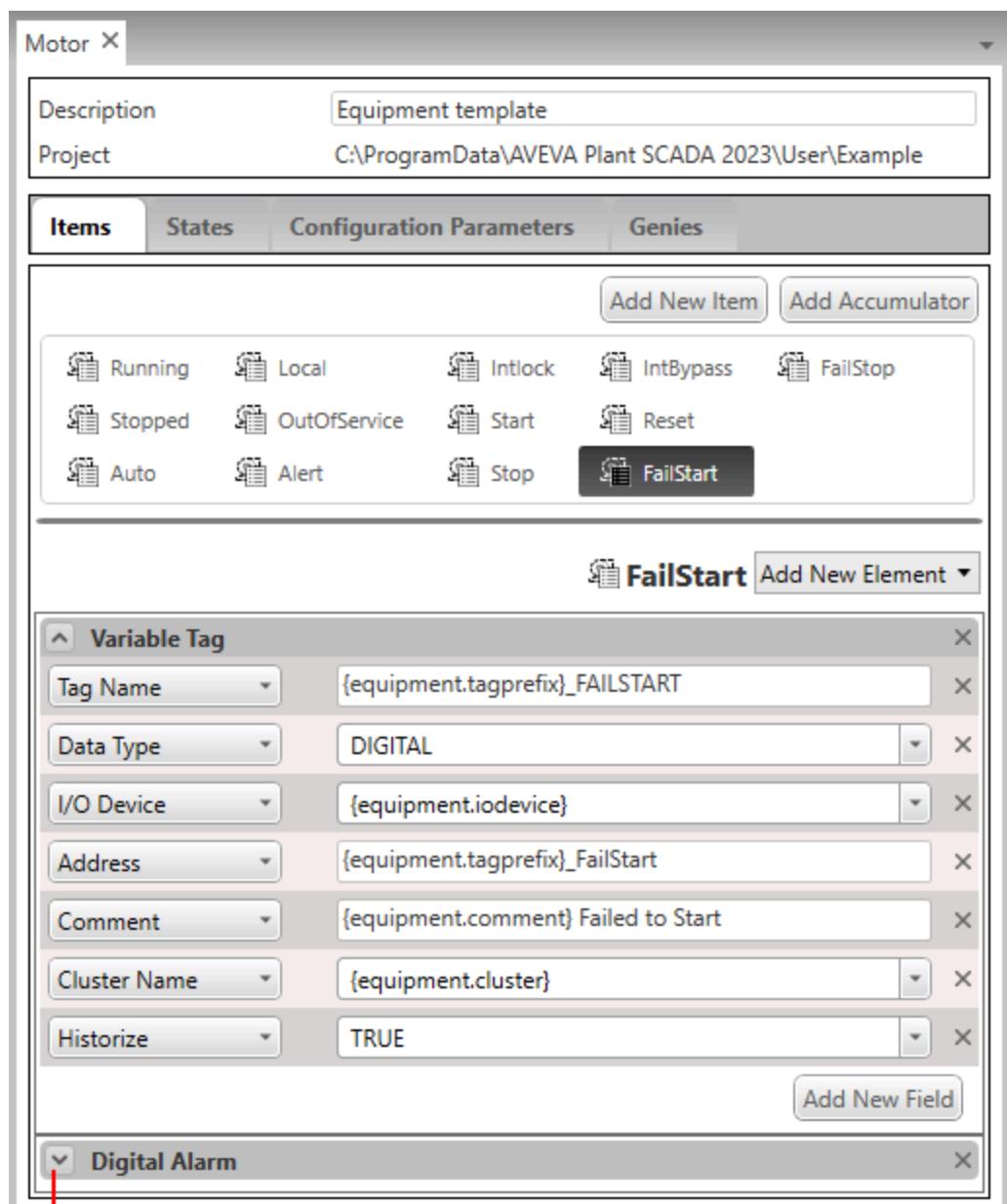
By default, some of the properties associated with a particular tag type or accumulator are not displayed. If required, you can add and remove properties to create a list that contains the fields you need.

You can also apply a value to a field, or use [Equipment Property Referencing](#).

### To edit the fields for an element or accumulator:

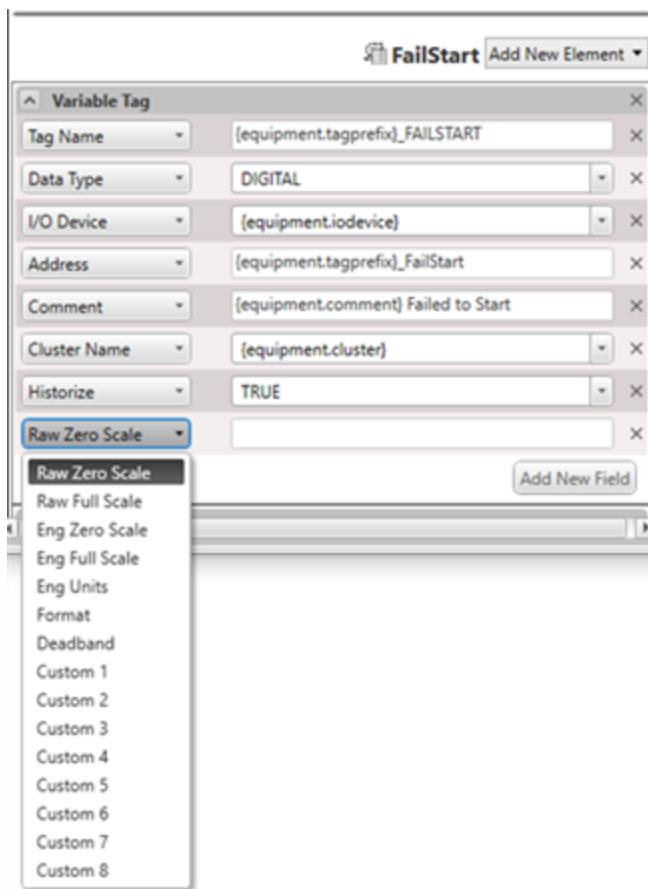
1. Open Equipment Editor and select the **Equipment Types** tab.
2. Open the equipment type you would like to edit (see [Open an Equipment Type](#)).
3. Select the **Items** tab.
4. On the Items tab, select the required item or accumulator. The associated elements/accumulators will display.

You can use the drop-down button on the title bar of each element or accumulator to show or hide the associated fields.



Click here to hide or display the fields for a particular element or accumulator.

5. To add an additional field, click the **Add New Field** button.  
The label on the new field is derived from the next available property for the tag type that is associated with the element.
6. To associate a field with a property other than the one indicated by the current label, click on the drop-down menu and select the required property.



To remove a field that is not required, click on the close button to the right of the field (x).

7. To apply a value to the field, select it and enter the value. You can use [Equipment Property Referencing](#) if required.  
In some cases, a menu of suggested options will appear when you select the field.
8. To save your changes, select **Save** from the **File** menu, or use the **Ctrl+S** keyboard shortcut.

## See Also

[Add an Element to an Item](#)

[Add an Accumulator to an Equipment Type](#)

## Add Configuration Parameters to an Equipment Type

[Configuration Parameters](#) are used to represent specific information about a physical piece of equipment, such as a set point or delay setting.

You can use Equipment Editor to add parameters to an equipment type as part of a created **group**, or you can add them individually as **ungrouped** parameters.

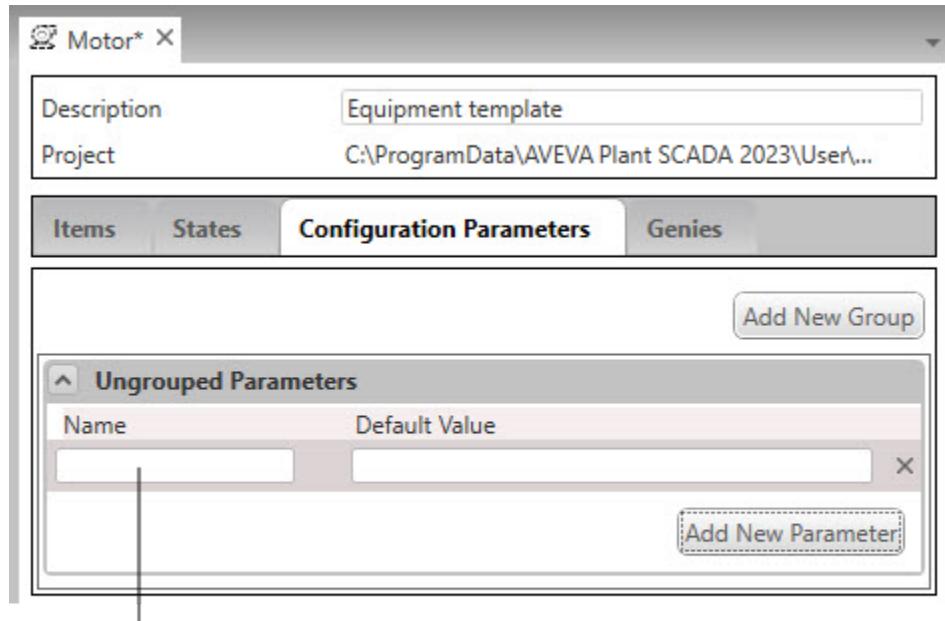
---

**Note:** When you access the Equipment Types tab, a dialog may appear asking if you want to migrate version 1 equipment types and instances. If this occurs, see [Migrate Custom Parameters to the Configuration Parameters Database](#).

### Add an ungrouped parameter to an equipment type:

1. Open Equipment Editor and select the **Equipment Types** tab.
2. Open the Equipment Type to which you would like to add a parameter.
3. Select the **Configuration Parameters** tab.
4. In the **Ungrouped Parameters** panel, click the **Add New Parameter** button.

A new entry is added to the list of ungrouped parameters.



Each time you click on the **Add New Parameter** button, a new entry is added to the list of ungrouped parameters.

5. Enter a **Name** for the parameter.

The name needs to be unique among the ungrouped parameters. If this is not the case, the Name field will display a red border. This indicates that you need to enter a different name.

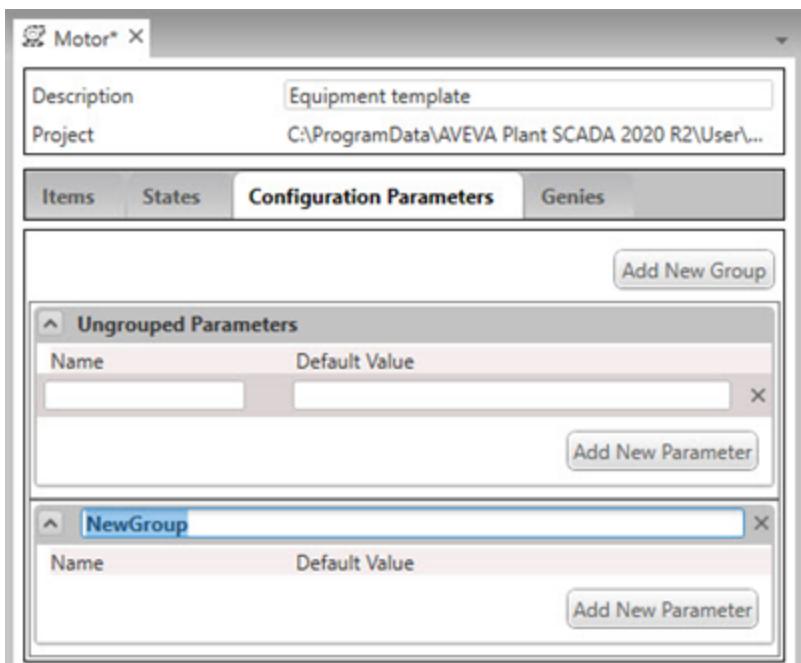
6. If required, enter a **Default Value** for the parameter.
7. To save your changes, select **Save** from the **File** menu, or use the **Ctrl+S** keyboard shortcut.

To delete a configuration parameter, click on the button marked with an "x" within the parameter row.

### Add a parameter to a parameter group:

1. Open Equipment Editor and select the **Equipment Types** tab.
2. Open the Equipment Type to which you would like to add a parameter.
3. On the **Configuration Parameters** tab, click the **Add New Group** button.

A new parameter group will appear with the name "NewGroup" highlighted, allowing you to enter a name. Each parameter group requires a unique name.



4. To add a parameter to the group, click the **Add New Parameter** button within the group you have created. A new entry is added to the parameters group.

5. Enter a **Name** for the parameter.

The name needs to be unique among the parameters in the group. If this is not the case, the Name field will display a red border. This indicates that you need to enter a different name.

6. If required, enter a **Default Value** for the parameter.

7. To save your changes, select **Save** from the **File** menu, or use the **Ctrl+S** keyboard shortcut.

To delete a configuration parameter group, click on the button marked with an "x" next to the group name.

**Note:** If you delete a configuration parameter (either directly, or by deleting its parent parameter group), any fields that reference the parameter will also be cleared.

## See Also

[Equipment Property Referencing](#)

## Link a Genie to an Equipment Type

You can use the Equipment Editor to link a genie to an equipment type. This can help simplify the process of creating graphics pages for your project, as you are able to deploy multiple equipment instances as genies from within the Equipment Editor's equipment hierarchy.

You may want to use the following genie to represent multiple instances of a motor on your graphics pages.



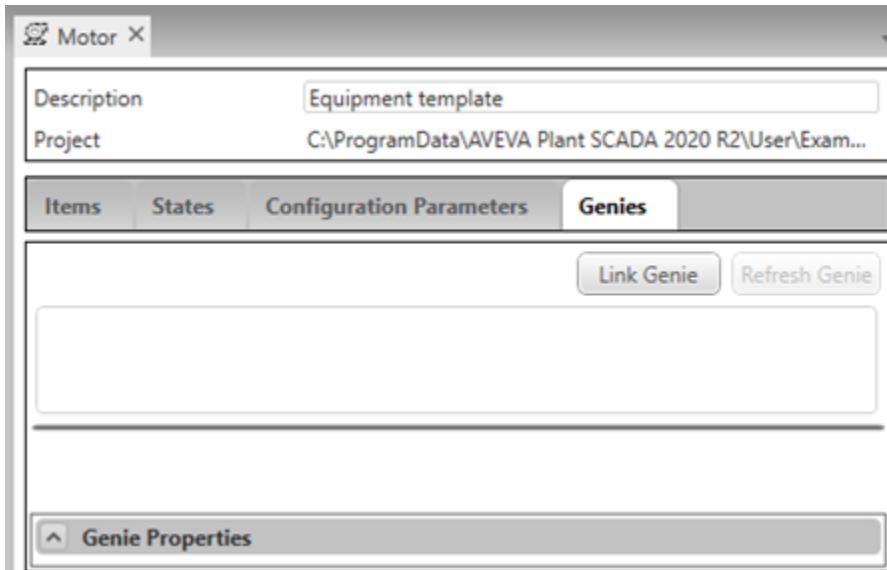
If you were to add multiple instances of the genie to a graphics page, you would have to manually configure the

properties for each copy.

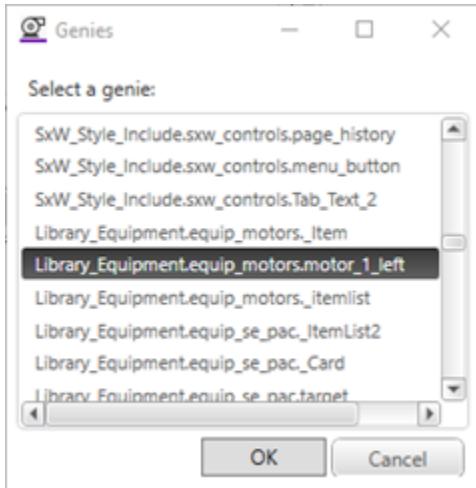
If you were to link the genie to an equipment type, you could deploy the genie from within the Equipment Editor. Any genie properties that are defined within the equipment type will be pre-configured in each pasted genie.

#### To configure the link to a Genie:

1. Open the required equipment type in Equipment Editor.
2. Select the **Genies** tab.

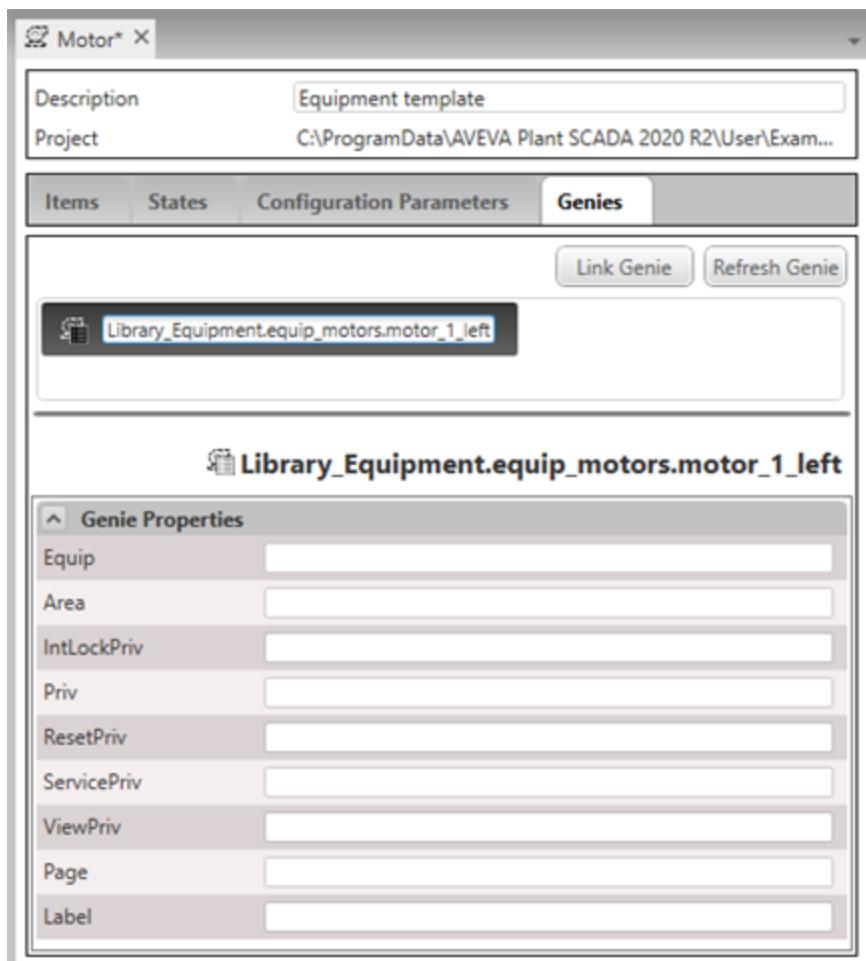


3. Click the **Link Genie** button to open the Genies dialog. The dialog includes a list of the genies defined in the current project and its included projects.



4. Select the genie you would like to link to the equipment type from the list included in the **Select a Genie** field. The list will display the full name for each available genie in the following format:  
<Project name>.<Library name>.<Genie name>
5. Click **OK** to confirm your selection.

The selected genie will appear on the Genies tab for the equipment type to which it is now linked. The associated Genie Properties will also appear.



6. You can edit the values for the **Genie Properties**, except for properties named "Equip" or "Equipment" as they are reserved for the equipment name.

The values you enter will be passed through to each associated equipment instance, and will be retained by the genie when it is pasted on to a graphics page.

If a linked genie has been modified in Graphics Builder, you can use the **Refresh Genie** button to update the Genie Properties.

7. Save your changes to the equipment type.

---

**Note:** You can only associate one genie with an equipment type.

---

#### To change the genie that is linked to an equipment type:

1. On the Genies tab, right-click on the current genie and select **Delete**.
2. Link to a new genie using the steps described in the procedure above.

Or:

1. On the Genies tab, right-click on the current genie and select **Edit**.

Or:

Select the current genie and press the **F2** key.

2. Manually enter the name of a different genie using the following format:

<Project name>.<Library name>.<Genie name>.

---

**Note:** If the specified genie does not exist in the main project or one of its included projects, the genie will display a red border. This indicates that the current genie name is invalid.

---

## See Also

[Paste a Linked Genie on a Graphics Page](#)

### Update Equipment in Equipment Editor

When you run an equipment update, Plant SCADA generates the tags associated with the equipment hierarchy and adds them to your project.

#### To run an equipment update from Equipment Editor:

1. Select **Update Equipment** from the **File** menu or use the keyboard shortcut **Ctrl+Q**.
2. After the equipment update is complete, click **Show Log** to view the log messages.

---

**Note:** If you duplicate an equipment instance with a new name (without changing the tag prefixes) and then run the equipment update, Plant SCADA does not generate tags for the new equipment due to duplicated tag names. The log file provides a record of the tags not generated.

---

By default, Plant SCADA performs an incremental equipment update. It includes only the modified equipment and equipment type after the last update.

If you disable the incremental updates, Plant SCADA regenerates every tag associated with the equipment hierarchy each time an update runs.

#### To disable incremental equipment updates:

1. On the Activity Bar, select **Options**.
2. The **Options** dialog box appears. Locate **Incremental equipment update** on the **Options** dialog box and clear its check box.

---

**Note:** If you clear the **Incremental equipment update** program option, an equipment update takes more time to complete. The time required to complete an update may increase significantly if the project includes a large equipment hierarchy.

---

## See Also

[Update Equipment in Plant SCADA Studio](#)

### Configure Equipment Instances Using Equipment Editor

To configure equipment instances in Equipment Editor, you use the **Equipment** view.

When displayed, it allows you to perform the following configuration tasks:

- [Add an Equipment Instance](#)

- Add an Equipment Instance to a New Hierarchy Level
- Open an Equipment Instance
- Delete an Equipment Instance
- Duplicate an Equipment Instance
- Edit the Field Values for an Equipment Instance
- Paste a Linked Genie on a Graphics Page.

## See Also

[Equipment Instances](#)

[Equipment Updates](#)

### Add an Equipment Instance

To add [Equipment Instances](#) to a project, you use the Equipment Editor's **Equipment** view.

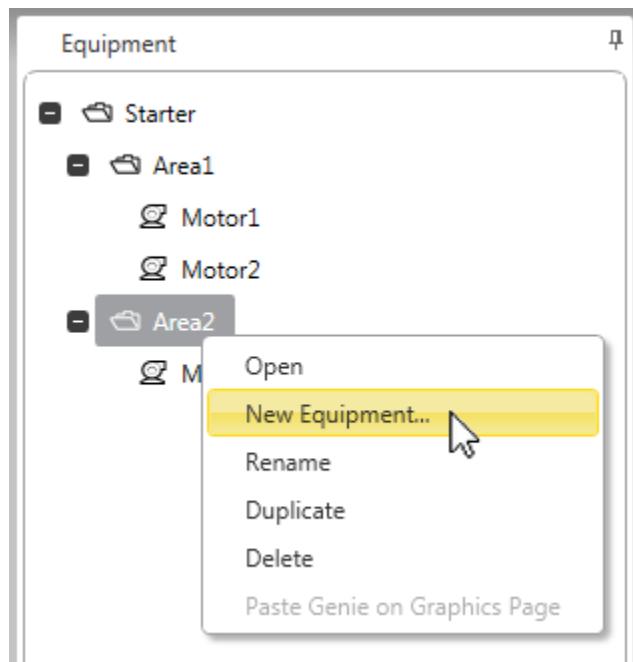
#### To add an equipment instance:

1. Open Equipment Editor.
2. Select the **Equipment** tab.

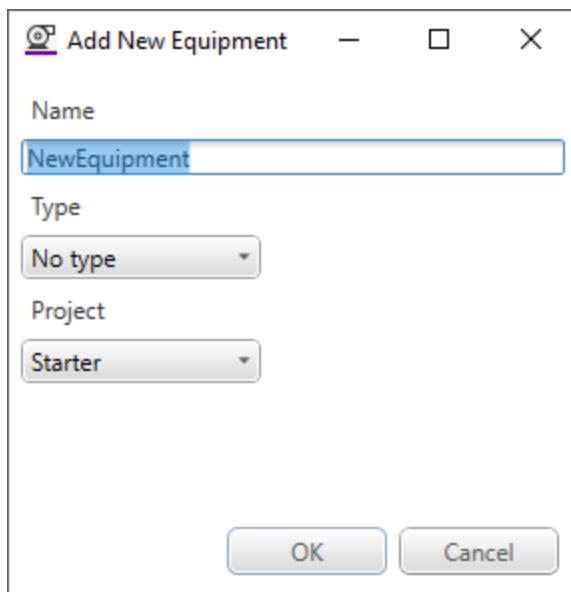
Or:

Select **Equipment** from **View** menu.

3. In the **Equipment** panel, select a location for the equipment instance within the [Equipment Hierarchy](#) for the current project.
4. Right-click and select **New Equipment**.



The Add New Equipment dialog will display.



5. Enter a **Name** for the equipment instance.

The equipment name needs to begin with either an alphabetical character (A-Z or a-z) or the underscore character (\_). Any following characters needs to be either alphanumeric characters (A-Z, a-z or 0-9), backslash characters (\), or underscore characters (\_). An equipment name does not support spaces.

---

**Note:** When defining an equipment name, avoid using the reserved words. If you use any of these, an error message will display when you compile your project.

---

6. In the **Type** field, use the drop-down list to select the equipment type on which the equipment instance will be based. If you select **No Type**, the equipment instance will not maintain an association with an equipment type.

---

**Note:** You cannot create an equipment instance using an equipment type that is currently open with unsaved changes.

---

7. Select the **Project** to which you would like to add the equipment instance. The drop-down list includes the current project and its included projects.
8. Click **OK**.

The new equipment instance will open for editing, and will appear within in the equipment hierarchy in the specified location.

You can also use the Add New Equipment dialog to add levels to the equipment hierarchy when new equipment is added. For more information, see [Add an Equipment Instance to a New Hierarchy Level](#).

## See Also

- [Equipment Instances](#)
- [Open an Equipment Instance](#)
- [Delete an Equipment Instance](#)
- [Duplicate an Equipment Instance](#)

## Open an Equipment Instance

When you open an equipment instance, it displays on a tab in the Equipment Editor.

### To open an equipment instance:

1. Open Equipment Editor and select the **Equipment** tab.
2. Double-click on the required equipment instance in the **Equipment** list.

Or:

Right-click on the equipment instance and select **Open** from the menu that appears.

Or:

Select the equipment instance and press the **Enter** key.

The equipment instance will display on a new tab.

3. Edit the field values as required. An asterisk will appear in the tab to indicate there are unsaved changes.  
To learn more about the fields, see [Define Equipment in Plant SCADA Studio](#).
4. To save the changes select **Save** or **Save All** from the **File** menu, or use the **Ctrl+S** keyboard shortcut.

---

**Note:** You cannot make changes to an equipment instance if its associated equipment type is currently open with unsaved changes.

## See Also

[Equipment Instances](#)

[Edit the Field Values for an Equipment Instance](#)

[Delete an Equipment Instance](#)

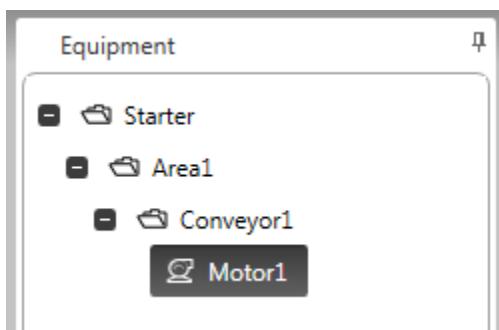
[Duplicate an Equipment Instance](#)

## Add an Equipment Instance to a New Hierarchy Level

When you add a new equipment instance to a project, you can include additional hierarchy levels by using the period character (.) within the equipment name. For example, you could create a new equipment instance in the root directory of a project with the following equipment name:

Area1.Conveyor1.Motor1

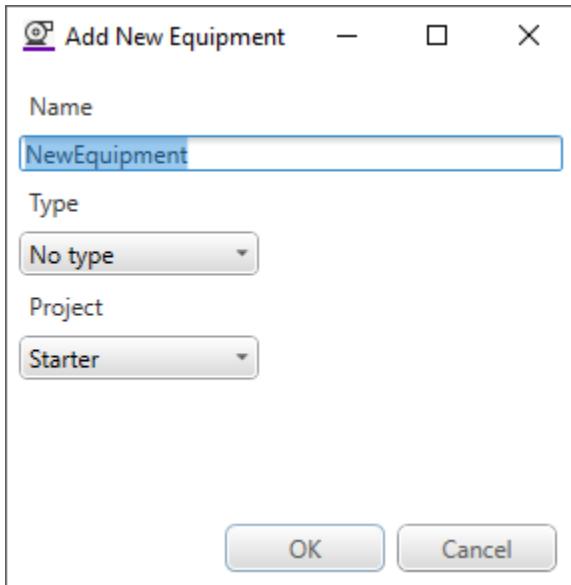
This would result in the following addition to the project's equipment hierarchy:



**To add an equipment instance to a new hierarchy level:**

1. Open Equipment Editor and select the **Equipment** tab.
2. In the **Equipment** panel, select a location for the equipment instance (and any additional hierarchy levels) within the **Equipment Hierarchy**.
3. Right-click and select **New Equipment**.

The Add New Equipment dialog will display.



4. Enter a **Name** for the equipment instance and any additional hierarchy levels using the following syntax:

<hierarchy level>.<hierarchy level>.<equipment name>

The maximum number of levels that can be defined in a hierarchy is 14.

Remember that the equipment name needs to begin with either an alphabetical character (A-Z or a-z) or the underscore character (\_). Any following characters needs to be either alphanumeric characters (A-Z, a-z or 0-9), backslash characters (\), or underscore characters (\_). An equipment name does not support spaces.

---

**Note:** You cannot use "me" as an equipment name, as it is reserved for use with Industrial Graphics.

5. In the **Type** field, use the drop-down list to select the equipment type on which the equipment instance will be based. If you select **No Type**, the equipment instance will not maintain an association with an equipment type.
6. Select the **Project** to which you would like to add the equipment type. The drop-down list includes the current project and its included projects.
7. Click **OK**.

The equipment instance will open for editing on a new tab, and will appear within in the equipment hierarchy in the specified location.

**See Also**

- [Open an Equipment Instance](#)
- [Delete an Equipment Instance](#)

## Duplicate an Equipment Instance

### Delete an Equipment Instance

#### To delete an equipment instance:

1. Right-click the required equipment instance in the Equipment hierarchy and select **Delete** from the menu that appears.

Or:

Select the instance and press the **Delete** key.

A confirmation dialog will appear.

2. Click **Yes** to continue with the deletion, or **No** to cancel.

**Note:** If you delete an equipment instance, any associated tags that were generated will also be deleted from your project the next time an equipment update is run.

### See Also

[Delete an Equipment Type](#)

### Duplicate an Equipment Instance

When you duplicate an equipment instance, the new instance will have "\_n" appended to its name (where "n" is the next integer required to create a unique name).

**Note:** You cannot duplicate an equipment instance if its associated equipment type is currently open with unsaved changes.

#### To duplicate an equipment instance:

- Right-click the required equipment instance in the Equipment panel hierarchy and select **Duplicate** from the menu that appears.

The duplicated instance will appear in the same hierarchy branch as the original (with the appended name). The name is highlighted, allowing you to enter a new name if required.

**Note:** If you duplicate an equipment instance with a new name (without changing the tag prefixes) and then run the equipment update, Plant SCADA does not generate tags for the new equipment due to duplicated tag names. The log file provides a record of the tags not generated.

### See Also

[Open an Equipment Instance](#)

### Edit the Field Values for an Equipment Instance

You can edit the equipment fields and custom parameter values for an equipment instance.

If the associated equipment type is configured to use equipment property referencing, the values you enter may be used to generate the tags, alarms and trends associated with the equipment type (see [Equipment Property](#)

Referencing).

---

**Note:** You cannot make changes to an equipment instance if its associated equipment type is currently open with unsaved changes.

**To edit the field values for an equipment instance:**

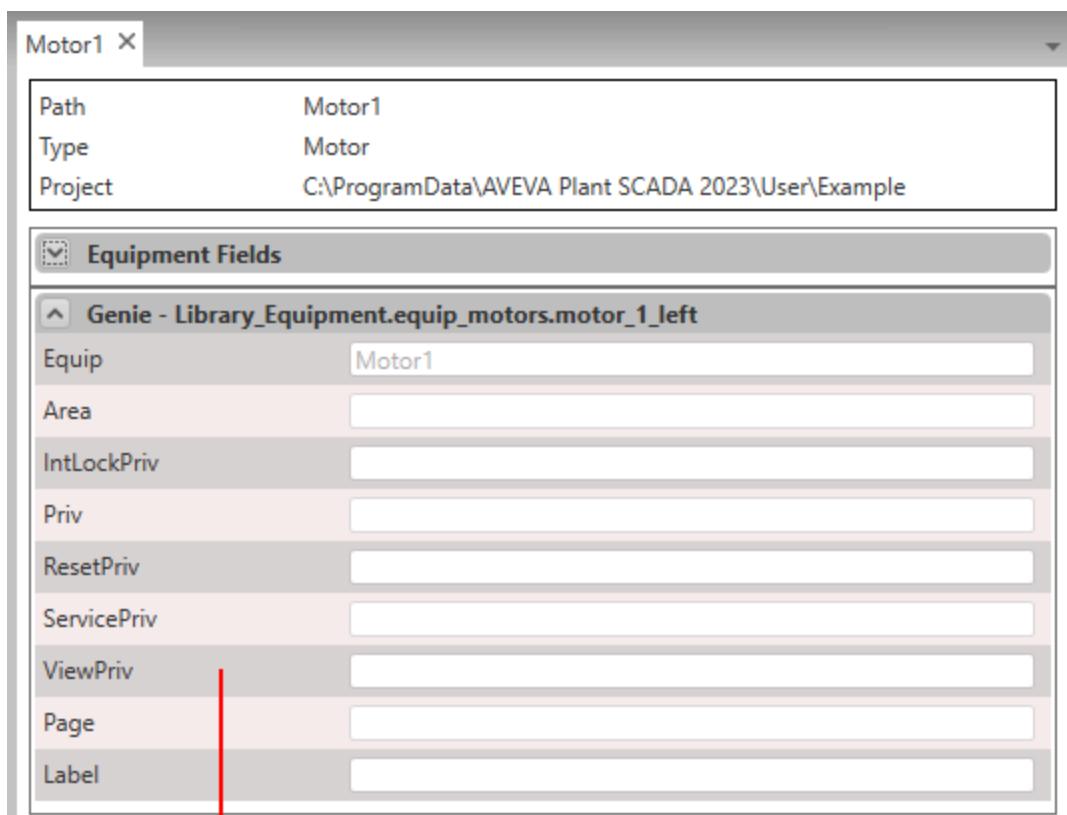
1. Open Equipment Editor and select the **Equipment** tab.
  2. Open the equipment instance for which you would like to edit the fields values (see [Open an Equipment Instance](#)).
- The equipment fields and configuration parameters that are associated with the equipment instance will be available to display. To learn more about the fields, see [Define Equipment in Plant SCADA Studio](#).
3. Locate the equipment field or custom parameter you would like to edit.
  4. Enter the required value.
  5. To save your changes, select **Save** from the **File** menu, or use the **Ctrl+S** keyboard shortcut.

**Paste a Linked Genie on a Graphics Page**

If an equipment instance is associated with an equipment type that has a genie linked to it, you can paste the genie on a graphics page using the Equipment Editor.

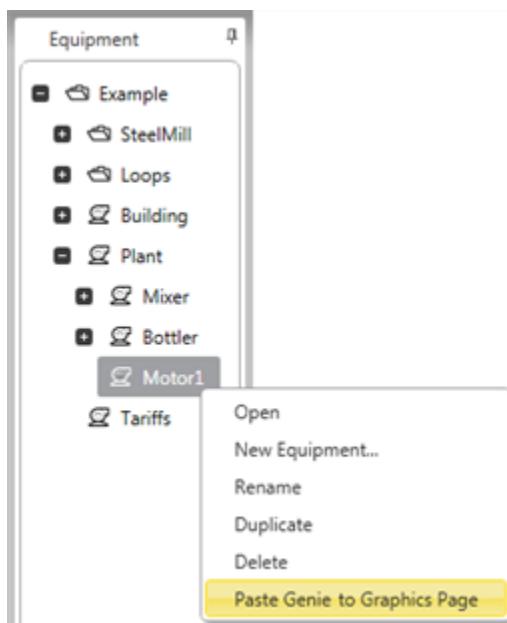
**To paste a linked genie on a graphics page for an equipment instance:**

1. In Graphics Builder, open the graphics page that will be the destination for the genie.
  2. In Equipment Editor, select the **Equipment** tab.
  3. In the Equipment tree, locate the equipment instance you would like to represent with a linked genie.
- If required, you can confirm that the associated equipment type has a genie linked to it by checking the equipment properties for a section named "Genie".



If an equipment instance has a set of Genie properties, it indicates that the associated equipment type has a Genie linked to it.

4. Right-click on the equipment instance in the Equipment tree and select **Paste Genie on to Graphics Page**. If this menu option is not available, it indicates that you need to open a page in Graphics Builder.



The linked genie will appear on the graphics page that is currently open in Graphics Builder.

If you repeat this process multiple times, the genies will be pasted on top of each on the destination graphics page. You will need to manually relocate each genie to display them individually.

If the paste is unsuccessful for a particular reason (for example, the linked genie no longer exists), a notification dialog will appear.

## See Also

[Link a Genie to an Equipment Type](#)

## Configure Equipment in Plant SCADA Studio

The way you configure equipment in Plant SCADA will initially depend on whether or not your project has existing tags. If you have a project with existing tags, you can use Plant SCADA Studio to associate equipment with your tags (see [Define an Equipment Association for a Tag](#)).

Equipment can be manually configured in the System Model activity of Plant SCADA Studio. Here you can perform the following tasks:

- [Define Equipment in Plant SCADA Studio](#)
- [Define Equipment Types in Plant SCADA Studio](#)
- [Define Equipment States in Plant SCADA Studio](#)
- [Define Equipment Configuration Parameters in Plant SCADA Studio](#)
- [Define Equipment References](#)
- [Define Equipment Runtime Parameters](#)
- [Update Equipment in Plant SCADA Studio](#).

---

**Note:** If you are starting a new project (or adding a new area or process to an existing project) you can use the Equipment Editor to generate tags (see [Equipment Editor](#)).

---

## Associate Existing Tags with an Equipment Hierarchy

Plant SCADA allows you to build an [Equipment Hierarchy](#) into a project that is already configured to include tags, alarms, trends, and so on.

When you build an equipment hierarchy into an existing system, you create a way to reference your tags using logical groupings that may reflect geographical areas or functional processes.

Associating equipment with tags in an existing project is a two-step process:

1. Define the required equipment.  
To achieve this, manually add the required equipment definitions to **Equipment** list in the **System Model** activity (see [Define Equipment in Plant SCADA Studio](#)).
2. Associate tags with the equipment you have defined.  
To achieve this, you set the Equipment property for a tag in the **System Model** activity (see [Define an Equipment Association for a Tag](#)).

You can also define equipment associations for your tags directly in the project database using Microsoft Excel™

(see [Use the Project DBF Add-In to Define Equipment Associations](#)).

## See Also

[Equipment](#)

### Define Equipment in Plant SCADA Studio

You can manually add equipment definitions to a Plant SCADA project that adhere to an [Equipment Hierarchy](#).

The hierarchy is created from the names you apply to the equipment you define. If you include a period (.) in a name, it specifies a hierarchy level. You can then use this hierarchy as a way to reference the tags in your project using logical equipment groupings (see [Define an Equipment Association for a Tag](#)).

To link equipment to equipment and equipment.items outside the equipment hierarchy, you can create equipment references (see [Define Equipment References](#)).

---

**Note:** If you are adding equipment to an existing project, it is recommended that you leave the **Type** field blank. An equipment type is primarily used to generate new tags from Equipment Editor, and this is not required if your tags already exist. If you want to define equipment that is associated with an equipment type, you should use the Equipment Editor instead of Plant SCADA Studio (see [Equipment Editor](#)).

#### To manually define equipment:

1. In the **System Model** activity, select **Equipment**.
2. On the menu below the Command Bar, select **Model**.  
The list of equipment definitions will display in the Grid Editor.
3. Add a row to the Grid Editor.
4. Type the required information in each column, or in the Property Grid (see below for a description of the properties).
5. Click **Save**.

The equipment you have defined will now be available to associate with the existing tags in your project (see [Define an Equipment Association for a Tag](#)).

You can also [Define Equipment States in Plant SCADA Studio](#) for the equipment to support scheduling.

## Equipment Properties

### General Properties

Property	Description
<b>Name</b>	<p>The name of the equipment. This can be a name that represents a single piece of equipment, or a name that also reflects the location of the equipment within an <a href="#">Equipment Hierarchy</a> (where a period (. ) is used to indicate levels in the hierarchy).</p> <p>The length of the complete name for the equipment is recommended to be less than 128 characters. The</p>

Property	Description
	<p>maximum length is 254 chars, however, long names can be harder to use within configuration expressions and Cicode.</p> <p>When naming equipment the following rules apply:</p> <ul style="list-style-type: none"> <li>• Equipment names need to be unique within the same cluster.</li> <li>• Equipment names cannot be the same as alarm tag names.</li> <li>• A period (.) may be used in equipment name to specify equipment hierarchy levels, for example:</li> <li>• Plant1.Filling.Tank1.</li> <li>• Each period delimiter denotes an extra level in the equipment hierarchy.</li> <li>• Up to 14 hierarchy levels are supported. The name applied to each hierarchy level may not exceed 63 characters.</li> <li>• Individual names (delimited by a period) within an equipment path need to follow tag name syntax. For example, characters such as '%', '?' and '/' are not allowed.</li> <li>• The name of root level equipment may not be the same as any cluster names.</li> <li>• The name of the root level equipment may not be a reserved word, such as IF, FOR, WHILE, END, OR, or AND.</li> </ul>
<b>Display Name</b>	A meaningful name for the equipment that can be used at runtime to easily identify it. Enter a value of 254 characters or less.
<b>Cluster Name</b>	The name of the cluster to which the equipment is assigned. The specified cluster should be configured otherwise a compile will not be successful. If no cluster name is set, the equipment will run on every cluster.
<b>Type</b>	The specific type of equipment in the system. This drop down box is populated from the types database created in the <a href="#">Equipment Types</a> settings.
<b>Location</b>	A string describing the location of the equipment. Enter a value of 254 characters or less.
<b>Page</b>	<p>The name of the page on which this equipment appears. Enter a value of 254 characters or less.</p> <p>This enables an operator to navigate directly to the page that hosts the piece of equipment. If the graphic</p>

Property	Description
	<p>object appears on more than one page, the page specified here should represent its primary operational context.</p> <p>Where duplicated variations of a page exist to suit HD1080 and UHD4K screen resolutions, the page that is used is determined by the resolution of the host workspace. This means you do not need to add the suffix "_HD1080" or "_UHD4K" to the end of a name to specify a particular page. For more information, see <a href="#">Configure a Project that Supports Multiple Screen Resolutions</a>.</p>
<b>Content</b>	<p>A comma-separated list that names the content associated with the piece of equipment. This can include pages, faceplates and documents. For example:</p> <p>Area1_Page_L1, Process1_Page_L2, Pump_Faceplate_FP</p> <p>This content will be available at runtime when the equipment comes into context.</p> <p>Where duplicated variations of a faceplate or page exist to suit HD1080 and UHD4K screen resolutions, the content that is called is determined by the resolution of the host workspace. This means you do not need to add the suffix "_HD1080" or "_UHD4K" to the end of a name. For more information, see <a href="#">Configure a Project that Supports Multiple Screen Resolutions</a>.</p>
<b>Help</b>	The help context string. Enter a value of 254 characters or less.
<b>Comment</b>	Any useful comment. Enter a value of 254 characters or less.
<b>Tag Prefix</b>	Designed to store a prefix that can be applied to the tags generated for the equipment instance. It can be used at runtime to find tags associated with the equipment.
<b>I/O Device</b>	The I/O device used to communicate with this piece of equipment. Specify redundant devices with a comma between primary and standby, for example, "IODev1,IODev2". Enter a value of 254 characters or less.

Property	Description
<b>Hidden</b>	<p>When set to true the Equipment will not be visible in the Equipment Tree at runtime.</p> <p><b>Note:</b> The functionality of the Equipment does not change and will continue to function as configured.</p>
<b>Unique ID</b>	<p>A unique ID is applied to each piece of equipment in a Plant SCADA project to provide consistent identification for integration with future software releases. This field is read-only in Plant SCADA Studio.</p> <p>Each unique IDs is a Globally Unique Identifier (GUID) generated by Plant SCADA using the following format: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX</p> <p>For more information, see <a href="#">Unique IDs for Variable Tags and Equipment</a>.</p> <p><b>Note:</b> It is highly recommended that you only use Plant SCADA to generate the unique identifiers for the equipment in your system. If for some reason this is not possible, use a reputable tool to generate GUIDs that match the format described above.</p> <p>If you use an external tool to modify the equipment database, the compiler will indicate if a Unique ID is invalid, duplicated or missing.</p>

### Security Properties

Property	Description
<b>Area</b>	<p>The area number or label to which this equipment belongs. Only users with access to this area (and any necessary privileges) will be able to perform operations on the equipment. For example, if you enter Area 1 here, operators need to have access to Area 1 (plus any necessary privileges) to perform operations on the equipment. Enter a value of 16 characters or less.</p>

### Custom Properties

Property	Description
<b>Custom1 .. Custom8</b>	<p>User-defined strings that can be used for filtering equipment when using the Cicode search functions (maximum 254 characters each). Labels or tags are not supported.</p>

### Scheduling Properties

**Note:** The following fields, **Default State** and **Scheduled**, are used when configuring schedules for equipment

(see [Configure Equipment for Scheduling](#)).

Property	Description
<b>Default State</b>	The default state applied to the equipment.
<b>Scheduled</b>	Specifies if the equipment is included in any scheduled actions. The options are "TRUE" or "FALSE". The following fields, <b>Schedule ID</b> and <b>Device Schedule</b> , are only required when using a schedule configured locally on a BACnet device (see <a href="#">Integrate BACnet Schedules into Scheduler</a> ). They require the <b>Scheduled</b> property to be set to "TRUE".
<b>Schedule ID</b>	The ID used to identify the required schedule on the BACnet device. This field is only required if the <b>Device Schedule</b> property is set to "TRUE".
<b>Device Schedule</b>	Specifies whether or not the report server should retrieve the required schedule from the BACnet device. The options are "TRUE" or "FALSE".

#### Deprecated Properties

Property	Description
<b>Parameters</b>	Prior to the release of Plant SCADA 2023, "custom parameters" were used to apply specific values to an equipment type. In Plant SCADA 2023, custom parameters were replaced with <a href="#">Configuration Parameters</a> . If an equipment instance is based on an equipment type that uses a version 1 template, this field will show the associated custom parameters. <b>Note:</b> Version 1 equipment types that support this parameter are flagged as deprecated and may not be supported in future release. To migrate your equipment types, see <a href="#">Migrate Custom Parameters to the Configuration Parameters Database</a> . Following migration, the content of this field will be populated at runtime from the content of the configuration parameters for this equipment instance. You should not make any changes to this field in the configuration environment.

#### Project Properties

Property	Description
<b>Project</b>	The project in which the equipment type is

Property	Description
	configured.

**Note:** Advanced users can use the Project DBF plug-in for Microsoft Excel to edit and save records directly in the EQUIP.DBF file (see [Use the Project DBF Add-In to Define Equipment Associations](#)).

## See Also

- [Define Equipment Types in Plant SCADA Studio](#)
- [Define Equipment States in Plant SCADA Studio](#)

### Define Equipment Types in Plant SCADA Studio

You can manually add equipment types to a Plant SCADA project. You would typically do this to incorporate an external XML template for an equipment type (see [Import Equipment Using XML Templates](#)).

#### To manually define an equipment type:

1. In the System Model activity, select **Equipment**.
2. On the menu below the Command Bar, select **Types**.  
The list of equipment types will display in the Grid Editor.
3. Add a row to the Grid Editor.
4. Type the required information in each column, or in the Property Grid (see below for a description of the properties).
5. Click **Save**.

**Note:** You can also use the Equipment Editor to create equipment types. These equipment type definitions will appear in the equipment type list alongside those you create manually. See [Add an Equipment Type](#).

## Equipment Type Properties

### General Properties

Property	Description
<b>Name</b>	A name that reflects the physical piece of equipment the equipment type represents, such as "pump", "light" or "valve". Enter a value of 254 characters or less. The names in this field populate the Type drop down in Equipment Properties (see <a href="#">Define Equipment in Plant SCADA Studio</a> ).
<b>Template</b>	The file name for the XML template that defines the equipment type. Enter a value of 254 characters or less.

	If the equipment type was created using Equipment Editor, this file will have the same name as the equipment type (with the .xml extension).  If you are using an externally created equipment XML template, enter the file name here to associate it with the equipment type (see <a href="#">Import Equipment Using XML Templates</a> for more information).
<b>Comment</b>	Any useful comment. Enter a value of 254 characters or less.
<b>Unique ID</b>	A unique ID is applied to each equipment type in a Plant SCADA project to provide consistent identification for integration with future software releases. This field is read-only in Plant SCADA Studio.  Each unique IDs is a Globally Unique Identifier (GUID) generated by Plant SCADA using the following format: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX  For more information, see <a href="#">Unique IDs for Variable Tags and Equipment</a> .  <b>Note:</b> It is highly recommended that you only use Plant SCADA to generate the unique identifiers for the equipment types in your system. If for some reason this is not possible, use a reputable tool to generate GUIDs that match the format described above.  If you use an external tool to modify the equipment types database, the compiler will indicate if a Unique ID is invalid, duplicated or missing.

### Project Properties

Property	Description
<b>Project</b>	The project in which the equipment state is configured.

### See Also

[Define Equipment in Plant SCADA Studio](#)

### Define Equipment States in Plant SCADA Studio

You can manually add [Equipment States](#) to a Plant SCADA project and associate them with your equipment definitions.

#### To manually define an equipment state:

1. In the System Model activity, select **Equipment**.

2. On the menu below the Command Bar, select **States**.  
The list of equipment states will display in the Grid Editor.
3. Add a row to the Grid Editor.
4. Type the required information in each column, or in the Property Grid (see below for a description of the properties).
5. Click **Save**.

**Note:** Updates made to the fields on Equipment States will come into effect at runtime once the project is compiled and reloaded.

## WARNING

### UNINTENDED EQUIPMENT OPERATION

Do not use the Scheduler in situations where equipment state actions must be executed. The engine does not ensure that defined actions are executed for every state transition. The Scheduler engine will not be able to run actions when other state actions for the same equipment take longer time to complete or a communication fault occurs.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Equipment State Properties

### General Properties

Property	Description
<b>Name</b>	Name of the State. Usually this would be an 'action' the equipment could be scheduled to do. For example, a light could have the action 'on' or 'active' defined.
<b>Cluster Name</b>	The name of the cluster to which this State will belong.
<b>Equipment</b>	The equipment associated with this state. For example: "Factory.Floor.Line1.Light1". See <a href="#">Equipment</a> . Equipment should be an existing item on the selected cluster.

### Actions Properties

Property	Description
<b>Entry Action</b>	The action the State will take when triggered. For example the entry action for light 1 could be tag_light1=1. When light 1 is scheduled, this will trigger the light to turn on. The Entry Action can be a Cicode expression.

Property	Description
	<p><b>Note:</b> You should not enter a Cicode function that relates to a graphics page in this field, as this type of function will only execute successfully on a display client.</p> <p>Be aware that long actions on this field may delay or prevent other scheduled actions from running. For example, if there is "action1" for startup and "action2" for repeat, if "action1" is an infinite loop, "action2" will be prevented from running. If "action2" takes a long time to complete, no new "action2" actions will run until the former finishes. If you need to run a long task, create an independent Cicode task using <a href="#">TaskNew</a> function.</p> <p>You can use the INI parameter <a href="#">[Scheduling]AlwaysExecuteEntryAction</a> to specify that the entry actions for all equipment resource states are executed, regardless of whether or not their internal states are the same. By default, this parameter is set to false (0), which means only changes that have been made to the selected equipment will take effect.</p>
<b>Delay</b>	<p>The time that needs to elapse before the state will initially become active.</p> <p>If delay is defined the Entry Action for the state will not occur until the delay period has elapsed. For example; this would allow you to turn on every light sequentially to avoid a power spike.</p>
<b>Repeat Action</b>	<p>An action set to run periodically when state is active. Repeat Actions can be a Cicode expression.</p> <p><b>Note:</b> You should not enter a Cicode function that relates to a graphics page in this field, as this type of function will only execute successfully on a display client.</p> <p>Be aware that long actions on this field may delay or prevent other scheduled actions from running. For example, if there is "action1" for startup and "action2" for repeat, if "action1" is an infinite loop, "action2" will be prevented from running. If "action2" takes a long time to complete, no new "action2" actions will run until the former finishes. If you need to run a long task, create an independent Cicode task using <a href="#">TaskNew</a> function.</p>
<b>Period</b>	<p>Sets the frequency of the 'Repeat Action'. For example set the repeat action to occur every hour.</p>

Property	Description
	<b>Note:</b> The repeat period will not start until the entry action has completed. The first repeat action will occur after the entry action completes plus one repeat time period. For example, if the Entry Action takes ten minutes to complete, the Repeat Action will not start until after ten minutes even though the Repeat Action is set to occur every two minutes.
<b>Priority</b>	Number used to determine which schedule entry will run if a schedule conflict occurs between more than one state.
<b>Comment</b>	Any useful comment.
<b>DR Mode</b>	Demand and Response (DR) mode (see <a href="#">Configure a Demand and Response Solution</a> ). The default DR mode value is 0.

### Project Properties

Property	Description
<b>Project</b>	The project in which the equipment state is configured.

**Note:** If the report server is run in single process mode and no user is logged in, all tag writes attempted by the scheduler when actions are run will not succeed. We recommend running the report server(s) in multi-process mode to avoid this issue (the server user will be logged in), or setting the citect.ini parameter [\[Client\]AutoLoginMode](#) to either 2 or 6 to make sure a user is always logged in.

### See Also

[Define Equipment in Plant SCADA Studio](#)

### Define Equipment Configuration Parameters in Plant SCADA Studio

Equipment configuration parameters can be applied to an equipment type to represent specific information about a physical piece of equipment, such as a set point or delay setting. A value can then be applied to this parameter for each associated equipment instance.

You can manually add equipment configuration parameters to a Plant SCADA project in Plant SCADA Studio.

#### To define an equipment configuration parameter:

1. In the System Model activity, select **Equipment**.
2. On the menu below the Command Bar, select **Configuration Parameters**.

A list of the configuration parameters included in the active project will display in the Grid Editor.

**Note:** When you access the Configuration Parameters view, a dialog may appear asking if you want to

migrate version 1 equipment types and instances. If this occurs, see [Migrate Custom Parameters to the Configuration Parameters Database](#).

3. Add a row to the Grid Editor.
4. Type the required information in each column, or in the Property Grid (see below for a description of the properties).
5. Click **Save**.

## Equipment Configuration Parameter Properties

### General Properties

Property	Description
<b>Cluster</b>	The name of the cluster to which the equipment is assigned.
<b>Equipment</b>	The equipment associated with this parameter. For example: "Factory.Floor.Line1.Light1". See <a href="#">Equipment</a> . Equipment should be an existing item on the selected cluster.
<b>Group</b>	If a parameter has been added to an equipment type as part of a created group, this field will indicate the group name. The field will be blank for any ungrouped parameters.
<b>Name</b>	A name for the parameter. Enter a value of 254 characters or less.
<b>Value</b>	Value assigned to the parameter. This could be a value or a tag name.

### Project Properties

Property	Description
<b>Project</b>	The project in which the equipment runtime parameter is configured.

## See Also

[Add Configuration Parameters to an Equipment Type](#)

### Define Equipment References

You can add equipment references to a Plant SCADA project. See [Equipment References](#) for more information.

**To manually define an equipment reference:**

1. In the System Model activity, select **Equipment**.
2. On the menu below the Command Bar, select **References**.  
The list of equipment references will display in the Grid Editor.
3. Add a row to the Grid Editor.
4. Type the required information in each column, or in the Property Grid (see below for a description of the properties).
5. Click **Save**.

**Equipment Reference Properties****General Properties**

Property	Description
<b>Cluster</b>	The name of the cluster to which the equipment is assigned.
<b>Equipment</b>	The name of the main piece of Equipment to which referenced equipment is linked from.
<b>Referenced Cluster</b>	The name of the cluster to which the referenced equipment is assigned.
<b>Referenced Equipment</b>	The name of the Equipment that you want to reference (link to).
<b>Referenced Item</b>	The name of the item belonging to the "referenced" Equipment. If the Item is listed only that referenced item will be returned using EquipRefBrowse functions. If this field is left blank items belonging to the referenced equipment will be available.
<b>Category</b>	A category that will be used to filter equipment references using EquipRefBrowse functions. You can use any separator between filter strings. For example, "Equipment.MT", "Equipment-MT" and "Equipment:MT" are all valid.  If you are configuring an interlock, the Situational Awareness Starter Project provides the following categories (see <a href="#">Interlocks</a> for definitions): <ul style="list-style-type: none"><li>• Interlock.Process</li><li>• Interlock.Safety</li><li>• Interlock.Permissive.</li></ul> Enter the whole name of the category type (for

Property	Description
	example, "Interlock.Process").
<b>Order</b>	A unique index per reference category that will be used in the ordering for the EquipRefBrowse functions.
<b>Association</b>	Controls the name of the association used in the <a href="#">AssEquipReferences</a> Cicode function. For each reference, a super-genie association is created which uses the name specified in the Association column. <b>Note:</b> Association names are also passed into the 'meo_selector_xx' genie instances on faceplates e.g ?MEO2?. See <a href="#">Configure Equipment Selection for Group Objects</a> for more information.
<b>Comment</b>	Any useful comment. Enter a value of 254 characters or less.

**Custom Properties**

Property	Description
<b>Custom1 .. Custom8</b>	User-defined strings that can be used for filtering equipment when using the EquipRefBrowse functions (maximum 254 characters each).

**Project Properties**

Property	Description
<b>Project</b>	Name of project

**See Also**

[Interlocks](#)  
[EquipRefBrowseOpen](#)  
[AlarmCountEquipment](#)

**Define Equipment Runtime Parameters**

Equipment runtime parameters are a set of parameters that are passed into content pages as associations. For example, a sub-page with a faceplate that displays information for multiple pieces of equipment.

---

**Note:** Equipment runtime parameters can be used with the [AssEquipParameters](#) and [EquipGetParameter](#) Cicode functions.

You can manually add equipment runtime parameters to a Plant SCADA project.

**To define an equipment runtime parameter:**

1. In the System Model activity, select **Equipment**.
2. On the menu below the Command Bar, select **Runtime Parameters**.  
The list of equipment runtime parameters will display in the Grid Editor.
3. Add a row to the Grid Editor.
4. Type the required information in each column, or in the Property Grid (see below for a description of the properties).
5. Click **Save**.

**Equipment Runtime Parameter Properties****General Properties**

<b>Property</b>	<b>Description</b>
<b>Cluster</b>	The name of the cluster to which the equipment is assigned.
<b>Equipment</b>	The equipment associated with this runtime parameter. For example: "Factory.Floor.Line1.Light1". See <a href="#">Equipment</a> . Equipment should be an existing item on the selected cluster.
<b>Name</b>	A name for the parameter. Enter a value of 254 characters or less.
<b>Value</b>	Value assigned to the parameter. This could be a value or a tag name.
<b>Is Tag</b>	By default, this is TRUE. If the value is set to a tag name, the tag value will be evaluated as such. Set this to FALSE if you have set a value manually instead of specifying a tag name. <b>Note:</b> If this is a tag name it can be used in conjunction with the Cicode function <a href="#">AssEquipParameters</a> to pass those tags as Super Genie associations.
<b>Comment</b>	Any useful comment. Enter a value of 254 characters or less.

**Project Properties**

<b>Property</b>	<b>Description</b>
<b>Project</b>	The project in which the equipment runtime

Property	Description
	parameter is configured.

## Define an Equipment Association for a Tag

You define equipment in a Plant SCADA project to create logical groupings that reflect the machinery or processes being monitored. When you specify an equipment association for a tag, you are simply adding the tag to one of the equipment groups you have defined.

This allows you to use a project's equipment hierarchy as a way to reference your tags.

You can associate equipment with the following tag types:

- Advanced alarms
- Analog alarms
- Multi-digital alarms
- Digital alarms
- Time stamped alarms
- Time stamped analog alarms
- Time stamped digital alarms
- Variable tags
- Trend tags
- SPC tags
- Accumulators

### To specify an equipment association for a tag:

1. Locate the tag record that you would like to update.
2. In the **Equipment** field, enter the name of the equipment with which you would like to associate the tag. You can select the required equipment name from the drop-down menu. The list includes the current equipment definitions.
3. If required, specify a **Cluster** for the tag. This is only useful if the associated equipment is defined for every cluster and you want to specify a specific cluster for this particular tag.
4. Click **Save** to update the tag with your changes.

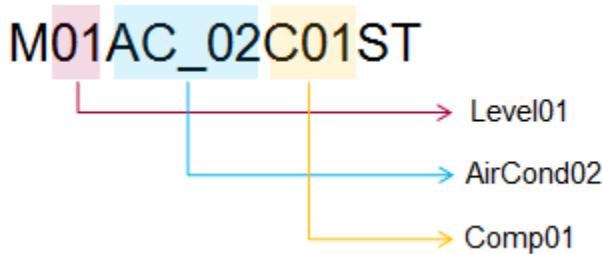
See [Define Equipment in Plant SCADA Studio](#).

If required, you can use Plant SCADA's Project DBF Add-In tool to define equipment associations for the tags in your project. This could be particularly useful if your project adheres to a tag naming convention that could be adopted as the basis of an equipment hierarchy (see [Use the Project DBF Add-In to Define Equipment Associations](#)).

## Example

You may have a project with tags that comply with a maintenance naming code. A typical tag name may be

"M01AC\_02C01ST". Using the naming code as a reference, it is possible to determine that this tag monitors the running status of a compressor on air conditioner 2 on level 1.



This demonstrates how an existing naming convention may be used as the framework for an equipment hierarchy. For example, you could associate the following equipment definition with the tag described above (and any others that apply to the same compressor):

Level01.AirCond02.Comp01

This would allow you to keep your site-based maintenance IDs while also providing a way to work with the data points using a logical, hierarchical structure.

## Update Equipment in Plant SCADA Studio

When you run an equipment update, Plant SCADA generates the tags associated with the equipment hierarchy and adds them to your project.

### To run an equipment update from the Plant SCADA Studio:

1. In the **System Model** activity, select **Equipment**.
2. Select **Update Equipment** from the Command Bar.
3. After the equipment update is complete, click **Show Log** to view the log messages.

**Note:** If you duplicate an equipment instance with a new name (without changing the tag prefixes) and then run the equipment update, Plant SCADA does not generate tags for the new equipment due to duplicated tag names. The log file provides a record of the tags not generated.

By default, Plant SCADA performs an incremental equipment update. It includes only the modified equipment and equipment type after the last update.

If you disable the incremental updates, Plant SCADA regenerates every tag associated with the equipment hierarchy each time an update runs.

### To disable incremental equipment updates:

1. On the Activity Bar, select **Options**.
2. The **Options** dialog box appears. Locate **Incremental equipment update** on the **Options** dialog box and clear its check box.

**Note:** If you clear the **Incremental equipment update** program option, an equipment update takes more time to complete. The time required to complete an update may increase significantly if the project includes a large equipment hierarchy.

## See Also

[Update Equipment in Equipment Editor](#)

### Use the Project DBF Add-In to Define Equipment Associations

You can use Plant SCADA's Project DBF Add-In to edit equipment associations directly in a project's database tables using Microsoft Excel™. This may allow you to rapidly deploy an equipment hierarchy for an existing project, particularly if your tags adhere to a naming convention that replicates a hierarchical architecture.

The Project DBF Add-In operates from within Excel. It connects you to the MASTER.DBF file for a selected project, and provides access to a list of included tables (see [Using Microsoft Excel to Edit .dbf Tables](#)).

#### To use the Project DBF Add-In to edit your equipment associations:

1. You will firstly need to [Define Equipment in Plant SCADA Studio](#) required to support your tags.  

---

**Note:** The **Type** and **Prefix** properties (or columns) are not required when configuring equipment for an existing project as they are only used when generating records in Equipment Editor.

---
2. You then need to open the table for a particular tag type to associate the **Equipment** property with one of your equipment definitions. The following tables include the Equipment column:
  - Advanced alarms
  - Analog alarms
  - Multi-digital alarms
  - Digital alarms
  - Time stamped alarms
  - Time stamped analog alarms
  - Time stamped digital alarms
  - Variable tags
  - Trend tags
  - SPC tags
  - Accumulators
3. Save your changes.

The compiler will check the existence of the equipment you have referenced and will generate an error message if required.

## See Also

[Define an Equipment Association for a Tag](#)

### Import Equipment Using XML Templates

Equipment tags can be created by importing them in much the same way as tags can be imported using TagGen. An XML template is used to specify the fields of input and output databases, and define filters and transformation rules that create tags from existing database fields.

Equipment XML templates are configuration files that are linked with equipment types that you create (see [Define Equipment Types in Plant SCADA Studio](#)). Each equipment type can have an associated template file specified.

The Equipment Update tool (accessible via the Command Bar when **Equipment** is selected in the **System Model** activity) processes the template for each piece of equipment to manage the associated configuration records. When updating the equipment configuration, the associated records are updated by modifying, adding or deleting as required. See [Update Equipment in Plant SCADA Studio](#).

For example, a set of associated records defined by the template may consist of a set of variable tags, alarms and trends. When an equipment record is added, the update tool will create the associated records. If the template is modified, the update tool will modify the associated records. If the equipment record is deleted, the update tool will delete the associated records.

## See Also

[Equipment XML Template](#)

[Calculator Examples](#)

## Equipment XML Template

The equipment template is an XML file that uses proprietary tags and attributes to specify the fields of input and output databases, and define filters and transformation rules that create tags from existing database fields. It uses the existing [TagGen XML Template](#) syntax rules.

The template file is linked to a specific equipment type. Equipment types can be supplied from different sources:

- The SCADA product
- An add-on pack
- External library
- User-defined.

Types defined by your project should use a project specific prefix to avoid conflicting with templates from another source. You should not use 'SE' or 'LIB' as prefixes for your template type names to avoid conflict with future product supplied templates.

It is recommended that the template file name should use the corresponding equipment type name for consistency. When adding the template to your project, the equipment type name should be the type name used by the template (see [Equipment Templates: Parameters](#)).

An equipment template XML file consists of five sections:

- [Equipment Templates: Header](#)
- [Equipment Templates: Parameters](#)
- [Equipment Templates: Input](#)
- [Equipment Templates: Outputs](#)
- [Equipment Templates: Footer](#)

Sample template files are located in the example project directory, and include Building Level.xml, Building Light Dimmer.xml, Building Light.xml, Building Room.xml, Plant Level.xml, Plant Light.xml, and Plant Line.xml.

## See Also

[Import Equipment Using XML Templates](#)

### Equipment Templates: Header

The template (XML) file should contain the header section that defines the:

- version
- start of the template node
- template description

The example below can be used without modification. However the header should reflect the actual encoding and the characters used. There are two ways to achieve this, either:

- change the encoding in the header e.g. to iso-8859-1 (latin-1) if using ANSI characters.  
or
- change any ANSI characters used to their Unicode equivalent.

## Example

```
<?xml version="1.0" encoding="utf-8"?>
<template desc="Equipment template">
```

## See Also

[Equipment XML Template](#)

[Import Equipment Using XML Templates](#)

### Equipment Templates: Parameters

The template (XML) file needs to contain a parameter section that defines the equipment type parameter. Other parameters can be added within this node as needed. Refer to the advanced section below or the 'TagGen' help topic in the product help for additional information.

The parameter string is used to define the template type name that is referenced in other parts of the template file. To use the example below, replace:

- [Type] with the name of your equipment type.

This should be unique across the templates used in your project(s). It needs to match the value specified in the name field of the record that defines the template type in the project.

- [TypeRef] with an abbreviation (if required) of the type name.

This is recommended to be less than 16 chars. When using an abbreviation, it should follow the same rules as the type name, such that it has to be unique across the templates used in your project(s).

## Example

```
<param name="type" desc="equipment type parameters">
<string name="name">[Type]</string>
<string name="ref">[TypeRef]</string>
</param>
```

## See Also

[Equipment XML Template](#)

[Import Equipment Using XML Templates](#)

### Equipment Templates: Input

The template (XML) file should contain an input section that defines the input node for the equipment database (DBF) file source. The input node is defined within the XML template node and contains the list of database fields that are to be available for use by the template. Only one input node can be defined within the template.

The example below can be used without modification and lists the complete set of the available data fields from the equipment database (DBF) file.

There are three built-in functions that can be used to manipulate strings in the XML template:

- SubString(string, start, count)
- Split(string, delimiter)
- ToProperty(string, separator, delimiter)

These functions are the same as used in the TagGen XML templates and are described in [Built-In Functions](#).

## Example

```
<!-- =====
Input Section - Equipment
=====

-->
<input name="equipment" file="equip.dbf" desc="Equipment Database">
<field name="name"></field>
<field name="cluster"></field>
<field name="type"></field>
<field name="iodevice"></field>
<array name="iodevice_list">{Split('{iodevice}',',')}</array>
<field name="tagprefix"></field>
<field name="area"></field>
<field name="page"></field>
<array name="page_list">{Split('{page}',',')}</array>
<field name="help"></field>
<field name="location"></field>
<field name="comment"></field>
<field name="scheduled"></field>
<field name="defstate"></field>
<field name="param"></field>
```

```
<array name="param_list">{ToProperty('{'param'}','=',';')}</array>
<field name="custom1"></field>
<field name="custom2"></field>
<field name="custom3"></field>
<field name="custom4"></field>
<field name="custom5"></field>
<field name="custom6"></field>
<field name="custom7"></field>
<field name="custom8"></field>
</input>
```

## See Also

[Equipment XML Template](#)

[Import Equipment Using XML Templates](#)

### Equipment Templates: Outputs

The template (XML) file can contain 1 or more output sections that define the configuration records that are to be created in other database (DBF) files. The output nodes are defined within the template node and contain the list of database fields that are to be updated in the new record. An [Equipment Templates: Output Node](#) should be added for each record that is to be created, such as a variable tag, alarm, trend, etc.

The values specified for each of the [Equipment Templates: Output Field](#) elements will be written to the database (DBF) field of the output file. Values from input fields and parameter strings can be referenced within this section to construct the resulting value for the output record field. Each output node should have a unique output name specified across the records in the template.

Refer to the following available output databases section for specific details on each of the available output databases and other advanced topics.

The example below shows a sample for an output to the variable database (DBF). For each output section, the field names specified should include the [Equipment Templates: Output Common Fields](#) and the fields specific to each [Output Database](#).

To use the example below, replace the following in your template:

- [FileRef] with a short string reference for the output destination database. Such as 'Var', 'Trend', 'AlmDig', 'Menu', etc. Examples are provided with the list of available databases
- [RefID] with a unique index (1, 2, 3, etc.) or a related string value for the entry (such as ON, STOP, etc.) for the specific output record.
- [TypeRef] with the abbreviation of the type name as defined in the parameter section. This may need to be an abbreviation of the full type name as the total length of [TypeRef].[RefID] should be 32 chars or less to fit the 'taggenlink' field.

## Example

```
<!-- =====
Output Section - <my variable tag>
=====
-->
<output name="[FileRef].[RefID]" file="variable.dbf"
```

```
filter="'{equipment.type}={type.name}'">
<field name="taggenlink" load="true">[TypeRef].[RefID]</field>
<field name="linked">1</field>
<field name="editcode">11</field>

<field name="name" key="true">{equipment.tagprefix}[Tag Suffix]</field>
<field name="type">[Tag Data Type]</field>
<field name="unit">{equipment.iodevice_list[0]}</field>
<field name="addr">[Tag Address]</field>
<field name="comment">[Tag Comment]</field>
</output>
```

## See Also

[Equipment XML Template](#)

[Import Equipment Using XML Templates](#)

# Equipment Templates: Output Node

The output node contains the following attributes:

- **Output Name**

The output node name attribute should be unique across the output nodes in the template. One way to achieve this consistently is to use the format of [FileRef]. [RefID] (as outlined in [Equipment Templates: Outputs](#)). The recommended value for [FileRef] is listed with each output database.

- **Output File**

The output node file attribute refers to the filename for the associated database (DBF) file.

- **Output Filter**

The output node filter attribute is used to match the equipment defined in the equipment database (DBF). The filter value needs to match the type name specified in the type field of the equipment. The recommended value for the filter is the parameter string for the type defined at the beginning of the template file.

Each output node defines the associated Output Common fields and Output Database fields for the record that is to be created.

## See Also

[Equipment Templates: Outputs](#)

# Equipment Templates: Output Field

The output node fields can reference the values from the associated equipment input record. The following is a list of the fields that can be used based on the input section of the template (XML) file. The value is referenced using the syntax '{equipment.<Field Name>}' where <Field Name> can be used from the list below:

Input Name	Description
Area	The required area
Cluster	The associated cluster
Comment	The configuration comment
custom1 .. 8	The custom runtime fields
Defstate	The default state for the scheduler
Help	The help reference
lodevice	The list of IO devices
iodevice_list[n]	The array of IO devices, where n in the zero based index
Location	The location reference
Name	The full equipment name
Page	The list of pages
page_list[n]	The array of pages, where n in the zero based index
Param	The list of parameters
param_list[name]	The list of parameters, where name is the specific parameter name
scheduled	The configuration of scheduler participation
Tagprefix	The prefix for the associated tags
Type	The equipment type

## See Also

[Equipment Templates: Outputs](#)

# Equipment Templates: Output Common Fields

There are three common fields that should be used with each output node:

- **Output Tag Gen Link**

The taggenlink field specifies a name that is used to identify the record in the database so that it is linked with the template. This allows the record to be found for an update or delete operation.

The value should include a unique reference to the template type and a reference to the output record within the template. The recommended way to achieve this consistently is to use the format of [TypeRef].[RefID] (as outlined in Template Section: Outputs).

- **Output Linked**

The linked field should be set to '1'.

- **Output Edit Code**

The edit code is a number that represents the bitmask of the database (DBF) fields based on the physical order in the DBF file. When the bit in the mask is set, the corresponding field will be marked read-only in the Project activity. If no fields are to be marked as read only, this field can be left blank or removed from the output definition.

The algorithm to determine an edit code is listed in [Equipment Templates: Output Database](#). The combined edit code for few fields is a sum of edit codes of all fields. For example, the edit code for first (edit code = 1), second (edit code = 2), fourth (edit code = 8) and 20th fields (edit code = 524288) is  $1 + 2 + 8 + 524288 = 524299$ , or in decimals: 0x000001 (edit code for first field) + 0x000002 (edit code for second field) + 0x000008 (edit code for third field) + 0x80000 (edit code for 20th field) = 0x8000B

## See Also

[Equipment Templates: Outputs](#)

# Equipment Templates: Output Database

Below is a list of available output databases and the corresponding output files and output file references to use in your equipment template.

Database	Output File	Output File Reference
Accumulators	file="accums.dbf"	Accum
Advanced Alarms	file="advalm.dbf"	AdAlm
Alarm Categories	file="category.dbf"	ACat
Analog Alarms	file="anaalm.dbf"	AAlm
Devices	file="devices.dbf"	Dev
Digital Alarms	file="digalm.dbf"	DAlm
Equipment	file="equip.dbf"	Equip
Equipment States	file="eqstate.dbf"	State
Events	file="events.dbf"	Event
Fonts	file="fonts.dbf"	Font

Database	Output File	Output File Reference
Groups	file="groups.dbf"	Group
Labels	file="labels.dbf"	Label
Local Variables	file="locvar.dbf"	LVar
Menu Configuration	file="pagemenu.dbf"	Menu
Multi-digital Alarms	file="argdig.dbf"	MDAIm
Parameters	file="param.dbf"	Param
ReMapping	file="remap.dbf"	Remap
Reports	file="reports.dbf"	Rep
SPC Tags	file="spc.dbf"	SPC
Time Stamped Alarms	file="hresalm.dbf"	TSAlm
Time Stamped Analog Alarms	file="tsana.dbf"	TAAlm
Time Stamped Digital Alarms	file="tsdig.dbf"	TDAIm
Trend Tags	file="trend.dbf"	Trend
Variable Tags	file="variable.dbf"	Var

## To determine acceptable field names for Tags and Alarms

To determine the acceptable field names and, respective edit codes, either open citect.frm and find the respective output file without an extension (for example, advalm for advanced alarms) or open the respective dbf in the DBF Add-In. Every field listed in the dbf are acceptable field names. The edit code of the first field in the table is initialized to 1. The edit code of the fields other than 1 equals the edit code of the previous field times 2 (i.e. the edit code of the second field is 2, third is 4, fourth is 8 and so on).

## See Also

[Equipment Templates: Outputs](#)

## Equipment Templates: Footer

The template (XML) file should contain a footer section to help so that the XML is formatted correctly. The footer contains the end marker for the template node.

The example below can be used without modification.

## Example

```
</template>
```

## See Also

[Equipment XML Template](#)

## Calculator Examples

The following examples demonstrate five cases where the calculator function could be applied to an equipment XML template. In each case, you need to add an additional line to the XML template:

```
<calculator>{parameter} + 1</calculator>
```

1. Calculator based on string parameter in the input section of the XML. This is only recommended if the calculated parameter is constant for each instance of equipment for this type; however, this is not a typical scenario.

```
<string name="MyString">1000</string>
<calculator name="MyCalc">{MyString} + 1</calculator>
```

See [Example 1](#).

2. Calculator based on equipment field, for example, {equipment.Custom1}.

```
<calculator name="MyCalc">{equipment.Custom1} + 1</calculator>
```

See [Example 2](#).

3. Calculator based on ungrouped parameter (defined in Equipment Editor).

```
<calculator name="MyCalc">{equipment.param_list[Param1]} + 1</calculator>
```

See [Example 3](#).

4. Calculator based on grouped parameter (defined in Equipment Editor).

```
<calculator name="MyCalc">{equipment.ParamGroup[Param2]} + 1</calculator>
```

These can then be used in fields for an element:

```
<field name="addr">MW_{MyCalc}_2</field>
```

See [Example 3](#).

5. Calculator based on grouped parameter (in the output section).

```
<output name="Element" file="variable.dbf" filter="'{equipment.type}={type.name}'>
<calculator name="TagAddress">{equipment.TagAddressing[StructOffset]} + 0</calculator>
...
</output>
```

These can then be used in fields for an element:

```
<field name="addr">MW_{MyCalc}_2</field>
```

See [Example 4](#).

## Example 1 - Type with Calculator based on STRING

```
<?xml version="1.0" encoding="utf-8"?>
<template desc="">
    <param name="type">
        <string name="name">TypeWithCalc</string>
        <string name="ref">TypeWithCalc</string>
        <string name="blank"></string>
        <string name="wildcard"></string>
        <string name="parameter-definitions">param_list.param1=;Addr.BaseAddr=</string>
    </param>
    <input name="equipment" file="equip.dbf" desc="Equipment Database">
        <field name="PARAM" />
        <field name="TAGPREFIX" />
        <field name="IODEVICE" />
        <field name="CLUSTER" />
        <field name="NAME" />
        <field name="AREA" />
        <field name="LOCATION" />
        <field name="COMMENT" />
        <field name="CUSTOM1" />
        <field name="CUSTOM2" />
        <field name="CUSTOM3" />
        <field name="CUSTOM4" />
        <field name="CUSTOM5" />
        <field name="CUSTOM6" />
        <field name="CUSTOM7" />
        <field name="CUSTOM8" />
        <field name="PAGE" />
        <field name="HELP" />
        <field name="DEFSTATE" />
        <field name="SCHEDULED" />
        <field name="COMPOSITE" />
        <field name="TYPE" />
        <array name="param_list">{ToProperty('{param}', '=', ';')}</array>
        <array name="Addr">{ToProperty('{equipment.param_list[Addr]}', ':', ',')}</array>
        <string name="MyAddrString">250</string>
        <calculator name="myaddrcalc">{MyAddrString}+5</calculator>
    </input>
    <output name="Element" file="variable.dbf" filter="{{equipment.type}={type.name}}">
        <field name="NAME" key="true">{equipment.TAGPREFIX}_Tag1</field>
        <field name="TYPE">INT</field>
        <field name="UNIT">{equipment.IODEVICE}</field>
        <field name="ADDR">{myaddrcalc}</field>
        <field name="ENG_ZERO">0</field>
        <field name="ENG_FULL">1000</field>
        <field name="CLUSTER" key="true">{equipment.CLUSTER}</field>
        <field name="EQUIP">{equipment.name}</field>
        <field name="ITEM">Tag1</field>
        <field name="TAGGENLINK" load="true">TypeWithCalc_0006</field>
        <field name="LINKED">1</field>
        <field name="EDITCODE">3938511</field>
    </output>
</template>
```

## Example 2 - Type with Calculator based on Equipment Field

```
<?xml version="1.0" encoding="utf-8"?>
<template desc="">
    <param name="type">
        <string name="name">TypeWithCalc</string>
        <string name="ref">TypeWithCalc</string>
        <string name="blank"></string>
        <string name="wildcard"></string>
        <string name="parameter-definitions">param_list.param1=;Addr.BaseAddr=</string>
    </param>
    <input name="equipment" file="equip.dbf" desc="Equipment Database">
        <field name="PARAM" />
        <field name="TAGPREFIX" />
        <field name="IODEVICE" />
        <field name="CLUSTER" />
        <field name="NAME" />
        <field name="AREA" />
        <field name="LOCATION" />
        <field name="COMMENT" />
        <field name="CUSTOM1" />
        <field name="CUSTOM2" />
        <field name="CUSTOM3" />
        <field name="CUSTOM4" />
        <field name="CUSTOM5" />
        <field name="CUSTOM6" />
        <field name="CUSTOM7" />
        <field name="CUSTOM8" />
        <field name="PAGE" />
        <field name="HELP" />
        <field name="DEFSTATE" />
        <field name="SCHEDULED" />
        <field name="COMPOSITE" />
        <field name="TYPE" />
        <array name="param_list">{ToProperty('{param}', '=', ';')}</array>
        <array name="Addr">{ToProperty('{equipment.param_list[Addr]}', ':', ',')}</array>
        <string name="MyAddrString">250</string>
        <calculator name="myaddrcalc">{equipment.CUSTOM1}+5</calculator>
    </input>
    <output name="Element" file="variable.dbf" filter="'{equipment.type}={type.name}'">
        <field name="NAME" key="true">{equipment.TAGPREFIX}_Tag1</field>
        <field name="TYPE">INT</field>
        <field name="UNIT">{equipment.IODEVICE}</field>
        <field name="ADDR">{myaddrcalc}</field>
        <field name="ENG_ZERO">0</field>
        <field name="ENG_FULL">1000</field>
        <field name="CLUSTER" key="true">{equipment.CLUSTER}</field>
        <field name="EQUIP">{equipment.name}</field>
        <field name="ITEM">Tag1</field>
        <field name="TAGGENLINK" load="true">TypeWithCalc_0006</field>
        <field name="LINKED">1</field>
        <field name="EDITCODE">3938511</field>
    </output>
</template>
```

### Example 3 - Type with Calculator based on Ungrouped and Grouped Parameter

```
<?xml version="1.0" encoding="utf-8"?>
<template desc="">
    <param name="type">
        <string name="name">TypeWithCalc</string>
        <string name="ref">TypeWithCalc</string>
        <string name="blank"></string>
        <string name="wildcard"></string>
        <string name="parameter-definitions">param_list.param1=;Addr.BaseAddr=</string>
    </param>
    <input name="equipment" file="equip.dbf" desc="Equipment Database">
        <field name="PARAM" />
        <field name="TAGPREFIX" />
        <field name="IODEVICE" />
        <field name="CLUSTER" />
        <field name="NAME" />
        <field name="AREA" />
        <field name="LOCATION" />
        <field name="COMMENT" />
        <field name="CUSTOM1" />
        <field name="CUSTOM2" />
        <field name="CUSTOM3" />
        <field name="CUSTOM4" />
        <field name="CUSTOM5" />
        <field name="CUSTOM6" />
        <field name="CUSTOM7" />
        <field name="CUSTOM8" />
        <field name="PAGE" />
        <field name="HELP" />
        <field name="DEFSTATE" />
        <field name="SCHEDULED" />
        <field name="COMPOSITE" />
        <field name="TYPE" />
        <array name="param_list">{ToProperty('{param}', '=', ';')}</array>
        <array name="Addr">{ToProperty('{equipment.param_list[Addr]}', ':', ',')}</array>
        <calculator name="myaddrcalc">{equipment.param_list[param1]}+5</calculator>
        <calculator name="myaddrcalc2">{equipment.Addr[BaseAddr]}+10</calculator>
    </input>
    <output name="Element" file="variable.dbf" filter="{{equipment.type}={type.name}}">
        <field name="NAME" key="true">{equipment.TAGPREFIX}_Tag1</field>
        <field name="TYPE">INT</field>
        <field name="UNIT">{equipment.IODEVICE}</field>
        <field name="ADDR">MW{myaddrcalc}_2</field>
        <field name="ENG_ZERO">0</field>
        <field name="ENG_FULL">1000</field>
        <field name="CLUSTER" key="true">{equipment.CLUSTER}</field>
        <field name="EQUIP">{equipment.name}</field>
        <field name="ITEM">Tag1</field>
        <field name="TAGGENLINK" load="true">TypeWithCalc_0006</field>
        <field name="LINKED">1</field>
        <field name="EDITCODE">3938511</field>
    </output>
</template>
```

## Example 4 - Type with Calculator in Output Section

```
<?xml version="1.0" encoding="utf-8"?>
<template desc="">
    <param name="type">
        <string name="name">TypeWithCalculator</string>
        <string name="ref">TypeWithCalculator</string>
        <string name="blank"></string>
        <string name="wildcard"></string>
        <string name="parameter-
definitions">TagAddressing.PLC.DBNumber=;TagAddressing.StructOffset=</string>
    </param>
    <input name="equipment" file="equip.dbf" desc="Equipment Database">
        <field name="PARAM" />
        <field name="TAGPREFIX" />
        <field name="NAME" />
        <field name="IODEVICE" />
        <field name="CLUSTER" />
        <field name="AREA" />
        <field name="LOCATION" />
        <field name="COMMENT" />
        <field name="CUSTOM1" />
        <field name="CUSTOM2" />
        <field name="CUSTOM3" />
        <field name="CUSTOM4" />
        <field name="CUSTOM5" />
        <field name="CUSTOM6" />
        <field name="CUSTOM7" />
        <field name="CUSTOM8" />
        <field name="PAGE" />
        <field name="HELP" />
        <field name="DEFSTATE" />
        <field name="SCHEDULED" />
        <field name="COMPOSITE" />
        <field name="TYPE" />
    <array name="param_list">{ToProperty('{param}', '=', ';')}</array>
    <array
        name="TagAddressing">{ToProperty('{equipment.param_list[TagAddressing]}',':',',',')}</array>
    </input>
    <output name="Element" file="variable.dbf" filter="{equipment.type}={type.name}">
        <calculator name="TagAddress">{equipment.TagAddressing[StructOffset]} +
        0</calculator>
        <field name="NAME" key="true">{equipment.tagprefix}_Enable</field>
        <field name="EQUIP">{equipment.name}</field>
        <field name="UNIT">{equipment.iodevice}</field>
        <field name="CLUSTER" key="true">{equipment.cluster}</field>
        <field name="TYPE">DIGITAL</field>
        <field name="COMMENT">Machine Enable</field>
        <field name="ADDR">DB{equipment.TagAddressing[PLC.DBNumber]},{TagAddress}.0</field>
        <field name="ITEM">Enable</field> <field name="TAGGENLINK" load="true">12345</field>
        <field name="LINKED">1</field> <field name="EDITCODE">3939343</field> </output>
    </template>
```

## See Also

[Equipment XML Template](#)

[Import Equipment Using XML Templates](#)

## Troubleshooting - Equipment XML Templates

- **After running an equipment update there are no records created in the output database files.**

Check that the type name defined in the [Equipment Templates: Parameters](#) of the template file is the same as the name defined for the [Define Equipment Types in Plant SCADA Studio](#) record of the project.

- **After running an equipment update there are duplicate records in the output database files.**

Check that the 'taggenlink' field in the [Equipment Templates: Outputs](#) of the template was not changed. When this field is changed, any previously generated entries need to be manually deleted.

- **After running an equipment update some tags were not created. Is there a log file I can check?**

You can check the log file called EquipGen.log; however you should set the ini parameter [\[CtEdit\]LogLevel](#) to turn it on.

[\[CtEdit\]LogLevel](#) increases the verbosity of the log file so that more information can be recorded. The following values are accepted:

Error=3

Summary=5

Details=10

Verbose=15 (set this to 15 when more detailed logging relating to EquipGen is required.)

The log file is located in the default log directory which is set using [\[CtEdit\]Logs](#). To access the log file, in Plant SCADA Studio select **Log Folder** from the [The System Menu](#).

---

**Note:** If you leave LogLevel on during runtime, it can affect system performance as other logging functions may be impacted.

- **What happens if the Equipment type XML file is corrupted?**

When the equipment type is saved in the Equipment Editor, a file called EquipType.xml.backup is created. This file is of the previous configuration before the last save. If the XML file is corrupted, it can be replaced with the xml.backup file.

## See Also

[Equipment XML Template](#)

[Import Equipment Using XML Templates](#)

## Migrate Custom Parameters to the Configuration Parameters Database

Prior to the release of Plant SCADA 2023, "custom parameters" were used to apply specific values to an equipment type. However, there was limitation of 256 characters on the database field that was used to store these parameters.

In Plant SCADA 2023, custom parameters were replaced with "configuration parameters". These parameters support the same functionality, but they are stored in a dedicated configuration parameter database that

removes the 256 character limitation.

Equipment types that support the configuration parameters database can be identified by a "2.0" version tag in the source XML.

```
<template version="2.0" desc="Equipment template">
```

### To migrate your equipment types (and any associated instances) to version 2.0:

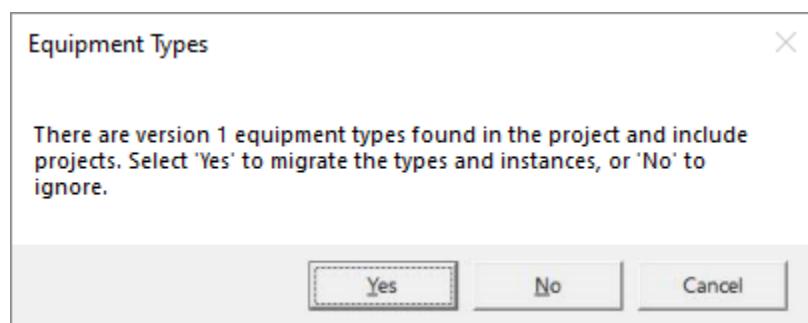
1. From the **Projects** view in Plant SCADA Studio, select **Migration Tool** on the command bar.  
The **Project Migration Tool** will display with the active project selected.
2. In the list of migration options, confirm that **Migrate equipment database, types and parameters** is selected.  
If required, you can clear the check boxes for any other migration options.
3. Click **Migrate**.

When the migration process is complete, an information dialog will appear listing the changes that have been made. This will include a tally for "Equipment types and instances upgraded".

4. Click **OK**.

The configuration parameters that were generated by the migration process will now be available to view in the **Equipment | Configuration Parameters** view in the **System Model** activity.

You can also initiate this process from **Equipment Editor**. If Plant SCADA detects a version 1 equipment type when you open the Equipment Editor, the following dialog will appear:



To migrate the version 1 equipment types, click **Yes** to open the Project Migration dialog and follow the steps above. To keep the existing version 1 templates, select **No**.

---

**Note:** If you are using equipment templates that were not created by Equipment Editor, you need to check the validity of these templates following the migration process. Before you launch runtime, perform an equipment update and review the configuration data associated with these templates (for example, the alarms and variable tags) to confirm that everything has migrated correctly.

## See Also

[Add Configuration Parameters to an Equipment Type](#)

[Define Equipment Configuration Parameters in Plant SCADA Studio](#)

## Variable Tags

A variable tag is a label that you can use to reference the current data value for an I/O device register at a

specified address.

This has several benefits:

- The address in the I/O device is defined only once. If you change the address, you only need to update the variable tag definition.
- You can use a tag name that is logical and descriptive.
- You can scale the raw data to an appropriate range in the same declaration.

Variable tags are a fundamental component of much of the functionality supported by a Plant SCADA system. For example, you can use variable tags to:

- Display production data on a graphics page (see [Display Tags on a Page](#)).
- Animate objects on a graphics page in response to tag value changes (see [Configure Graphics Objects](#)).
- Define commands that control equipment and processes (see [Interact with Graphics Pages at Runtime](#)).
- Create alarms that monitor variables tags for specific value changes (see [Alarms](#)).
- Store tag data for trending and analysis (see [Trends](#)).

When you define a variable tag, you need to give it a [Tag Names](#) and specify its [Tag Data Types](#). The most common variables supported by I/O devices are digital and integer. However, Plant SCADA also supports REAL, STRING, BYTE, BCD, LONG and LONGBCD data types.

Variable tags also support a set of [Tag Elements](#) that provide different views of a tag's data. This includes extended data values, quality validation and timestamp information. You can read and write to tag elements using [Tag Extensions](#).

You can use the following types of variable tags:

1. [Arrays](#) allow you to use a single variable tag to represent a set of variables are stored in consecutive memory registers on an I/O device.
2. [Local Variables](#) allow you to store data in memory when you start your runtime system.
3. [Calculated Variables](#) allow you to generate a tag value at runtime that is the result of a Cicode expression.

---

**Note:** You can use Cicode to create a tag browsing session at runtime. See [Configure Tag Browsing](#).

---

## See Also

[Configure Variable Tags](#)

[Refer to Variable Tags](#)

[Link Tags to an External Data Source](#)

## Tag Names

Plant SCADA places some restrictions on the names you can use for variable tags:

- Tags names need to adhere to specific syntax (see [Tag Name Syntax](#)).
- There is a set of reserved words that you need to avoid when naming a tag (see [Reserved Words](#)).
- Do not give a tag the same name as a Cicode function that is used within the project or its included projects.

A compile error message will occur if a tag has the same name as a Cicode function and is placed on a graphic page.

You can reference variable tag names (as a string) in your project, using the following formats:

- <tag name>
- <cluster>.<tag name> (used in multi-cluster systems).

---

**Note:** If your Plant SCADA project incorporates an equipment hierarchy, you may need to include a location in the hierarchy when you refer to a tag. This is reflected in the tag reference by using a period (.) to indicate levels in the equipment hierarchy (for example, "Plant.Line1.Pump.Tag1"). For more information, see [Equipment Hierarchy](#).

---

## See Also

[Structured Tag Names](#)

## Tag Name Syntax

Plant SCADA tags (variable tags, alarm tags and trend tags) need to use the following syntax:

- The fields are not case sensitive and need to begin with either an alpha character (A-Z or a-z) or the underscore character (\_).
- Any following characters need to be either alpha characters (A-Z or a-z), digit characters (0 - 9), backslash characters (\), or underscore characters (\_).

The use of any other characters will result in a compile error message.

For example:

- "\_MyTag123" and "my\New\Tag" are both valid tag names.
- "\NewTag\" is invalid.

Tag names that begin with a numeric character, such as '12TagName', are only valid if the INI parameter **[General]TagStartDigit** is set to 1 (the default value is zero).

---

**Note:** In the case of alarms, the **Alarm Tag** property needs to follow this syntax, but the **Alarm Name** property does not.

---

## See Also

[Structured Tag Names](#)

[Reserved Words](#)

## Structured Tag Names

The following naming convention is recommended for a system, to obtain maximum benefit when using features such as Genies and Super Genies. (If you are already using a naming system that differs from the following convention, you can still use Genies and Super Genies supplied with Plant SCADA by modifying the Genies that you want to use.)

Each tag name can contain up to 79 characters. To establish a convention, you need to divide the characters in the tag name into sections that describe characteristics of the tag, for example, the area where the tag is located, the type of variable, and any specific attributes.

Four basic sections are suggested for a Plant SCADA naming convention:

*Area\_Type\_Occurrence\_Attribute*

## See Also

[Reserved Words](#)

## Reserved Words

You cannot use certain reserved words when configuring a Plant SCADA project. You need to check the following lists before you create a name for any element:

### Tag Names

You cannot use Cicode function names and the following reserved words when defining a tag name for a variable tag:

&	*	**	-
/	+	<	<=
>=	<>	=	>
ACTION	AND	ANY_NUM	AREA
ARRAY	BOOL	CASE	CONFIGURATION
CONSTANT	DATE	DINT	DO
DUT	DWORD	ELSE	EN
END_CASE	END_CONFIGURATION	END_FOR	END_IF
END_PROGRAM	END_VAR	END WHILE	END
EXIT	FALSE	FOR	FUNCTION
IF	INT	ME	MOD
NOT	OF	OR	PROGRAM
REAL	REPEAT	STRING	TASK
TIME	TO	TRUE	UNTIL
VAR_EXTERNAL	VAR_GLOBAL	VAR_IN_OUT	VAR_INPUT
VAR_OUTPUT	WHILE	WORD	XOR

**Note:** When defining an alarm tag name, following are some additional reserved words that you need to avoid:

- ServerStatus
- SummaryRecordId
- RDBTimeInDB
- CitectArchiveObject

If you use any of these reserved words for alarm tag, an error message will display when you compile your project.

### Cluster Names

You cannot use the following reserved words when defining a cluster name:

&	*	**	-
/	+	<	<=
>=	<>	=	>
ACTION	AND	ANY_NUM	AREA
ARRAY	BOOL	CASE	CONFIGURATION
CONSTANT	DATE	DINT	DO
DUT	DWORD	ELSE	EN
END_CASE	END_CONFIGURATION	END_FOR	END_IF
END_PROGRAM	END_VAR	END WHILE	END
EXIT	FALSE	FOR	FUNCTION
IF	INT	ME	MOD
NOT	OF	OR	PROGRAM
REAL	REPEAT	STRING	TASK
TIME	TO	TRUE	UNTIL
VAR_EXTERNAL	VAR_GLOBAL	VAR_IN_OUT	VAR_INPUT
VAR_OUTPUT	WHILE	WORD	XOR

### Equipment Names

You cannot use the following reserved words when defining an equipment name:

&	*	**	-
/	+	<	<=
>=	<>	=	>
ACTION	AND	ANY_NUM	AREA

ARRAY	BOOL	CASE	CONFIGURATION
CONSTANT	DATE	DINT	DO
DUT	DWORD	ELSE	EN
END_CASE	END_CONFIGURATION	END_FOR	END_IF
END_PROGRAM	END_VAR	END WHILE	END
EXIT	FALSE	FOR	FUNCTION
IF	INT	ME	MOD
NOT	OF	OR	PROGRAM
REAL	REPEAT	STRING	TASK
TIME	TO	TRUE	UNTIL
VAR_EXTERNAL	VAR_GLOBAL	VAR_IN_OUT	VAR_INPUT
VAR_OUTPUT	WHILE	WORD	XOR

**Note:** When defining an Item Name, there are a number of reserved words that you need to avoid. These reserved words relate to syntax that is used for [Tag Extensions](#). They are:

- field
- valid
- override
- overridemode
- controlmode
- status
- v
- vt
- q
- qt
- t

If you use one of these reserved words, an error message will display when you compile your project.

**Note:** Using a reserved word for a name may generate a compile error message.

## Tag Data Types

Tags in Plant SCADA hold data values that can be defined as one of the following data types.

Data Type	Variable	Size	Allowed Values
BCD	Binary- Coded Decimal	2 bytes	0 to 9,999
BYTE	Byte	1 byte	0 to 255

Data Type	Variable	Size	Allowed Values
DIGITAL	Digital	1 byte	0 or 1
INT	Integer	2 bytes	-32,768 to 32,767
UINT	Unsigned Integer	2 bytes	0 to 65,535
LONG	Long Integer	4 bytes	-2,147,483,648 to 2,147,483,647
ULONG	Unsigned Long Integer (Only for display on a screen. Arithmetic operations are not supported.)	4 bytes	0 to 4,294,967,295
LONGBCD	Long Binary-Coded Decimal	4 bytes	0 to 99,999,999
REAL	Floating Point	4 bytes	-3.4E38 to 3.4E38
STRING	String	256 bytes (maximum)	ASCII (null terminated) <b>Note:</b> Non printable characters within string tag values will be substituted with spaces. For example, string [ 0x01, 0x41, 0x10, 0x42 ] will appear as "A B" so cache loading continues to operate.

While numeric variables are more common, some I/O devices also support ASCII strings. You can use strings to store text data (for example, from a bar code reader). All strings need to be NULL-terminated in the I/O device, as Plant SCADA uses the NULL character to check for the end of a string. If no NULL character is present, Plant SCADA reads (and displays) any extra characters in memory, after the end of the string.

When you are using a disk I/O device, you can also specify a string data type for storage of recipes, or for operator display information.

Not every I/O device supports strings. However, if your I/O device does support integer data types, Plant SCADA can use these integer registers to store ASCII strings in your I/O device. Plant SCADA strings can only be stored in contiguous blocks (consecutive registers), and are stored as an array.

A Cicode variable of INT data type can be used to store Tag data types: BCD, BYTE, DIGITAL, INT, UINT, LONG, ULONG and LONGBCD.

A Cicode variable of the QUALITY or TIMESTAMP data type can be used to store the Tag quality and timestamp items.

Tag values can be used with the Cicode variable data types (see [Declaring the Variable Data Type](#)).

## Tag Elements

A variable tag is a representation of data elements. Each element provides access to a view of the data value for the tag.

Each variable tag can be used on its own or by referencing a particular element to access the following information:

Element	Description
.field	The field element, which represents the latest field data received from the device.
.valid	The valid element, which represents the last field data which had 'Good' quality.
.override	The override element, which represents the overridden tag value.
.overridemode	The override mode, which is used to set the override behavior of the tag.
.controlmode	The control mode, which is used to set the control inhibit mode of the tag.
.status	The tag status element, which is used to represent the current status of the tag.

The tag and each element have items that can be referenced to access the following information:

Item	Description
v	The value item. Use this to retrieve the current data value of the tag or element.
vt	The value timestamp item. Use this to retrieve a timestamp that represents when the value last changed.
q	The quality item. Use this to retrieve the quality of the value.  The quality item differs to other elements as it offers several layers of information. For more information, see <a href="#">The Quality Item</a> .
qt	The quality timestamp item. Use this to retrieve a timestamp that represents when the quality last changed.
t	The timestamp item. This will access the timestamp of when the tag or element was last updated.

To refer to tag elements in your project, you use [Tag Extensions](#).

## The Quality Item

The majority of data which is contained within the variable tag is represented as an element which includes the items "Value", "ValueTimestamp", "Quality", "QualityTimestamp" and "Timestamp". With the exception of "Quality" these items are absolutes of a single value. However the "Quality" item has several layers of information within it. These layers of information are covered by the following categories:

The primary layer identifies the **General Quality Values**.

Cicode Label Name	Value	Description
QUAL_BAD	0x00	Value is not useful for reasons indicated by the Substatus Bit Field.
QUAL_UNCR	0x01	The Quality of the value is uncertain for reasons indicated by the Substatus Bit Field.
QUAL_GOOD	0x03	The Quality of the value is Good.

These can be accessed from a text object on a graphics page at runtime, or with the Cicode function [QualityGetPart](#) using the necessary value for the *Part* parameter.

Within each of these quality values is a substatus layer, and a third extended substatus layer. These layers of information can also be accessed using the [QualityGetPart](#) Cicode function.

## Tag Extensions

Tag extensions allow you to refer to the [Tag Elements](#) of a tag. They support the following functionality:

- Provide access to the quality and time stamp information associated with a tag value.
- Provide extended data to Plant SCADA client components, such as a display client, trend server, alarm server, report server, Cicode and CtAPI.
- Allow you to apply an override to a tag, and prohibit writing to the tag value.
- Allow you to display and trend tag values even if their quality is not "good".
- Enable tag value persistence, and replication of tag data.

The tag reference syntax used for tag extensions is as follows:

**[Cluster.]Tag[.Element][.Item][ [n]]**

Where:

Cluster	The optional <a href="#">Clusters</a> name.
Tag	The tag name or Super Genie <a href="#">Super Genies</a> .
Element	The optional element name. If the element name is not specified, the requested element will be

	determined at runtime.
Item	The optional item name. If the item name is not specified, the whole element is referenced.
n	The optional array index if the tag is defined as an <a href="#">Arrays</a> .

The array index is at the end of the reference (MyArray.v[n], MyArray.Field[n], MyArray.Field.v[n]). There is only a single quality and timestamp for each array, each member will return the same quality and timestamp.

**Note:** Consider the impact on network traffic when configuring tag extensions, as the distribution of quality and value timestamps increases the amount of data being sent between servers.

You can use tag extensions to access tag data in the following ways:

- Reference tag data using only the tag name, for example:

"MyTag" (unqualified tag reference).

This will provide default access to the **.field** element information, unless the tag is in one of the override modes.

- Reference tag data using the tag name and the item name, for example:

"MyTag.q" (unqualified tag reference)

This will provide access to the item information for the tag, either default from **.field** or **.override** element.

- Reference the tag data by using the tag name and the element name, for example:

"MyTag.field" (qualified tag reference)

This reference will provide access to the specified tag element information.

- Reference the tag data by using the tag name, element name and the item name, for example:

"MyTag.field.vt" (qualified tag reference)

This reference will provide access to the specific tag element item.

You can also access alarm data in a similar way. See [Using Alarm Properties as Tags](#).

**Note:** Consider the impact on network traffic when configuring tag extensions, as the distribution of quality and value timestamps increases the amount of data being sent between servers.

## Controlling Tag Extension Behavior

By default, the tag data referenced without an element will provide access to the data value when the value is of quality is good and an error (#BAD, #COM, etc) when the quality is bad. The configuration parameter [\[Page\]IgnoreValueQuality](#) can be used to change this behavior, including automatically changing the background color of text and number graphics objects on a page with changes in quality of the tag.

Tag extension behavior is controlled by several citect.ini file settings which are described in the Parameters Reference. For each of these settings there is a corresponding setting in the project database parameters (param.dbf).

A citect.ini file setting specifies behavior for a particular machine and a parameter database setting is applied system-wide.

**Note:** By default, the TagSubscribe Cicode function is set to retrieve "lightweight" tag values that exclude quality

---

and value timestamps. If you need a subscription to retrieve timestamp data, you need to set the "bLightweight" argument to 0 (false).

---

#### See Also

[Refer to Tag Extensions](#)

## Arrays

An array is a collection of variables (all of the same data type) that are stored in consecutive memory registers in an I/O device.

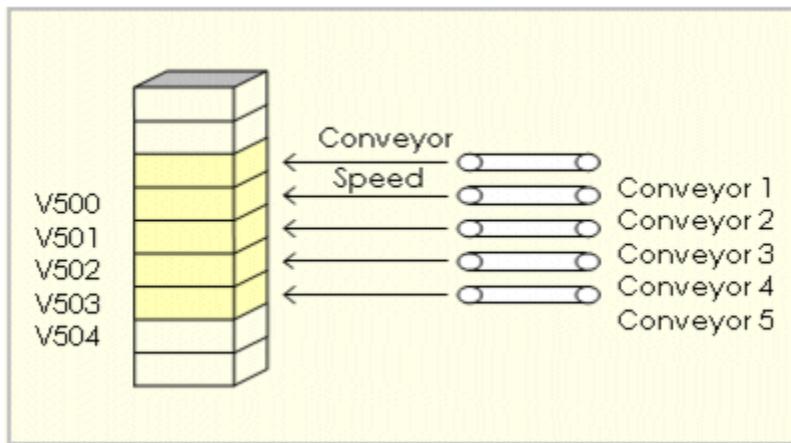
Plant SCADA supports two types of arrays:

- Numeric arrays
- String arrays.

## Numeric Arrays

Numeric arrays are useful when you have separate devices (or processes) performing similar functions. You can program the I/O device to store, in consecutive memory registers, variables that relate to each of these processes.

In the following example, five consecutive variables (V500, V501, V502, V503, and V504) store the conveyor speed of five conveyors (1 to 5).



Instead of configuring five separate variable tags (one for each conveyor), you can configure a single tag as an array.

To specify a single variable tag for an array, define the first address and add the size of the array (the number of consecutive addresses) to the register address.

Variable Tag Name	Conveyor_Speed
Address	V500[5]

Here, five register addresses are referred to by the variable tag Conveyor\_Speed.

## String Arrays

If you are using a Plant SCADA string data type, you need to specify an array of integer data types. One INT register stores two string characters.

To calculate the size of an array (for string data types), use the following formula:

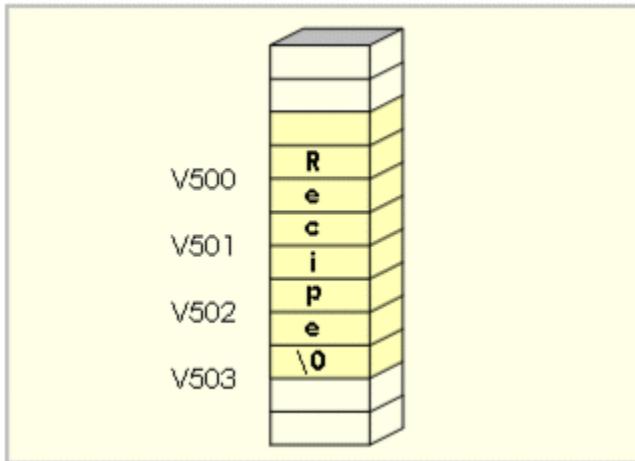
$$\text{Size of Array} = \frac{(\text{number of characters} + 1)}{2}$$

The last element of the array is always used to store the null character '\0'. This character marks the end of the string.

To store the word "Recipe", you need to specify an array with 4 elements, for example:

Variable Tag Name	Recipe_Tag
Address	V500[4]

Two characters are stored in each register, that is:



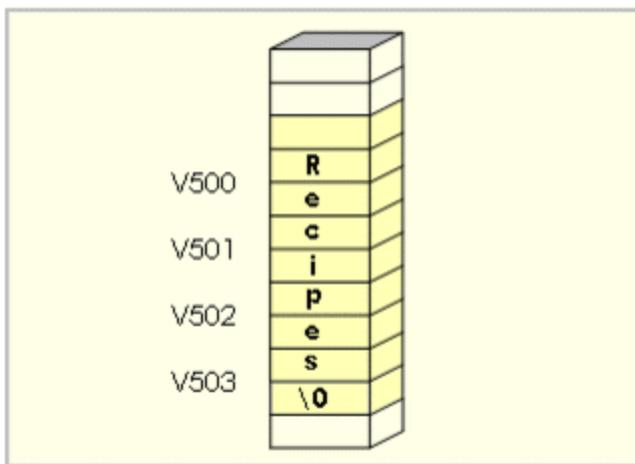
You can then refer to the entire string by specifying the tag:

<Tag Name>

For example:

Variable Tag	Recipe_Tag
--------------	------------

To store the word "Recipes", you would also specify an array with 4 elements. The characters are stored as follows:



**Note:** If your I/O device supports string data types, you need to specify the address in the format determined by the I/O Device you are using.

## See Also

[Refer to Array Elements](#)

## Local Variables

Local variables allow you to store data in memory when you start your runtime system. They are created each time the system starts, and therefore do not retain their values when you shut down.

Every Plant SCADA process has its own copy of the local variables configured in a project. This is useful when you need each process to have a separate copy of the associated data. The values in a local variable are only available to the process in which they were written.

Local variables may be of any data type supported by Plant SCADA, including two-dimensional arrays.

## See Also

[Add a Local Variable](#)

## Calculated Variables

A calculated variable allows you to do the following:

- Generate a tag value at runtime that is the result of a Cicode expression.
- Call an internal Cicode function as a variable tag.

Calculated variables are evaluated on the I/O server only when a subscription to the calculated variable exists.

The following data types are supported for calculated variables:

- Int
- Long

- Real
- Digital
- String

To configure a calculated variable, you use a variable tag that addresses an I/O device with its protocol set to "CICODE". If the variable tag address is a valid Cicode expression, the Cicode I/O device will calculate the result of the expression and present it as the tag value.

**Note:**

- The same Cicode function can generate different results depending on whether or not the Cicode function was called from a process, an event, or if it was called from a Cicode I/O device.
- When used as an address of a calculated variable, if there is at least one argument in #BAD then a function will return #BAD even if it executed properly.

If the expression in the **Address** field reads from or writes to another tag in the system, the calculated variable is added towards the point count.

**Note:** When a primary I/O server is starting up, it loads and initializes all of its I/O devices. For large projects, this process can have a heavy impact on system performance especially when there are Cicode devices supporting calculated variables. To minimize the potential impact on display client updates, you can apply a delay that will stop client tag subscriptions from switching to a primary device while the I/O server is still starting up. See Use a Client Tag Subscription Delay for Calculated Variables.

## Example 1

The Plant SCADA Example Project includes a number of tags that support the Steel Mill page. These tags include two tags that identify the length and width of the coil that is currently loaded on the uncoiling station.

Tag Name	I/O Device	Address	Data Type	Units
Entry_CoilLength	SteelMillDisk	R10	REAL	meters
Entry_CoilWidth	SteelMillDisk	R13	REAL	millimeters

Using a calculated variable, you could determine how many square meters of steel will be produced by the coil that is currently uncoiling. To do this, you could create the following calculated variable on an I/O device called "CicodeIODevice".

Tag Name	I/O Device	Address	Data Type	Units
Entry_CoilArea	CicodeIODevice	Entry_CoilLength * (Entry_CoilWidth)/1000	REAL	m2

You could then use the value generated by this tag at runtime to estimate the production potential of the current coil.

## Example 2

Using a calculated variable, you can call a Cicode function for diagnostics and determine the state of an I/O device.

Tag Name	I/O Device	Address	Data Type
IOServer_Mixer_PM800_S state	CicodeIODevice	IODeviceInfo("IOServer_Mixer_PM800", 10)	STRING

In the example above, a CICODE function has been used in a calculated variable to determine the state of the specified mixer (for example, online or offline).

**Note:** Cicode functions used for calculated variables need to be non-blocking functions.

### Example 3

Using a calculated variable, you can trigger an alarm when an I/O device goes offline. In this case, you will need to:

- create a variable tag with a calculated variable to determine the state of an I/O device
- create an advanced alarm with an expression that will trigger an alarm when the expression evaluates to TRUE.

Tag Name	I/O Device	Address	Data Type
IOServer_Mixer_PM800_S state	Mixer	IODeviceInfo("IOServer_Mixer_PM800", 10)	STRING

Alarm Tag	I/O Device	Expression
IOServer_Mixer_PM800_Offline	Mixer	(StrToInt(IOServer_Mixer_PM800_State) = 16) OR (StrToInt(IOServer_Mixer_PM800_State) = 32)

In the example above, the variable tag calculates the state of the mixer. The alarm will be triggered if the tag is calculated to be 16 (offline) or 32 (disabled).

### Example 4

Using a calculated variable, you can perform mathematical calculations. In this example, the variable tag calculates the quotient of two integers and in the event of the divisor being 0 returns an error code.

Tag Name	I/O Device	Address	Data Type
Divide_Tag_Value	CicodeIODevice	Divide(4,9)	REAL

```
REAL FUNCTION Divide( INT iDividend, INT iDivisor)
    REAL iResult;
    INT iError;
    Errset(1);
    iResult = iDividend / iDivisor
    iError = isError();
```

```
Errset(0);
RETURN iResult;
END
```

## See Also

[Add a Calculated Variable](#)

## Tag Data Persistence

Each I/O device has a tag data cache on its I/O server. This cache is periodically saved to disk to maintain the last known value (LKV) for each tag. This allows the tags on an I/O device to be initialized to their corresponding LKVs when the device is restarted. On load, the original persisted quality is substituted by the BadLastKnown quality value.

The tag cache file is saved as an [XML DataSource Schema](#), preserving the necessary information about each tag's value, quality, and timestamp. The file is saved in the Plant SCADA Data Folder with the following default name:

<ClusterName>.<IOServerName>.<IODeviceName>.cache

When the time comes to save a tag cache file, a temporary file is used. After it is written, it simply gets renamed to the original file name. This is done to avoid partial master file updates.

If during the file load the schema validation is unsuccessful, the entire file is considered invalid and the tags LKVs are lost. If during the file load a particular tag's value, quality, or timestamp is invalid, the corresponding invalid entry is simply ignored (and logged as such).

Persistence can be configured using the **Persist**, **Persist Period** and **File Name** properties (see [Add an I/O Device](#)).

## Minimum Update Rate

An I/O device can send tag update notifications to subscription clients after a pre-defined period of time expires. This is configured via the **Min Update Rate** property.

Devices that are configured for redundant operation needs to have minimum update rate set according to the rules specified in the table below.

Primary Device - <b>Min Update Rate Value</b>	Standby Device - <b>Min Update Rate Value</b>	<b>Compilation Result</b>
Blank (Default)	Blank (Default)	OK
Blank (Default)	Value Defined	Error
Value Defined	Blank (Default)	OK
Value Defined	Same value as primary	OK
Value Defined	Value is different than on primary	Error

## Staleness Period

You can specify a "staleness period" for tag values received from an I/O device. If a subscription client does not

receive a tag update from the I/O device within this specified period of time, the tag element value is considered to be "stale". The [The Quality Item](#) substatus **QUAL\_EXT\_STALE** will indicate this condition. Processing of staleness period for tags is performed on the client side of the connection.

A staleness period can be configured in the following ways:

- On the **server** side you can set the staleness period for a particular I/O device by setting the **Staleness Period** property (see [Add an I/O Device](#)). The value you enter should represent the total number of seconds that can elapse after the last update before **QUAL\_EXT\_STALE** is set to "stale".

For devices configured for redundant operation, the Staleness Period settings must conform to the same rules outline for Min Update Rate in the table above.

- On the **server** or **client**, you can use the [\[Client\]StalenessPeriod](#) parameter. This specifies the total number of seconds that can elapse following the last update before **QUAL\_EXT\_STALE** is set to "stale".
- On the **client** only, you can use the [\[Client\]StalenessPeriodTolerance](#) parameter. Staleness period tolerance specifies an amount of time as a percentage of the total staleness period, after which a check of stale client tag elements will be performed. The default value is 10%. This means that if the staleness period is set to 600 seconds, a check for stale client tag elements will be performed every 60 seconds.

## XML DataSource Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xss:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns="http://www.mycompany.com/Platform/PSI/DataSource/PersistenceCache/V1/"
    xmlns:dsps="http://www.mycompany.com/Platform/PSI/DataSource/PersistenceCache/V1/"
    elementFormDefault="qualified"
    targetNamespace="http://www.mycompany.com/Platform/PSI/DataSource/PersistenceCache/V1/">
    <xss:simpleType name="DataType">
        <xss:restriction base="xs:string">
            <xss:enumeration value="Boolean" />
            <xss:enumeration value="SByte" />
            <xss:enumeration value="Byte" />
            <xss:enumeration value="Char" />
            <xss:enumeration value="Double" />
            <xss:enumeration value="Int16" />
            <xss:enumeration value="Int32" />
            <xss:enumeration value="Int64" />
            <xss:enumeration value="Single" />
            <xss:enumeration value="String" />
            <xss:enumeration value="UInt16" />
            <xss:enumeration value="UInt32" />
            <xss:enumeration value="UInt64" />
            <xss:enumeration value="Decimal" />
            <xss:enumeration value="DateTime" />
        </xss:restriction>
    </xss:simpleType>
    <xss:simpleType name="ElementName">
        <xss:restriction base="xs:string">
            <xss:enumeration value="" />
            <xss:enumeration value="Field" />
            <xss:enumeration value="Valid" />
            <xss:enumeration value="Override" />
            <xss:enumeration value="OverrideMode" />
        </xss:restriction>
    </xss:simpleType>

```

```
<xs:enumeration value="ControlMode" />
<xs:enumeration value="Status" />
</xs:restriction>
</xs:simpleType>
<xs:complexType name="DataSource">
<xs:sequence minOccurs="1" maxOccurs="1">
<xs:element name="properties" type="PropertyCollection">
<xs:unique name="UniquePropertyName">
<xs:selector xpath="dsps:property" />
<xs:field xpath="@name" />
</xs:unique>
</xs:element>
<xs:element name="tags" type="TagCollection">
<xs:unique name="UniqueTagName">
<xs:selector xpath="dsps:tag" />
<xs:field xpath="@name" />
</xs:unique>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="PropertyCollection">
<xs:sequence minOccurs="0" maxOccurs="unbounded">
<xs:element name="property" type="Property" />
</xs:sequence>
</xs:complexType>
<xs:complexType name="TagCollection">
<xs:sequence minOccurs="0" maxOccurs="unbounded">
<xs:element name="tag" type="Tag" />
</xs:sequence>
</xs:complexType>
<xs:complexType name="Property">
<xs:simpleContent>
<xs:extension base="xs:string">
<xs:attribute name="name" type="xs:string" use="necessary" />
<xs:attribute name="type" type="DataType" use="necessary" />
</xs:extension>
</xs:simpleContent>
</xs:complexType>
<xs:complexType name="TagElement">
<xs:sequence minOccurs="1" maxOccurs="1">
<xs:element name="v" type="Value" />
<xs:element name="q" type="Quality" />
<xs:element name="t" type="xs:dateTime" />
<xs:element name="qt" type="xs:dateTime" />
<xs:element name="vt" type="xs:dateTime" />
</xs:sequence>
<xs:attribute name="name" type="ElementName" use="necessary" />
</xs:complexType>
<xs:complexType name="Value">
<xs:sequence minOccurs="1" maxOccurs="unbounded">
<xs:element name="item" type="xs:string" />
</xs:sequence>
<xs:attribute name="type" type="DataType" use="necessary" />
<xs:attribute name="size" type="xs:positiveInteger" use="necessary" />
</xs:complexType>
<xs:complexType name="Quality">
```

```
<xss:sequence minOccurs="1" maxOccurs="1">
    <xss:element name="generic" type="xs:integer" />
    <xss:element name="specific" type="xs:integer" />
</xss:sequence>
</xss:complexType>
<xss:element name="datasource" type="DataSource" />
</xss:schema>
```

## Configure Variable Tags

You can configure the following types of variables in a Plant SCADA project:

- [Add a Variable Tag](#)
- [Add a Calculated Variable](#)
- [Add a Local Variable.](#)

## See Also

[Refer to Variable Tags](#)

### Add a Variable Tag

Variable tags allow you to reference a specific I/O device variable.

**To add a variable tag:**

1. In the **System Model** activity, select **Variables**.
2. On the menu below the Command Bar, select **Variables**.
3. Add a row to the Grid Editor.
4. Type the required information in each column, or in the fields in the Property Grid.  
For a description of the properties, see below.
5. Click **Save**.

To successfully configure a variable, you need to complete the **Tag Name**, **I/O Device**, **Address** and **Data Type** fields. All other fields are optional.

### Variable Tags Properties

**Note:** If a variable tag was generated by the Equipment Editor, a number of fields on the properties form will be shaded. To configure these fields, you will need to use [Equipment Editor](#).

#### Equipment Properties

Property	Description
Equipment	The name of the equipment associated with the variable tag. Select a name from the drop-down list of existing equipment definitions, or enter a name.

Property	Description
	<p>There is a limit of 254 characters across the <b>Equipment</b> and <b>Item Name</b> fields, including any separating periods (.).</p>
<b>Item Name</b>	<p>The name of the item with which the variable tag is associated.</p> <p>Items form part of an equipment hierarchy. They can be used to associate tags, alarms and trends with a particular attribute of a physical piece of equipment (see <a href="#">Items</a>).</p> <p>There is a limit of 254 characters across the <b>Equipment</b> and Item Name, including any separating periods (.).</p> <p><b>Note:</b> When entering an item name, avoid using any <a href="#">Reserved Words</a>. If you use any of these, an error message will display when you compile your project.</p>

#### General Properties

##### **WARNING**

###### **UNINTENDED EQUIPMENT OPERATION**

Do not mix the use of odd and even variable addresses as boundaries when you are concatenating registers.  
**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Field	Description
<b>Tag Name</b>	<p>You can use any name for a tag (79 characters) provided it follows the <a href="#">Tag Name Syntax</a> rules and is not the same as the name of a Cicode function within the project (or any included projects). See <a href="#">Reserved Words</a> for a list of restrictions on naming.</p> <p>If you have many tags, use a naming convention (see <a href="#">Structured Tag Names</a>). This makes it easier to find and debug your tags. When the tag name is referenced on a graphics page, Cicode, etc., it can be used with or without a specific tag element or item.</p> <p>If you are using distributed servers, the name needs to be unique to the cluster (for example, you cannot have the same variable tag name in more than one cluster).</p> <p><b>Note:</b> Plant SCADA will automatically resolve tags without a cluster context specified if every tag name in the project is unique. See <a href="#">Clusters</a>.</p>
<b>Cluster Name</b>	<p>The name of the cluster that runs the variable tag. This field needs to be defined if your project has more than one cluster.</p> <p>You can leave this field blank in a multi-cluster system if cluster replication is enabled (see the parameter [General]ClusterReplication).</p>
<b>I/O Device</b>	<p>The name of the I/O device where the variable is stored (31 characters). If using I/O device redundancy, you need to specify the primary I/O device name here, not the standby.</p> <p><b>Note:</b> Avoid changing the tag's I/O device name from this FORM as the variable.dbf will not be updated. If you do change the tag's I/O device name you will need to clear the tag's reference fields.</p>
<b>Data Type</b>	<p>The type of I/O device variable (16 characters). I/O devices support several data types that are used to exchange data with Plant SCADA. Because of the lack of an industry standard, many I/O device manufacturers use individual naming conventions for their I/O device variables. However, every variable corresponds to one of the <a href="#">Tag Data Types</a>.</p> <p><b>Note:</b> If you do not specify a range for your tag, then an out of range alert message will be generated if you write a value which is outside the range of the type.</p> <p>You need to specify the correct data type that</p>

Field	Description
	<p>corresponds to the data type of the I/O device variable you are configuring. Each data type has a unique address format. You need to use this format when you are specifying the address of the variable (as the Address property).</p> <p>You will want to verify that you only use data types that are valid for your I/O device. If you do not specify a data type, the variable will be treated as 16-bit integer.</p> <p>Plant SCADA supports concatenation of I/O device registers. For example, you can define a real data type (in Plant SCADA) as two contiguous int data types (in the I/O device). Plant SCADA reads across the boundary of the two INTs and returns a REAL.</p> <p>If you use concatenation of registers, the address of the variables needs to be on odd boundaries or even boundaries. You cannot mix address boundaries. For example, V1, V3, V5 are valid addresses.</p> <p>Be careful when using this feature: the I/O device needs to maintain the integrity of the second register. (If the I/O device writes to the second int, the value of the real could be corrupted.) The structure of some I/O devices might not support this feature, and might therefore behave unpredictably.</p>
<b>Address</b>	<p>The register address in the I/O device where the variable is stored (254 characters). The format and prefix of an address will depend on the protocol configured for the I/O device, which can be determined by checking the <b>Protocol</b> field on the I/O devices view in Plant SCADA Studio.</p> <p><b>Note:</b> The address of the variables for all I/O devices using the same protocol needs to be on odd boundaries or even boundaries. You cannot mix address boundaries.</p>
<b>Comment</b>	Any useful comment (254 characters).
<b>Deadband</b>	(11 characters). A deadband allows the value of a variable tag to fluctuate within a defined threshold without updates being sent through the system. This may be useful if a tag produces many small, insignificant value changes. The threshold is represented as a percentage of the tag's engineering range. The default value is 0 (zero), which captures every value change.

Field	Description
	<p>For example, if a variable tag has an engineering range of zero to 10000, a deadband of 1 would mean a change in value would have to be greater than 100 (or 1 percent of the range) to be recognized. If the current value was 5600, the tag in the PLC would have to change to a value greater than 5700 or less than 5500 before an update would be sent through the system.</p> <p>The deadband setting for a variable tag will not apply to an associated trend tag. If a trend tag requires a deadband setting, it should be configured independently in the trend tag properties.</p> <p><b>Note:</b> Some data types support an engineering range that includes negative values. For example, a two-byte integer has a range of -32,768 to 32,767. In this case, the deadband percentage specified would need to be measured against a possible range of 65535. See the topic <a href="#">Tag Data Types</a> to determine the default range for each of the different data types supported by Plant SCADA.</p>
<b>Eng Units</b>	(8 characters.) The engineering units that the value represents (for example %, deg, mm/sec, etc.). This property is optional. If you do not specify engineering units, no engineering units are used.
<b>Format</b>	<p>(11 characters). The display format of the value (of the variable) when it is displayed on a graphics page, written to a file, or passed to a function (that expects a string). This property is optional. If you do not specify a format, the format defaults to #####.##.</p> <p>For more information, see <a href="#">Format Specifiers</a>.</p>
<b>Unique ID</b>	<p>A unique ID is applied to each of the variable tags in a Plant SCADA project to provide consistent identification for integration with future software releases. This field is read-only in Plant SCADA Studio.</p> <p>Each unique IDs is a Globally Unique Identifier (GUID) generated by Plant SCADA using the following format: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX</p> <p>For more information, see <a href="#">Unique IDs for Variable Tags and Equipment</a>.</p> <p><b>Note:</b> It is highly recommended that you only use Plant SCADA to generate the unique identifiers for the variables in your system. If for some reason this is not possible, use a reputable tool to generate GUIDs that</p>

Field	Description
	<p>match the format described above.</p> <p>If you use an external tool to modify the variables database, the compiler will indicate if a Unique ID is invalid, duplicated or missing.</p>

**Scale Properties**

Field	Description
<b>Raw Zero Scale / Raw Full Scale</b>	<p>The unscaled (raw) values (of the variable) that represent the zero point and full scale point for the data (11 characters). The raw values are the values that Plant SCADA reads from the I/O device.</p> <p><b>Note:</b> If not defined, the <b>Raw Zero Scale</b> and <b>Raw Full Scale</b> will default to the full range based on the data type. See the topic <a href="#">Tag Data Types</a> to determine the default range for each of the different data types supported by Plant SCADA.</p>
<b>Eng Zero Scale / Eng Full Scale</b>	<p>The scaled values that Plant SCADA calculates from the raw values (11 characters). The Raw Zero Scale is scaled to the Eng Zero Scale and the Raw Full Scale is scaled to the Eng Full Scale. These properties are represented in engineering units and are used as the upper and lower limits of trends and bar graphs.</p> <p>Many I/O devices return an integer to indicate the value of an analog input. To return a usable value, the I/O device converts an input signal (usually 4-20 mA) to a raw scale variable, usually in the range 6400 to 32000.</p> <p>To present this variable as a meaningful value, you can specify a scaling calculation. Plant SCADA then scales every value accordingly, as in the following diagram.</p>

Field	Description
	<p>The scaled value of the variable (engineering value), not its raw value, is used throughout the system.</p> <p>The scaling properties are optional. If you do not specify scaling, Eng Zero Scale defaults to Raw Zero Scale, and Eng Full Scale defaults to Raw Full Scale; that is, no scaling occurs.</p> <p>A value that is below the specified Raw Zero Scale or above the specified Raw Full Scale causes an "Out of Range" alert message in your runtime system.</p>

**Custom Properties**

Field	Description
<b>Custom 1 to Custom 8</b>	A user-defined string can be used for storage of static information. Information contained within custom1...8 can be used for categorization purposes, as such, it does not usually change during runtime.

**Security Properties**

Field	Description
<b>Write Roles</b>	<p>Used to determine which Roles have write permissions for a variable tag when using an OPC UA or Industrial Graphics client. Only users included in the specified role(s) will be able to send values to a variable tag from a client application.</p> <p>Use the drop-down menu to select one of the roles configured in your Plant SCADA project, or manually enter a comma-separated list to include multiple roles.</p> <p>For more information, see <a href="#">Enable Tag Writes for Industrial Graphics Applications</a>.</p>

**Historian Properties**

Field	Description
<b>Historize</b>	<p>This field enables you to automatically historize and publish the specified variable tag in CitectHistorian. If you set this field to "TRUE", the variable will be included in an automated configuration process within the Historian environment. If you set the field to "FALSE" (or leave it blank), the variable will not be included.</p> <p>See <a href="#">Integration with Historian</a>.</p>

## Project Properties

Property	Description
Project	The project in which the variable tag is configured.

## See Also

[Configure Variable Tags](#)

### Unique IDs for Variable Tags and Equipment

A unique ID is applied to each of the following in a Plant SCADA project:

- Variable tags
- Equipment instances
- Equipment types.

This is done to provide consistent identification for integration with future software releases.

Each unique ID is a Globally Unique Identifier (GUID) generated by Plant SCADA using the following format:

XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

A unique ID is automatically generated whenever the following occurs:

- A new variable, equipment instance or equipment type is created in Plant SCADA Studio.
- A new variable, equipment instance or equipment type is generated by Equipment Editor during an [Equipment Updates](#).
- The Migration Tool is run with the **Create unique IDs for Variables and Equipment** option selected.
- A variables database is imported as a CSV file (see [Import from Microsoft Excel](#)).
- Tags are imported from a device (see [Import Variable Tags from an External Data Source](#)).

---

**Note:** It is highly recommended that you only use Plant SCADA to generate the unique identifiers for the variables in your system. If for some reason this is not possible, use a reputable tool to generate GUIDs that match the format described above.

---

Using an external tool to modify the variables and/or equipment databases can result in unique IDs that are invalid or duplicated. The Plant SCADA compiler will use the following messages to indicate if this occurs.

#### E2182 - Invalid Unique ID format

To resolve this, do one of the following:

- If a small number of invalid IDs are detected, you can copy an identified variable into a new row in Plant SCADA Studio. The paste function will create a new ID. You can then delete the original variable with the invalid ID.
- If a large number of invalid IDs are detected, you can export the variables database to Excel and use the compiler results to delete the invalid IDs. New IDs will be created when you import the variables again.

#### E2183 - Duplicate Unique ID for variable

To resolve this, do one of the following:

- If a small number of duplicate IDs are detected, you can copy one of the identified variables into a new row in Plant SCADA Studio. The paste function will create a new ID. You can then delete the original variable with the duplicated ID.
- If a large number of duplicated IDs are detected, you can export the variables database to Excel and use the compiler results to delete the duplicated IDs. New IDs will be created when you import the variables again.

**Note:** If the compiler generates a warning message stating that some equipment or variable tags do not have a unique ID, it indicates that a project has not been migrated to the current version of Plant SCADA. To resolve this, run the [Project Migration Tool](#) with the **Create unique IDs for Variables and Equipment** option selected.

## See Also

[Repair A Project's Unique IDs](#)

## Format Specifiers

Format specifiers are keyboard characters that you can use to define the format for a numeric variable. The specifiers that you can use are:

Specifier	Function	Description
#	The number of characters to display	<p>The hash character specifies how far digits display to the right of an animation point, for example:</p> <p>Format: #####</p> <p>This will display at least four digits (or spaces) to the right of the animation point. If the number contains more than four digits, the first digit is located at the animation point.</p> <p><b>Note:</b> As an alternative to the hash (#) notation, you can use shortform notation (see below).</p>
0	Padding	<p>When a number contains fewer digits than your format specifies, Plant SCADA only displays the meaningful digits. You can use the padding character, zero (0) as the second character in the format string to fill the number with zeros, for example:</p> <p>Format: #0##</p> <p>This will display:</p>

Specifier	Function	Description
		0075 for 75 0005 for 5
-	Justification	By default, numeric variables are right-justified (within their field). You can change the default justification by using a minus (-) sign as the second character in the format string, for example: Format: #-###
.	Decimal notation	To specify decimal notation, use a period (.), for example: Format: ###.## In this example, Plant SCADA displays three digits before the decimal point and two digits after. If the variable is 12.3, Plant SCADA displays 12.30. All numbers are automatically rounded, for example: 12.306 displays as 12.31.
EU	Engineering units	You can specify engineering units (such as %, deg, rpm, M, mm/sec, and so on) when you define a variable tag. To include these units in the format of the number, type <b>EU</b> in the appropriate position. Format: #####.##EU <b>Note:</b> If you do not specify an engineering unit for the variable, only the number is displayed (or logged).
S	Exponential notation	To specify exponential notation, include the exponential character ( <b>s</b> ), for example: Format: #s### In this example, Plant SCADA displays the number in exponential notation format, for example: 1.234e+012.

You can combine format specifiers, for example:

Format: #0-###.##EU

This format string displays six digits before the decimal point and two digits after. The number is left justified, padded, and displayed with engineering units (if specified).

## Shortform Notation

As an alternative to the hash (#) notation, you can use shortform notation. With shortform notation, you use a number to specify the format of the numeric variable, for example:

Format: 3.2

This format string displays three digits before the decimal point and two digits after. This format string is equivalent to the ###.## format specification. You can also include engineering units with shortform notation, for example:

Format: 6.0EU

This format string displays six digits before the decimal point and no digits after. The number is displayed with engineering units (if specified). This format string is equivalent to the #####EU format specification.

You cannot use padding or left justification with shortform notation.

---

**Note:** If the value of a numeric variable is extremely large, it is displayed in exponential notation (for example 1.2345E012). If no format is specified for a variable, the default ####.# (or 4.1) is used.

---

## Add a Calculated Variable

[Calculated Variables](#) allows you to generate a tag value at runtime that is the result of a Cicode expression.

---

**Note:** A calculated variable will contribute to the point count if it refers to variable tag(s) in its Cicode expression that has been configured in its address field. If the calculated variable only refers to Cicode functions, and these functions do not refer to variable tags, then it will not contribute to the point count.

---

To configure calculated variables, you need to perform two procedures:

- Create an I/O device with its protocol set to "CICODE".
- Create a variable tag on a Cicode I/O device that has a Cicode expression as its address.

---

**Note:** Projects created from a starter project in Plant SCADA 2016 (or later) will include a pre-configured Cicode I/O device.

---

## Create a Cicode I/O Device

You can use one of the following methods to create a Cicode I/O device:

### Create a Cicode I/O device in the Topology activity

1. In the **Topology** activity, select **I/O Devices**.  
The I/O devices list will display in the Grid Editor.
2. In the **Project** field, select the project that will host the Cicode I/O device.
3. In the **Server Name** field, select the I/O server that will host the Cicode I/O device.
4. In the **Name** field, enter a name for the Cicode I/O device. The name needs to be unique to the selected I/O server.

5. In the **Number** field, enter a unique number for the Cicode I/O device (0-16383).
6. In the **Protocol** field, select "CICODE".
7. The remaining fields are not required for a Cicode I/O device.
8. Click **Save**.

### Create a Cicode I/O device using the Device Communications Wizard.

1. In the **Topology** activity, select **I/O Devices**.
2. On the Command Bar, select **New Device**.  
The [Using the Device Communications Wizard](#) will display.
3. When asked to select the "type of the I/O Device", select **Persisted Memory I/O Device**.
4. When asked to select the "manufacturer, model and method of communication" for the I/O device, select **Cicode Protocol**.
5. When you have finished stepping through the wizard, the Cicode I/O device will appear in the I/O Devices list.

---

**Note:** The **Startup Mode** field should be configured appropriately if device redundancy is required for the Cicode I/O device you have created.

---

### Create a Calculated Variable on a Cicode I/O Device

#### To create a calculated variable on a Cicode I/O device:

1. In the **System Model** activity, select **Variables**.
2. On the menu below the Command Bar, select **Variables**.
3. The variables list will display in the Grid Editor.
4. Add a row to the Grid Editor.
5. In the **Tag Name** field, enter a name that describes the calculated variable.
6. In the **I/O Device** field, select the name of the Cicode I/O device you would like to use.
7. In the **Data Type** field, select a data type that is appropriate to the value that will be returned by the specified calculation or Cicode expression.
8. In the **Address** field, enter a valid Cicode expression.

---

**Note:** Cicode functions used for calculated variables need to be non-blocking functions.

---

9. Click **Save**.

---

**Note:** If an error is trapped in the Cicode that has been placed in the **Address** field of a calculated variable (either internally via a built-in Cicode function or by calling the function ErrTrap), the variable tag will have bad quality and will raise a Cicode error when the tag is read. After reading the tag, call IsError() to get the error number and clear it.

---

### See Also

[Configure Variable Tags](#)

## Add a Local Variable

[Local Variables](#) allow you to store data in memory when you start your runtime system.

### To add a local variable:

1. In the **System Model** activity, select **Variables**.
2. On the menu below the Command Bar, select **Local**.
3. The local variables list will display in the Grid Editor.
4. Add a row to the Grid Editor.
5. Type the required information in each column, or in the fields in the Property Grid.  
For a description of the properties, see below.
6. Click **Save**.

## Local Variables Properties

### General Properties

Field	Description
<b>Name</b>	You can use any name for a tag (79 characters) provided it follows <a href="#">Tag Name Syntax</a> and is not the same as the name of a Cicode function within the project or any included projects. See <a href="#">Reserved Words</a> for a list of restrictions on naming.
<b>Data Type</b>	The data type of the variable (16 characters). Select one of the following supported data types: <ul style="list-style-type: none"><li>• BCD (Binary) - Coded Decimal - 2 bytes (0 to 9,999)</li><li>• BYTE (Byte) - 1 byte (0 to 255)</li><li>• DIGITAL (Digital) - 1 bit or 1 byte (0 or 1)</li><li>• INT (Integer) - 2 bytes (-32,768 to 32,767)</li><li>• UINT (Unsigned Integer) - 2 bytes (0 to 65,535)</li><li>• LONG (Long Integer) - 4 bytes (-2,147,483,648 to 2,147,483,647)</li><li>• LONGBCD (Long Binary Coded Decimal) - 4 bytes (0 to 99,999)</li><li>• REAL (Floating Point) - 8 bytes (-1.7e308 to 1.7e308)</li><li>• STRING (String) - up to 256 bytes</li><li>• SCII (null terminated)</li></ul> Numeric and digital variables have a default value

Field	Description
	of 0 and string variables default to "" (empty string). If you do not specify a data type, the local variable will be treated as 16-bit integer.
<b>Array Size</b>	<p>In the <b>Array Size</b> field (8 characters), enter the size of the array (number of elements) used to store the local variable. The array will be of the data types specified in the <b>Data Type</b> field. The array can be one or two-dimensional.</p> <p>Up to 32766 elements can be used, which, for a two dimensional array, means that the size of the first dimension multiplied by the size of the second is less than or equal to 32766. When specifying a multi-dimensional array, separate the dimensions with a comma, for example "20,30".</p>
<b>Eng Units</b>	The engineering units that the value represents (for example, %, deg, or mm/sec). This property is optional. If you do not specify engineering units, no engineering units are used.
<b>Format</b>	<p>(11 characters). The display format of the value (of the variable) when it is displayed on a graphics page, written to a file, or passed to a function (that expects a string). This property is optional. If you do not specify a format, the format defaults to #####.##.</p> <p>For more information, see <a href="#">Format Specifiers</a>.</p>
<b>Comment</b>	Any useful comment (254 characters).

### Scale Properties

Field	Description
<b>Zero Scale</b>	Enter the value that represents the zero point for the local variable's data (11 characters). The zero scale value is used as the lower limit for trend and bar graphs, and values below the zero scale value will cause an "Out of Range" alert message in the runtime system.
<b>Full Scale</b>	Enter the value that represents the full scale point for the local variable's data (11 characters). The full scale value is used as the upper limit for trend and bar graphs, and values above the full scale value will cause an "Out of Range" alert message in the runtime

Field	Description
	system.

### Project Properties

Property	Description
Project	The project in which the local tag is configured.

## See Also

[Configure Variable Tags](#)

## Refer to Variable Tags

There are a number of ways you can reference the information that is made available by the variables within a Plant SCADA system.

- If a set of variables are configured as an array, see [Refer to Array Elements](#).
- If the elements of a tag are configured as tag extensions, see [Refer to Tag Extensions](#).
- To access a tag associated with an equipment item, see [Use an 'Equipment.Item' Reference](#).
- To browse a list of variable tags at runtime, see [Configure Tag Browsing](#).

## See Also

[Variable Tags](#)

## Refer to Array Elements

Each entry in an array is referred to by an index. You can extract individual variables (from the array) by specifying the tag name and index using the following format:

`<Tag Name>[Index]`

For example, to refer to the third variable of the array "Conveyor\_Speed", use the following syntax:

`Conveyor_Speed[2]`

The index of the first entry in an array is always 0 (zero).

Note the following when using arrays:

- Do not define large arrays, because each time an individual array entry is requested, Plant SCADA reads the entire array from the I/O device. With large arrays, system performance could be reduced.
- The size of the array needs to be less than the maximum request length for the driver used to connect to the I/O device. To determine the maximum request length, refer to the driver documentation.
- Declare digital arrays on a 16 bit boundary. Plant SCADA rounds each digital array down to the nearest 16 bit boundary. Consequently, digits in the array are changed. For example, if you declare an array Test to start

at bit 35, and Plant SCADA starts the array at bit 32. The index to the array also starts at bit 32; Test[0] refers to bit 32, not bit 35.

- Each individual array entry has its own data value for the field, valid and override elements, but the entries within an array share the override and control mode elements, quality and timestamps. Reading the quality or timestamp for an indexed array entry will return the quality or timestamp for the entire array. Writing to the value for an indexed array entry will change the timestamps and quality for the entire array. Changing the override mode and the control mode for an array entry will change it for the entire array.

## See Also

[Tag Extensions](#)

## Refer to Tag Extensions

You can reference [Tag Extensions](#) in the following ways:

- [Read a Tag Value](#)
- [Write to a Tag Value](#)
- [Set a Tag to Control Inhibit Mode](#)
- [Set a Tag to Override Mode](#)
- [Determine Tag Status.](#)

## See Also

[Refer to Variable Tags](#)

## Read a Tag Value

Data received from the field is represented by the Field and Valid tag elements.

- The **Field** element represents the latest tag field data received from the device.
- The **Valid** element represents the last field data which had "Good" quality.

**Note:** The Valid tag element initially has bad quality. The quality will only become good when the first read request is successfully finished for the tag, which will only occur when the IO device is online and either background polling is enabled, the tag has been subscribed to, or a TagRead has been initiated.

You can access these elements by using a qualified tag reference (a tag referenced by the tag name and the element name, for example 'MyTag.Field'). The following tables describe each of these elements.

## Field Tag Element

Reference Syntax	Plant SCADA Data type	Description
TagName.Field	INT	Implied value of Field element

Reference Syntax	Plant SCADA Data type	Description
	REAL STRING	
TagName.Field.V	INT REAL STRING	Value
TagName.Field.VT	TIMESTAMP	Timestamp of when the value last changed
TagName.Field.Q	QUALITY	Quality
TagName.Field.QT	TIMESTAMP	Timestamp of when the quality last changed
TagName.Field.T	TIMESTAMP	Timestamp of when the element was last updated

## Valid Tag Element

Reference Syntax	Plant SCADA Data type	Description
TagName.Valid	INT REAL STRING	Implied value of Valid element
TagName.Valid.V	INT REAL STRING	Value
TagName.Valid.VT	TIMESTAMP	Timestamp of when the value last changed
TagName.Valid.Q	QUALITY	Quality
TagName.Valid.QT	TIMESTAMP	Timestamp of when the quality last changed
TagName.Valid.T	TIMESTAMP	Timestamp of when the element was last updated

If you determine that the tag value is incorrect because of a sensor or communication becoming inoperative, the

tag value may need to be overridden. The Override tag element is used to store the overridden tag value. For further information, refer to [Set a Tag to Override Mode](#).

## See Also

[Set a Tag to Control Inhibit Mode](#)

### Write to a Tag Value

The tag elements which have read/write access can be modified. However, these elements can only be updated as a whole; writes to an individual element item is not allowed.

The following table shows the result of writing to each of the tag elements:

Element Name	Write allowed	Write Result
Not specified (unqualified tag reference)	Yes	Write to the Field element and propagate the value to the I/O device.
Field	Yes	Write to the Field element and propagate the value to the I/O device.
Valid	No	Will not succeed.
Override	Yes	Write to the Override element.
Override Mode	Yes	Write to the Override Mode element. See the appropriate function references for the available override mode values. <b>Note:</b> When the tag is in Override mode, an unqualified tag reference will refer to the Override element.
Control Mode	Yes	Write to the Control Mode element The Value item can be one of the following: 0 – Control inhibit mode is Off. 1 – Control inhibit mode is On. <b>Note:</b> When Control inhibit mode is On, writing to the Field element is prohibited.
Tag Status	No	Will not succeed.

If you determine that the tag value is incorrect because of a sensor or communication interruption, the tag value

may need to be overridden. The Override element tag element is used to store the overridden tag value. For further information, refer to [Set a Tag to Override Mode](#).

## See Also

[Set a Tag to Control Inhibit Mode](#)

### Set a Tag to Control Inhibit Mode



#### PERFORMING A CONTROL OPERATION WHICH DOES NOT IN FACT OCCUR.

When the tag is put into the control inhibit mode, writing to the Field element value is prohibited. If the system is configured in such a way that it does not give the operator any indication that the tag is in the control inhibit mode, the operator may assume that he is performing a control operation which does not, in fact, occur.

Configure the system in such a way that it provides a visual indication to the operator that a tag is in the control inhibit mode.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

To provide a visual indication to the operator that a tag is in the control inhibit mode the following can be done:

Set any of the following Citect.ini parameters in such a way that control inhibit mode will be indicated by changing the background color or overlaying the numeric or text graphics objects and symbol set objects with a dithered pattern:

- [Page]ControlInhibitDitheringColor
- [Page]ControlInhibitDitheringDensity
- [Page]ControlInhibitTextBackgroundColor

Then set [Page]IgnoreValueQuality parameter to a value of 0 or 2.

Or:

Use the Control Mode element value (0 if the tag is not in the control inhibit mode or 1 otherwise).

Or:

Use the Field element quality Tag Status ControlInhibit bit (1 if the tag is in the control inhibit mode or 0 otherwise).

## Control Mode

Control inhibit mode allows you to prohibit writing to the Field tag element. Setting a tag in Control inhibit mode is applied system wide. Writing to the Field element will be prohibited for each of the I/O data consumers.

Control inhibit mode is controlled by the "Control Mode" element which is described in the following table.

Reference Syntax	Plant SCADA Data type	Description
TagName.ControlMode	INT	Value of Control Mode element.
TagName.ControlMode.V	INT	Value. Allowed values are: 0 – Control inhibit mode is Off. 1 – Control inhibit mode is On. Default value: 0.
TagName.ControlMode.VT	TIMESTAMP	Timestamp of when the value last changed.
TagName.ControlMode.Q	QUALITY	The Control Mode quality is QUAL_GOOD if the tag was put into the control inhibit mode on the primary server and it was propagated to redundant servers. Otherwise the general quality status will be QUAL_UNCR and the extended substatus will be QE_NOT_REPLICATED to indicate that not every redundant server is aware of the fact that the tag is in control inhibit mode.
TagName.ControlMode.QT	TIMESTAMP	Timestamp of when the quality last changed.
TagName.ControlMode.T	TIMESTAMP	Time of when the tag was put in or out of the control inhibit mode (equal to 0 at startup).

**Note:** The quality of tags referenced by items (Tag1.v or Tag1.Field.t) is GOOD and its timestamps are 0 (INVALID TIMESTAMP). Therefore they give no visual indication of any not good quality, error or change in handling state such as control inhibit or override mode regardless of the setting used for the [Page]IgnoreValueQuality parameter.

## See Also

[Set a Tag to Override Mode](#)

### Set a Tag to Override Mode



**PERFORMING A CONTROL OPERATION BASED ON INACCURATE DATA.**

When the tag is put into the override mode, the default tag element value will be equal to the Override element value instead of the Field element value. If the system is configured in such a way that it does not give the operator any indication that the tag is in override mode, the operator may assume that he is seeing the Field element value (live data) and consequently may operate the plant inappropriately.

Configure the system in such a way that it provides a visual indication to the operator that a tag is in override mode.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

To provide a visual indication to the operator that a tag is in the override mode the following can be done:

Set any of the following Citect.ini parameters in such a way that override mode will be indicated by changing the background color or overlaying the numeric or text graphics objects and symbol set objects with a dithered pattern:

- [\[Page\]OverrideDitheringColor](#)
- [\[Page\]OverrideDitheringDensity](#)
- [\[Page\]OverrideTextBackgroundColor](#)

Then set [\[Page\]IgnoreValueQuality](#) parameter to a value of 0 or 2.

Or:

Use the Override Mode element value (0 if the tag is not in the Override Mode or 1,2,3,4 otherwise).

Or:

Use the default or override element quality Tag Status Override bit (1 if the tag is in the override mode or 0 otherwise).

## Using Override Mode

If you determine that the tag value is incorrect because of a sensor or communication interruption, the tag value may need to be overridden. The Override tag element is used to store the overridden tag value.

You can put the tag into the Override Mode which will cause each unqualified tag reference (a tag referenced only by the tag name, for example 'MyTag') in every I/O Data Consumer (Control Client, Trend Server, Alarm server, and so on) to return the Override element. The quality of the tag will indicate that the tag is in override mode.

Tag override is controlled by the Override and Override Mode tag elements. The Override element contains the override value and the Override Mode tag element is used to set the specific override mode.

The following tables describe each of these elements.

## Override Tag Element

Reference Syntax	Plant SCADA data type	Description
TagName.Override	INT REAL STRING	Implied value of Override element
TagName.Override.V	INT REAL STRING	Value
TagName.Override.VT	TIMESTAMP	Timestamp of when the value last changed
TagName.Override.Q	QUALITY	Quality. The Override value has "Good" quality except when it is out of range
TagName.Override.QT	TIMESTAMP	Timestamp of when the quality last changed
TagName.Override.T	TIMESTAMP	Timestamp of when the element was last updated

**Note:** The quality of tags referenced by items (Tag1.v or Tag1.Field.t), is constantly GOOD and its timestamps are constantly 0 (INVALID TIMESTAMP). Therefore they give no visual indication of any not good quality, error or change in handling state such as control inhibit or override mode regardless of the setting used for the [\[Page\]IgnoreValueQuality](#) parameter.

## Override Mode Tag Element

Reference Syntax	Plant SCADA data type	Description
TagName.OverrideMode	INT	Value of Override Mode element.
TagName.OverrideMode.V	INT	Override mode value. See the appropriate function references for the available override mode values.
TagName.OverrideMode.VT	TIMESTAMP	Timestamp of when the value last changed.
TagName.OverrideMode.Q	QUALITY	The Override Mode quality is

Reference Syntax	Plant SCADA data type	Description
		QUAL_GOOD if the tag was put into the override mode on the primary server and it was propagated to any redundant servers. Otherwise the general quality status will be QUAL_UNCR and the extended substatus will be QE_NOT_REPLICATED to indicate that some redundant servers are unaware of the fact that the tag has been overridden.
TagName.OverrideMode.QT	TIMESTAMP	Timestamp of when the quality last changed.
TagName.OverrideMode.T	TIMESTAMP	Time when the tag was put in or out of override mode. (equal to 0 at start up).

## Override Modes

Override Mode Value	Description
0	The override mode is Off.
1	The tag is in static override mode and the override value is initially set to the value of the tag's Field element.
2	The tag is in static override mode and the override value is initially set to the value of the tag's Valid element.
3	The tag is in static override mode and the override will remain at the previously set value of the tag's Override element.
4	The tag is in the dynamic override mode and the override value will continually track the value of the tag's Valid element. The ability to write to the value of the tag's override element is disabled in

Override Mode Value	Description
	this mode.

## See Also

[Set a Tag to Control Inhibit Mode](#)

### Determine Tag Status

The Tag Status element is used to represent the current status of the tag. The value of this element is affected only for those tag operations that involve a physical device (for example, writing to the Field element will affect the status element, but writing to the Control Mode will not). Tag Status element is reset after I/O server restart. The element value contains a set of bit flags.

The following table describes the Tag Status element:

Reference Syntax	Plant SCADA Data type	Description
TagName.Status	INT	Value of Tag Status element
TagName.Status.V	INT	Value. A set of bit flags representing the following data: - 0x1: Read Pending. - 0x2: Write Pending. - 0x4: Write Successful. - 0x8: Write Unsuccessful. Default value: 0.
TagName.Status.VT	TIMESTAMP	Timestamp of when the value last changed.
TagName.Status.Q	QUALITY	Quality.
TagName.Status.QT	TIMESTAMP	Timestamp of when the quality last changed.
TagName.Status.T	TIMESTAMP	Timestamp of when the element was last updated.

## Use an 'Equipment.Item' Reference

You can reference a variable tag using 'equipment.item' instead of a tag name.

For example, an equipment type named "Motor" includes the item "FailStart". FailStart includes a variable tag definition that links to an I/O point on the physical motor. To refer to this variable tag, you can use "Motor.FailStart" instead of the tag name. (For more information, see [Items](#).)

To do this, use the following syntax:

[Cluster.]Equipment.Item[.ExtItem]

Where:

Cluster	The cluster name (optional).
Equipment.Item	A substitution for a tag name using an "equipment.item" reference.
ExtItem	The <a href="#">Tag Extensions</a> item name (optional). If the item name is not specified, the value of the property is referenced.

This means you can:

- Use equipment.item in place of variable tags in all expression fields in graphic pages.
- Use equipment.item to reference variable tags in Genie and Super Genie substitutions.
- Use equipment.item in the value field to represent a variable tag when associating super genie substitutions via metadata.
- Pass equipment.item to Cicode functions in which the variable tag name is an argument.
- Pass equipment.item to CTAPI functions in which the variable tag name is an argument.

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

If you update the Equipment Name or Equipment Item for a variable tag and restart the I/O Server, these changes will not be automatically reflected in tag references in Equipment.Item syntax within the system. After updating the Equipment or Item name for a tag, restart all servers and clients in the system.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** You can also access alarm data in a similar way. See [Using Alarm Properties as Tags](#).

## **See Also**

[Tag Extensions](#)

## **Configure Tag Browsing**

Tag browsing enables you to browse a list of variable tags owned by a data source at runtime, in the same way that you can with alarm and trend browsing. You can create a tag browsing page in the Graphics Builder that will be available to the runtime operator. The tag browsing page can present any of the following information to the operator:

- Any property available for a [Add a Variable Tag](#), including a newly added equipment field.
- Some fields available from the Cicode functions [TagInfo](#) and [TagInfoEx](#).
- Tags which are in [Set a Tag to Override Mode](#).

- Tags which are in Set a Tag to Control Inhibit Mode.
- The Number of current subscriptions.

Tag browsing enables you to browse to the tag element level only, for example, '.field'. Refer to the [TagBrowseOpen](#) Cicode function for a list of supported fields.

For more details on the above Value, Quality, Timestamp tag elements and sub elements refer to [Tag Extensions](#).

## See Also

[The Browsing Session](#)

[Open a Browsing Session](#)

### The Browsing Session

In order to access the tag information at runtime, the client computer initiates a connection to the servers specifying the browsing session parameters such as required fields and filters. It is possible to open and maintain multiple browsing sessions simultaneously.

Each server in a session prepares a snapshot of its data and sends records in batches to the client. As the server records arrive, the client reconciles them against each other in the event that duplicate or conflicting data arrives from different servers.

The records reconciliation is based on records with the "best device status" which are selected, and the others are discarded.

"Best device status" is calculated as follows:

- Data Source online state (ONLINE is picked over OFFLINE), then
- Data Source active state (ACTIVE is picked over INACTIVE), then
- Data Source priority (higher priority is picked over a lower one, with highest being 1 – for primary I/O device)

If records from different I/O servers fully match on "best device status" the browsing records are taken on the first-come basis. Be aware that there may be project inconsistencies (on different machines) which can affect tag browsing functionality. This can occur if a project is being updated across a number of servers, but only gradually, so there may exist several versions of the project across the network. Tag Browsing will try to reconcile the conflicting data the best way possible, but in such cases check that the returned data is correct.

For each server session the client keeps track of how many records are expected, and what the last record received was. The [TagBrowseNext](#) Cicode function will either cause the client to look at the next cached record which was received earlier and fully reconciled, or request the next batch of records from the I/O servers.

If a communication disruption occurs during the browsing process, or when the client explicitly closes the browsing session, the session will be terminated.

## See Also

[Open a Browsing Session](#)

## Open a Browsing Session

Open a browsing session by using the [TagBrowseOpen](#) Cicode function. The server then builds a snapshot of the data requested (representing the state of the data at the time TagBrowseOpen was called).

A snapshot from a given server contains the same number of records as the number of tags available from all I/O devices. Requesting only the "CLUSTER" field with 10000 tags belonging to three clusters will produce a snapshot with 10000 records where entries Cluster1, Cluster2, Cluster3 will be repeating a number of times.

The snapshot will exist on the server until all the records have been fetched by the client, or the browse session is closed, or an existing connection is lost, or a new connection is established.

When you use the [TagBrowseOpen](#) function you have the option to specify several parameters which will determine which tags will be displayed. These parameters are used to select tags to display in the browsing session.

## See Also

[The Browsing Session](#)

[Specify Browsing Field](#)

[Browsing Filter](#)

[Browsing Clusters](#)

[Specify a Sort Order](#)

## Browsing Filter

A browsing filter can be used to specify the records to return during the browsing session, using a set of predefined field names, shown below, within a regular expressions separated by a semicolon.

For example, consider the following filter:

NAME=Tag\_IO\_\*;TYPE<2

This would request all tags from all clusters where the tag name matches the pattern "Tag\_IO\_\*", and tag type is 0 (DIGITAL) or 1 (INTEGER). If you do not specify a filter string then all tag records will be displayed.

If any of the filter names are invalid, opening a browsing session will not succeed and will return an invalid handle.

## See Also

[The Browsing Session](#)

[Open a Browsing Session](#)

[Specify Browsing Field](#)

[Browsing Clusters](#)

[Specify a Sort Order](#)

## Specify Browsing Field

Fields are specified using a set of predefined field names, separated by a comma. For example,

"TAG,TYPE,ENG\_UNITS". Each browsing record arriving from the server in response to the request will be represented as a list of key-value pairs. At least one field should be specified. If any of the fields are invalid, opening a browsing session does not succeed.

Only the fields explicitly asked for will be available for the [TagBrowseGetField](#) function. The only exception to this are the fields mentioned within the sort order since record sorting takes place on both the server and the client and sorting cannot be done on a field which is not available on the client.

## See Also

[The Browsing Session](#)

[Open a Browsing Session](#)

[Browsing Filter](#)

[Browsing Clusters](#)

[Specify a Sort Order](#)

## Specify a Sort Order

You can optionally include a preferred sort order for the fields returned in the browsing session by including A (ascending) or D (descending). For example, "TYPE:D".

By default tag browsing records are sorted by the tag primary key which is 'Cluster.TagName', which is implicitly appended to the list of sorting fields since the other fields may end up not being unique.

Tag records are sorted on the server and arrive at the client in the order specified by the sort order parameter, or the default sorting order.

## See Also

[The Browsing Session](#)

[Open a Browsing Session](#)

[Browsing Filter](#)

[Browsing Clusters](#)

[Specify Browsing Field](#)

## Browsing Clusters

An optional parameter can be included using a comma delimited string that specifies the clusters containing tags to be included in the browse session. An empty string indicates that all clusters will be browsed.

## See Also

[The Browsing Session](#)

[Open a Browsing Session](#)

[Browsing Filter](#)

[Specify Browsing Field](#)

[Specify a Sort Order](#)

## Link Tags to an External Data Source

Linking is an I/O device specific operation. When you add an I/O device to Plant SCADA, you can choose to link it to an external data source (see [Link an I/O Device to an External Data Source](#)).

The external data source is where the tag data was saved when the actual I/O device was programmed. If you link to the external data source, Plant SCADA automatically creates variable tag records for every tag in the I/O device.

These variable tag records are dynamically linked to the tags in the external data source. Plant SCADA can update whenever one of the external tags is changed.

For example, if you change the address of a tag using a third party I/O device programming package, the external data source is changed. Then, when you compile your project in Plant SCADA, the change is copied from the external database to Plant SCADA's variable tags database.

If a tag is linked and you change a field, your change will not be overwritten when the link is next refreshed. Generally, however, any field which takes its value from the external data source is disabled.

---

**Note:** Some properties defined for the external tags will not be relevant to Plant SCADA. Also, some will not be in a format that Plant SCADA can read. Each I/O device has an associated format file in Plant SCADA. It is this file that determines what information is copied to Plant SCADA's variable tags database and how this information is to be converted. In most cases, you will not have to edit this file.

---

## Linked Unity Tags

Note the following for Plant SCADA tags linked to Unity tags:

1. The addition of a new tag on either side, will result in the addition of a new tag on the other side.
2. The deletion of an existing tag on either side, will result in the deletion of the corresponding tag on the other side.
3. The modification of an existing tag on either side, will result in the modification of the corresponding tag on the other side.
4. The delete operation in either Unity or Plant SCADA will take precedence over other operations (such as modify).
5. If there is some contention between the Unity database and the Plant SCADA database for the same tag, that is modifications are done from both databases at the same time, the Unity database item will take precedence over the Plant SCADA database item.
6. The linked I/O Device live update synchronization will be triggered in the following cases:
  - Unity configuration update.
  - Plant SCADA configuration update.
  - Manual refresh.

See [Unity Support Matrixes](#) for information about levels of support between Unity and Plant SCADA.

## See Also

[Refresh the Tags for a Linked I/O Device](#)

[Breaking the Link to the External Data Source](#)

## Unity Support Matrixes

The following sections show the levels of support between Unity and Plant SCADA:

- Imported Tags
- Exported Tags
- Linking Tags

## See Also

[Link Tags to an External Data Source](#)

### Imported Tags

For Unity SpeedLink Dynamic and Static:

Unity PLCs	Unity Data Types	Plant SCADA Protocols	Support Status
QUANTUM	BOOL	Unite	Not Supported
	EBOOL	Modnet	Supported (Except BYTE type)
	BYTE	Modbus	Supported (Except BYTE type)
	WORD	MBPlus	Supported (Except BYTE type)
	DWORD	OPC	Not Supported
PREMIUM	INT	Unite	Supported
	DINT	Modnet	Supported (Except BYTE type)
	UINT	Modbus	Supported (Except BYTE type)
	UDINT	MBPlus	Supported (Except BYTE type)
	REAL	OPC	Not Supported
* Denotes supported by Unity SpeedLink Dynamic only.			

Unity PLCs	Unity Data Types	Plant SCADA Protocols	Support Status
	STRING		
M340	BOOL EBOOL BYTE WORD DWORD INT DINT UINT UDINT REAL TIME * DATE * TOD * DT * STRING	Unite  Modnet  Modbus  MBPlus  OPC	Not Supported  Supported (Except BYTE type)  Supported (Except BYTE type)  Not Supported  Not Supported

**Note:** Plant SCADA recommends that you use the Unity SpeedLink Dynamic driver when importing tags from Unity Pro into a Plant SCADA project.

**Note:** Arrays are supported for every data type except DATE, TOD, DT and STRING data types. For Unity data types marked with \*, Plant SCADA does not have data types that directly match the Unity types. Instead use supported data types and Cicode to convert and simulate those data types.

In addition, the following Unity data types are not supported:

- Structured data types.
- Arrays which are not zero-based.
- Multi-dimensional arrays.
- Arrays of Unity BOOL data type are not supported for all protocols except OPC and will be excluded from the import.

## See Also

[Unity Support Matrixes](#)

## Exported Tags

For SpeedLink Static:

Unity PLCs	Unity Data Types	Plant SCADA Protocols	Support Status
QUANTUM	BOOL EBOOL BYTE WORD	Unite	Not Supported
		Modnet	Supported (Except BYTE type)

Unity PLCs	Unity Data Types	Plant SCADA Protocols	Support Status
	DWORD INT DINT UINT UDINT REAL TIME * DATE * TOD * DT * STRING	Modbus	Supported (Except BYTE type)
		MBPlus	Supported (Except BYTE type)
		OPC	Not Supported
PREMIUM	BOOL EBOOL BYTE WORD DWORD INT DINT UINT UDINT REAL TIME * DATE * TOD * DT * STRING	Unite	Supported
		Modnet	Supported (Except BYTE type)
		Modbus	Supported (Except BYTE type)
		MBPlus	Supported (Except BYTE type)
		OPC	Not Supported
M340	BOOL EBOOL BYTE WORD DWORD INT DINT UINT UDINT REAL TIME * DATE * TOD * DT * STRING	Unite	Not Supported
		Modnet	Supported (Except BYTE type)
		Modbus	Supported (Except BYTE type)
		MBPlus	Not Supported
		OPC	Not Supported

**Note:** Arrays are supported for data types except DATE, TOD, DT and STRING data types. For Unity data types marked with \*, Plant SCADA does not have data types that directly match the Unity types. Instead use Technical supported data types and Cicode to convert and simulate those data types.

In addition, the following Unity data types are not supported:

- Structured data types.
- Arrays which are not zero-based.
- Multi-dimensional arrays.
- Arrays of Unity BOOL data type are not supported for all protocols except OPC.

## See Also

[Unity Support Matrixes](#)

### Linking Tags

Linked I/O Device Live Update.

For Unity SpeedLink Dynamic:

Unity PLCs	Unity Data Types	Plant SCADA Protocols	Support Status
QUANTUM	BOOL	Unite	Not Supported
	EBOOL	Modnet	Supported (Except BYTE type)
	BYTE	Modbus	Supported (Except BYTE type)
	WORD	MBPlus	Supported (Except BYTE type)
	DWORD	OPC	Not Supported
PREMIUM	BOOL	Unite	Supported
	EBOOL	Modnet	Supported (Except BYTE type)
	BYTE	Modbus	Supported (Except BYTE type)
	WORD	MBPlus	Supported (Except BYTE type)
	DWORD	OPC	Not Supported

Unity PLCs	Unity Data Types	Plant SCADA Protocols	Support Status
	STRING		
M340	BOOL EBOOL BYTE WORD DWORD INT DINT UINT UDINT REAL TIME * DATE * TOD * DT * STRING	Unite  Modnet  Modbus  MBPlus  OPC	Not Supported  Supported (Except BYTE type)  Supported (Except BYTE type)  Not Supported  Not Supported

**Note:** Arrays are supported for data types except DATE, TOD, DT and STRING data types. For Unity data types marked with \*, Plant SCADA does not have data types that directly match the Unity types. Instead use supported data types and Cicode to convert and simulate those data types.

In addition, the following Unity data types are not supported:

- Structured data types.
- Arrays which are not zero-based.
- Multi-dimensional arrays.
- Arrays of Unity BOOL data type are not supported for all protocols except OPC.

## See Also

[Unity Support Matrixes](#)

## Alarms

An alarm indicates that an abnormal condition has occurred within your Plant SCADA system. For example, you can configure an alarm to notify you if a piece of hardware is not responding to input commands, or if a variable tag value has moved beyond a specified limit.

Alarms are a fundamental component of a control system, as they indicate to an operator that the system requires attention. When you configure your Plant SCADA system, you should confirm that alarms will be reported to an operator in a way that allows them to respond in an appropriate manner.



### LOSS OF CONTROL

- Do not use Plant SCADA's alarming system as the primary alert mechanism for safety-related alarms or notifications (that is, those classified as critical to process safety, the protection of the plant and equipment, or the protection of human life).
- Do not use Plant SCADA's audible alarm functionality as a replacement for safety-related warning systems critical to process safety, the protection of the plant and equipment or the protection of human life.
- Safety-related alarms and notifications should be independent and separate from the process control system. They should be implemented in the form of a stand-alone physical alarm system driving individual discrete alarm annunciators.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Plant SCADA supports two types of alarms:

- **Defined alarms**

Defined alarms are added to a project by a system engineer to report specific runtime conditions that require an operator's attention. Each alarm is associated with one or more tags, depending on the type of alarm that has been configured.

See [Defined Alarm Types](#).

- **Hardware alarms**

Plant SCADA regularly runs diagnostic routines to monitor the performance of internal operations associated with any connected peripheral equipment. If the system detects any abnormal conditions, such as an I/O device that is not responding to input commands, a hardware alarm is generated.

See [Hardware Alarms](#).

Plant SCADA monitors a system for the conditions that define an alarm. This is evaluated by an [alarm server process](#), which generates an alarm whenever it detects the conditions that define an alarm.

When the alarm server generates an alarm, it is reported to an operator via the runtime interface. If your project is based on a Plant SCADA starter project, alarms are reported in the following ways:

- The three most recent alarms are displayed on the alarm banner
- Alarm lists and alarm counts are displayed on specific alarm pages
- A historical list of alarms and events is displayed on the Sequence of Events page (see [Events](#)).

For more information, see [Viewing Alarms](#).

You can also add content to your graphics pages that are animated when specific alarm conditions occur (see [Using Alarm Properties as Tags](#)).

Information about alarms is provided in the following sections:

- [Manage Alarms](#)
- [Configure Alarms](#)
- [Action Alarms at Runtime](#)
- [Archive the Alarms History](#)
- [Customize Alarm Pages](#).

## Defined Alarm Types

The following table describes the types of defined alarms you can add to your Plant SCADA project. Within these types, two different processes are used to determine alarm conditions.

- With most alarm types, polling is used to monitor a system for the conditions that define an alarm. The system subscribes to the tags associated with each configured alarm, and polls them at a rate specified by the Citect.ini parameter [\[Alarm\]ScanTime](#).

When a change in value is detected, a notification is sent to the alarm server for processing. The alarm server evaluates each notification against the conditions that define each alarm, and generates an alarm if the conditions are met.

- For time stamped digital alarms, time stamped analog alarms, and double point status alarms, Plant SCADA supports the retrieval of time-stamped data directly from field devices. This is enabled via the Cicode function [AlarmNotifyVarChange](#), or via a set of specific drivers that support the Driver Runtime Interface (see [Retrieving Time-stamped Data from I/O Devices](#)).

---

**Note:** Different types of alarms, even those which are not designated as "timestamped", can retrieve timestamp information from different sources. To avoid any discrepancies in the reporting of alarms, it is recommended that your system uses an accurate clock synchronisation mechanism.

---

Alarm Type	Description
Digital alarms	<p>Digital alarms activate in response to the state of one or two digital variables.</p> <p>When you define the variable tags for a digital alarm, you can precede a tag name with the logical operator "NOT". This specifies that the OFF state (0) is the triggering condition for the variable tag.</p> <p><b>Example</b></p> <p>A digital alarm is configured to activate in response to the following variable tags:</p> <ul style="list-style-type: none"> <li>• Variable Tag A = RFP3_TOL</li> <li>• Variable Tag B = NOT MCOL3043</li> </ul> <p>In this scenario, the alarm is triggered when the state of both variables changes to the active state:</p> <ul style="list-style-type: none"> <li>• RFP3_TOL changes to ON (1)</li> <li>• MCOL304 changes to OFF (0).</li> </ul> <p>You can specify a delay for a digital alarm, which means the alarm only becomes active when the triggering condition spans the duration of the specified delay period.</p> <p>See <a href="#">Add a Digital Alarm</a>.</p>
Time stamped alarms	<p>Time stamped alarms are similar to digital alarms, except that a counter variable is used to provide an external time stamp of when a triggering condition occurs, rather than the poll time. Time stamped alarms can only be associated with a single digital variable.</p> <p>See <a href="#">Add a Time Stamped Alarm</a>.</p>
Analog alarms	<p>Analog alarms are triggered when an analog variable changes beyond one or more specified limits. Each alarm can be configured as any combination of the following types:</p> <ul style="list-style-type: none"> <li>• <b>High</b> and <b>high high</b> alarms - where the value reaches an atypical high</li> <li>• <b>Low</b> and <b>low low</b> alarms - where the value reaches an atypical low</li> <li>• <b>Deviation</b> alarm - where the values move away from a predefined set point</li> </ul>

Alarm Type	Description
	<ul style="list-style-type: none"> <li>• <b>Rate of change</b> alarm - where a dramatic value change occurs within a specified period of time.</li> </ul> <p>See <a href="#">Add an Analog Alarm</a>.</p>
Advanced alarms	<p>An advanced alarm activates when the result of an associated expression changes to true.</p> <p>See <a href="#">Add an Advanced Alarm</a>.</p>
Multi-digital alarms	<p>Multi-digital alarms use the output for up to three digital variables to define eight states. These states represent different combination of true and false values. You can specify which of these states will trigger an alarm.</p> <p>The following eight states represent the possible tag value combinations. The tags are represented in the order tag C, tag B, tag A.</p> <ul style="list-style-type: none"> <li>• State 000 - all 3 tags are false.</li> <li>• State 00A - tags C and B are false, tag A is true.</li> <li>• State 0B0 - tags C and A are false, tag B is true.</li> <li>• State 0BA - tag C is false, tags B and A are true.</li> <li>• State C00 - tag C is true, and tags B and A are false.</li> <li>• State COA - tags C and A are true, tag B is false.</li> <li>• State CB0 - tags C and B are true, tag A is false.</li> <li>• State CBA - all 3 tags are true.</li> </ul> <p>You can indicate whether a particular state will trigger an alarm by applying a value of 1 (indicates an alarm state) or 0 (indicates no alarm will be triggered).</p> <p>See <a href="#">Add a Multi-Digital Alarm</a>.</p>
Time stamped digital alarms	<p>Time stamped digital alarms operate via a process where the alarm server is notified if a value changes to a specified digital variable using the Cicode function <a href="#">AlarmNotifyVarChange</a>.</p> <p>The alarm server uses this information to update alarms that monitor the variable, allowing an accurate time stamp to be associated with an alarm condition.</p> <p>Time stamped digital alarms activate in response to the state of one or two digital variable tags. When you specify the variable tags, you can precede a tag name</p>

Alarm Type	Description
	<p>with the logical operator "NOT". This specifies that the OFF state (0) is the triggering condition.</p> <p>Events trends can be used in conjunction with time stamped digital alarms to provide millisecond accuracy for both trend and alarm data. See the <a href="#">TrnEventSetTable</a> and <a href="#">TrnEventSetTableMS</a> Cicode functions.</p> <p>See <a href="#">Add a Time Stamped Digital Alarm</a>.</p> <p><b>Note:</b> When a time stamped alarm is used in conjunction with some protocols (such as OPC, OFSOPC and DNP3), a time stamp will be pushed from a tag into an alarm without the need to use <code>AlarmNotifyVarChange</code>. See <a href="#">Retrieving Time-stamped Data from I/O Devices</a>.</p>
Time stamped analog alarms	<p>Time stamped analog alarms operate via a process where the alarm server is notified of any value changes to a specified variable using the Cicode function <code>AlarmNotifyVarChange</code>.</p> <p>The alarm server uses this information to update the alarms that monitor the variable, allowing an accurate time stamp to be associated with an alarm condition.</p> <p>Time stamped analog alarms are triggered when an analog variable changes beyond one or more specific limits. Each alarm can be configured as any combination of the following types:</p> <ul style="list-style-type: none"> <li>• <b>High</b> and <b>high high</b> alarms - where the value reaches an atypical high</li> <li>• <b>Low</b> and <b>low low</b> alarms - where the value reaches an atypical low</li> <li>• <b>Deviation</b> alarm - where the values moves away from a predefined set point</li> <li>• <b>Rate of change</b> alarm - where a dramatic value change occurs within a specified period of time.</li> </ul> <p>Events trends can be used in conjunction with time stamped analog alarms to provide millisecond accuracy for both trend and alarm data. See the <a href="#">TrnEventSetTable</a> and <a href="#">TrnEventSetTableMS</a> Cicode functions.</p> <p>See <a href="#">Add a Time Stamped Analog Alarm</a>.</p> <p><b>Note:</b> When a time stamped alarm is used in conjunction with some protocols (such as OPC,</p>

Alarm Type	Description
	OFSOPC and DNP3), a time stamp will be pushed from a tag into an alarm without the need to use AlarmNotifyVarChange. See <a href="#">Retrieving Time-stamped Data from I/O Devices</a> .
Double point status alarms	<p>A double point status alarm responds to eight states in a field device that are represented in Plant SCADA as a single integer tag.</p> <p>For each of the eight states you can specify the following:</p> <ul style="list-style-type: none"> <li>• <b>State Name &lt;n&gt;</b> — a name that identifies the condition that the state represents.</li> <li>• <b>State Type &lt;n&gt;</b> — the action that occurs when the device transitions to a defined state.</li> </ul> <p>See <a href="#">Add a Double Point Status Alarm</a>.</p> <p><b>Note:</b> When a double point status alarm is used in conjunction with some protocols (such as OPC UA, IEC870IP or DNP3), a time stamp will be pushed from a tag into an alarm (see <a href="#">Retrieving Time-stamped Data from I/O Devices</a>). Alternatively, you can trigger a double point status alarm using the Cicode function <a href="#">AlarmNotifyVarChange</a>.</p>

For more information on adding defined alarms to a Plant SCADA project, see [Configure Alarms](#).

## Hardware Alarms

Hardware alarms notify an operator of circumstances such as:

- Interrupted communication to a device
- An inoperative server
- Cicode not executing.

Hardware alarms are fully integrated in Plant SCADA, which means there is no need to manually configure them in a project. However, you need to carefully consider how they are reported to an operator.

If your project is based on one of the Plant SCADA starter projects, a hardware alarm count can be viewed on the alarm banner, and a list is displayed on the dedicated hardware alarms page.

---

**Note:** Do not allow your system to have any recurring hardware alarms.

There are two hardware alarm fields that are not always shown on the hardware alarms pages. ERRPAGE will display the name of the page that was displayed when the error was detected. This is useful for finding errors caused by improperly programmed animations. ERRDESC provides information that is specific to the type of the alarm. For example, if the alarm is an I/O Device error, ERRDESC shows the name of the device.

---

**Note:** A timeout applies on hardware alarms, so that old hardware alarms will be inactive as per [\[Alarm\]HardHoldTime](#) parameter. Unlike configured alarms, all inactive hardware alarms will be removed immediately.

---

## Hardware Alarms for Certificates

The Hardware Alarms page displays hardware alarms when a secure connection cannot be established due to certificates. Some scenarios where a hardware alarm may be generated include the following.

- The binding certificate is inaccessible because of incorrect permissions.
- The chain of trust to the binding certificate is not valid.
- A client cannot trust the certificate with which a server's connections is signed.
- A certificate's common name check is unsuccessful or other policy errors are encountered.

To rectify these alarms, check the error logs and if required contact your network administrator. See [Troubleshooting Certificate Error Messages](#).

## Alarm Server Process

An alarm server process is responsible for evaluating the conditions that define an alarm.

The I/O server sends a notification to the alarm server process each time the value changes for a variable tag that is associated with an alarm. If a notification results in an alarm condition occurring, the alarm server process will generate an alarm.

An alarm server process evaluates the notifications it receives from an I/O server at a rate that is double that used by the I/O server to poll the variable tags (that is, the rate set by the Citect.ini parameter [\[Alarm\]ScanTime](#)).

For information on how to add an alarm server process to a project, see the topic [Add an Alarm Server Process](#).

A Plant SCADA system can have more than one alarm server process. Multiple alarm server processes may be required in the following circumstances:

- Redundant systems – alarm servers can be paired and synchronized so alarm data remains online if a primary server becomes inactive (see [Alarm Server Redundancy](#) in the chapter on [Redundancy](#))
- Clustered systems – multiple alarm servers can be added to a distributed system that includes multiple clusters (see [Clusters](#)).

If required, you can also configure an alarm server to operate in **Extended Memory** mode. This allows the alarm server process to utilize memory beyond a 4GB limit. It does this by running the process in 64 bit.

---

**Note:** When operating in Extended Memory mode, an alarm server process does not support the following:

- ActiveX objects
  - Calls from CitectVBA
  - Cicode functions or alarm expressions that call a DLL compiled in 32 bit.
- 

Extended Memory mode enables efficient query handling under the following circumstances:

- On systems with a large archive of historical alarm data
- On high-capacity systems that can generate a large number of alarms.

To enable this mode of operation, you need to set the alarm server's **Extended Memory** property to TRUE (see [Add an Alarm Server Process](#)). You also need to recompile your project and restart the runtime system for the change to take effect.

## See Also

[Manage Alarms](#)

## Viewing Alarms

If you create a project using one of Plant SCADA's starter projects, the following components will be available to monitor alarms at runtime.

- [Alarm Pages](#)
- [Alarm Banner](#)
- [Alarm Equipment Tree](#).

These components are automatically added to a project as they are included in the templates on which a starter project is based. This is the case for both the Tab Style Templates and the SxW Style Templates.

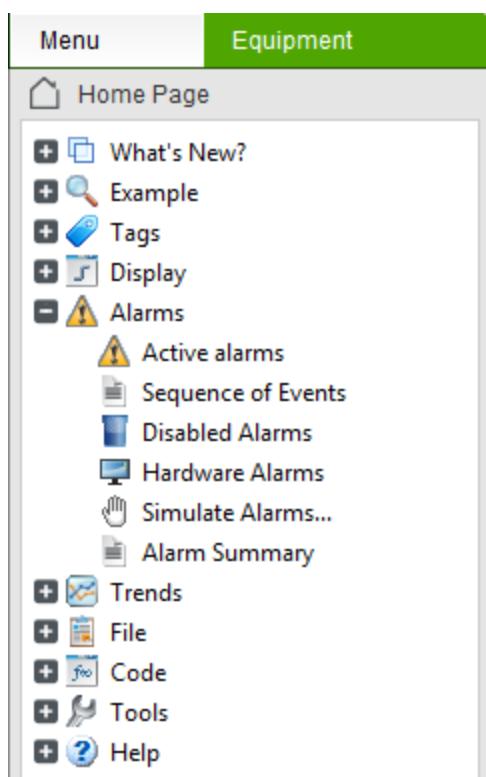
## Alarm Pages

Alarm pages present a list of alarms to an operator. There are different types of alarm pages, each designed to suit a specific purpose. For example, an alarm page may only display alarms that are currently active, or it may include a historical summary of all alarms that have occurred.

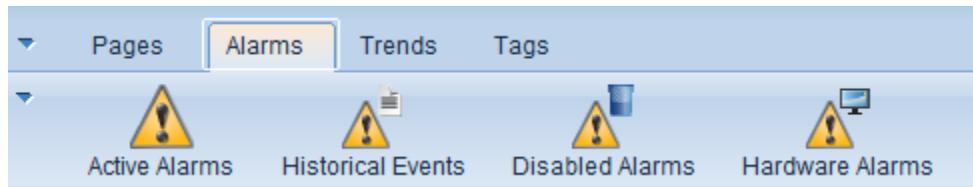
---

**Note:** For information about the alarm pages that are available in a project based on a Situational Awareness Starter Project, see the topic [Default Alarm Pages](#).

Viewing alarm pages at runtime varies according to the templates used to create the alarm pages. For example, if your project uses the SxW templates the Alarm pages are automatically included in the runtime system available from the list of alarm pages in the **Menu** panel.



If your project is based on a Tab Style starter project, you can access the Alarms page at runtime via the **Alarms** tab.



The following alarm pages will be available at runtime:

## Active Alarms Page

The Active Alarms page displays all alarms that are unacknowledged, or acknowledged and still in an alarm state. The active state of a digital alarm is ON, while the active state of an analog alarm varies, depending on how the alarm is configured. For example, the alarm state could be HIGH, LOW, or so on.

By default, the following columns are displayed:

- **Date** — the day on which the alarm occurred
- **Time** — the time at which the alarm occurred
- **Tag** — the tag associated with the alarm (if applicable)
- **Name** — the name of the tag associated with the alarm (if applicable)
- **State** — the current state of the tag associated with the alarm
- **User Comment** — a comment attached to the alarm at runtime (see [Add a Comment to the SOE Page](#)).

You can also include additional columns that display information such as specific alarm property values or quality

flags (see [Add a Column to an Alarms List](#)).

When an alarm is selected, you can perform the following actions:

- **Acknowledge** the alarm (see [Acknowledge Alarms](#))
- **Disable** the alarm (see [Disable Alarms](#))
- **Disable for** a specified number of hours (see [Shelve Alarms](#))
- **Disable until** a specified date and time (see [Shelve Alarms](#))
- View **Information** about the alarm.

## Hardware Alarms Page

The Hardware Alarms page displays a list of any hardware alarms that are unacknowledged, or acknowledged and still in alarm state.

The following columns are displayed:

- **Tag** — the tag associated with the alarm (if applicable)
- **Name** — the name of the tag associated with the alarm (if applicable)
- **Description** — a description of the hardware alarm
- **Source** — a description of the location where the error originated . For example, it could be the name of a page or a Cicode function.
- **Error Details** — Additional information that is specific to the error.

When a hardware alarm is selected, you can perform the following actions:

- **Acknowledge** the alarm (see [Acknowledge Alarms](#)).

## Disabled Alarms Page

The Disabled Alarms page displays a list of alarms that are disabled in the system for reasons such as a maintenance shutdown.

By default, the following columns are displayed:

- **Date** — the day on which the alarm occurred
- **Time** — the time at which the alarm occurred
- **Tag** — the tag associated with the alarm (if applicable)
- **Name** — the name of the tag associated with the alarm (if applicable)
- **State** — the current state of the tag associated with the alarm
- **User Comment** — a comment attached to the alarm at runtime (see [Add a Comment to the SOE Page](#)).

You can also include additional columns that display information such as specific alarm property values or quality flags (see [Add a Column to an Alarms List](#)).

When an alarm is selected, you can perform the following actions:

- **Enable** the alarm (see [Disable Alarms](#))

- View [Information](#) about the alarm.

## Alarm Summary Page

The Alarm Summary page displays a historical log of alarms that you can use for long term analysis and troubleshooting. It displays the date and time each alarm reached ON state and OFF state.

**Note:** The alarm summary includes historical data, which means you should manually refresh the page to view the latest events before you analyze the data it includes.

By default, the following columns are displayed:

- **Tag** — the tag associated with the alarm (if applicable)
- **Name** — the name of the tag associated with the alarm (if applicable)
- **On Date** — the day on which the alarm reached an ON state
- **On Time** — the time at which the alarm reached an ON state
- **Off Date** — the day on which the alarm reached an OFF state
- **Off Time** — the time at which the alarm reached an OFF state
- **Delta Time** — the difference between OFF time and ON time
- **Sum State** — the state of the tag associated with the alarm when it occurred
- **User Comment** — a comment attached to the alarm at runtime (see [Add a Comment to the SOE Page](#)).

You can also include additional columns that display information such as specific alarm property values or quality flags (see [Add a Column to an Alarms List](#)).

When an alarm is selected, you can perform the following actions:

- **Acknowledge** the alarm (see [Acknowledge Alarms](#))
- Add a **Comment**
- **Disable** the alarm (see [Disable Alarms](#))
- **Disable for** a specified number of hours (see [Shelve Alarms](#))
- **Disable until** a specified date and time (see [Shelve Alarms](#))
- **Enable** a disabled alarm (see [Enable a Disabled Alarm](#))
- View [Information](#) about the alarm.

## Sequence of Events (SOE) Page

The Sequence of Events (SOE) page displays a historical view of all system events that have occurred. In the case of alarms, an event is logged when an alarm changes state, or when an alarm is enabled or disabled.

By default, the following columns are displayed:

- **Date** — the day on which the event occurred
- **Time** — the time at which the event occurred
- **Tag** — the tag associated with the event (if applicable)
- **Name** — the name of the tag associated with the event (if applicable)

- **Message** — a message describing the event
- **State** — the current state of the tag associated with the event
- **Classification** — the type of event (see Events)
- **User Name** — the name of the user that initiated the event (including "System")
- **User Location** — the IP address or computer name of the client computer that generated the event.

You can also include additional columns that display information such as specific alarm property values or quality flags (see Add a Column to an Alarms List).

**Note:** You can only use the **Date**, **Time** and **Milliseconds** columns to sort the SOE page (see [Sort Alarms](#)).

When an event is selected, you can perform the following actions:

- Manually add an event (see Add an Event to the SOE page)
- Add a comment to an event (see Add a Comment to the SOE page)
- View **Information** about the alarm.

Information included on the alarm pages is presented in a table. This allows you to make simple runtime changes to the page, as you can adjust, rearrange and remove columns (see [Add a Column to an Alarms List](#)).

Each alarm page supports a variety of built in features that an operator can use at runtime. For example, if you used a starter project an operator will be able to perform the following tasks:

- Acknowledge alarms
- Enable and disable alarms
- Shelve alarms
- Filter an alarms list
- Sort an alarms list
- Change the fields displayed in an alarms list
- View a count of the active and disabled alarm on all pages
- Add user events to a sequence of events.

For more information, see [Action Alarms at Runtime](#).

## See Also

[Alarm Banner](#)

[Alarm Equipment Tree](#)

### Alarm Banner

An alarm banner is a reduced version of the active alarms page that displays three alarms. If you use a starter project as the foundation of your Plant SCADA project, the alarm banner is displayed at the top of the runtime view for the system.

	10/04/2014 04:30:00 PM PLT_LINE1_PM800\MSSTAT1\MaxW	Plant Mixer Peak Demand N...
	10/04/2014 04:30:00 PM PLT_LINE2_PM800\MSSTAT1\MaxW	Plant Bottler Peak Demand N...
	10/04/2014 04:25:46 PM PLT_LINE2_PM800\PDT_MSSTAT1\AvW	Plant Bottler Predicted Peak...

The alarm banner that is included in the starter project based on the SxW templates.

This means the alarm banner appears as a permanent feature of the runtime interface, allowing an operator to be alerted to alarm conditions even if an alarm page is not displayed.

The alarm banner also includes alarm counts that summarize the current activity for a number of alarm types including active alarms, unacknowledged alarms and disabled alarms. Alarm counts are created using the Cicode function `AlarmCount`.

	20	22/04/2014 03:27:50 PM PLT_LINE2_PM800\PDT_MSSTAT1\AvW
	25	11/04/2014 02:15:00 PM PLT_LINE1_PM800\MSSTAT1\MaxW
	3	22/04/2014 03:28:50 PM Entry_Fault_Multi

Alarm counts.

The Alarm Banner allows an operator to perform the following actions at runtime:

- Acknowledge an alarm
- Disable an alarm
- Display information about an alarm.

For more information, see [Action Alarms at Runtime](#).

## See Also

[Alarm Pages](#)

[Alarm Equipment Tree](#)

## Alarm Equipment Tree

If you use a starter project as the foundation of your Plant SCADA project, an alarms equipment tree will automatically be included on the following alarm pages:

- Active Alarms
- Sequence of Events
- Disabled Alarms
- Alarms Summary.

The equipment tree displays in a panel to the left of these alarm page. It allows you to locate the alarms listed on the current page within the system's equipment hierarchy.

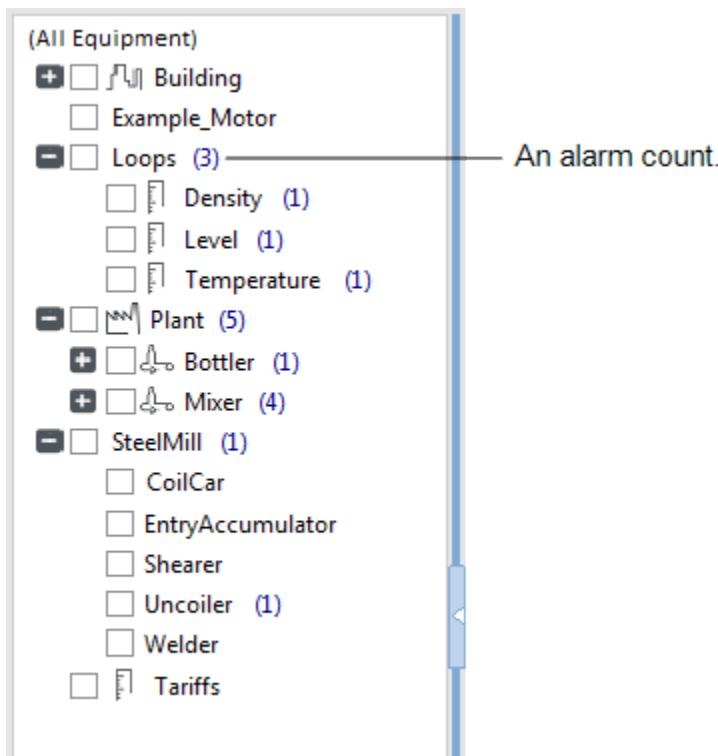
On the Active Alarms and Disabled Alarms pages, an alarm count is applied to each branch of the equipment

tree. If a value appears next to a branch, it indicates the number of alarms from the current list that are included within the branch.

If you would like the alarm count to include alarms defined for the equipment and the equipment it references set the Including Reference.parameter to 1 when configuring the AlarmCountEquipment Cicode function.

**Note:** The Alarm Summary page will not list the alarms belonging to referenced equipment.

The example below shows the equipment tree on the Active Alarms page.



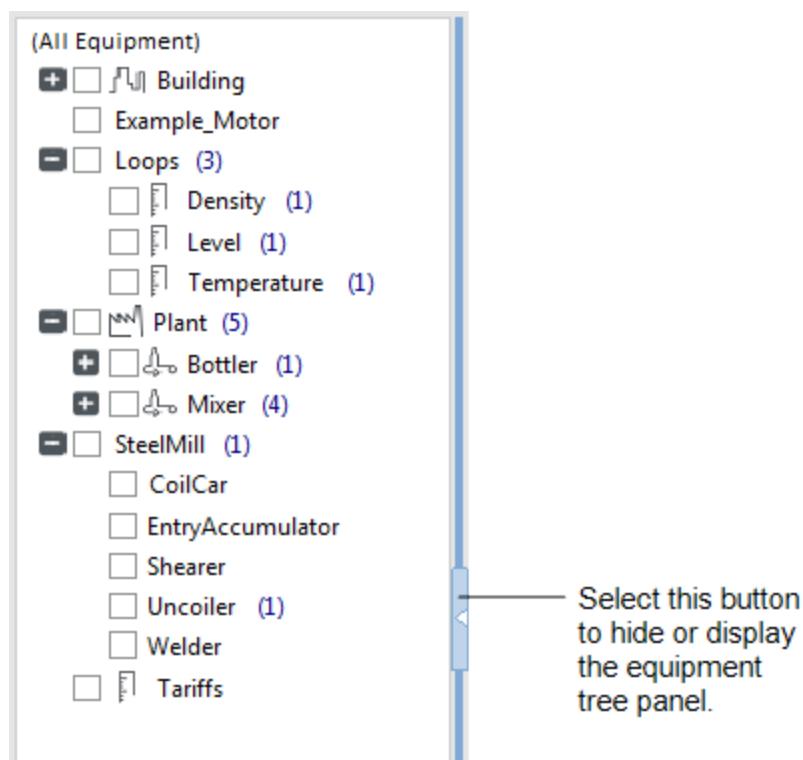
Observe that the alarm count on the "Loops" branch is displaying the value "3", which represents the total number of the alarms included within the branch.

**Note:** The Alarm Summary and Sequence of Events pages will not include alarm count information on the equipment tree as they present historical data.

You can create your own alarm counts using the Cicode function AlarmCount.

The alarm equipment tree allows an operator to perform the following actions at runtime:

- Filter the alarms list by selecting the check boxes to the left of each branch
- Hide and display the equipment tree panel.



Select this button  
to hide or display  
the equipment  
tree panel.

## See Also

[Alarm Pages](#)

[Alarm Banner](#)

## Events

Events provide notification of occurrences within a Plant SCADA system that do not require any action to be taken. Examples of such occurrences could include a particular user logging on, or an alarm being acknowledged.

The events that are generated by a Plant SCADA system are grouped according to the following classifications:

- **Action** — events that log an action performed on an alarm by an operator
- **<Alarm Type>** — events associated with a particular type of alarm (for example, ON, OFF and ACK transitions)
- **System** — events that represent an occurrence that affects the entire system, such as the server being stopped and restarted
- **Configuration** — events that occur when configuration changes are loaded (for example, alarm tags being added or removed)
- **Comment** — events that indicate when a user adds a comment to the sequence of events (see [Add a Comment to the SOE Page](#))
- **Interface Events** — events that indicate when a user adds an event to the sequence of events (see [Add an Event to the SOE Page](#)).

Events can be viewed at runtime via the Sequence of Events (SOE) page (see [Alarm Pages](#)).

---

**Note:** Plant SCADA also supports triggering events that allow you to initiate a particular action when a specified set of circumstances occurs. For more information on this type of event, see [Triggering Events](#).

---

## Manage Alarms

You should carefully consider alarm management when you configure a Plant SCADA project, as it could determine if an operator is able to respond to all the alarms that a system generates.

When an operator receives an alarm notification, there are several actions they may need to perform. These include:

- Acknowledge the alarm
- Navigate to the relevant system data
- Analyze the data
- Perform the required actions
- Confirm that the alarm condition is resolved.

If a system generates alarm notifications at a rate that does not allow the operator enough time to perform these tasks, it raises the possibility of alarms being missed.

There are guidelines on the number of alarms which can be successfully managed by an operator. For example, the ISA 18.2 Alarm Management standard provides metrics regarding manageable average alarm rates. It suggests that an average of one alarm per ten minutes is very likely to be acceptable, with a maximum manageable level of two alarms per ten minutes.

The following topics include information you can use to keep the alarms in your system at a manageable level:

- [Prioritize Alarms](#)
- [Categorize Alarms](#)
- [Provide Cause and Response Information to Operators](#)
- [Use Alarm Indicators](#)
- [Set the Alarm Scan Rate](#).

If required, you can also configure an alarm server to operate in **Extended Memory** mode. This allows the alarm server process to utilize memory beyond a 4GB limit. For more information, see [Alarm Server Process](#).

### Prioritize Alarms

During periods of high activity, a control system can generate a large number of alarms within a short period of time. If you prioritize the alarms in your Plant SCADA system, you can indicate to an operator which alarms they should address first.

There are two things to consider when you plan alarm priorities:

- The severity of the consequence that could occur if the operator does not take the appropriate corrective action to the alarm
- The amount of time that the operator has available, compared to the amount of time that it would take to perform a corrective action.

Alarm management standards provide guidelines on how to effectively prioritize alarms in a control system. For example, the EEUMA 191 standard recommends that three alarm priorities should be used within one alarm display.

However, if your control system incorporates more than one alarm system (for example, if your site includes a separate fire or gas alarm panel), it is recommended that the alarm priorities you use are consistent with any existing systems.

Alarm priorities are managed in Plant SCADA through alarm categories. Each alarm category can be assigned a priority value (from 1 to 255), which can be used to determine the order in which alarms are displayed, acknowledged and enabled.

If required, you can configure optional display properties for an alarm priority that will allow you to:

- Specify a name for a priority value to provide a meaningful representation of its purpose.
- Define background and foreground colors that provide a visual representation of priority on a graphics page.
- Associate Genies with a priority that you can use as an alarm flag or icon.

These additional properties are configured in the **Setup** activity (see [Configure Display Properties for an Alarm Priority](#)).

You can also [Create Priority and State Symbols for an Alarms List](#).

## See Also

[Categorize Alarms](#)

### Categorize Alarms

Alarm categories allow you to group the alarms in your Plant SCADA system. They are a key part of an alarm management strategy, as the priorities that are applied to alarms are implemented through the alarm categories you create (see [Prioritize Alarms](#)).

When you create an alarm category, you can define the following settings for all the alarms that are added to the category:

- The priority applied to an alarm
- The font that is used when an alarm displays
- The formatting that is used when an alarm displays
- The action that is triggered (via a Cicode expression) when an alarm reaches a specified status (on, off or acknowledged)
- The logging that occurs for an alarm.

Alarms can also be filtered and sorted based on their category.

A Plant SCADA project can support up to 16376 alarm categories. Each individual alarm category is represented by a numeric value. There are three special cases:

- Category 0 is reserved as a default category
- Category 254 is reserved for user-created alarm summary entries
- Category 255 is reserved for hardware alarms.

You can also create a label to refer to an alarm category. This allows you use a meaningful name to identify the category instead of a numeric value. When you define a label for an alarm category, the label's **Expression** field should include the numeric value associated with the category you would like the label to represent. For more information, see [Define a Label](#).

## See Also

[Add an Alarm Category](#)

## Provide Cause and Response Information to Operators

Plant SCADA allows you to associate cause and response information with an alarm tag. This means you can describe the circumstances that are likely to cause an alarm, and the appropriate course of action required to address the alarm. This information can then be presented to an operator at runtime.

The properties you can associate with an alarm tag include:

- **Cause** — a description of the cause of an alarm.
- **Response** — a description of the appropriate response to an alarm.
- **Response Time** — the period of time in which the specified response needs to be acted upon.
- **Consequence** — a description of the likely outcome if the suggested response is not acted upon within the specified response time.

You can configure up to eight sets of cause and response properties for each alarm tag.

This information is presented to an operator at runtime using the following Cicode functions:

- [AlarmGetDsp](#)
- [AlarmGetFieldRec](#)
- [AlmBrowseGetField](#).

For example, you could use the following Cicode to retrieve cause and response information from a displayed alarm list:

```
// Example of getting alarm responses from an Animation Number of a displayed alarm list
FUNCTION GetAlarmResponsesForAN(INT AN)
    INT i;
    STRING cause, response, time, consequence;
    INT Count = AlarmGetDsp(AN, "ResponseNum");
    FOR i = 1 to Count DO
        cause = AlarmGetDsp(AN, "Cause" + IntToStr(i));
        response = AlarmGetDsp(AN, "Response" + IntToStr(i));
        time = AlarmGetDsp(AN, "RespTime" + IntToStr(i));
        consequence = AlarmGetDsp(AN, "Consequence" + IntToStr(i));
        // DO something with the retrieved fields
        // ...
    END
END
```

If your project is based on the SxW or Tab Style templates, you can also view cause and response information at runtime via the Cause and Response dialog. See [Display Cause and Response Information](#).

To specify cause and response information for an alarm tag, you need to use the **Alarming** view in the **Setup** activity. See [Add Cause and Response Information to Alarms](#).

## Use Alarm Indicators

Alarm indicators consist of a border and flag that appear around the extent of an object group or Genie. You can use them to provide a prominent visual indication of alarm occurrences at runtime.



### Alarm border

An alarm border will appear around an object group or Genie when an associated alarm state occurs.

### Alarm flag

An alarm flag provides additional information about an alarm by presenting:

- a flag color
- a flag shape
- a flag code that can indicate the alarm priority or the alarm type.

The example below shows the shape, color and flag codes that are used to indicate emergency alarms, high priority alarms and low priority alarms in a project created from the Situational Awareness Starter Project.



If you need to configure your own alarm flags, see [Create a Custom Flag for an Alarm Indicator](#).

As well as using color to indicate the priority of an alarm, the appearance of an alarm indicator can be used to determine the current state of an alarm. For example, if an alarm is unacknowledged, the indicator will flash. For more information, see [Alarm States](#).

You can position an alarm flag at a number of locations around the outside of an object's alarm border. To indicate a specific alarm state, an alarm indicator may flash or display a lighter version of a color.

To configure an alarm indicator for an object group or Genie, you need to perform the following tasks:

1. Confirm that the required alarm priorities are assigned to an alarm category. See [Categorize Alarms](#).
2. Define the display properties for the alarm priorities that an indicator will represent. See [Configure Display Properties for an Alarm Priority](#).

3. For each object group or Genie, define the Alarm Indicator properties in Graphics Builder. See [Alarm Indicator](#).

## See Also

[Prioritize Alarms](#)

## Alarm States

Five alarm states can be represented by an alarm indicator:

1. **Normal** — no alarms are active.
2. **On Unacknowledged** — an active alarm has not been acknowledged.
3. **On Acknowledged** — an active alarm has been acknowledged.
4. **Off Unacknowledged** — the object has gone in and out of an alarm state without being acknowledged.
5. **Shelved** — the alarm is currently shelved (which means it is disabled for a specified period of time).

These five state are demonstrated in the following table. The examples show the default appearance of emergency priority alarms in a project based on the Situational Awareness Starter Project. For other priorities, the color and flag shape would be different (see [Use Alarm Indicators](#)).

State	Example	Behavior
Normal		<ul style="list-style-type: none"><li>• No alarms are active.</li></ul>
On Unacknowledged		<ul style="list-style-type: none"><li>• The alarm border and flag are shown around the object.</li><li>• The border and flag both flash between colored and white fill.</li></ul>
On Acknowledged		<ul style="list-style-type: none"><li>• The alarm border remains around the object, but is filled with half-intensity color.</li><li>• The alarm flag retains its usual appearance.</li><li>• There is no flashing for either element.</li><li>• If the alarm condition clears, the object returns to normal state.</li></ul>

State	Example	Behavior
Off Unacknowledged		<ul style="list-style-type: none"> <li>The alarm border appears around the object filled with half-intensity color.</li> <li>The alarm flag retains its usual appearance.</li> <li>The alarm flag flashes.</li> <li>The alarm border does not flash.</li> </ul>
Shelved		<ul style="list-style-type: none"> <li>A white alarm border is shown with a generic alarm flag symbol; this is shown while alarms are shelved for the object (that is, temporarily disabled).</li> </ul>

## See Also

[Create a Custom Flag for an Alarm Indicator](#)

### Create a Custom Flag for an Alarm Indicator

An [Use Alarm Indicators](#) includes a flag that can be used to reflect the priority and current state of the most critical active alarm.

By default, this flag is constructed from the following set of Genies:

- One Genie for each alarm priority that is flagged
- One Genie for the Disabled/Shelved alarm mode.

To create a custom flag for an alarm indicator, you need to create the required Genies and assign them to the associated alarm priority or alarm mode.

#### Create a Genie to use as a flag

The Genie requires a root level object named "Shape".

- In Graphics Builder, create a new Genie.
- Add the object you would like use as an alarm flag.
- Display the Properties dialog for the object.
- On the **Appearance | General** tab, select **Filled**.

**Note:** If you are configuring a Genie for the Disabled/Shelved alarm mode, set the fill color you would like to use on the **Appearance | General** tab. You can then skip step 5 as the Genie will not need to change color.

5. On the **Fill | Color** tab, configure the following:
  - Set the **Type** to **Array**.
  - Enter the following **Array Expression**:

```
 dspAnGetMetaData(dspGetAnCur(), "State")
```
  - Specify the **Array Colors** you want to use for 1 – 3, based to the following states:
    - 1 = On and Unacknowledged
    - 2 = Off and Unacknowledged
    - 3 = On and AcknowledgedYou can configure flashing colors via the [Edit Favorite Colors Dialog Box](#), which is accessible via the **Edit** button on the Color Picker.
6. On the **Access | General** tab, enter the **Name** "Shape".
7. On the **Metadata | General** tab, enter the following items:
  - Name: Equipment; Value: <blank>
  - Name: Label; Value: <blank>
  - Name: State; Value: 0

---

**Note:** If you are configuring a Genie for the Disabled/Shelved alarm mode, replace "Name: State; Value: 0" with "Name: Priority; Value: <blank>".
8. Click **OK**.

Alarm flag Genies use functionality provided by the Cicode function DspSym. You need to perform the remaining steps to enable DspSym support for the Genie.

1. Display the Page Properties for the Genie. To do this:

On the **File** menu, select **Properties**.

Or:

Right-click on the Genie background and select **Page Properties**.

2. On the Page Properties dialog, select **I want to use this Genie with the Cicode function DspSym**.
  3. Click **OK**.
  4. Save the Genie to a library.
- 

**Note:** When saving an alarm flag Genie, use a name that is unique across all included projects that share the same primary project.

## Add a text label

The Genie can also include an optional text label. The procedure described below adds a text label that displays the **Short Name** specified for an Alarm Priority.

1. Add a text object to an alarm flag Genie.
2. Display the Text Properties dialog.
3. On the **Appearance | Display Value** tab, configure the following:
  - Set the **Type** to **Numeric**. This will allow you to specify the number of characters used for the text label.
  - Select the required **Format** for the label.

This should reflect the number of characters used for the **Short Name** specified in the Alarm Priority properties.

- Enter the following **Numeric Expression**:

```
dspGetMetaDataFromName( ".Shape", "Label" )
```

4. On the **Fill|Color** tab, configure the following:

- Set the **Type** to **Array**.

- Enter the following **Array Expression**:

```
dspGetMetaDataFromName( ".Shape", "State" )
```

- Specify the **Array Colors** you want to use for 1 – 3, based to the following states:

1 = On and Unacknowledged

2 = Off and Unacknowledged

3 = On and Acknowledged

5. Click **OK**.

6. Save the Genie.

### Associate a Genie with an alarm priority

1. In Plant SCADA Studio's **Setup** activity, select **Alarming**.
2. On the menu below the Command Bar, select **Alarm Priorities**.
3. Locate the Alarm Priority you would like to associate with a Genie.
4. In the **Genie Name** field, enter the name of the Genie you would like to associate with the alarm priority.
5. In the **Library Name** field, enter the name of the library that includes the Genie.
6. Click **Save**.

### Associate a Genie with the Shelved/Disabled alarm mode

1. In Plant SCADA Studio's **Setup** activity, select **Alarming**.
2. On the menu below the Command Bar, select **Alarm Modes**.
3. Locate the Shelved/Disabled mode in the **Display Name** column.
4. In the **Genie Name** field, enter the name of the Genie you would like to associate with the alarm mode.
5. In the **Library Name** field, enter the name of the library that includes the Genie.
6. Click **Save**.

### Example

The default alarm flag Genies that are provided with Plant SCADA are located in the SA\_Include project in a library called "sa\_priorities". They are named as follows:

- Priority 1 – "sa\_p1\_normal"
- Priority 2 – "sa\_p2\_normal"
- Priority 3 – sa\_p3\_normal"
- Disabled/Shelved – "sa\_disabled\_normal".

## See Also

[Use Alarm Indicators](#)

## Set the Alarm Scan Rate

You can use the Citect.ini parameter [\[Alarm\]ScanTime](#) to adjust the rate at which the alarm server receives updates from the I/O server. By default, the rate is set to 500 ms. This means the minimum time between alarm transitions will be 500 ms.

If you increase the scan time setting, it reduces the subscription rate used to poll tags that are associated with alarms. This may reduce the load on the I/O server. Be aware, however, that this will only work if no other servers have subscribed to a device at a higher rate.

## See Also

[Alarm Server Process](#)

## Configure Alarms

You can add alarms to a project that allow specific conditions to be brought to an operator's attention at runtime (see [Alarms](#)).

There are two ways to add configured alarms to a Plant SCADA project:

- Use Equipment Editor to generate alarms that are a part of the hierarchical architecture defined for the equipment in your system (see [Use Equipment Editor to Configure Alarms](#))
- Using the System Model activity in Plant SCADA Studio to add defined alarms (see [Use Plant SCADA Studio to Configure Alarms](#)).

## See Also

[Using Alarm Properties as Tags](#)

## Use Equipment Editor to Configure Alarms

Alarms can be added to a Plant SCADA system using the [Equipment Editor](#). This will incorporate the alarms you create into the hierarchical architecture defined for the equipment in your system.

When you define an equipment type in Equipment Editor, you can associate alarms with the items it includes. These alarms are passed through to each instance of the equipment type, and the required alarm tags are generated when an equipment update is run. Any changes to an equipment type can also be instantiated across all existing instances of the associated equipment.



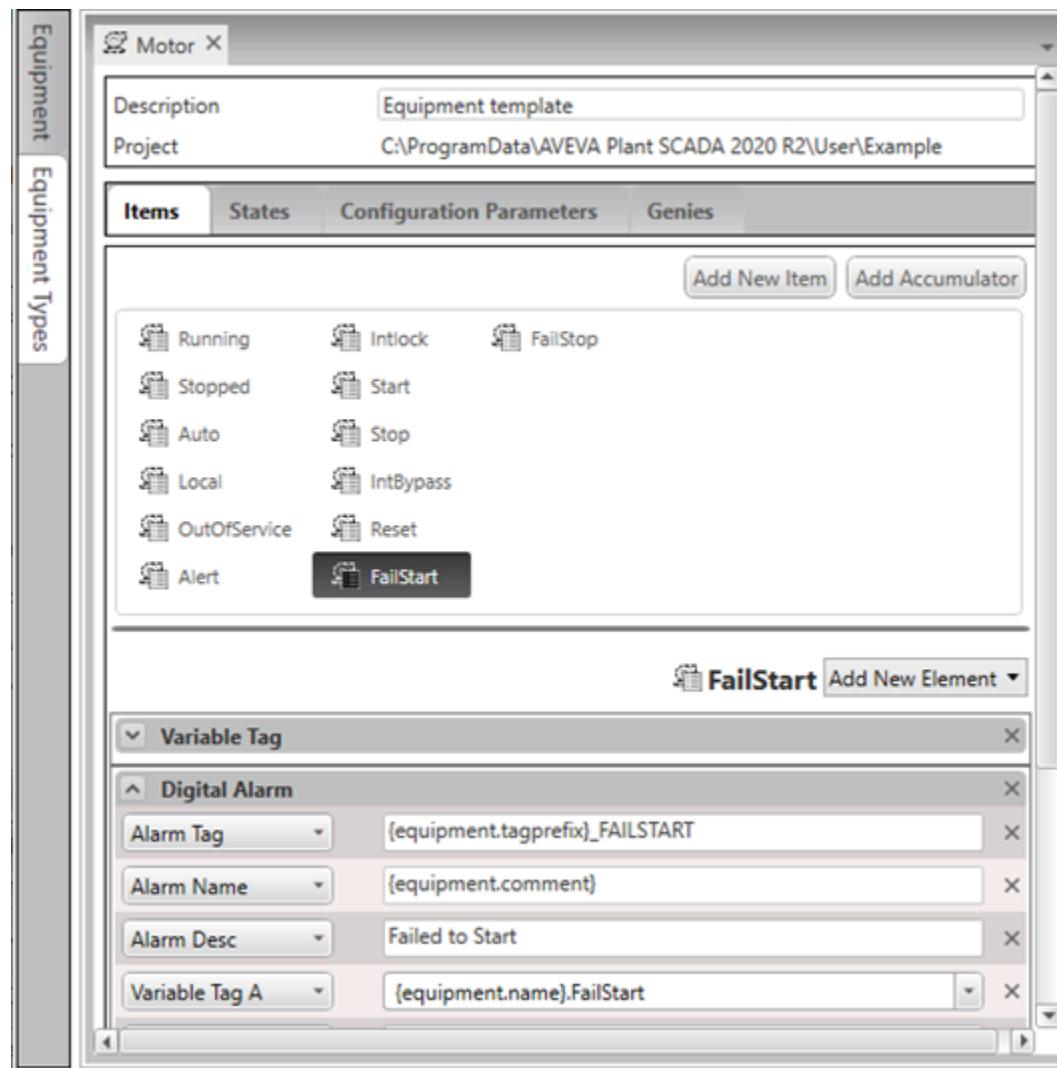
**UNINTENDED EQUIPMENT OPERATION**

On updating the Equipment Name or Equipment Item for an alarm, the update will not be reflected on the client side until the alarm servers have been restarted. The reload procedure will indicate a reload error on Runtime Manager. Reloadlog.dat and Syslog.dat of the server process will show the details of the unsupported changes. You can use this information to revert the changes, or restart all servers after deploying your updated project.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Example

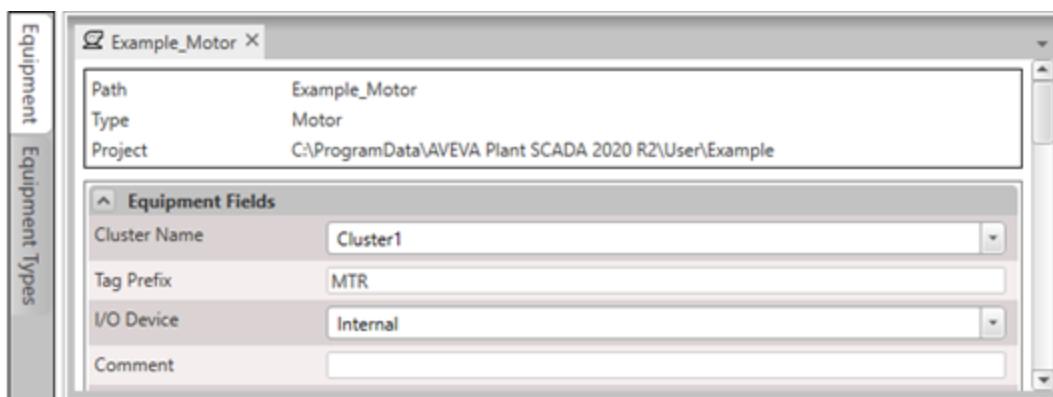
The following example shows the Equipment Editor displaying the configuration for an equipment type named "Motor".



The "FailStart" item includes a digital alarm as one of its elements, and the **Alarm Tag** property is defined as follows:

{equipment.tagprefix}\_FAILSTART

An instance of this equipment type named "Example\_Motor" is created.



The **Tag Prefix** property (which is referenced by the **Alarm Name** property in the associated equipment type) is defined as "MTR". This means a digital alarm called "MTR\_FAILSTART" will be generated when an equipment update is run.

The Digital Alarms grid will display the corresponding alarm record.

System Model		Equipment	Variables	Alarms	Trends	Accumulators	SPC	
		Save	Discard	Copy	Paste	Delete Row(s)	Export All	Import All
<b>Digital Alarms</b>								
Row	Equipment	Item Name	Alarm Tag	Alarm Name	Cluster Name			
71	Plant.Bottler.Filler	Fault	Bottler_Filler_FAULT	Bottler Filler Fault	Cluster1			
72	Plant.Bottler.Capper	Fault	Bottler_Cap_FAULT	Bottler Capper Fault	Cluster1			
73	Plant.Bottler.Labeller	Fault	Bottler_Label_FAULT	Bottler Labelller Fa	Cluster1			
74	Example_Motor	FailStart	MTR_FAILSTART		Cluster1			
75	Example_Motor	FailStop	MTR_FAILSTOP		Cluster1			

Alarm properties generated by the Equipment Editor that are read-only appear as gray text, which indicates that they cannot be manually updated in Plant SCADA Studio.

**Note:** You can also associate defined alarms with equipment using Plant SCADA Studio. See [Associate an Alarm with Equipment](#).

## Use Plant SCADA Studio to Configure Alarms

You can use the System Model activity in Plant SCADA Studio to add alarms to your project. Each alarm record is added using a properties form that reflects a particular alarm type (see [Alarms](#)).

There are properties that are common to all alarm types. These include:

- Alarm Tag (see [Name an Alarm Tag](#))
- Alarm Name (see [Use an Alarm Name to Describe an Alarm Tag](#))
- Category (see [Assign an Alarm to an Alarm Category](#))
- Cluster Name (see [Assign an Alarm to a Cluster](#))
- Equipment properties (see [Associate an Alarm with Equipment](#)).

Each configured alarm type also includes specific properties. For more information, see:

- [Add a Digital Alarm](#)
- [Add a Time Stamped Alarm](#)
- [Add an Analog Alarm](#)
- [Add an Advanced Alarm](#)
- [Add a Multi-Digital Alarm](#)
- [Add a Time Stamped Digital Alarm](#)
- [Add a Time Stamped Analog Alarm](#)
- [Add a Double Point Status Alarm](#)

## See Also

[Use Alarm Delay](#)

[Using Alarm Properties as Tags](#)

### Name an Alarm Tag

When configuring an alarm, you use the **Alarm Tag** property to specify a name for the alarm tag. The name you use needs to meet the following requirements:

1. The tag name needs to use [Tag Name Syntax](#).
2. The tag name needs to be unique within a cluster (see [Assign an Alarm to a Cluster](#)).

If the tag name does not meet these requirements, it may not be recognized.

---

**Note:** If your project has a large number of tags, use a naming convention to easily find and debug your tags (see [Structured Tag Names](#)).

---

#### To name an alarm tag:

1. In the **System Model** activity, select **Alarms**.
2. On the menu below the Command Bar, select the required alarm type. The alarms list will display in the Grid Editor.
3. If you would like to update an existing alarm, locate the required record.
4. In the **Alarm Tag** field displayed in the Property Grid, enter an appropriate name for the alarm tag.
5. Click **Save**.

For more information on how to configure a particular alarm type, select one of the following topics:

- [Add a Digital Alarm](#)
- [Add a Time Stamped Alarm](#)
- [Add an Analog Alarm](#)
- [Add an Advanced Alarm](#)
- [Add a Multi-Digital Alarm](#)

- Add a Time Stamped Digital Alarm
- Add a Time Stamped Analog Alarm

## See Also

[Use an Alarm Name to Describe an Alarm Tag](#)

### Use an Alarm Name to Describe an Alarm Tag

When configuring an alarm, you can use the **Alarm Name** property to include a description of the alarm. For example, you may include the name of the physical device with which the alarm is associated.

The name you enter can be used to identify an alarm when it is displayed on screen or logged to a device.

#### To add an Alarm Name to an alarm tag:

1. In the **System Model** activity, select **Alarms**.
2. On the menu below the Command Bar, select the required alarm type. The alarms list will display in the Grid Editor.
3. If you would like to update an existing alarm, locate the required record.
4. In the **Alarm Name** field displayed in the Property Grid, enter a name that describes the alarm tag.
5. Click **Save**.

For more information on how to configure a particular alarm type, select one of the following topics:

- Add a Digital Alarm
- Add a Time Stamped Alarm
- Add an Analog Alarm
- Add an Advanced Alarm
- Add a Multi-Digital Alarm
- Add a Time Stamped Digital Alarm
- Add a Time Stamped Analog Alarm

## See Also

[Assign an Alarm to an Alarm Category](#)

### Assign an Alarm to an Alarm Category

To assign a configured alarm to an alarm category, you use the **Alarm Category** property (see [Categorize Alarms](#)).

To do this, you would typically enter a numeric value that represents one of the categories defined in your project. However, you can also enter a label (if one has been associated with a particular category).

If a value is not specified in the **Alarm Category** field, the default category zero (0) will be used.

**To assign an alarm to an alarm category:**

1. In the **System Model** activity, select **Alarms**.
2. On the menu below the Command Bar, select the required alarm type. The alarms list will display in the Grid Editor.
3. If you would like to update an existing alarm, locate the required record.
4. In the **Alarm Category** field in the Property Grid, enter a number or label that represents one of the categories defined in your project.
5. Click **Save**.

For more information on how to configure a particular alarm type, select one of the following topics:

- [Add a Digital Alarm](#)
- [Add a Time Stamped Alarm](#)
- [Add an Analog Alarm](#)
- [Add an Advanced Alarm](#)
- [Add a Multi-Digital Alarm](#)
- [Add a Time Stamped Digital Alarm](#)
- [Add a Time Stamped Analog Alarm](#)

**See Also**

[Assign an Alarm to a Cluster](#)

**Assign an Alarm to a Cluster**

When you assign an alarm to a cluster you specify the cluster that will run the alarm. To assign an alarm to a cluster, you use the **Cluster Name** property.

---

**Note:** If your system has more than one cluster, you need to specify a cluster for each alarm.

---

**To assign an alarm to a cluster:**

1. In the **System Model** activity, select **Alarms**.
2. On the menu below the Command Bar, select the required alarm type. The alarms list will display in the Grid Editor.
3. If you would like to update an existing alarm, locate the required record.
4. In the **Cluster Name** field displayed in the Property Grid, enter the name of the cluster that will run the alarm. You can select an existing cluster from the drop-down menu to the right of the field.
5. Click **Save**.

For more information on how to configure a particular alarm type, select one of the following topics:

- [Add a Digital Alarm](#)
- [Add a Time Stamped Alarm](#)
- [Add an Analog Alarm](#)

- Add an Advanced Alarm
- Add a Multi-Digital Alarm
- Add a Time Stamped Digital Alarm
- Add a Time Stamped Analog Alarm

## See Also

[Associate an Alarm with Equipment](#)

### Associate an Alarm with Equipment

To associate an alarm with equipment in Plant SCADA Studio, you use the **Equipment** and **Item Name** properties:

- The **Equipment** property defines the equipment association that you can use to reference the alarm
- The **Item Name** property is associate the alarm with a particular attribute of the equipment.

**Note:** The steps described here are recommended if you are associating an existing alarm tag with equipment. If you are creating new alarm tags, you should use Equipment Editor to manage your equipment associations. For more information, see [Use Equipment Editor to Configure Alarms](#).

#### **WARNING**

##### **UNINTENDED EQUIPMENT OPERATION**

On updating the Equipment Name or Equipment Item for an alarm, the update will not be reflected on the client side until the alarm servers have been restarted. The reload procedure will indicate a reload error on Runtime Manager. Reloadlog.dat and Syslog.dat of the server process will show the details of the unsupported changes. You can use this information to revert the changes, or restart all servers after deploying your updated project.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

### To associate an alarm with equipment:

1. In the **System Model** activity, select **Alarms**.
2. On the menu below the Command Bar, select the required alarm type. The alarms list will display in the Grid Editor.
3. Locate the alarm record you would like to update.
4. In the **Equipment** field displayed in the Property Grid, enter the name of the equipment with which you would like to associate the alarm. You can select an existing equipment definition from the drop-down menu to the right of the field.
5. If required, specify an **Item Name** for the alarm.
6. Click **Save** to save the changes to the selected alarm record.

For more information on how to configure a particular alarm type, select one of the following topics:

- [Add a Digital Alarm](#)

- Add a Time Stamped Alarm
- Add an Analog Alarm
- Add an Advanced Alarm
- Add a Multi-Digital Alarm
- Add a Time Stamped Digital Alarm
- Add a Time Stamped Analog Alarm
- Add a Double Point Status Alarm.

### Add a Digital Alarm

Digital alarms activate in response to the state of one or two digital variables. The alarm becomes active when the triggering condition spans the duration of a specified delay period.

#### To add a digital alarm:

1. In the **System Model** activity, select **Alarms**.
2. On the menu below the Command Bar, select **Digital Alarms**.
3. Add a row to the Grid Editor.
4. Type the required information in each column, or in the Property Grid (see below for a description of the properties).
5. Click **Save**.

**Note:** Configure the [\[Alarm\]UseConfigLimits](#) parameter to force the Plant SCADA alarms server to use digital alarm property values from the RDB, rather than using the values which may be stored in the database file.

### Digital Alarms Properties

**Note:** If an alarm was generated by the Equipment Editor, a number of fields on the properties form will be shaded. To configure these fields, you will need to use Equipment Editor (see [Use Equipment Editor to Configure Alarms](#)).

#### Equipment Properties

Property	Description
<b>Equipment</b>	The name of the equipment associated with the digital alarm. Select a name from the drop-down list of existing equipment definitions, or enter a name. There is a limit of 254 characters across the Equipment and <b>Item Name</b> fields, including any separating periods (.).
<b>Item Name</b>	The name of the item with which the alarm is associated. Items form part of an equipment hierarchy. They can be used to associate tags, alarms and trends with a particular attribute of a physical piece of equipment

Property	Description
	<p>(see <a href="#">Items</a>).</p> <p>There is a limit of 254 characters across the <b>Equipment</b> and Item Name, including any separating periods (.).</p> <p>If you leave this field blank, the last 63 characters of the <b>Alarm Tag</b> field will be used for the Item Name. Be aware that the Alarm Tag allows 79 characters, while Tag Item has a maximum of 63 characters. This may result in compiler errors if the combination of '&lt;Equipment&gt;.&lt;TagItem&gt;' is not unique.</p> <p><b>Note:</b> When defining an item name, avoid using the <a href="#">Reserved Words</a>. If you use any of these, an error message will display when you compile your project.</p>

### General Properties

Property	Description
<b>Alarm Tag</b>	<p>The name of the alarm tag. The name needs to be unique to the cluster and adhere to <a href="#">Tag Name Syntax</a> rules.</p> <p>If your project includes a large number of tags, a naming convention can be helpful when searching and debugging tags (see <a href="#">Structured Tag Names</a>).</p>
<b>Alarm Name</b>	<p>A meaningful description of the alarm, for example, a name that includes the physical device associated with the alarm. The name is used when details of the alarm are displayed on the screen or logged to a device. This field does not support variable data.</p>
<b>Cluster Name</b>	<p>The name of the cluster that runs the alarm. This field needs to be defined if your project has more than one cluster.</p> <p>You can leave this field blank in a multi-cluster system if cluster replication is enabled (see the parameter <a href="#">[General]ClusterReplication</a>).</p>
<b>Category</b>	<p>The alarm category to which the alarm is assigned, defined as either a category number or a label (see <a href="#">Categorize Alarms</a>).</p> <p>If not specified, the category defaults to category 0.</p>
<b>Alarm Desc</b>	<p>A meaningful description of the alarm condition. This description is used when details of the alarm are displayed on screen or logged to a device.</p>

Property	Description
	<p>This field can support variable data (by enclosing an expression in braces). For example:</p> <p>Line Broken Alarm at Line Speed {LineSpeed1}</p> <p>See <a href="#">Add Variable Data to Alarm Messages</a>.</p>
<b>Delay</b>	<p>The alarm delay period (see <a href="#">Use Alarm Delay</a>).</p> <p>A digital alarm becomes active when the state of the triggering condition remains true for the duration of the delay period.</p> <p>The delay period needs to be entered in the following format:</p> <p>HH:MM:SS (Hours:Minutes:Seconds).</p> <p>The value needs to be between 0 seconds (00:00:00) and 24 hours (24:00:00).</p>
<b>Help</b>	<p>The name of the graphics page that displays when the <code>AlarmHelp()</code> function is called by a user-defined command. If not specified, no action occurs when the <code>AlarmHelp()</code> function is called.</p>
<b>Comment</b>	<p>Any useful comment.</p>

#### Source Properties

Property	Description
<b>Variable Tag A / Variable Tag B</b>	<p>Configure digital alarms to activate based on the state of a single or multiple digital variable tags. For a single variable tag use field Var Tag A. For multiple variable tags use fields Var Tag A and Var Tag B.</p> <p>See <a href="#">Alarms</a> for more information on digital alarms.</p>

#### Custom Properties

Property	Description
<b>Custom 1 to Custom 8</b>	<p>A user-defined string for filtering active alarms (maximum 64 characters).</p> <p>Used in a custom Cicode query function as search criteria, the custom alarm filter enables operators to identify and display a subset of active alarms.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>• Custom filters are visible only when the Digital Alarms form is open in Extended mode.</li> <li>• The fields are not case-sensitive and can contain</li> </ul>

Property	Description
	'A'..'Z', 'a'..'z', '0' .. '9', and the underscore '_'. • A custom filter cannot start with a digit.

**Paging Properties**

Property	Description
<b>Paging</b>	A read/write property that indicates whether the alarm will be paged. When the value is 1 (TRUE) the alarm will be paged. The default value is 0 (FALSE). This property can be read using alarm tag browsing and read or modified when tag properties are enabled using the tag name "myCluster.myAlarm.paging".
<b>Paging Group</b>	A read only text string that indicates the paging group to which the alarm belongs. Maximum length is 80 characters. See your third-party paging system documentation for information on how to use this Paging Group string. This property can be read using alarm tag browsing or when tag properties are enabled read using the tagname "myCluster.myAlarm.paginggroup". For example, assign the value of PagingGroup to a variable: <code>myString = myCluster.Alarm_1.paginggroup</code>

**Security Properties**

Field	Description
<b>Area</b>	The Area to which the alarm belongs. If an operator does not have access to an area, the alarm is not visible on the alarm display. For example, if you enter <b>Area 1</b> here, operator need to have access to Area 1 (plus any necessary privileges) to acknowledge or disable this alarm.  The area and privilege fields defined here needs to be designed to work in conjunction. A privilege defined on a button (say) will ignore the alarm defined area.
<b>Privilege</b>	Privilege necessary by an operator to acknowledge or disable the alarm.  If you assign an acknowledgment privilege to an alarm, you should also check the privilege that is assigned to the command(s) that acknowledge the

Field	Description
	alarm. If you assign a different privilege to the commands, an operator needs to have both privileges to acknowledge the command. More importantly, the area defined here may be ignored.

**Historian Properties**

Property	Description
<b>Historize</b>	This field enables you to automatically historize and publish the specified digital alarm in CitectHistorian. If you set this field to "TRUE", the variable will be included in an automated configuration process within the Historian environment. If you set the field to "FALSE" (or leave it blank), the variable will not be included.  See <a href="#">Integration with Historian</a> .

**Project Properties**

Property	Description
<b>Project</b>	The project in which the alarm is configured.

**See Also**[Alarms](#)**Add a Time Stamped Alarm**

Time stamped alarms use a counter to provide an accurate time stamp of when a triggering condition occurred, rather than the time the variable was polled. They can be associated with a single digital variable.

See [Alarms](#) for more information on time stamped alarms.

**To add a time stamped alarm:**

1. In the **System Model** activity, select **Alarms**.
2. On the menu below the Command Bar, select **Time Stamped Alarms**.
3. Add a row to the Grid Editor.
4. Type the required information in each column, or in the Property Grid (see below for a description of the properties).
5. Click **Save**.

**Note:** Configure the [\[Alarm\]UseConfigLimits](#) parameter to force the Plant SCADA alarms server to use digital

---

alarm property values from the RDB, rather than using the values which may be stored in the database file.

---

## Time Stamped Alarms Properties

**Note:** If an alarm was generated by the Equipment Editor, a number of fields on the properties form will be shaded. To configure these fields, you will need to use Equipment Editor (see [Use Equipment Editor to Configure Alarms](#)).

---

### Equipment Properties

Property	Description
<b>Equipment</b>	The name of the equipment associated with the time stamped alarm. Select a name from the drop-down list of existing equipment definitions, or enter a name. There is a limit of 254 characters across the <b>Equipment</b> and <b>Item Name</b> fields, including any separating periods (.).
<b>Item Name</b>	The name of the item with which the alarm is associated. Items form part of an equipment hierarchy. They can be used to associate tags, alarms and trends with a particular attribute of a physical piece of equipment (see <a href="#">Items</a> ). There is a limit of 254 characters across the <b>Equipment</b> and Item Name, including any separating periods (.). If you leave this field blank, the last 63 characters of the <b>Alarm Tag</b> field will be used for the Item Name. Be aware that the Alarm Tag allows 79 characters, while Tag Item has a maximum of 63 characters. This may result in compiler errors if the combination of '<Equipment>.<TagItem>' is not unique. <b>Note:</b> When defining an item name, avoid using the reserved words. If you use any of these, an error message will display when you compile your project.

### General Properties

Property	Description
<b>Alarm Tag</b>	<p>The name of the alarm tag. The name needs to be unique to the cluster and adhere to <a href="#">Tag Name Syntax</a> rules.</p> <p>If your project includes a large number of tags, a naming convention can be helpful when searching and debugging tags (see <a href="#">Structured Tag Names</a>).</p>
<b>Alarm Name</b>	<p>A meaningful description of the alarm, for example, a name that includes the physical device associated with the alarm. The name is used when details of the alarm are displayed on the screen or logged to a device. This field does not support variable data.</p>
<b>Cluster Name</b>	<p>The name of the cluster that runs the alarm. This field needs to be defined if your project has more than one cluster.</p> <p>You can leave this field blank in a multi-cluster system if cluster replication is enabled (see the parameter <a href="#">[General]ClusterReplication</a>).</p>
<b>Category</b>	<p>The alarm category to which the alarm is assigned, defined as either a category number or a label (see <a href="#">Categorize Alarms</a>).</p> <p>If not specified, the category defaults to category 0.</p>
<b>Alarm Desc</b>	<p>A meaningful description of the alarm condition. This description is used when details of the alarm are displayed on screen or logged to a device.</p> <p>This field can support variable data (by enclosing an expression in braces). For example:</p> <p>Line Broken Alarm at Line Speed {LineSpeed1}</p> <p>See <a href="#">Add Variable Data to Alarm Messages</a>.</p>
<b>Timer Expr.</b>	<p>The variable tag or Cicode expression that represents the counter (or millisecond timer) configured in the I/O Device. The counter needs to be configured and maintained by the program in the I/O Device; it is read only when the alarm is triggered.</p> <p>You can use one of three types of counter or timer to record the triggering of time-stamped alarms:</p> <p><b>Continuous counter</b> - A continuous counter is read in the unit to determine the sequence in which the alarms are triggered. The alarms are sorted based on the value of the counter when the alarm was triggered (the exact time is not recorded). Program the counter</p>

Property	Description
	<p>(in the unit) to count continually to its limit, reset, and again count to its limit.</p> <p><b>Millisecond counter</b> - If your unit supports a millisecond counter, program a counter (in the unit) to count (in milliseconds) for 24 hours, and then reset (at midnight). The value of this timer variable (in the unit) is read to determine the exact time when the alarm was triggered.</p> <p><b>LONGBCB timer</b> - Using a LONGBCD timer, you can log the exact time when a Time-stamped alarm becomes active. This variable is read, along with the alarm tag when the alarm activates. You need to program the LONGBCD variable in the following format with the range specified as hexadecimal numbers, excluding numbers containing alphabetic characters:</p> <ul style="list-style-type: none"> <li>1st BYTE - Hours - 00-23 (most significant byte)</li> <li>2nd BYTE - Minutes - 00-59</li> <li>3rd BYTE - Seconds - 00-59</li> <li>4th BYTE - 100th/sec - 00-99 (least significant byte)</li> </ul> <p>Use 'Timer' to handshake with the PLC code: The PLC is informed that it has read the timer register and now the PLC can overwrite the last value. For example, with the following code saved in a Cicode file:</p> <pre> INT FUNCTION AlarmTimerReset(INT iTimer, STRING sTimerTrigger) TagWrite(sTimerTrigger, 0); //Reset the trigger RETURN iTimer; //Return the timer value to the alarm system END </pre> <p><b>Example:</b></p> <p>Where:</p> <p><b>Variable Tag is AlmTrigger1 AND Timer is AlarmTimerReset(AlmTimer1, "AlmTrigger1")</b></p> <p>In this example <b>AlmTimer1</b> is the PLC register that stores the alarm time, and <b>AlmTrigger1</b> is the alarm trigger bit.</p> <p>When AlmTrigger1 is set to 1, the alarm is triggered, and the Cicode function is called. On calling the function, the AlmTimer1 register is read. The function resets the trigger bit (handshaking), and the value of</p>

Property	Description
	<p>AlmTimer1 is returned.</p> <p><b>Note:</b> <code>AlarmTimerReset()</code> is a user function that does not exist in Plant SCADA.</p>
<b>Timer Group</b>	<p>This field is only applicable if the parameter <code>[Alarm]HresType</code> is set to 1 (continuous counter).</p> <p>It allows you to group alarms so that the alarm server can keep track of the counter sequence for each group independently. This may be useful if your system requires multiple alarm count sequences (to support multiple devices).</p>
<b>Help (Optional)</b>	The name of the graphics page that displays when the <code>AlarmHelp()</code> function is called by a user-defined command. If not specified no action occurs when the <code>AlarmHelp()</code> function is called.
<b>Comment</b>	Any useful comment.

**Source Properties**

Property	Description
<b>Variable Tag</b>	The variable tag that triggers the alarm.

**Custom Properties**

Property	Description
<b>Custom 1 to Custom 8</b>	<p>A user-defined string for filtering active alarms (maximum 64 characters).</p> <p>Used in a custom Cicode query function as search criteria, the custom alarm filter enables operators to identify and display a subset of active alarms.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>Custom filters are visible only when the MultiDigital Alarm form is open in Extended mode.</li> <li>The fields are not case-sensitive and can contain 'A'..'Z', 'a'..'z', '0'..'9', and the underscore '_'.</li> <li>A custom filter cannot start with a digit.</li> </ul>

**Paging Properties**

Property	Description
<b>Paging</b>	A read/write property that indicates whether the alarm will be paged. When the value is 1 (TRUE) the alarm will be paged. The default value is 0 (FALSE).

Property	Description
	This property can be read using alarm tag browsing and read or modified when tag properties are enabled using the tag name "myCluster.myAlarm.paging".
<b>Paging Group</b>	<p>A read only text string that indicates the paging group to which the alarm belongs. Maximum length is 80 characters. See your third-party paging system documentation for information on how to use this Paging Group string. This property can be read using alarm tag browsing or when tag properties are enabled read using the tagname "myCluster.myAlarm.paginggroup". For example, assign the value of PagingGroup to a variable:</p> <pre>myString = myCluster.Alarm_1.paginggroup</pre>

### Security Properties

Property	Description
<b>Area</b>	<p>The area to which the alarm belongs. If an operator does not have access to an area, the alarm is not visible on the alarm display. For example, if you enter <b>Area 1</b> here, operator need to have access to Area 1 (plus any necessary privileges) to acknowledge or disable this alarm.</p> <p>The area and privilege fields defined here needs to be designed to work in conjunction. A privilege defined on a button (say) will ignore the alarm defined area.</p>
<b>Privilege</b>	<p>Privilege necessary by an operator to acknowledge or disable the alarm.</p> <p>If you assign an acknowledgment privilege to an alarm, you should also check the privilege that is assigned to the command(s) that acknowledge the alarm. If you assign a different privilege to the commands, an operator needs to have both privileges to acknowledge the command. More importantly, the area defined here may be ignored.</p>

### Historian Properties

Property	Description
<b>Historize</b>	<p>This field enables you to automatically historize and publish the specified time-stamped alarm in CitectHistorian.</p> <p>If you set this field to "TRUE", the variable will be</p>

Property	Description
	<p>included in an automated configuration process within the Historian environment. If you set the field to "FALSE" (or leave it blank), the variable will not be included.</p> <p>See <a href="#">Integration with Historian</a>.</p>

**Project Properties**

Property	Description
<b>Project</b>	The project in which the alarm is configured.

**See Also**[Alarms](#)**Add an Analog Alarm**

Analog alarms are triggered when an analog variable changes beyond one or more specific limits.

To configure an alarm that triggers when the associated tag value reaches a specified limit, you use the following properties:

- High
- High High
- Low
- Low Low

To configure an alarm that triggers when the tag value moves away from a predefined set point, you use:

- Setpoint
- Deviation

To configure an alarm that monitors the rate of change for a tag value across a set period of time, you use:

- Rate

For more information on analog alarms, see [Alarms](#).

**To add an analog alarm:**

1. In the **System Model** activity, select **Alarms**.
2. On the menu below the Command Bar, select **Analog Alarms**.
3. Add a row to the Grid Editor.
4. Complete the properties (see below for a description of the properties).

For a description of the properties, see below.

5. Click **Save**.

**Note:** Configure the [\[Alarm\]UseConfigLimits](#) parameter to force the Plant SCADA Alarm Server to use digital alarm property values from the RDB, rather than using the values which may be stored in the database file.

## Analog Alarms Properties

**Note:** If an alarm was generated by the Equipment Editor, a number of fields on the properties form will be shaded. To configure these fields, you will need to use Equipment Editor (see [Use Equipment Editor to Configure Alarms](#)).

### Equipment Properties

Property	Description
<b>Equipment</b>	The name of the equipment associated with the analog alarm. Select a name from the drop-down list of existing equipment definitions, or enter a name. There is a limit of 254 characters across the Equipment and <b>Item Name</b> fields, including any separating periods (.).
<b>Item Name</b>	The name of the item with which the alarm is associated. Items form part of an equipment hierarchy. They can be used to associate tags, alarms and trends with a particular attribute of a physical piece of equipment (see <a href="#">Items</a> ). There is a limit of 254 characters across the <b>Equipment</b> and Item Name, including any separating periods (.). If you leave this field blank, the last 63 characters of the <b>Alarm Tag</b> field will be used for the Item Name. Be aware that the Alarm Tag allows 79 characters, while Tag Item has a maximum of 63 characters. This may result in compiler errors if the combination of '<Equipment>.<TagItem>' is not unique. <b>Note:</b> When defining an item name, avoid using the <a href="#">Reserved Words</a> . If you use any of these, an error message will display when you compile your project.

### General Properties

Property	Description
<b>Alarm Tag</b>	The name of the alarm tag. The name needs to be unique to the cluster and adhere to <a href="#">Tag Name Syntax</a> rules. If your project includes a large number of tags, a

Property	Description
	naming convention can be helpful when searching and debugging tags (see <a href="#">Structured Tag Names</a> ).
<b>Alarm Name</b>	A meaningful description of the alarm, for example, a name that includes the physical device associated with the alarm. The name is used when details of the alarm are displayed on the screen or logged to a device. This field does not support variable data.
<b>Cluster Name</b>	<p>The name of the cluster that runs the alarm. This field needs to be defined if your project has more than one cluster.</p> <p>You can leave this field blank in a multi-cluster system if cluster replication is enabled (see the parameter [General]ClusterReplication).</p>
<b>Category</b>	<p>The alarm category to which the alarm is assigned, defined as either a category number or a label (see <a href="#">Categorize Alarms</a>).</p> <p>If not specified, the category defaults to category 0.</p>
<b>Setpoint</b>	<p>An analog variable tag, expression or base value that determines if a deviation alarm is to be triggered. This property is optional. If a setpoint is not specified, it will default to 0 (zero).</p> <p><b>Note:</b> If set to an analog variable tag, it needs to be a different tag to the Variable Tag property of the alarm.</p>
<b>High High Delay</b>	<p>Delay period for High High Alarms. Alarm will activate if its triggering condition is met for the duration of the delay period (see <a href="#">Use Alarm Delay</a>).</p> <p>If no value is set, the high high alarm will be activated as soon as the tag exceeds the high high value.</p> <p><b>Note:</b> The delay period needs to be entered in the format HH:MM:SS (hours:minutes:seconds). The value needs to be between 0 seconds (00:00:00) and 24 hours (24:00:00).</p>
<b>High Delay</b>	<p>The delay period for high alarms. Alarm will activate if its triggering condition is met for the duration of this period. If no value is set, the high alarm will be activated as soon as the tag exceeds the high value.</p> <p><b>Note:</b> The delay period needs to be entered in the format HH:MM:SS (hours:minutes:seconds). The value needs to be between 0 seconds (00:00:00) and 24 hours (24:00:00).</p>

Property	Description
	<p>When a tag value increases from high to high high within the high delay period, the delay timer is reset. The high high alarm is only activated if the value remains in the high high range for the delay period.</p> <p>When the value increases from high to high high after the high delay period has expired, a high alarm is activated and then the delay period for the high high alarm begins.</p> <p>If the tag value exceeds the high high value and then falls below it before the high high delay period expires, at the time it falls, the high alarm is triggered immediately.</p> <p>It has an ON time of when the tag value exceeded the high high value. These points also apply to tag values traveling between Low and Low Low ranges.</p>
<b>Low Delay</b>	<p>The delay period for Low Alarms. The alarm will only activate if its triggering condition is met for the duration of this period. This property is optional. If no value is set, the Low Alarm is activated as soon as the tag drops below the Low value.</p> <p><b>Note:</b> The delay period needs to be entered in the format HH:MM:SS (Hours:Minutes:Seconds). The value needs to be between 0 seconds (00:00:00) and 24 hours (24:00:00).</p>
<b>Low Low Delay</b>	<p>The delay period for Low Low Alarms. The alarm will only activate if its triggering condition is met for the duration of this period. If no value is set, the Low Low Alarm is activated as soon as the tag drops below the Low Low value.</p> <p><b>Note:</b> The delay period needs to be entered in the format HH:MM:SS (Hours:Minutes:Seconds). The value needs to be between 0 seconds (00:00:00) and 24 hours (24:00:00).</p>
<b>Variable Tag</b>	<p>The analog variable (tag) or expression that triggers the alarm.</p>
<b>Deviation</b>	<p>The value used as the triggering condition for a Deviation Alarm. A Deviation Alarm is activated when the value of the Variable Tag remains outside the deviation range (determined by the Setpoint) for the duration of the Deviation Delay period. This property is optional. If a deviation is not specified, no Deviation Alarm is activated.</p>

Property	Description
<b>Deviation Delay</b>	<p>The delay period for Deviation Alarms. The alarm will only activate if its triggering condition is met for the duration of this period. If no value is set, the Deviation Alarm is activated as soon as the Variable Tag falls outside the deviation range.</p> <p><b>Note:</b> The delay period needs to be entered in the format HH:MM:SS (Hours:Minutes:Seconds). The value needs to be between 0 seconds (00:00:00) and 24 hours (24:00:00).</p>
<b>Rate</b>	<p>By dividing this value by the alarm period, the "maximum rate" at which the value of the variable tag can change is determined. At each Scan Time, the value of the tag is checked. If its rate of change is greater than the maximum rate, a Rate of Change Alarm is triggered.</p> <p>For example, to verify that a tank does not fill too quickly, you might configure a rate of change alarm, using a Rate of <b>300</b> liters, an <b>[Alarm]Period of 60 seconds</b>, and an <b>[Alarm]ScanTime of 1 second</b>. This means that the maximum allowable rate of change for the tank level is <b>5 l/sec</b> (300 liters / 60 seconds). The actual rate of change at each <b>ScanTime</b> is calculated. That is, every second, it checks the current level of the tank and compares it to the level recorded a second earlier. If the actual rate of change is, say, <b>8 l/sec</b>, a Rate of Change Alarm is triggered immediately.</p> <p>The variable tags deadband % needs to be less than the alarms rate divided by the engineering scale of the variable tag. Otherwise, the rate of change alarm will only go off when the change in the variable tag exceeds the deadband value. If no value is set, no Rate of Change Alarm is activated.</p>
<b>Deadband</b>	Value the Variable Tag needs to return to before the Alarm becomes inactive.
<b>Format</b>	The display format of the value (of the variable) when it is displayed on a graphics page, written to a file or passed to a function (that expects a string). If a format is not specified, the format defaults to the format specified for Variable tag.
<b>Help</b>	The name of the graphics page that displays when the <b>AlarmHelp()</b> function is called by a user-defined command. If not specified no action occurs when the

Property	Description
	AlarmHelp() function is called.
Comment	Any useful comment.

**Source Properties**

Property	Description
Variable Tag	The analog variable (tag) or expression that triggers the alarm.

**Limits Properties**

Property	Description
<b>High High</b>	<p>Value used as the triggering condition for a high high alarm. If the value of the variable tag exceeds this value for the duration of the high high delay period the alarm will be triggered. The active alarm will have an ON time of when the tag exceeded the high high value. Because a high alarm needs to precede a high high alarm, when the high high alarm is triggered it replaces the high alarm.</p> <p>To display more than one state on the alarm page at the same time, configure a separate alarm for each state. (Each alarm would monitor the same tag.)</p>
<b>High</b>	The value used as the triggering condition for a high alarm. The high alarm becomes active when the value of the variable tag exceeds this value for the duration of the high delay period. The active alarm has an ON time of when the tag exceeded the high value.
<b>Low</b>	The value used as the triggering condition for a Low Alarm. Alarm becomes active when the value of the Variable Tag drops below this value and remains there for the duration of the Low Delay period. The active alarm has an ON time of when the tag fell below the Low value.
<b>Low Low</b>	<p>The value used as the triggering condition for a Low Low Alarm. A Low Low Alarm becomes active when the value of the Variable Tag drops below this value and remains there for the duration of the Low Low Delay period. The active alarm has an ON time of when the tag fell below the Low Low value.</p> <p>Because a Low Alarm needs to precede a Low Low Alarm, when the Low Low Alarm is triggered it</p>

Property	Description
	replaces the Low Alarm. If you want an analog alarm to display more than one state on the alarm page at the same time, configure a separate alarm for each state. (Each alarm would monitor the same tag.)

**Custom Properties**

Property	Description
<b>Custom 1 to Custom 8</b>	<p>A user-defined string for filtering active alarms (maximum 64 characters).</p> <p>Used in a custom Cicode query function as search criteria, the custom alarm filter enables operators to identify and display a subset of active alarms.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>• Custom filters are visible only when the Digital Alarms form is open in Extended mode.</li> <li>• The fields are not case-sensitive and can contain 'A'..'Z', 'a'..'z', '0'..'9', and the underscore '_'.</li> <li>• A custom filter cannot start with a digit.</li> </ul>

**Paging Properties**

Property	Description
<b>Paging</b>	A read/write property that indicates whether the alarm will be paged. When the value is 1 (TRUE) the alarm will be paged. The default value is 0 (FALSE). This property can be read using alarm tag browsing and read or modified when tag properties are enabled using the tag name "myCluster.myAlarm.paging".
<b>Paging Group</b>	A read only text string that indicates the paging group to which the alarm belongs. Maximum length is 80 characters. See your third-party paging system documentation for information on how to use this Paging Group string. This property can be read using alarm tag browsing or when tag properties are enabled read using the tagname "myCluster.myAlarm.paginggroup". For example, assign the value of PagingGroup to a variable: <code>myString = myCluster.Alarm_1.paginggroup</code>

**Security Properties**

Field	Description
<b>Area</b>	The Area to which the alarm belongs. If an operator

Field	Description
	<p>does not have access to an area, the alarm is not visible on the alarm display. For example, if you enter <b>Area 1</b> here, operator need to have access to Area 1 (plus any necessary privileges) to acknowledge or disable this alarm.</p> <p>The area and privilege fields defined here needs to be designed to work in conjunction. A privilege defined on a button (say) will ignore the alarm defined area.</p>
<b>Privilege</b>	<p>Privilege necessary by an operator to acknowledge or disable the alarm.</p> <p>If you assign an acknowledgment privilege to an alarm, you should also check the privilege that is also assigned to the command(s) that acknowledge the alarm. If you assign a different privilege to the commands, an operator needs to have both privileges to acknowledge the command. More importantly, the area defined here may be ignored.</p>

### Historian Properties

Field	Description
<b>Historize</b>	<p>This field enables you to automatically historize and publish the specified analog alarm in Citect Historian.</p> <p>If you set this field to "TRUE", the variable will be included in an automated configuration process within the Historian environment. If you set the field to "FALSE" (or leave it blank), the variable will not be included.</p> <p>See <a href="#">Integration with Historian</a>.</p>

### Project Properties

Property	Description
<b>Project</b>	The project in which the alarm is configured.

## See Also

[Alarms](#)

### Add an Advanced Alarm

Advanced alarms trigger when the result of an expression changes to true. To achieve this, you enter the required Cicode expression in the alarm's **Expression** property.

**To add an advanced alarm:**

1. In the **System Model** activity, select **Alarms**.
2. On the menu below the Command Bar, select **Advanced Alarms**.
3. Add a row to the Grid Editor.
4. Complete the property information in each column, or in the fields in the Property Grid (see below for a description of the properties).
5. Click **Save**.

**Note:** Configure the [\[Alarm\]UseConfigLimits](#) parameter to force the Plant SCADA alarms server to use digital alarm property values from the RDB, rather than using the values which may be stored in the database file.

## Advanced Alarms Properties

**Note:** If an alarm was generated by the Equipment Editor, a number of fields on the properties form will be shaded. To configure these fields, you will need to use Equipment Editor (see [Use Equipment Editor to Configure Alarms](#)).

### Equipment Properties

Property	Description
<b>Equipment</b>	The name of the equipment associated with the alarm. Select a name from the drop-down list of existing equipment definitions, or enter a name. There is a limit of 254 characters across the <b>Equipment</b> and <b>Item Name</b> fields, including any separating periods (.).
<b>Item Name</b>	The name of the item with which the alarm is associated. Items form part of an equipment hierarchy. They can be used to associate tags, alarms and trends with a particular attribute of a physical piece of equipment (see <a href="#">Items</a> ). There is a limit of 254 characters across the <b>Equipment</b> and Item Name, including any separating periods (.). If you leave this field blank, the last 63 characters of the <b>Alarm Tag</b> field will be used for the Item Name. Be aware that the Alarm Tag allows 79 characters, while Tag Item has a maximum of 63 characters. This may result in compiler errors if the combination of '<Equipment>.<TagItem>' is not unique. <b>Note:</b> When defining an item name, avoid using the reserved words. If you use any of these, an error message will display when you compile your project.

**General Properties**

Property	Description
<b>Alarm Tag</b>	<p>The name of the alarm tag. The name needs to be unique to the cluster and adhere to <a href="#">Tag Name Syntax</a> rules.</p> <p>If your project includes a large number of tags, a naming convention can be helpful when searching and debugging tags (see <a href="#">Structured Tag Names</a>).</p>
<b>Alarm Name</b>	<p>A meaningful description of the alarm, for example, a name that includes the physical device associated with the alarm. The name is used when details of the alarm are displayed on the screen or logged to a device. This field does not support variable data.</p>
<b>Cluster Name</b>	<p>The name of the cluster that runs the alarm. This field needs to be defined if your project has more than one cluster.</p> <p>You can leave this field blank in a multi-cluster system if cluster replication is enabled (see the parameter <a href="#">[General]ClusterReplication</a>.</p>
<b>Category</b>	<p>The alarm category to which the alarm is assigned, defined as either a category number or a label (see <a href="#">Categorize Alarms</a>).</p> <p>If not specified, the category defaults to category 0.</p>
<b>Alarm Desc</b>	<p>A meaningful description of the alarm condition. This description is used when details of the alarm are displayed on screen or logged to a device.</p> <p>This field can support variable data (by enclosing an expression in braces). For example:</p> <p>"Line Broken Alarm at Line Speed {LineSpeed1}"</p> <p>See <a href="#">Add Variable Data to Alarm Messages</a>.</p>
<b>Delay</b>	<p>The alarm delay period (see <a href="#">Use Alarm Delay</a>).</p> <p>An alarm becomes active when the state of the triggering condition remains true for the duration of the delay period. The active alarm has an ON time of when the state became true. If not specified the alarm is active as soon as it is triggered by the digital tag(s).</p> <p>The delay period needs to be entered in the format HH:MM:SS (Hours:Minutes:Seconds).</p> <p>The value needs to be between 0 seconds (00:00:00) and 24 hours (24:00:00).</p>

Property	Description
<b>Help</b>	The name of the graphics page that displays when the AlarmHelp() function is called by a user-defined command. If not specified no action occurs when the AlarmHelp() function is called.
<b>Comment</b>	Any useful comment.

**Source Properties**

Property	Description
<b>Expression</b>	The Cicode expression that triggers the alarm. Whenever the result of the expression changes to TRUE, the alarm is triggered.

**Custom Properties**

Property	Description
<b>Custom 1 to Custom 8</b>	<p>A user-defined string for filtering active alarms (maximum 64 characters).</p> <p>Used in a custom Cicode query function as search criteria, the custom alarm filter enables operators to identify and display a subset of active alarms.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>Custom filters are visible only when the Advanced Alarms form is open in Extended mode.</li> <li>The fields are not case-sensitive and can contain 'A'..'Z', 'a'..'z', '0'..'9', and the underscore '_'.</li> <li>A custom filter cannot start with a digit.</li> </ul>

**Paging Properties**

Property	Description
<b>Paging</b>	A read/write property that indicates whether the alarm will be paged. When the value is 1 (TRUE) the alarm will be paged. The default value is 0 (FALSE). This property can be read using alarm tag browsing and read or modified when tag properties are enabled using the tag name "myCluster.myAlarm.paging".
<b>Paging Group</b>	A read only text string that indicates the paging group to which the alarm belongs. Maximum length is 80 characters. See your third-party paging system documentation for information on how to use this Paging Group string. This property can be read using alarm tag browsing or when tag properties are

Property	Description
	enabled read using the tagname "myCluster.myAlarm.paginggroup". For example, assign the value of PagingGroup to a variable: <code>myString = myCluster.Alarm_1.paginggroup</code>

**Security Properties**

Property	Description
<b>Area</b>	Area the alarm belongs to. If an operator does not have access to an area, the alarm is not visible on the alarm display. For example, if you enter <b>Area 1</b> here, operator need to have access to Area 1 (plus any necessary privileges) to acknowledge or disable this alarm.  The area and privilege fields defined here needs to be designed to work in conjunction. A privilege defined on a button (say) will ignore the alarm defined area.
<b>Privilege</b>	Privilege necessary by an operator to acknowledge or disable the alarm.  If you assign an acknowledgment privilege to an alarm, you should check the privilege that is assigned to the command(s) that acknowledge the alarm. If you assign a different privilege to the commands, an operator needs to have both privileges to acknowledge the command. More importantly, the area defined here may be ignored.

**Historian Properties**

Property	Description
<b>Historize</b>	This field enables you to automatically historize and publish the specified advanced alarm in CitectHistorian.  If you set this field to "TRUE", the variable will be included in an automated configuration process within the Historian environment. If you set the field to "FALSE" (or leave it blank), the variable will not be included.  See <a href="#">Integration with Historian</a> .

**Project Properties**

Property	Description
<b>Project</b>	The project in which the alarm is configured.

## See Also

[Alarms](#)

### Add a Multi-Digital Alarm

Multi-digital alarms use the output from three digital variables to define eight states. These states represent every possible combination of true and false values. You can then specify which of these states will trigger an alarm.

See [Alarms](#) for more information on multi-digital alarms.

**To add a multi-digital alarm:**

1. In the **System Model** activity, select **Alarms**.
2. On the menu below the Command Bar, select **Multi-Digital Alarms**.
3. Add a row to the Grid Editor.
4. Type the required information in each column, or in the fields in the Property Grid (see below for a description of the fields).
5. Click **Save**.

**Note:** Configure the [\[Alarm\]UseConfigLimits](#) parameter to force the Plant SCADA alarms server to use digital alarm property values from the RDB, rather than using the values which may be stored in the database file.

## Multi-digital Alarms Properties



### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Do not put a blocking Cicode function in the ON Function or OFF Function fields. A blocking function will affect the polling of alarms, and may result in slow or delayed alarm processing.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** If an alarm was generated by the Equipment Editor, a number of fields on the properties form will be shaded. To configure these fields, you will need to use Equipment Editor (see [Use Equipment Editor to Configure Alarms](#)).

#### Equipment Properties

Field	Description
<b>Equipment</b>	The name of the equipment associated with the multi-digital alarm. Select a name from the drop-down list of existing equipment definitions, or enter a name.  There is a limit of 254 characters across the Equipment and <b>Item Name</b> fields, including any separating periods (.).
<b>Item Name</b>	The name of the item with which the alarm is associated.  Items form part of an equipment hierarchy. They can be used to associate tags, alarms and trends with a particular attribute of a physical piece of equipment (see <a href="#">Items</a> ).  There is a limit of 254 characters across the <b>Equipment</b> and Item Name, including any separating periods (.).  If you leave this field blank, the last 63 characters of the <b>Alarm Tag</b> field will be used for the Item Name. Be aware that the Alarm Tag allows 79 characters, while Tag Item has a maximum of 63 characters. This may result in compiler errors if the combination of '<Equipment>.<TagItem>' is not unique.  <b>Note:</b> When defining an item name, avoid using the reserved words. If you use any of these, an error message will display when you compile your project.

**General Properties**

Field	Description
<b>Alarm Tag</b>	The name of the alarm tag. The name needs to be unique to the cluster and adhere to <a href="#">Tag Name Syntax</a> rules.  If your project includes a large number of tags, a naming convention can be helpful when searching and debugging tags (see <a href="#">Structured Tag Names</a> ).
<b>Alarm Name</b>	A meaningful description of the alarm, for example, a name that includes the physical device associated with the alarm. The name is used when details of the alarm are displayed on the screen or logged to a device. This field does not support variable data.
<b>Cluster Name</b>	The name of the cluster that runs the alarm. This field needs to be defined if your project has more than one cluster.

Field	Description
	You can leave this field blank in a multi-cluster system if cluster replication is enabled (see the parameter [General]ClusterReplication).
<b>Category</b>	<p>The alarm category to which the alarm is assigned, defined as either a category number or a label (see <a href="#">Categorize Alarms</a>).</p> <p>If not specified, the category defaults to category 0.</p>
<b>Alarm Desc</b>	<p>A meaningful description of the alarm condition. This description is used when details of the alarm are displayed on screen or logged to a device.</p> <p>This field can support variable data (by enclosing an expression in braces). For example:</p> <p>"Line Broken Alarm at Line Speed {LineSpeed1}"</p> <p>See <a href="#">Add Variable Data to Alarm Messages</a>.</p>
<b>On function</b> (254 chars)	<p>A Cicode function that is executed when the Multi-Digital Alarm becomes active, for example:</p> <p><b>ON Function</b> is <b>STOP_PROCESS = 1;</b></p> <p>In this example the digital variable STOP_PROCESS is set to ON when the alarm is triggered.</p> <p><b>Note:</b> Do not put a blocking Cicode function in this field. The alarm system executes ON or OFF actions within the polling loop. A blocking function will affect the polling of alarms, and may result in slow or delayed alarm processing.</p> <p>A special case of this command occurs when the Alarm ON Function is self-referring, for example:</p> <p>TAG1 = TAG1 + 1.</p> <p>This command will not work properly since tags are not read again before processing the Alarm On function (for performance reasons). Therefore this particular command will initially set the value of TAG1 to 1 rather than incrementing it.</p> <p>To correctly run a command of this type in the Alarm ON Function, use TaskNew() to run your own Cicode function to perform the tag command:</p> <p><b>ON function</b> is <b>TaskNew("MyFunc","Data",5);</b></p>
<b>OFF function</b> (254 chars)	<p>A Cicode function that is executed when a Multi-Digital Alarm becomes inactive. For example:</p> <p><b>OFF Function</b> is <b>ENABLE_PROCESS = 1;</b></p>

Field	Description
	<p>In this example the digital variable ENABLE_PROCESS is set to ON when an alarm in this category is reset.</p> <p><b>Note:</b> Do not put a blocking Cicode function in this field. The alarm system executes ON or OFF actions within the polling loop. A blocking function will affect the polling of alarms, and may result in slow or delayed alarm processing.</p>
Help	The name of the graphics page that displays when the AlarmHelp() function is called by a user-defined command. If not specified, no action occurs when the AlarmHelp() function is called.
Comment	Any useful comment.

**Source Properties**

Field	Description
Variable Tag A	
Variable Tag B	
Variable Tag C	The three digital variables used to define eight states. Each state represents a different combination of tag values for these three variables.

**States Properties**

Field	Description
States and Descriptions	<p>The following eight states represent any possible tag value combinations. The variable tags are presented in the order tag C, tag B, tag A.</p> <ul style="list-style-type: none"> <li>• State 000 - all 3 tags are false.</li> <li>• State 00A - tags C and B are false, tag A is true.</li> <li>• State 0B0 - tags C and A are false, tag B is true.</li> <li>• State 0BA - tag C is false, tags B and A are true.</li> <li>• State C00 - tag C is true, and tags B and A are false.</li> <li>• State C0A - tags C and A are true, tag B is false.</li> <li>• State CB0 - tags C and B are true, tag A is false.</li> <li>• State CBA - all 3 tags are true.</li> </ul> <p>For each state, there are two fields you can configure. In the first field enter a description (for example, "Healthy" or "Stopped") with a maximum of eight characters.</p> <p>In the second field, indicate whether the state will trigger an alarm. A value of 1 indicates an alarm state, and a 0 indicates no alarm will be triggered.</p>

**Suppression Properties**

Field	Description
<b>Suppression</b>	<p>The number of the Suppression Group to which the alarm belongs.</p> <p>Integer value between 0–32767.</p> <p>Alarms in the same group display the same value in this field. Use in conjunction with Suppression Level.</p> <p><b>Note:</b> To assign a name to a Suppression Group, define the name as a label with an integer value.</p>
<b>Level</b>	<p>The level of an alarm within its Suppression Group (integer value). This is a value between 0 and 255, where a lower level represents a higher priority.</p> <p>This property enables an active alarm which is in an ON state (irrespective of whether it is acknowledged or not) to suppress lower priority alarms within the same Suppression Group. When this occurs, only the higher priority (lower level) alarms are displayed.</p> <p>Alarms with lower priorities (higher levels) will only activate and display when the higher priority (lower level) alarms become inactive.</p>

**Custom Properties**

Field	Description
<b>Custom 1 to Custom 8</b>	<p>A user-defined string for filtering active alarms (maximum 64 characters).</p> <p>Used in a custom Cicode query function as search criteria, the custom alarm filter enables operators to identify and display a subset of active alarms.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>• Custom filters are visible only when the MultiDigital Alarm form is open in Extended mode.</li> <li>• The fields are not case-sensitive and can contain 'A'..'Z', 'a'..'z', '0'..'9', and the underscore '_'.</li> <li>• A custom filter cannot start with a digit.</li> </ul>

**Paging Properties**

Field	Description
<b>Paging</b>	A read/write property that indicates whether the alarm will be paged. When the value is 1 (TRUE) the alarm will be paged. The default value is 0 (FALSE). This property can be read using alarm tag browsing and read or modified when tag properties are enabled using the tag name "myCluster.myAlarm.paging".
<b>Paging Group</b>	A read only text string that indicates the paging group to which the alarm belongs. Maximum length is 80 characters. See your third-party paging system documentation for information on how to use this Paging Group string. This property can be read using alarm tag browsing or when tag properties are enabled read using the tagname "myCluster.myAlarm.paginggroup". For example, assign the value of PagingGroup to a variable: <code>myString = myCluster.Alarm_1.paginggroup</code>

**Security Properties**

Field	Description
<b>Area</b>	The Area to which the alarm belongs. If an operator does not have access to an area, the alarm is not visible on the alarm display. For example, if you enter <b>Area 1</b> here, operator need to have access to Area 1 (plus any necessary privileges) to acknowledge or disable this alarm.  The area and privilege fields defined here needs to be designed to work in conjunction. A privilege defined on a button (say) will ignore the alarm defined area.
<b>Privilege</b>	Privilege necessary by an operator to acknowledge or disable the alarm.  If you assign an acknowledgment privilege to an alarm, you should also check the privilege that is assigned to the command(s) that acknowledge the alarm. If you assign a different privilege to the commands, an operator needs to have both privileges to acknowledge the command. More importantly, the area defined here may be ignored.

**Historian Properties**

Field	Description
<b>Historize</b>	This field enables you to automatically historize and publish the specified multi-digital alarm in

Field	Description
	<p>CitectHistorian.</p> <p>If you set this field to "TRUE", the variable will be included in an automated configuration process within the Historian environment. If you set the field to "FALSE" (or leave it blank), the variable will not be included.</p> <p>See <a href="#">Integration with Historian</a>.</p>

**Project Properties**

Field	Description
Project	The project in which the alarm is configured.

**See Also**[Alarms](#)**Add a Time Stamped Digital Alarm**

Time stamped digital alarms use notifications from the alarm server to determine when an alarm is triggered. This provides an accurate time stamp of when an alarm condition occurs.

**Note:** Time stamped digital alarms are timestamp based and designed to work with the Driver Runtime Interface (DRI). For a list of supported drivers, see [Retrieving Time-stamped Data from I/O Devices](#). Alternatively, you can configure the Cicode function [AlarmNotifyVarChange](#), as it notifies the alarm server of any value changes for the associated variable. It also passes the timestamp with the notification message. The alarm server maintains a queue for notification messages and runs a separate task to check this queue.

A time stamped digital alarm activates in response to the state of one or two digital variables (specified in the alarm's **Variable Tag A** and **Variable Tag B** properties). When you define the variable tags, you can precede a tag name with the logical operator "NOT". This specifies that the OFF state (0) is the triggering condition for the variable tag.

The alarm becomes active when the triggering condition spans the amount of time specified in the **Delay** property.

**To add a time stamped digital alarm:**

1. In the **System Model** activity, select **Alarms**.
2. On the menu below the Command Bar, select **Time Stamped Digital Alarms**.
3. Add a row to the Grid Editor.
4. Type the required information in each column, or in the fields in the Property Grid (see below for a description of the fields).
5. Click **Save**.

**Note:** Configure the [\[Alarm\]UseConfigLimits](#) parameter to force the Plant SCADA alarms server to use digital alarm property values from the RDB, rather than using the values which may be stored in the database file.

## Time-stamped Digital Alarms Properties

**Note:** If an alarm was generated by the Equipment Editor, a number of fields on the properties form will be shaded. To configure these fields, you will need to use Equipment Editor (see [Use Equipment Editor to Configure Alarms](#)).

### Equipment Properties

Field	Description
<b>Equipment</b>	The name of the equipment associated with the digital alarm. Select a name from the drop-down list of existing equipment definitions, or enter a name. There is a limit of 254 characters across the Equipment and <b>Item Name</b> fields, including any separating periods (.).
<b>Item Name</b>	The name of the item with which the alarm is associated. Items form part of an equipment hierarchy. They can be used to associate tags, alarms and trends with a particular attribute of a physical piece of equipment (see <a href="#">Items</a> ). There is a limit of 254 characters across the <b>Equipment</b> and Item Name, including any separating periods (.). If you leave this field blank, the last 63 characters of the <b>Alarm Tag</b> field will be used for the Item Name. Be aware that the Alarm Tag allows 79 characters, while Tag Item has a maximum of 63 characters. This may result in compiler message if the combination of '<Equipment>.<TagItem>' is not unique. <b>Note:</b> When defining an item name, avoid using the reserved words. If you use any of these, an error message will display when you compile your project.

### General Properties

Field	Description
<b>Alarm Tag</b>	The name of the alarm tag. The name needs to be unique to the cluster and adhere to <a href="#">Tag Name Syntax</a> rules. If your project includes a large number of tags, a naming convention can be helpful when searching and debugging tags (see <a href="#">Structured Tag Names</a> ).
<b>Alarm Name</b>	A meaningful description of the alarm, for example, a name that includes the physical device associated with

Field	Description
	the alarm. The name is used when details of the alarm are displayed on the screen or logged to a device. This field does not support variable data.
<b>Cluster Name</b>	The name of the cluster that runs the alarm. This field needs to be defined if your project has more than one cluster.  You can leave this field blank in a multi-cluster system if cluster replication is enabled (see the parameter <a href="#">[General]ClusterReplication</a> ).
<b>Category</b>	The alarm category to which the alarm is assigned, defined as either a category number or a label (see <a href="#">Categorize Alarms</a> ).  If not specified, the category defaults to category 0.
<b>Alarm Desc</b>	A meaningful description of the alarm condition. This description is used when details of the alarm are displayed on screen or logged to a device.  This field can support variable data (by enclosing an expression in braces). For example:  Line Broken Alarm at Line Speed {LineSpeed1} See <a href="#">Add Variable Data to Alarm Messages</a> .
<b>Delay</b>	The alarm delay period (see <a href="#">Using Alarm Delay</a> ).  A time stamped digital alarm becomes active when the state of the triggering condition remains true for the duration of the delay period.  The delay period needs to be entered in the following format:  HH:MM:SS (Hours:Minutes:Seconds).  The value needs to be between 0 seconds (00:00:00) and 24 hours (24:00:00).
<b>Help</b>	The name of the graphics page that displays when the <code>AlarmHelp()</code> function is called by a user-defined command. If not specified no action occurs when the <code>AlarmHelp()</code> function is called.
<b>Comment</b>	Any useful comment.

**Source Properties**

Field	Description
<b>Variable Tag A /</b>	Configure time stamped digital alarms to activate

Field	Description
<b>Variable Tag B</b>	based on the state of a single or multiple digital variable tags. For a single variable tag use field Var Tag A. For multiple variable tags use fields Var Tag A and Var Tag B.  See <a href="#">Alarms</a> for more information on time stamped digital alarms.

**Custom Properties**

Field	Description
<b>Custom 1 to Custom 8</b>	A user-defined string for filtering active alarms (maximum 64 characters).  Used in a custom Cicode query function as search criteria, the custom alarm filter enables operators to identify and display a subset of active alarms.  <b>Note:</b> <ul style="list-style-type: none"> <li>The fields are not case-sensitive and can contain 'A'..'Z', 'a'..'z', '0'..'9', and the underscore '_'. </li> <li>A custom filter cannot start with a digit.</li> </ul>

**Paging Properties**

Field	Description
<b>Paging</b>	A read/write property that indicates whether the alarm will be paged. When the value is 1 (TRUE) the alarm will be paged. The default value is 0 (FALSE). This property can be read using alarm tag browsing and read or modified when tag properties are enabled using the tag name "myCluster.myAlarm.paging".
<b>Paging Group</b>	A read only text string that indicates the paging group to which the alarm belongs. Maximum length is 80 characters. See your third-party paging system documentation for information on how to use this Paging Group string. This property can be read using alarm tag browsing or when tag properties are enabled read using the tagname "myCluster.myAlarm.paginggroup". For example, assign the value of PagingGroup to a variable:  myString = myCluster.Alarm_1.paginggroup

**Security Properties**

Field	Description
<b>Area</b>	The Area to which the alarm belongs. If an operator

Field	Description
	<p>does not have access to an area, the alarm is not visible on the alarm display. For example, if you enter <b>Area 1</b> here, operator need to have access to Area 1 (plus any necessary privileges) to acknowledge or disable this alarm.</p> <p>The area and privilege fields defined here needs to be designed to work in conjunction. A privilege defined on a button (say) will ignore the alarm defined area.</p>
<b>Privilege</b>	<p>Privilege necessary by an operator to acknowledge or disable the alarm.</p> <p>If you assign an acknowledgment privilege to an alarm, there is no need to assign a privilege to the command(s) that acknowledge the alarm. If you assign a different privilege to the commands, an operator needs to have both privileges to acknowledge the command. More importantly, the area defined here may be ignored.</p>

**Historian Properties**

Field	Description
<b>Historize</b>	<p>This field enables you to automatically historize and publish the specified digital alarm in CitectHistorian.</p> <p>If you set this field to "TRUE", the variable will be included in an automated configuration process within the Historian environment. If you set the field to "FALSE" (or leave it blank), the variable will not be included.</p> <p>See <a href="#">Integration with Historian</a>.</p>

**Project Properties**

Property	Description
<b>Project</b>	The project in which the time-stamped digital alarm is configured.

**See Also**

[Alarms](#)

**Add a Time Stamped Analog Alarm**

Time stamped analog alarms use notifications from the alarm server to determine when an alarm is triggered.

This provides an accurate time stamp of when an alarm condition occurs.

**Note:** Time stamped analog alarms are timestamp based and designed to work with the Driver Runtime Interface (DRI). For a list of supported drivers, see [Retrieving Time-stamped Data from I/O Devices](#). Alternatively, you can configure the Cicode function [AlarmNotifyVarChange](#), as it notifies the alarm server of any value changes for the associated variable. It also passes the timestamp with the notification message. The alarm server maintains a queue for notification messages and runs a separate task to check this queue.

To configure an alarm that triggers when the associated tag value reaches a specified limit, you use the following properties:

- High
- High High
- Low
- Low Low

To configure an alarm that triggers when the tag value moves away from a predefined set point, you use:

- Setpoint
- Deviation

To configure an alarm that monitors the rate of change for a tag value across a set period of time, you use:

- Rate

#### To add a time stamped analog alarm:

1. In the **System Model** activity, select **Alarms**.
2. On the menu below the Command Bar, select **Time Stamped Analog Alarms**.
3. Add a row to the Grid Editor.
4. Type the required information in each column, or in the fields in the Property Grid.  
For a description of the properties, see below.
5. Click **Save**.

**Note:** Configure the [\[Alarm\]UseConfigLimits](#) parameter to force the Plant SCADA alarms server to use digital alarm property values from the RDB, rather than using the values which may be stored in the database file.

## Time Stamped Analog Alarms Properties

**Note:** If an alarm was generated by the Equipment Editor, a number of fields on the properties form will be shaded. To configure these fields, you will need to use Equipment Editor (see [Use Equipment Editor to Configure Alarms](#)).

### Equipment Properties

Field	Description
Equipment	The name of the equipment associated with the digital alarm. Select a name from the drop-down list

Field	Description
	<p>of existing equipment definitions, or enter a name. There is a limit of 254 characters across the Equipment and <b>Item Name</b> fields, including any separating periods (.).</p>
Item Name	<p>The name of the item with which the alarm is associated.</p> <p>Items form part of an equipment hierarchy. They can be used to associate tags, alarms and trends with a particular attribute of a physical piece of equipment (see <a href="#">Items</a>).</p> <p>There is a limit of 254 characters across the <b>Equipment</b> and Item Name, including any separating periods (.).</p> <p>If you leave this field blank, the last 63 characters of the <b>Alarm Tag</b> field will be used for the Item Name. Be aware that the Alarm Tag allows 79 characters, while Tag Item has a maximum of 63 characters. This may result in compiler errors if the combination of '&lt;Equipment&gt;.&lt;TagItem&gt;' is not unique.</p> <p><b>Note:</b> When defining an item name, avoid using the <a href="#">Reserved Words</a>. If you use any of these, an error message will display when you compile your project.</p>

### General Properties

Field	Description
Alarm Tag	<p>The name of the alarm tag. The name needs to be unique to the cluster and adhere to <a href="#">Tag Name Syntax</a> rules.</p> <p>If your project includes a large number of tags, a naming convention can be helpful when searching and debugging tags (see <a href="#">Structured Tag Names</a>).</p>
Alarm Name	<p>A meaningful description of the alarm, for example, a name that includes the physical device associated with the alarm. The name is used when details of the alarm are displayed on the screen or logged to a device. This field does not support variable data.</p>
Cluster Name	<p>The name of the cluster that runs the alarm. This field needs to be defined if your project has more than one cluster.</p> <p>You can leave this field blank in a multi-cluster system if cluster replication is enabled (see the parameter</p>

Field	Description
	<a href="#">[General]</a> <a href="#">ClusterReplication</a> ).
Category	<p>The alarm category to which the alarm is assigned, defined as either a category number or a label (see <a href="#">Categorize Alarms</a>).</p> <p>If not specified, the category defaults to category 0.</p>
Setpoint	<p>An analog variable tag that determines if a deviation alarm is to be triggered. This property is optional. If no value is set, it will default to 0 (zero).</p> <p><b>Note:</b> If set to an analog variable tag, it needs to a different tag to the Variable Tag property of the alarm.</p>
High High Delay	<p>Delay period for High High Alarms. Alarm will activate if its triggering condition is met for the duration of the delay period (see <a href="#">Use Alarm Delay</a>).</p> <p>If no value is set, the high high alarm will be activated as soon as the tag exceeds the high high value.</p> <p><b>Note:</b> The delay period needs to be entered in the format HH:MM:SS (hours:minutes:seconds). The value needs to be between 0 seconds (00:00:00) and 24 hours (24:00:00).</p>
High Delay	<p>The delay period for high alarms. Alarm will activate if its triggering condition is met for the duration of this period. If no value is set, the high alarm will be activated as soon as the tag exceeds the high value.</p> <p><b>Note:</b> The delay period needs to be entered in the format HH:MM:SS (hours:minutes:seconds). The value needs to be between 0 seconds (00:00:00) and 24 hours (24:00:00).</p> <p>When a tag value increases from high to high high within the high delay period, the delay timer is reset. The high high alarm is only activated if the value remains in the high high range for the delay period.</p> <p>When the value increases from high to high high after the high delay period has expired, a high alarm is activated and then the delay period for the high high alarm begins.</p> <p>If the tag value exceeds the high high value and then falls below it before the high high delay period expires, at the time it falls, the high alarm is triggered immediately.</p> <p>It has an ON time of when the tag value exceeded the high high value. These points also apply to tag values</p>

Field	Description
	traveling between Low and Low Low ranges.
Low Delay	<p>The delay period for Low Alarms. The alarm will only activate if its triggering condition is met for the duration of this period. This property is optional. If no value is set, the Low Alarm is activated as soon as the tag drops below the Low value.</p> <p><b>Note:</b> The delay period needs to be entered in the format HH:MM:SS (Hours:Minutes:Seconds). The value needs to be between 0 seconds (00:00:00) and 24 hours (24:00:00).</p>
Low Low Delay	<p>The delay period for Low Low Alarms. The alarm will only activate if its triggering condition is met for the duration of this period. If no value is set, the Low Low Alarm is activated as soon as the tag drops below the Low Low value.</p> <p><b>Note:</b> The delay period needs to be entered in the format HH:MM:SS (Hours:Minutes:Seconds). The value needs to be between 0 seconds (00:00:00) and 24 hours (24:00:00).</p>
Deviation	<p>The value used as the triggering condition for a Deviation Alarm. A Deviation Alarm is activated when the value of the Variable Tag remains outside the deviation range (determined by the Setpoint) for the duration of the Deviation Delay period. This property is optional. If no deviation is set, no Deviation Alarm is activated.</p>
Deviation Delay	<p>The delay period for Deviation Alarms. The alarm will only activate if its triggering condition is met for the duration of this period. If no value is set, the Deviation Alarm is activated as soon as the Variable Tag falls outside the deviation range.</p> <p><b>Note:</b> The delay period needs to be entered in the format HH:MM:SS (Hours:Minutes:Seconds). The value needs to be between 0 seconds (00:00:00) and 24 hours (24:00:00).</p>
Rate	<p>By dividing this value by the alarm period, the "maximum rate" at which the value of the variable tag can change is determined. At each Scan Time, the value of the tag is checked. If its rate of change is greater than the maximum rate, a Rate of Change Alarm is triggered.</p>

Field	Description
	<p>For example, to verify that a tank does not fill too quickly, you might configure a rate of change alarm, using a Rate of <b>300</b> liters, an <b>[Alarm]Period of 60 seconds</b>, and an <b>[Alarm]ScanTime of 1 second</b>. This means that the maximum allowable rate of change for the tank level is <b>5 l/sec</b> (300 liters / 60 seconds). The actual rate of change at each <b>ScanTime</b> is calculated. That is, every second, it checks the current level of the tank and compares it to the level recorded a second earlier. If the actual rate of change is, say, <b>8 l/sec</b>, a Rate of Change Alarm is triggered immediately.</p> <p>The variable tags deadband % needs to be less than the alarms rate divided by the engineering scale of the variable tag. Otherwise, the rate of change alarm will only go off when the change in the variable tag exceeds the deadband value. If no value is set, no Rate of Change Alarm is activated.</p>
Deadband	Value the variable tag needs to return to before the alarm becomes inactive.
Format	The display format of the value (of the variable) when it is displayed on a graphics page, written to a file or passed to a function (that expects a string). If no format is specified, it defaults to the format specified for variable tag.
Help	The name of the graphics page that displays when the <b>AlarmHelp()</b> function is called by a user-defined command. If not specified no action occurs when the <b>Alarm()</b> function is called.
Comment	Any useful comment.

**Source Properties**

Field	Description
Variable Tag	The analog variable tag that triggers the alarm.

**Limits Properties**

Field	Description
High High	The value used as the triggering condition for a high high alarm. The high high alarm becomes active when the value of the variable tag exceeds this value for the duration of the high high delay period. The active

Field	Description
	<p>alarm has an ON time of when the tag exceeded the high high value.</p> <p>Because a high alarm needs to precede a high high alarm, when the high high alarm is triggered it replaces the high alarm. If you want an analog alarm to display more than one state on the alarm page at the same time, configure a separate alarm for each state. (Each alarm would monitor the same tag.)</p>
High	<p>The value used as the triggering condition for a high alarm. The high alarm becomes active when the value of the variable tag exceeds this value for the duration of the high delay period. The active alarm has an ON time of when the tag exceeded the high value.</p>
Low	<p>The value used as the triggering condition for a Low Alarm. Alarm becomes active when the value of the Variable Tag drops below this value and remains there for the duration of the Low Delay period. The active alarm has an ON time of when the tag fell below the Low value.</p>
Low Low	<p>The value used as the triggering condition for a Low Low Alarm. A Low Low Alarm becomes active when the value of the Variable Tag drops below this value and remains there for the duration of the Low Low Delay period. The active alarm has an ON time of when the tag fell below the Low Low value.</p> <p>Because a Low Alarm needs to precede a Low Low Alarm, when the Low Low Alarm is triggered it replaces the Low Alarm. If you want an analog alarm to display more than one state on the alarm page at the same time, configure a separate alarm for each state. (Each alarm would monitor the same tag.)</p>

#### Custom Properties

Field	Description
Custom 1 to Custom 8	<p>A user-defined string for filtering active alarms (maximum 64 characters).</p> <p>Used in a custom Cicode query function as search criteria, the custom alarm filter enables operators to identify and display a subset of active alarms.</p> <p><b>Note:</b></p>

Field	Description
	<ul style="list-style-type: none"> <li>The fields are not case-sensitive and can contain 'A'..'Z', 'a'..'z', '0'..'9', and the underscore '_'. </li> <li>A custom filter cannot start with a digit.</li> </ul>

**Paging Properties**

Field	Description
Paging	A read/write property that indicates whether the alarm will be paged. When the value is 1 (TRUE) the alarm will be paged. The default value is 0 (FALSE). This property can be read using alarm tag browsing and read or modified when tag properties are enabled using the tag name "myCluster.myAlarm.paging".
Paging Group	A read only text string that indicates the paging group to which the alarm belongs. Maximum length is 80 characters. See your third-party paging system documentation for information on how to use this Paging Group string. This property can be read using alarm tag browsing or when tag properties are enabled read using the tagname "myCluster.myAlarm.paginggroup". For example, assign the value of PagingGroup to a variable: <code>myString = myCluster.Alarm_1.paginggroup</code>

**Security Properties**

Field	Description
Area	The Area the alarm belongs to. If an operator does not have access to an area, the alarm is not visible on the alarm display. For example, if you enter <b>Area 1</b> here, operator need to have access to Area 1 (plus any necessary privileges) to acknowledge or disable this alarm.  The area and privilege fields defined here needs to be designed to work in conjunction. A privilege defined on a button (say) will ignore the alarm defined area.
Privilege	Privilege necessary by an operator to acknowledge or disable the alarm.  If you assign an acknowledgment privilege to an alarm, there is no need to assign a privilege to the command(s) that acknowledge the alarm. If you assign a different privilege to the commands, an operator

Field	Description
	needs to have both privileges to acknowledge the command. More importantly, the area defined here may be ignored.

**Historian Properties**

Field	Description
Historize	This field enables you to automatically historize and publish the specified analog alarm in CitectHistorian. If you set this field to "TRUE", the variable will be included in an automated configuration process within the Historian environment. If you set the field to "FALSE" (or leave it blank), the variable will not be included.  See <a href="#">Integration with Historian</a> .

**Project Properties**

Property	Description
Project	The project in which the time-stamped analog alarm is configured.

**See Also**[Alarms](#)**Add a Double Point Status Alarm**

A double point status alarm responds to eight states in a field device that are represented in Plant SCADA as a single integer tag.

For each of the eight states you can specify the following properties:

- **State Name <n>** — a name that identifies the condition that the state represents.
- **State Type <n>** — the action that occurs when the device transitions to the defined state. The options are:
  - "Alarm" - an alarm is added to the Active Alarms page and an event is logged to the SOE page. The **State Name** will appear in the **State** field.
  - "Event" - indicates an OFF state has occurred. An event is logged on the SOE page, even if an alarm state transition did not occur. The **State Name** will appear in the **State** field.
  - "None" - indicates an OFF state has occurred. The **State Name** will appear in the **State** field on both the SOE and Active Alarms pages when an alarm transition occurs.

**Note:** If an alarm is currently in an "Event" or "None" state, a further transition to a "None" state will not be reported.

This means when the value of the integer tag is 0, the alarm will go into "State Name 0". When the value of the tag is 1, it will go into "State Name 1", and so on. Out of range values will be ignored.

**Note:** Double point status alarms are timestamp based and designed to work with the Driver Runtime Interface (DRI). For a list of supported drivers, see [Retrieving Time-stamped Data from I/O Devices](#). Alternatively, you can trigger a double point status alarm using the Cicode function `AlarmNotifyVarChange` as it notifies the alarm server of any value changes for the associated variable. It also passes the timestamp with the notification message.

### Example

The following table demonstrates the configuration of a double point status alarm associated with a single integer tag that represents four states.

Tag Value	Alarm Property	Value	Alarm	Event
0	State Name 0	Unknown	Yes	No
	State Type 0	Alarm		
1	State Name 1	Open	No	Yes
	State Type 1	Event		
2	State Name 2	Closed	No	Yes
	State Type 2	Event		
3	State Name 3	Jammed	Yes	No
	State Type 3	Alarm		

This configuration will result in following:

- An alarm will be triggered whenever the device transitions to an "Unknown" or "Jammed" state. This is determined by the **State Type <n>** property, which is set to "Alarm" for these two states. When an alarm occurs, the associated **State Name <n>** value will display in the State field on the Active Alarms page and the SOE page.
- If a transition occurs between two states that both have **State Type <n>** set to "Event" (for example, a transition from "Open" to "Closed"), this will be recorded on the SOE page without an alarm being raised.
- If the alarm transitions between two states that both have **State Type <n>** set to "Alarm", the state of the alarm will change to reflect the new state, and the time of the alarm will be updated.
- If an alarm transitions from "Alarm" state to an "Event" state, the alarm will be moved to an OFF state on the Active Alarms page and an event will be logged on the SOE page.

### To add a double point status alarm:

- In the **System Model** activity, select **Alarms**.
- On the menu below the Command Bar, select **Double Point Status Alarms**.
- Add a row to the Grid Editor.
- Type the required information in each column, or in the fields in the Property Grid.  
For a description of the properties, see below.
- Click **Save**.

**Note:** Configure the [\[Alarm\]UseConfigLimits](#) parameter to force the Plant SCADA alarms server to use double point alarm property values from the RDB, rather than using the values which may be stored in the database file.

## Double Point Status Alarm Properties

**Note:** If an alarm was generated by the Equipment Editor, a number of fields on the properties form will be shaded. To configure these fields, you will need to use Equipment Editor (see [Use Equipment Editor to Configure Alarms](#)).

### Equipment Properties

Field	Description
Equipment	<p>The name of the equipment associated with the double point status alarm. Select a name from the drop-down list of existing equipment definitions, or enter a name.</p> <p>There is a limit of 254 characters across the <b>Equipment</b> and <b>Item Name</b> fields, including any separating periods (.).</p>
Item Name	<p>The name of the item with which the alarm is associated.</p> <p>Items form part of an equipment hierarchy. They can be used to associate tags, alarms and trends with a particular attribute of a physical piece of equipment (see <a href="#">Items</a>).</p> <p>There is a limit of 254 characters across the <b>Equipment</b> and Item Name, including any separating periods (.).</p> <p>If you leave this field blank, the last 63 characters of the <b>Alarm Tag</b> field will be used for the Item Name. Be aware that the Alarm Tag allows 79 characters, while Tag Item has a maximum of 63 characters. This may result in compiler errors if the combination of '&lt;Equipment&gt;.&lt;TagItem&gt;' is not unique.</p> <p><b>Note:</b> When defining an item name, avoid using the <a href="#">Reserved Words</a>. If you use any of these, an error message will display when you compile your project.</p>

### General Properties

Field	Description
Alarm Tag	<p>The name of the alarm tag. The name needs to be unique to the cluster and adhere to <a href="#">Tag Name Syntax</a> rules.</p> <p>If your project includes a large number of tags, a</p>

Field	Description
	naming convention can be helpful when searching and debugging tags (see <a href="#">Structured Tag Names</a> ).
Alarm Name	A meaningful description of the alarm, for example, a name that includes the physical device associated with the alarm. The name is used when details of the alarm are displayed on the screen or logged to a device. This field does not support variable data.
Cluster Name	<p>The name of the cluster that runs the alarm. This field needs to be defined if your project has more than one cluster.</p> <p>You can leave this field blank in a multi-cluster system if cluster replication is enabled (see the parameter <a href="#">[General]ClusterReplication</a>).</p>
Category	<p>The alarm category to which the alarm is assigned, defined as either a category number or a label (see <a href="#">Categorize Alarms</a>).</p> <p>If not specified, the category defaults to category 0.</p>
Alarm Desc	<p>A meaningful description of the alarm condition. This description is used when details of the alarm are displayed on screen or logged to a device.</p> <p>This field can support variable data (by enclosing an expression in braces).</p> <p>For example:</p> <p>"Line Broken Alarm at Line Speed {LineSpeed1}"</p> <p>See <a href="#">Add Variable Data to Alarm Messages</a>.</p>
Delay	<p>The alarm delay period (see <a href="#">Use Alarm Delay</a>).</p> <p>An alarm becomes active when the state of the triggering condition remains true for the duration of the delay period.</p> <p>The delay period needs to be entered in the following format:</p> <p>HH:MM:SS (Hours:Minutes:Seconds).</p> <p>The value needs to be between 0 seconds (00:00:00) and 24 hours (24:00:00).</p>
Help	The name of the graphics page that displays when the <code>AlarmHelp()</code> function is called by a user-defined command. If not specified no action occurs when the <code>Alarm()</code> function is called.

Field	Description
Comment	Any useful comment.

**Source Properties**

Field	Description
Variable Tag	An integer tag that represents up to eight different states in a field device.

**State Properties**

Field	Description
State Name <1> to State Name <7>	Enter a name that describes the associated state. If an alarm or event is triggered for this state, this name will appear in the <b>State</b> field of the Active Alarms page and/or the SOE page.
State Type <1> to State Type <7>	<p>Determines the action that occurs when the device transitions to the defined state. The options are:</p> <ul style="list-style-type: none"> <li>• <b>Alarm</b> - an alarm is added to the Active Alarms page and an event is logged to the SOE page. The <b>State Name</b> will appear in the <b>State</b> field.</li> <li>• <b>Event</b> - indicates an OFF state has occurred. An event is logged on the SOE page, even if an alarm state transition did not occur. The <b>State Name</b> will appear in the <b>State</b> field.</li> <li>• <b>None</b> - indicates an OFF state has occurred. The <b>State Name</b> will appear in the <b>State</b> field on both the SOE and Active Alarms pages when an alarm transition occurs.</li> </ul> <p><b>Note:</b> If an alarm is currently in an "Event" or "None" state, a further transition to a "None" state will not be reported.</p>

**Note:** The **State Type** is stored as a numeric value in the alarms database. In some circumstances, you will need to use a numeric value to represent a particular State Type. This will include:

- When using **Equipment Editor** to associate a double point status alarm with an Equipment Type item.
- When using the **Project DBF Add-in** to manually edit the alarms database.
- When using filtering in Plant SCADA Studio.
- When using the **Export All / Import All** feature in the Alarms activity.

When a numeric value is required, use one of the following: 0 = None; 1 = Event; 2 = Alarm.

**Custom Properties**

Field	Description
Custom 1 to Custom 8	<p>A user-defined string for filtering active alarms (maximum 64 characters). Used in a custom Cicode query function as search criteria, the custom alarm filter enables operators to identify and display a subset of active alarms.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>The fields are not case-sensitive and can contain 'A'..'Z', 'a'..'z', '0'..'9', and the underscore '_'.</li> <li>A custom filter cannot start with a digit.</li> </ul>

**Paging Properties**

Field	Description
Paging	<p>A read/write property that indicates whether the alarm will be paged. When the value is 1 (TRUE) the alarm will be paged. The default value is 0 (FALSE). This property can be read using alarm tag browsing and read or modified when tag properties are enabled using the tag name "myCluster.myAlarm.paging".</p>
Paging Group	<p>A read only text string that indicates the paging group to which the alarm belongs. Maximum length is 80 characters. See your third-party paging system documentation for information on how to use this Paging Group string. This property can be read using alarm tag browsing or when tag properties are enabled read using the tagname "myCluster.myAlarm.paginggroup". For example, assign the value of PagingGroup to a variable:</p> <pre>myString = myCluster.Alarm_1.paginggroup</pre>

**Security Properties**

Field	Description
Area	<p>The Area the alarm belongs to. If an operator does not have access to an area, the alarm is not visible on the alarm display. For example, if you enter <b>Area 1</b> here, operator need to have access to Area 1 (plus any necessary privileges) to acknowledge or disable this alarm.</p>

Field	Description
	The area and privilege fields defined here needs to be designed to work in conjunction. A privilege defined on a button (say) will ignore the alarm defined area.
Privilege	<p>Privilege necessary by an operator to acknowledge or disable the alarm.</p> <p>If you assign an acknowledgment privilege to an alarm, there is no need to assign a privilege to the command(s) that acknowledge the alarm. If you assign a different privilege to the commands, an operator needs to have both privileges to acknowledge the command. More importantly, the area defined here may be ignored.</p>

### Historian Properties

Field	Description
Historize	<p>This field enables you to automatically historize and publish the specified analog alarm in CitectHistorian. If you set this field to "TRUE", the variable will be included in an automated configuration process within the Historian environment. If you set the field to "FALSE" (or leave it blank), the variable will not be included.</p> <p>See <a href="#">Integration with Historian</a>.</p>

### Project Properties

Property	Description
Project	The project in which the time-stamped digital alarm is configured.

### See Also

[Alarms](#)

### Add an Alarm Category

You can use alarm categories to group the alarms in your Plant SCADA system (see [Categorize Alarms](#)). Up to 16376 alarm categories can be created.

#### To add an alarm category to a project:

1. In the **Setup** activity, select **Alarming**.

2. On the menu below the Command Bar, select **Alarm Categories**.
3. Type the required information in each column, or in the fields in the Property Grid (see below for a description of the fields).
4. Click **Save**.

You will now be able to assign alarms to the category you have created in the System Model activity (see [Assign an Alarm to an Alarm Category](#)).

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

When configuring an alarm category, do not use a blocking Cicode function in the following fields:

- ON Action
- OFF Action
- ACK Action.

A blocking function or a lengthy operation will affect the polling of alarms, and may result in slow or delayed alarm processing. Therefore, do not configure long running functions in these fields but keep the functions simple or they may not be executed correctly. To execute complicated functions, it is recommended to use TaskNew() function to initiate a new task which performs the required function in a separate task.

If the configuration of these fields is updated and reloaded by the alarm server, the new actions will be executed. However, the updated configuration can not call any new or updated user functions which did not exist before reloading, even if a new task is used to run it. If used, nothing will happen until the alarm server is restarted. Hence, confirm that the updated actions do not refer to brand new or updated functions or tasks prior to reloading the alarm server.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## **Alarm Category Properties**

### **General Properties**

Field	Description
<b>Category</b>	The alarm category as a numeric value (0-16375). The following category numbers are reserved: <ul style="list-style-type: none"><li>• Category 0 is reserved for selecting all the alarms with all categories.</li><li>• Category 254 is reserved for user-created alarm summary entries.</li><li>• Category 255 is reserved for hardware alarms.</li></ul> This field supports up to 16 characters.
<b>Priority</b>	The priority applied to alarms assigned to this alarm

Field	Description
	<p>category (0- 255). Alarm priority determines the order in which alarms are displayed, acknowledged, enabled and so on.</p> <p>Priority 1 is the highest priority, and priority 255 is the lowest. For example, if alarms with priorities 1 to 8 were displayed, priority 1 alarms would be displayed first in their time/date order, then priority 2 alarms, then priority 3, and so on up to priority 8.</p> <p>Priority 0 is the default priority. It is reserved for selecting alarms of all priorities when referencing priorities. For example, if you use the Cicode function <a href="#">AlarmSetInfo</a> to change an alarms list so that alarms of all priorities are displayed, you would use the following settings:</p> <p><i>Type = 7, Value = 0.</i></p> <p><b>Note:</b> When priority 0 is used to display alarms of priorities, priority 0 only alarms will display first, followed by priority 1 alarms, then priority 2, and so on. You can also customize the order in which alarms are displayed on an alarm summary page using the SummarySort and SummarySortMode parameters. (This order will override the alarm category priority order.)</p>
<b>Show on Active</b>	<p>Determines if alarms assigned to this category display on active alarms pages.</p> <p>You can set this field to TRUE or FALSE. The default value is TRUE.</p> <p><b>Note:</b> For Situational Awareness projects, <b>Show On Active</b> must be blank (TRUE) for associated alarm counts to appear on the Page Navigation Zone and Tree Views.</p>
<b>Show on Summary</b>	<p>Determines if alarms assigned to this category display on alarm summary pages.</p> <p>You can set this field to TRUE or FALSE. The default value is TRUE.</p>
<b>Comment</b>	Any useful comment.

**Font Properties**

Field	Description
<b>UnAck On Font</b>	Defines the font used to display alarms that are unacknowledged and active.

Field	Description
	<p>This property is optional. If no font is specified, the font defaults to 10pt YELLOW.</p> <p>This field supports up to 16 characters.</p>
<b>UnAck Off Font</b>	<p>Defines the font used to display alarms that are unacknowledged and no longer active.</p> <p>This property is optional. If no font is specified, the font defaults to 10pt BROWN.</p> <p>This field supports up to 16 characters.</p>
<b>Ack On Font</b>	<p>Defines the font used to display alarms that have been acknowledged and are still active.</p> <p>This property is optional. If no font is specified, the font defaults to 10pt CYAN.</p> <p>This field supports up to 16 characters.</p>
<b>Ack Off Font</b>	<p>Defines the font used to display alarms that have been acknowledged and are no longer active.</p> <p>This property is optional. If no font is specified, the font defaults to 10pt WHITE.</p> <p>This field supports up to 16 characters.</p>
<b>Disabled Font</b>	<p>Defines the font used to display disabled alarms.</p> <p>This property is optional. If no font is specified, the font defaults to 10pt WHITE.</p> <p>This field supports up to 16 characters.</p>

### Format Properties

Field	Description
<b>Alarm Format</b>	<p>Defines the screen display format used on active alarm pages for alarms in this category.</p> <p><b>Note:</b> Alarm pages based on the tab-style templates will not consider individual formats for each category. The screen display format used for an alarm page is set by the first available of the following definitions:</p> <ul style="list-style-type: none"> <li>• [Format]Alarm parameter</li> <li>• The Alarm Format specified for category 0</li> <li>• The default alarm display format.</li> </ul> <p>When applied to pages based on a standard template, Alarm Format specifies how the data for alarms in this category are displayed on alarms pages (on the screen only). Each alarm displays on the alarms page in a single line, for example:</p>

Field	Description
	<p>2:32:21RFP3 Raw Feed pump 3 Overload</p> <p>When alarms are displayed using variable width fonts (such as Arial or Helvetica), the alarm fields may not align properly across different rows. This can be avoided by using a field separator in the alarm format configuration instead of just a space.</p> <p>Tab characters, denoted by either "<code>^t</code>" (horizontal tab) or "<code>^v</code>" (vertical tab), between the alarm fields will act as alignment points in your alarm display.</p> <p>If you leave the Alarm Format field blank, the format defaults to:</p> <p><b>{Time,12} {Tag,10} {Name,20} {Desc,32}</b></p> <p>You can change this default setting with the parameter <b>[Alarm]DefDspFmt</b>.</p> <p>See <a href="#">Alarm Format Fields</a> for details about each field type.</p> <p><b>Note:</b> If an alarm value is longer than the field it is to be displayed in, it will be truncated or replaced with the #OVR ("overflow of format width") alert message. When the alarm is logged to a device (i.e. printed or written to a file or database), the format specified for the logging device overrides the display format.</p>
<b>Summary Format</b>	<p>Defines the screen display format used on alarm summary pages for alarms in this category.</p> <p><b>Note:</b> Alarm summary pages based on the tab-style templates will not consider individual formats for each category. The screen display format used for an alarm summary page is set by the first available of the following definitions:</p> <ul style="list-style-type: none"> <li>• [Format]Alarm parameter</li> <li>• The Alarm format specified for category 0</li> <li>• Default Alarm display format.</li> </ul> <p>When applied to pages based on a standard template, Summary Format is defined the same way as <b>Alarm Format</b> (see above). However, you can also use additional data fields.</p> <p>See <a href="#">Alarm Format Fields</a> for details about each field type.</p> <p>If you leave the Summary Format field blank, the format defaults to:</p> <p><b>{Name,20} {OnTime,8} {OffTime,8} {DeltaTime,8} {Comment,22}</b></p>

Field	Description
	<p>You can change this default setting with the parameter <a href="#">[Alarm]DefSumFmt</a>.</p> <p><b>Note:</b> When an alarm is logged to a summary device (i.e. printed or written to a file or database), the format specified for the logging device overrides the display format.</p>
<b>SOE Format</b>	<p>Defines the screen display format used on sequence of event (SOE) pages for alarms in this category.</p> <p><b>Note:</b> SOE pages based on the tab-style templates will not consider individual formats for each category. The screen display format used for an SOE page is set by the first available of the following definitions:</p> <ul style="list-style-type: none"> <li>• <a href="#">[Format]Alarm</a> parameter</li> <li>• The Alarm format specified for category 0</li> <li>• Default Alarm display format.</li> </ul> <p>When applied to pages based on a standard template, SOE Format is defined the same way as <b>Alarm Format</b> (see above). However, you can also use additional data fields.</p> <p>See <a href="#">Alarm Format Fields</a> for details about each field type.</p> <p>If you leave the SOE Format field blank, the format defaults to:</p> <pre>{DATE,16} {TIME,16} {TAG,10} {NAME,15} {MESSAGE,64} {STATE,16} {CLASSIFICATION,13} {USERNAME,16} {USERLOCATION,16}</pre> <p>You can change this default setting with the parameter <a href="#">[Alarm]DefSOEFmt</a>.</p>

### Actions Properties

Field	Description
<b>ON Action</b>	<p>Specifies a Cicode command that is executed when an alarm of this Category becomes active (ON).</p> <p><b>Example:</b> Where <b>ON Action</b> is <b>STOP_PROCESS = 1;</b> In this example the digital variable STOP_PROCESS is set to ON when an alarm in this category is triggered.</p> <p><b>Note:</b> Do not put a blocking Cicode function in this field. The alarm system executes ON, OFF, or ACK actions within the polling loop. A blocking function will affect the polling of alarms, and may result in slow</p>

Field	Description
	<p>or delayed alarm processing.</p> <p>A special case of this command occurs when the ON Action is self-referring, with a form such as TAG1 = TAG1 + 1. This command will not work properly since tags are not reread before processing the ON action (for performance reasons). This particular command will therefore initially set the value of TAG1 to 1 rather than incrementing it.</p> <p>To correctly run a command of this type in the ON Action, use TaskNew() to run your own Cicode function to perform the tag command. For example: ON Action is TaskNew("MyFunc","Data",5);</p>
<b>OFF Action</b>	<p>Specifies a Cicode command that is executed when an alarm of this Category is reset (OFF).</p> <p><b>Example:</b> Where <b>OFF Action</b> is <b>ENABLE_PROCESS = 1;</b> In this example the digital variable <b>ENABLE_PROCESS</b> is set to ON when an alarm in this category is reset.</p> <p><b>Note:</b> Do not put a blocking Cicode function in this field. The alarm system executes ON, OFF, or ACK actions within the polling loop. A blocking function will affect the polling of alarms, and may result in slow or delayed alarm processing.</p>
<b>ACK Action</b>	<p>A Cicode command that is executed when an alarm of this Category is acknowledged.</p> <p><b>Note:</b> Do not put a blocking Cicode function in this field. The alarm system executes ON, OFF, or ACK actions within the polling loop. A blocking function will affect the polling of alarms, and may result in slow or delayed alarm processing.</p>

### Storage Properties

Field	Description
<b>Alarm Acquisition Error</b>	<p>Acquisition error is an error related to acquiring data from I/O sub-system for the underlying tag and/or expression, for example, Device Offline, Tag Unknown, etc. (Integer length 6).</p> <p>All acquisition errors are stored for each alarm record separately within the alarm server as a new AcqError field. They are not logged and no hardware alarm is raised from them.</p>

Field	Description
	The AcqError field stores the error as the Plant SCADA error code (for example, NO ERROR). This field is made accessible through the alarm browse functionality. If there are multiple acquisition errors for a single alarm record, then the first of these is stored within the system.
<b>Summary Device</b>	Specified the device to which the alarm summary is sent.  The format specified in the device is used instead the display format. If not specified, alarm summaries are not logged.
<b>Log Device</b>	Specifies the device to which alarm state changes are logged. An alarm entry is made in the log device each time an alarm assigned to the category changes state (on, off, acknowledged, enabled, or disabled).  When the alarm is printed, or written to a file or device, the format specified in the device overrides the display format.  If not specified, alarm state changes are not logged.
<b>Log Transitions: ON</b>	Determines if the alarm details are logged when an alarm assigned to this category becomes active.  You can set this field to TRUE or FALSE. The default value is TRUE.
<b>Log Transitions: OFF</b>	Determines if the alarm details are logged when an alarm assigned to this category becomes inactive.  You can set this field to TRUE or FALSE. The default value is TRUE.
<b>Log Transitions: ACK</b>	Determines if the alarm details are logged when an alarm assigned to this category is acknowledged.  You can set this field to TRUE or FALSE. The default value is TRUE.

**Project Properties**

Property	Description
<b>Project</b>	The project in which the specified alarm category is included.

## See Also

[Prioritize Alarms](#)

### Use Alarm Delay

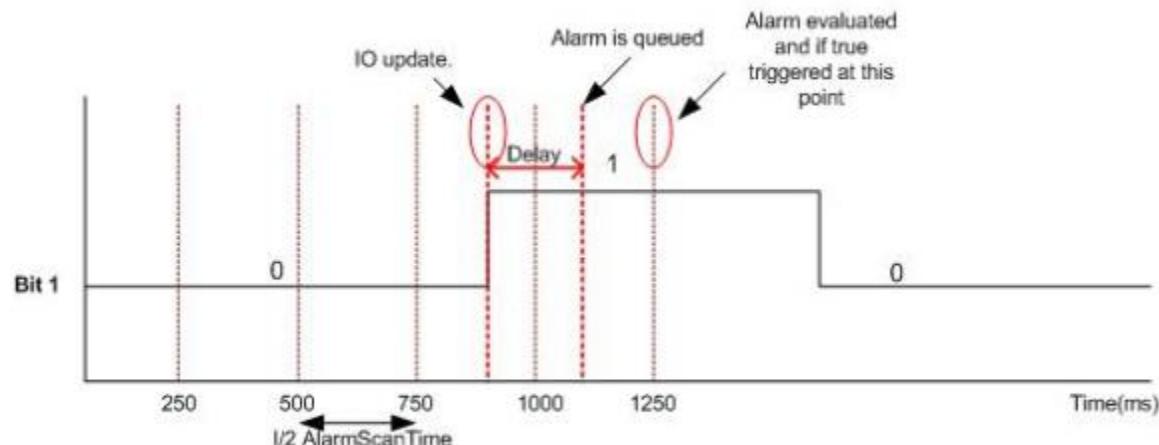
The **Alarm Delay** property allows you to configure digital, analog, and advanced alarms so that they will not be activated unless their triggering conditions remain true for a specified period.

For analog alarms, this means the analog variable needs to fall within a specified range for the duration of the alarm delay period before an alarm will activate. In the case of digital and advanced alarms, the triggering condition of the digital variable or Cicode expression need to remain true for the delay period.

#### Example

An alarm delay could be useful in monitoring temperature. By setting a delay period, you can filter out momentary alarms caused by a temperature moving in and out of different ranges.

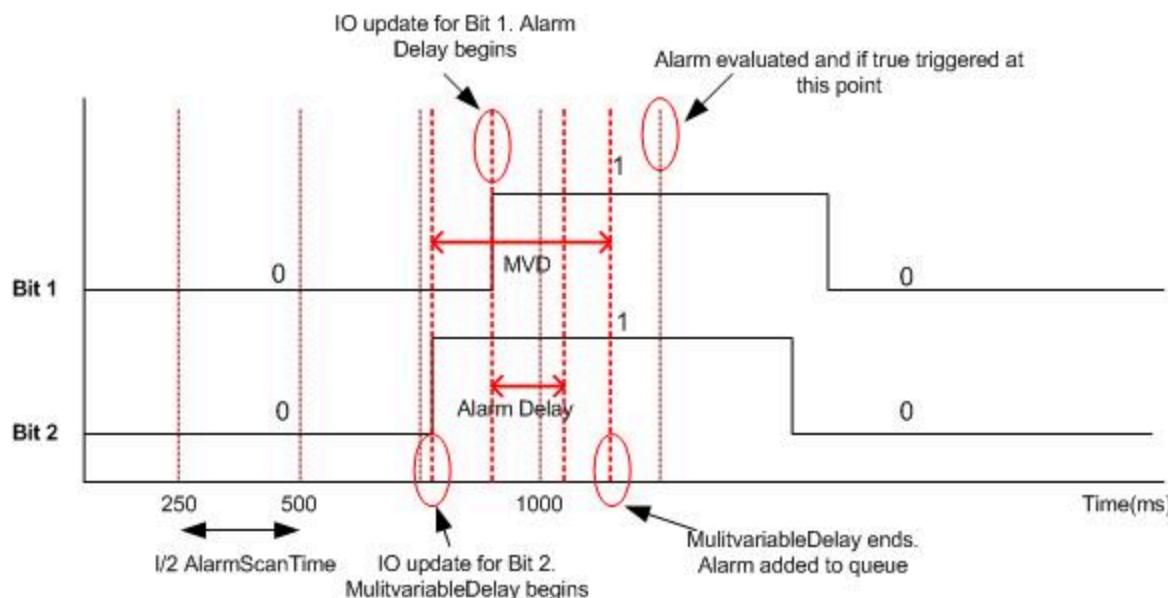
If 'Bit 1 = 1' then raise an alarm



When IO update occurs, alarm delay period will start. Once complete the alarm is queued.

For alarms with multiple variables configured, if you use the **Alarm Delay** property the delay period will start when the most recent I/O update has occurred. This delay can run concurrently with the internal multi-variable delay (MVD) which starts at the first IO update. The alarm will not be added to the queue until the period of the longest delay setting has elapsed. The alarm will then be triggered at the following 1/2 alarm scan time.

If 'Bit 1 AND Bit 2 = 1' then raise an alarm



Alarm delay starts after the most recent IO update has occurred and can run concurrently with the internal Multivariable delay (MVD), which starts at the first IO update has occurred. The alarm will not be added to the queue until the time of the longest delay setting has elapsed.

## See Also

[Add an Analog Alarm](#)

[Add a Digital Alarm](#)

[Add an Advanced Alarm](#)

## Customize Alarm Pages

The Plant SCADA starter projects generate a set of alarm pages that are designed to support the requirements of a typical control system. This section of the help describes the different ways you can customize these pages to suit the requirements of a particular project.

If required, you can add additional alarm pages to a Plant SCADA project in Graphics Builder. For instructions on how to add a new page to a project, see [Create a Graphics Page](#).

---

**Note:** To display a custom alarm page (with a non-standard name), use the [PageDisplay](#) function to display the page, followed by the [AlarmSetInfo](#) function. You can create a keyboard command or a button to call the page, or add a touch command to an existing screen object.

---

If your project includes multiple clusters, you may want to visually indicate to an operator the status of alarm data across all clusters (see [Confirm that Alarms are Updated for All Clusters](#)).

You can also customize alarm pages in the following ways:

- Adjust the way alarms display on an alarms list (see [Format an Alarm Display](#))
- Customize fonts (see [Fonts](#))

- Adjust the default sort order (see [Alarms List Default Sort Order](#)).

It is also possible to define keywords for your alarm tags that allow you to perform customized queries on your alarm data (see [Using Custom Alarm Filters](#)).

## Confirm that Alarms are Updated for All Clusters

In a multi-cluster system, alarm data may appear on a display client in different time frames due to the proximity of the alarm server associated with each cluster.

For example, an alarm page may retrieve alarm data for one cluster from a local alarm server, while the alarm server for another cluster may be located remotely on a slow network. This could cause moments where alarm data is only partially retrieved for an alarms list, an SOE list or alarm summary.

If this situation is a possibility, you can use the Cicode function [AlarmGetInfo](#) to provide a visual indication of the status of your alarm data across all clusters. By calling this function from an animation object on a page, you can indicate the following to an operator:

- If data has been retrieved for no clusters, some clusters or all clusters
- If data has been retrieved for a specific cluster
- If any timeouts have occurred.

To do this:

1. Determine where you would like to implement AlarmGetInfo. You should use the Cicode function in a way that will indicate to an operator how alarm data retrieval is progressing for all clusters (see the example below).
2. Specify the **AN** associated with the alarms list that you would like the function to monitor.
3. Specify a **Type** of 13, 14 or 16, where:
  - 13 - the return value is used to indicate if data has been retrieved for all, some, or no clusters
  - 14 - the return value is used to indicate if data has been updated for a particular cluster (specified using the ClusterName argument)
  - 16 - the return value is used to indicate if a timeout occurred.

## Example

The following call of AlarmGetInfo() could be used to determine if the alarm data for the list at AN 21 has been retrieved for all clusters.

`AlarmGetInfo(21,13)`

- If zero (0) is returned, it would indicate that data is not ready for any clusters.
- If 1 is returned, it would indicate that data is ready for only some clusters.
- If 2 is returned, it would indicate that data is now ready for all clusters.

These return values could be implemented on a graphics page to indicate the completeness of the alarms list. For example, you could create an object that represents a traffic light, where "0" displays red, "1" displays amber, and "2" displays green.

## Format an Alarm Display

You can adjust the way alarms display on an alarms list.

The display format specifies how alarms are displayed on screen for the alarms.

The way you format the appearance of your alarms lists depends on the template used to create the pages in your project.

- [Format Alarm Pages in a Starter Project](#)
- [Format Alarms in a Standard Project](#)

### See also

[Create Priority and State Symbols for an Alarms List](#)

### Format Alarm Pages in a Starter Project

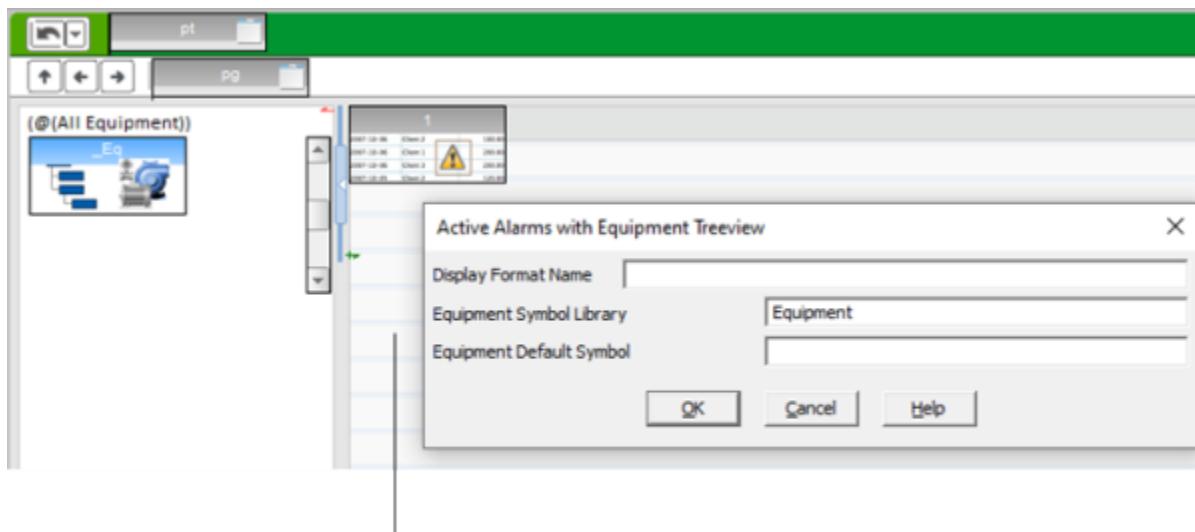
If you would like to customize the content included in each of the table columns on an alarm page, you can configure a new display format using [Alarm Format Fields](#).

To do this, you use the parameter `[Format]FormatName`. This parameter allows you to create and name a customized alarm display format that you can then associate with an alarm page.

**Note:** If you create a customized alarm display format for a page, be aware that the arrangement of columns may only be temporary. An operator will be able to rearrange the columns at runtime, and their changes will be restored each time they log in.

### To associate a customized alarm display format with an alarm page:

1. Use the `[Format]FormatName` parameter to create a customized alarm display format.
2. In Graphics Builder, open the alarm page that you would like to customize.
3. Double-click on the page. A dialog will appear displaying the page parameters.



4. In the **Display Format Name** field, enter the name of the alarm display format you created.
5. Click **OK** and **Save** the alarm page.

**Note:** The Hardware Alarm page does not display alarms in a table. To modify the alarm display format for a Hardware Alarms page, use the techniques described in the topic [Format Alarms in a Standard Project](#).

You can also use the following parameters to globally customize the alarm pages in a starter project:

- [Tab\\_Style Template Parameters](#)
- [Alarm Heading Parameters](#)

Advanced users may also use the [\[TabAlarm.Custom\] Parameters](#) to set a function that returns the value of a specified alarm field.

## See Also

[Format Alarms in a Standard Project](#)

### Format Alarms in a Standard Project

This topic describes how to format alarms in a standard project. The display format specifies how alarms are displayed on screen for the Active Alarms and Alarm Summary pages.

### Include System Data in an Alarm Field

Include data in an alarm display by specifying the field name and width for each field to display. You need to enclose each field in braces {} and use the following syntax:

{<field name>, [width[, justification]]}

For example:

Format	{Tag,8} {Name,32}
--------	-------------------

In this case, data displays in two fields: Tag, with 8 characters; and Name, with 32 characters. The width specifier is optional; if it is not used, the width of the field is determined by the number of characters between the braces.

Format	Name of Alarm:{Name }
--------	-----------------------

In this case, Name is followed by four spaces; the data {Name} displays with 8 characters.

**Note:** The screen resolution of your computer determines the total number of characters (and therefore the number of fields) that can be displayed on the alarms page.

### Include Fixed Text in an Alarm Field

You can include fixed text by specifying the text exactly as it will display; for example:

Format	Name of Alarm:
--------	----------------

Any spaces that you use in a text string are also included in the display.

## Set the Text Justification in an Alarm Field.

To set the justification of the text in each field, use a justification specifier. You can use three justification characters, L (Left), R (Right), and N (None); for example:

Format	Name of Alarm:{Name,32,R} {Tag,8,L}
--------	-------------------------------------

The justification specifier is optional; if it is omitted, the field is left justified. If you use a justification specifier, you need to also use the width specifier.

To display field text in columns, use the tab character (^t); for example:

Format	{Tag,8}^t{Name,32}^t{Desc,8}
--------	------------------------------

This format aligns the tag, name and description fields of the alarm when using a proportional font to display the alarms.

### See Also

[Alarm Format Fields](#)

### Alarm Format Fields

Alarm format fields define syntax you can use to customize the way alarms appear in an alarms list.

In a standard project, you would typically apply the syntax to the **Alarm Format** and **Summary Format** fields in the Alarm Category Properties (see [Add an Alarm Category](#)) to define how the alarms associated with a particular category will display (category zero (0) being the default for all categories).

In projects based on a starter project, some parameters allow you to use alarm format fields to customize an alarms list. For example, [\[Format\]FormatName](#).

The syntax associated with the alarm format fields are described in the following topics:

- [Alarm Display Fields](#)
- [Alarm SOE Fields](#)
- [Alarm Summary Fields](#).

### See Also

[Format an Alarm Display](#)

## Alarm Display Fields

You can use any of the fields listed below, or the [Alarm Summary Fields](#), to format an alarm display and an alarm log device (see [Format an Alarm Display](#)).

Field Name	Description
{AcqDesc,n}	Textual representation of Alarm Acquisition Error.
{AcqError, n}	Numeric representation of Alarm Acquisition Error.

Field Name	Description
{AlarmType,n}	Alarm type (string), not localized. Values are: Digital, Analog, Advanced, Multi-Digital, Time Stamped, Time Stamped Digital, Time Stamped Analog.
{AlmComment,n}	The text entered into the Comment field of the alarm properties dialog.
{Area,n}	Area <b>Note:</b> Set the value of this field between 0 and 255.
{Category,n}	Alarm Category
{Cluster,n}	Cluster Name
{CUSTOM1,n} {CUSTOM2,n} {CUSTOM3,n} {CUSTOM4,n} {CUSTOM5,n} {CUSTOM6,n} {CUSTOM7,n} {CUSTOM8,n}	Alarm custom fields as configured.
{Date,n}	The date on which the alarm changed state (dd:mm:yyyy). Be aware that you can change the format used via the parameter <a href="#">[Alarm]ExtendedDate</a> .
{DateExt,n}	The date on which the alarm changed state in extended format.
{Deadband,n}	Deadband
{Desc,n}	Alarm description
{Deviation,n}	Deviation alarm trigger value
{DisableEndTime,n}	Indicates when a shelved alarm will no longer be disabled.  This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.
{ErrDesc,n}	Text string associated with a protocol (communication) error. This field is only associated with hardware errors and contains extra information associated with whatever error is detected (for example if the error is associated with a device, the

Field Name	Description
	device name is returned; if the error is associated with a Cicode function, the function name is returned; if the error is associated with an I/O Device, the I/O Device's alert message is returned).
{ErrPage,n}	The page, device, etc. associated with the alarm.
{Format,n}	Display format of the Variable Tag
{Help,n}	Help Page
{High,n}	High Alarm trigger value
{HighHigh,n}	High High Alarm trigger value
{LocalTimeDate,n}	Alarm date and time in the form: "yyyy-mm-dd hh:mm:ss[.ttt]"
{LogState,n}	The last state that the alarm passed through. (This is useful when logging alarms to a device.)
{Low,n}	Low Alarm trigger value
{LowLow,n}	Low Low Alarm trigger value
{Name,n}	<p>Alarm Name</p> <p><b>Note:</b> If the <b>Name</b> field is configured to support long names (up to 79 characters), it might cause overlap in an alarm display. Use a smaller display font if long names are expected.</p>
{Native_Desc,n}	Alarm Description in the native language
{Native_Name,n}	<p>Alarm Name in the expression</p> <p><b>Note:</b> If the <b>Native_Name</b> field is configured to support long names (up to 79 characters), it might cause overlap in an alarm display. Use a smaller display font if long names are expected.</p>
{Paging,n}	Indicates whether the alarm has to be paged. When the value is TRUE the alarm will be paged. The default value is FALSE.
{PagingGroup, n}	Indicates the paging group to which the alarm belongs. Maximum length is 80 characters.
{Priority,n}	Alarm category's priority
{Priv,n}	Privilege

Field Name	Description
{Rate,n}	Rate of change trigger value
{State,n}	<p>The current state of the alarm. This field may be used for</p> <p>Alarm Display Only. It is not applicable to Alarm Summary.</p> <p>ON OFF DEVIATION RATE LOW LOWLOW HIGH HIGHHIGH ON State 2 ON State 3 ON State 4 ON State 5 ON State 6 ON State 7 CLEARED</p>
{State_desc, n}	The configured description (for example healthy or stopped) of a particular state. This description is entered when configuring the Multi-Digital Alarm Properties
{Tag,n}	<p>Alarm Tag</p> <p><b>Note:</b> If the <b>Tag</b> field is configured to support long names (up to 79 characters), it might cause overlap in an alarm display. Use a smaller display font if long names are expected.</p>
{TagEx,n}	<p>Alarm Tag with Cluster Name prefix</p> <p><b>Note:</b> If the <b>TagEx</b> field is configured to support long names (up to 79 characters), it might cause overlap in an alarm display. Use a smaller display font if long names are expected.</p>
{Time,n}	<p>The time at which the alarm changed state. This can be in 12 hour format (for example, h:mm:ss tt) or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.</p> <p>Set the <a href="#">[Alarm]SetTimeOnAck</a> parameter to use this field for the time the alarm is acknowledged.</p>

Field Name	Description
{Type,n}	The type of alarm or condition: ACKNOWLEDGED Cleared Disabled Unacknowledged
{TypeNum,n}	Alarm type number (use AlarmType to get string value instead). Values are: -1 Invalid 0 Digital 1 Analog 2 Advanced 3 Multi-Digital 4 ArgAna 5 User Event 6 timestamped 7 hardware 8 timestamped digital 9 timestamped analog
{Value,n}	The current value of the analog variable
{Classification,n}	The class of the event. For example: - "Action" - "Comment" - "Configuration" - "System" - "<alarm type>"
{Equipment,n}	The name of the equipment the alarm is associated with.
{Message,n}	The event message.
{Millisecond,n}	Adds milliseconds to the {Time,n} field
{RecordId,n}	String that uniquely identifies SOE records within the cluster. On the Alarm Summary table, this field references the associated SOE record.
{State,n}	State of the event. The value of this field indicates the action that triggers the event, similar to the value returned by the existing LogState field.
{UserLocation,n}	The IP of the machine which last raised, or performed an action on alarm.

Where n specifies the display field size.

**Note:**

- Any of the above fields can be displayed for any type of alarm. Where not applicable for a particular alarm type, zero or an empty string will be displayed.
- If an alarm value is longer than the field it is to be displayed in (n), it will be truncated or replaced with the #OVR ("overflow of format width") alert message.
- For summary pages use {SumState}. To log the state to a device, use {LogState}. State is the current state of the alarm, SumState is the state of the alarm when it occurred, and Log State is the state of the alarm at the transition.

**See Also**[Alarm Summary Fields](#)

## Alarm SOE Fields

You can use any fields listed below (or a combination) to format an alarm SOE display. Those fields specific to the event journal can be archived.

Format the alarm SOE for an entire category of alarms by specifying field names in the **SOE Format** field of the Alarm Category Properties dialog box.

**Note:** To change the font and format of the system events on the SOE page, use alarm category 0 (zero).

You can also use the [\[Alarm\]DefSOEFmt](#) parameter to format the alarm SOE, particularly if your alarm SOE formats are to be the same.

Field Name	Description	Archived
{AcqDesc,n}	Textual representation of Alarm Acquisition Error.	No
{AcqError,n}	Acquisition error. Separate event (error code) in SOE.	No
{AlarmType,n}	Alarm type (text). "Digital", "Analog", "Advanced", "Multi-Digital", "Time Stamped", "Time Stamped Digital", "Time Stamped Analog"	No
{AlmComment,n}	Alarm comment	No
{Area,n}	Alarm area. Numeric value (integer)	No
{Category,n}	Alarm category. Numeric value (integer)	No
{Classification,n}	The class of the event. For example: Action — The event was logged due to an action being performed. An	Yes

Field Name	Description	Archived
	<p>action can be triggered manually by an operator, or automatically.</p> <p>Comment — The event was logged due to a comment being created.</p> <p>Configuration — The event was logged due to configuration changes being made (for example, a new item being created, the configuration of an existing item being modified, an item being renamed or deleted from the database).</p> <p>System — The event has been logged due to an occurrence that affects the entire system, such as the server being stopped and restarted. The System category is also used for miscellaneous events that are not suited to the other categories.</p> <p>Alarm type - Type of Alarm (for example, digital, multi-digital etc.)</p>	
{Cluster,n}	The cluster to which the tag belongs.	No
{CUSTOM1,n} {CUSTOM2,n} {CUSTOM3,n} {CUSTOM4,n} {CUSTOM5,n} {CUSTOM6,n} {CUSTOM7,n} {CUSTOM8,n}	Alarm custom fields from 1 to 8	No
{Date,n}	Part of event record time in event journal.	Yes
{DateExt,n}	Part of event record time in event journal.	Yes
{DeadBand,n}	Alarm deadband. For Analog, and Timestamped Analog alarms: Numeric value (real).	No
{Deviation,n}	Alarm deviation. For Analog, and	No

Field Name	Description	Archived
	Timestamped Analog alarms: Numeric value (real).	
{Desc,n}	Alarm Description	Yes
{Equipment,n}	The name of the equipment the alarm is associated with.	No
{FullName,n}	The full name of the user (Full Name) who was logged on and performed some action on the alarm (for example acknowledging the alarm or disabling the alarm, etc.). When the alarm is first activated, the full name is set to "system" (because the operator did not trip the alarm). This field is not localized.	No
{Group,n}	Alarm group. For Argyle Digital (Multi-digital) alarms: Numeric value.	No
{Help,n}	Alarm help. The name of the help page.	No
{High,n}	High Alarm trigger value	No
{HighHigh,n}	High High Alarm trigger value	No
{LocalTimeDate,n}	Alarm date and time in the form: "yyyy-mm-dd hh:mm:ss[.sss]"	Yes
{LogState,n}	The last state that the alarm passed through. (This is useful when logging alarms to a device.)	Yes
{Low,n}	Low Alarm trigger value	No
{LowLow,n}	Low Low Alarm trigger value	No
{Message,n}	The event message.	Yes
{MilliSec,n}	Alarm milliseconds. Numeric value (integer)	Yes
{Name,n}	Alarm Name	No
{Native_SumDesc,n}	A description of the alarm	No

Field Name	Description	Archived
	summary, in the native language.	
{Paging,n}	Alarm paged flag. A flag to indicate that the alarm is going to be paged. Values are 1 (TRUE) or 0 (FALSE).	No
{PagingGroup,n}	Paging group for alarm. A free form text field indicating the sequence of people to notify in the event the alarm occurred.	No
{Priority,n}	Alarm category priority. Numeric value (integer)	No
{Priv,n}	Alarm privilege. Numeric value (integer)	No
{Quality, n}	The I/O server will pass the quality of the tag to the Alarm server when a tag's value is changed. If the changed tag value causes a state transition the alarm server will pass the quality to the database. This field displays the raw value (2 bytes numeric). The field TSQuality relates to this field and displays high byte of the 2 byte value in a string.	No
{Rate,n}	Alarm rate. For Analog, and Timestamped Analog alarms: Numeric value (real)	No
{ReceiptDate,n}	Date the master station received the event. For example, Date the RTU received the event.	
{ReceiptLocalDateTime,n}	This field displays the alarm's occurrence time (if known), otherwise it displays the receipt time of the alarm.	
{ReceiptTime,n}	Time the event was logged into the database. This can be in 12 hour format (for example, h:mm:ss tt) or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on	No

Field Name	Description	Archived
	the local computer.	
{State,n}	Current state of alarm.	Yes
{State_Desc,n}	The configured description (for example healthy or stopped) of a particular state. This description is entered when configuring the Multi-Digital Alarm Properties.	No
{State_Desc0,n} {State_Desc1,n} {State_Desc2,n} {State_Desc3,n} {State_Desc4,n} {State_Desc5,n} {State_Desc6,n} {State_Desc7,n}	Argyle state 0-7. For Argyle Digital (Multi-digital) alarms: "OFF". For others: "INVALID"	No
{SumDesc,n}	Alarm description text. Analog alarm - Alarm state text. Otherwise - Configured event [or alarm in case event not defined] description.	No
{SumState,n}	Describes the state of the alarm when it occurred.	No
{Tag,n}	Alarm Tag <b>Note:</b> If the Tag field is configured to support long names (up to 79 characters), it might cause overlap in an alarm display. Use a smaller display font if long names are expected.	Yes
{TagItem,n}	Alarm Tag with Cluster Name prefix <b>Note:</b> If the TagEx field is configured to support long names (up to 79 characters), it might cause overlap in an alarm display. Use a smaller display font if long names are expected.	No
{TagEx,n}	Alarm source	Yes
{Time,n}	This can be in 12 hour format (for example, h:mm:ss tt) or 24 hour format (for example, H:mm:ss)	Yes

Field Name	Description	Archived
	<p>depending on the regional date and time settings on the local computer.</p> <p><b>Note:</b> This works between UTC time 1/1/1980 - UTC time 31/12/2037, otherwise "".</p>	
{TimeDate,n}	<p>Special SQL formatted time. HH:MM:SS.</p> <p><b>Note:</b> The format can be configured in the Citect.ini file by specifying TimeDate under the section [Alarm].</p>	Yes
{TSQuality,n}	<p>The IO server will pass the quality of the tag to the Alarm server when a tag's value is changed. If the changed tag value causes a state transition the alarm server will pass the quality to the database. The field displays high byte of the 2 byte value (available under the Quality field) in a string . Expected Timestamp Quality strings are: Time Good, Time Uncertain, Clock Not Synchronized, or empty string if no match is found.</p>	Yes
{Type,n}	<p>The type of alarm or condition: ACKNOWLEDGED, DISABLED, UNACKNOWLEDGED</p>	Yes
{TypeNum,n}	<p>Alarm type number (use AlarmType to get string value instead). Values are: -1 Invalid, 0 Digital 1, Analog 2, Advanced 3, Multi-Digital 4, ArgAna 5, User Event 6, timestamped 7, hardware 8, timestamped digital 9, timestamped analog</p>	No
{UserLocation,n}	<p>The IP address where the event was triggered.</p>	Yes
{UserName,n}	<p>The name of the user (User Name) who was logged on and performed some action on the alarm (for example acknowledging the alarm or disabling the alarm, etc.). When</p>	Yes

Field Name	Description	Archived
	the alarm is first activated, the user name is set to "system" (because the operator did not trip the alarm).	

Where "n" specifies the display field size.

## See Also

[Alarms List Default Sort Order](#)

## Alarm Summary Fields

You can use any fields listed below (or a combination) to format an alarm summary display and an alarm summary device.

Format the alarm summary for an entire category of alarms by specifying field names in the **Summary Format** field of the Alarm Category Properties dialog box (category zero (0) being the default for all categories).

You can also use the [\[Alarm\]DefSumFmt](#) parameter to format the alarm summary, particularly if your alarm summary formats are to be the same.

Field Name	Description
{AckDate,n}	The date when the alarm was acknowledged.
{AckDateExt,n}	The date (in extended format) when the alarm was acknowledged (dd/mm/yyyy).
{AckTime,n}	The time when the alarm was acknowledged. This can be in 12 hour format (for example, h:mm:ss tt) or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.
{DeltaMilli,n}	Adds milliseconds precision to the DeltaTime.
{DeltaTime,n}	The time difference between OnDate/OnTime and OffDate/OffTime, in seconds.
{FullName,n}	The full name of the user (Full Name) who was logged on and performed some action on the alarm (for example acknowledging the alarm or disabling the alarm, etc.). When the alarm is first activated, the full name is set to "system" (because the operator did not trip the alarm).
{Native_SumDesc,n}	A description of the alarm summary, in the native language.

Field Name	Description
{OffDate,n}	The date when the alarm returned to its normal state.
{OffDateExt,n}	The date (in extended format) when the alarm returned to its normal state (dd/mm/yyyy).
{OffMilli,n}	Adds milliseconds to the time the alarm returned to its normal state.
{OffTime,n}	The time when the alarm returned to its normal state. This can be in 12 hour format (for example, h:mm:ss tt) or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.
{OnDate,n}	The date when alarm was activated.
{OnDateExt,n}	The date (in extended format) when the alarm was activated (dd/mm/yyyy).
{OnMilli,n}	Adds milliseconds to the time the alarm was activated.
{OnTime,n}	The time when the alarm was activated. This can be in 12 hour format (for example, h:mm:ss tt) or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.
{SumDesc,n}	A description of the alarm summary.
{SumState,n}	Describes the state of the alarm when it occurred.
{SumType,n}	Type of alarm summary (similar to alarm "Type"). Values are ACKNOWLEDGED, DISABLED, UNACKNOWLEDGED.
{UserDesc,n}	The text related to the user event. If you specify an asterisk '*' as the first letter of the tag argument in AlarmSumAppend, the field value is the text following the asterisk '*' in the tag argument.
{UserLocation,n}	The IP of the machine which last raised, performed an action on or modified the alarm. Note: If the last action performed is from a System machine, the IP will not over ride the last user machine IP address.
{UserName,n}	The name of the user (User Name) who was logged on and performed some action on the alarm (for example, acknowledging the alarm, disabling the alarm, and so on). When the alarm is first activated,

Field Name	Description
	the user name is set to "system" (because the operator did not trip the alarm).
{Value,n}	Formatted alarm value.

Where n specifies the display field size

**Note:** You can also include in your Alarm Summary any alarm display field other than **State**.

## See Also

[Alarms List Default Sort Order](#)

### Create Priority and State Symbols for an Alarms List

Plant SCADA allows you to add dynamic symbols to an alarms list that indicates the priority and state of an alarm.

Date	Time	Name	Priority	State
21/11/2017	02:28:34 PM	Drive1_V_Feedback_H_P3	3	HIGH
21/11/2017	02:28:34 PM	Drive1_OP_TRK_H_P2	2	HIGH
21/11/2017	02:28:34 PM	Drive1_V_OP_HH_P1	1	HIGH HIGH
8/11/2017	11:29:03 AM	Drive1_Running_False_P1	1	ON
Priority and State				

To achieve this, you need to perform the following tasks.

#### Create a Genie for each alarm priority that you want highlight with a symbol

The Genie requires a root level object named "Shape".

1. In Graphics Builder, create a new Genie.
2. Add the object you would like use as a symbol.  
A small, simple shape that supports a fill color is recommended.
3. Display the Properties dialog for the object.
4. On the **Appearance | General** tab, select **Filled**.
5. On the **Fill | Color** tab, configure the following:
  - Set the **Type** to **Array**.
  - Enter the following **Array Expression**:
 

```
dspAnGetMetaData(dspGetAnCur(), "State")
```
  - Specify the **Array Colors** you want to use for 6 – 9, based to the following states:
    - 6 = On and Unacknowledged
    - 7 = Off and Unacknowledged

8 = On and Acknowledged

9 = Disabled/Shelved.

If required, you can configure flashing colors for a state via the [Edit Favorite Colors Dialog Box](#), which is accessible via the **Edit** button on the drop-down Color Picker.

**Note:** Array colors 1 - 4 are reserved for use with the flag on an Alarm Indicator.

6. On the **Access | General** tab, enter the **Name** "Shape".
7. On the **Metadata | General** tab, enter the following items:
  - Name: Equipment; Value: <blank>
  - Name: Label; Value: <blank>
  - Name: State; Value: 0
8. Click **OK**.

Alarm flag Genies use functionality provided by the Cicode function DspSym. You need to perform the remaining steps to enable DspSym support for the Genie.

1. Display the Page Properties for the Genie. To do this:

On the **File** menu, select **Properties**.

Or:

Right-click on the Genie background and select **Page Properties**.

2. On the Page Properties dialog, select **I want to use this Genie with the Cicode function DspSym**.
3. Click **OK**.
4. **Save** the Genie to a library.

**Note:** When saving an alarm flag Genie, use a name that is unique across all included projects that share the same primary project.

## Associate a Genie with an alarm priority

1. In Plant SCADA Studio's **Setup** activity, select **Alarming**.
2. On the menu below the Command Bar, select **Alarm Priorities**.
3. Locate the Alarm Priority you would like to associate with a Genie.
4. In the **Small Genie Name** field, enter the name of the Genie you would like to associate with the alarm priority.
5. In the **Library Name** field, enter the name of the library that includes the Genie.
6. Click **Save**.

## Add a Priority and State column to an alarms list

To add a Priority and State column to an alarms list, you need to add the display field "{PriorityAndState}" to the alarm format used by the list.

For example, in a project created from a Situational Awareness Starter Project, the parameter [\[Format\]FormatName](#) is used as a project database parameter to define the following alarm formats:

Format Name	Display Fields
Alarm	{PriorityAndState,50}{Date,80}{Time,90}{Name,250}{Cluster,100}{State,40}{Type,70}{UserName,100}{Tag,250}{AlmComment,250}{Priority,60}{Quality,60}
Top5Alarm	{PriorityAndState,24}{Date,90}{Time,90}{Name,220}{Desc,300}{State,50}
SOE	{PriorityAndState,50}{Date,160}{Time,160}{Tag,100}{Message,640}{State,160}{Classification,130}{UserName,160}{UserLocation,160}
TopActiveAlarms_UHD4K	{PriorityAndState,24}{Date,120}{Time,130}{Name,190}{Desc,180}{State,50}
InfoAlarm_HD1080	{PriorityandState,24}{Time,90}{Item,160}{Name,180}

This places the Priority and State symbol in the first column of these alarm lists.

You can edit the alarm formats for a Situational Awareness project in the **Setup** activity (see [Project Database Parameters](#)).

For other projects, see [Format an Alarm Display](#).

A default set of priority and state symbols are provided in the SA\_Include project in a library called "sa\_priorities" (see [Alarms](#)). They represent the top three alarm priorities, and are named as follows:

- Priority 1 – "sa\_p1\_small"
- Priority 2 – "sa\_p2\_small"
- Priority 3 – "sa\_p3\_small".

---

**Note:** The Genie "sa\_p4\_small" is provided in the "sapriorities" library of the SA\_Controls project to represent alarms in the fourth-highest priority. If you would like to use this additional Genie, you need to correctly configure the fourth alarm priority. See [Configure Display Properties for a Fourth Alarm Priority](#).

## See Also

[Add an Alarms List to a Page](#)

[Configure Display Properties for an Alarm Priority](#)

## Add Variable Data to Alarm Messages

The **Alarm Desc** field of digital, advanced and time-stamped alarms can be used to display variable data. An expression (variable tag, function etc.) can be embedded into the text of the **Alarm Desc** field. This expression is evaluated when the alarm goes ON/OFF, returning the instant value of any associated variable tags.

Enclosing the expression in braces separates the variable data from the static text. For example:

Alarm Desc	Line Broken Alarm at Line Speed {LineSpeed1}
------------	--

When *LineSpeed1* is a variable tag, this expression will produce the following output on the alarm display or

alarm log:

Line Broken Alarm at Line Speed 1234

The following alarm entry uses an expression instead of a tag:

Alarm Desc	High Level at Total Capacity {Tank1+Tank2+Offset()}
------------	---

When *Tank1* and *Tank2* are variable tags, and *Offset* is a Cicode function, this expression produces the following output:

**High Level at Total Capacity 4985 liters**

**Note:** The result is formatted according to the formatting specified for the first variable tag in the expression. Standard variable formatting specifiers can be used to define the format for the numeric variable, over-riding the default format specified in Variable Tags.

## See Also

[Alarm Display Fields](#)

### Alarms List Default Sort Order

The default sort order for an alarms list depends on the type of alarm page and the [\[Alarm\]SortMode](#) parameter. In the case of the Alarm Summary page, the equivalent parameter is [\[Alarm\]SummarySortMode](#).

Type of view	[Alarm] SummarySortMode	Default sorting order
Sequence of events	0	{TIME,Descending}
Sequence of events	1	{TIME,Ascending}
Alarm Summary	0	{Ack,Ascending}{Active,Descending} {TIME,Descending}
Alarm Summary	1	{Ack,Ascending}{Active,Descending} {TIME,Ascending}

Whatever sort order you specify, the system will append the following default set of order by fields:

Alarm list display	Appended default sort order
Sequence of events	RecordTime (ASC or DESC, depending on [Alarm]SortMode) SeqNo (DESC) Cluster (ASC) CommentNo (ASC)
Alarm Summary	As per [Alarm]SummarySort and [Alarm]SummarySortMode
Active, Disabled, Acknowledged, Off, On, Unacknowledged	(ASC) Active (DESC) As per [Alarm]Sort and [Alarm]SortMode

Alarm list display	Appended default sort order
Configuration (non-hardware alarms in system)	Tag (ASC)

### Example

According to the order you specified, if two records result in being equal, the records will adhere to the default sort orders above. In the example where tags have no equipment specified (and therefore are ordered equally) the alarms will be ordered according to Ack, Active, Time.

If you specify Active (ASC), then the resultant order on the active page will be:

Active (ASC), Acknowledged (ASC), [Alarm]Sort / [Alarm]SortMode.

You can customize the order in which alarms are displayed on the alarm summary page by using the SummarySort and SummarySortMode parameters.

The SummarySort parameter allows you to display alarms according to OnTime, OffTime, and AckTime. SummarySortMode determines if the alarms will be arranged in ascending or descending order. (The order set using these parameters will override the alarm category priority order.)

## Action Alarms at Runtime

An operator can perform the following actions on an alarm page at runtime.

- [Acknowledge Alarms](#)
- [Disable Alarms](#)
- [Enable a Disabled Alarm](#)
- [Shelve Alarms](#)
- [Display Cause and Response Information](#)
- [Filter Alarms](#)
- [Sort Alarms](#)
- [Add a Column to an Alarms List](#)
- [Save and Restore an Alarms List View](#)
- [Export an Alarms List](#)
- [Print an Alarms List](#)
- [Add a Comment to the SOE Page](#)
- [Add an Event to the SOE Page.](#)

### Acknowledge Alarms

When an operator acknowledges an alarm, it indicates that they have viewed the alarm and accepted responsibility for resolving it. The appearance of the font used to describe the alarm will change to let others know that the alarm is already being investigated.

### Acknowledge alarms in a Situational Awareness project

Alarms can be acknowledged during runtime in the following locations:

- Active Alarms page (see Default Alarm Pages)
- Hardware Alarms page (see Default Alarm Pages)
- Active Alarms Zone (UHD4K projects only)
- Top 5 Active Alarm list on the Header Bar
- Information Zone Alarms tab.

#### To acknowledge a selected alarm:

1. In an alarms list, right-click the alarm that you would like to acknowledge.
2. Select **Acknowledge** from the menu that appears.

	On Date	On Time	Name	State	Cluster
◇	2/5/2021	05:17:57 PM	TS03 Flavoured Milk Temperature Hi...	OFF	Cluster1
◇	2/5/2021	04:48:01 PM	TS02 Skim Milk Temperature High Hi...	OFF	Cluster1
◇	2/5/2021	04:42:57 PM	TS01 Milk Temperature Low Hi...	High	Cluster1
△	2/5/2021	05:29:14 PM	AC01 Compressor 1 Pressure Low	ON	Cluster1
△	2/5/2021	05:29:14 PM	AC01 Compressor 1 Pressure Low	ON	Cluster1
△	2/5/2021	05:26:52 PM	TS01 Milk Temperature Low Low	Low	Cluster1
△	2/5/2021	05:26:39 PM	CM01 Compressor 1 Motor Current	High	Cluster1
△	2/5/2021	05:26:39 PM	CM01 Compressor 1 Motor Current	OFF	Cluster1

A context menu is displayed over the third row of the table, listing options: Navigate, Find In Model, Acknowledge, Shelve for..., and Shelve until... The 'Acknowledge' option is highlighted with a blue background.

#### Acknowledge alarms in a project based on the SxW templates

Alarms can be acknowledged during runtime on the following alarm pages:

- Active Alarms
- Hardware Alarms.

#### To acknowledge selected alarms:

1. On the Active Alarms or Hardware Alarms page, right-click the alarm(s) that you would like to acknowledge.
2. Select **Acknowledge** from the menu that appears.

The screenshot shows the 'Active Alarms' page with a list of alarms. A context menu is open over the 5th alarm in the list, which has the following options:

- Acknowledge
- Disable
- Disable for ...
- Disable until ...
- Information
- Cause and Response

Date	Time	Tag	Name
2/5/2021	05:37:58 PM	TopMilk_TS03_PV...	TS03 Flavoured Mil
2/5/2021	05:37:51 PM	TopMilk_TS01_PV_H	TS01 Full Milk Terr
2/5/2021	05:37:46 PM	TopMilk_TS03_PV_H	TS03 Flavoured Mil
2/5/2021	05:29:14 PM	TopMilk_AC01_PV_L	AC01 Flavoured Mil
2/5/2021	05:29:14 PM	TopMilk_AC01_PV_L	AC01 Flavoured Mil
2/5/2021	05:29:14 PM	TopMilk_AC01_PV_L	AC01 Flavoured Mil
2/5/2021	05:36:59 PM	TopMilk_TS01_PV_H	TS01 Full Milk Terr
2/5/2021	05:36:57 PM	TopMilk_TS03_PV_H	TS03 Flavoured Mil
2/5/2021	05:36:49 PM	TopMilk_TS03_PV_H	TS03 Full Milk Terr

### To acknowledge all alarms on a page:

1. Select **Acknowledge Page** on the page Command Bar. You can just click on the Command Bar button, or select the command from the button's drop-down menu.

The screenshot shows the 'Active Alarms' page with a list of alarms. A context menu is open over the 5th alarm in the list, with the following options:

- Acknowledge page
- Disable Page
- Disable page for ...
- Disable page until ...

Date	Time	Tag	Name
2/5/2021	05:29:14 PM	TopMilk_AC01_PV_L	AC01 Flavoured Mil
2/5/2021	05:29:14 PM	TopMilk_AC01_PV_L	AC01 Flavoured Mil
2/5/2021	05:29:14 PM	TopMilk_AC01_PV_L	AC01 Flavoured Mil
2/5/2021	05:38:27 PM	TopMilk_TS01_PV_H	TS01 Full Milk Terr
2/5/2021	05:38:24 PM	TopMilk_TS03_PV_H	TS03 Flavoured Mil
2/5/2021	05:38:14 PM	TopMilk_TS03_PV_H	TS03 Flavoured Mil
2/5/2021	05:36:59 PM	TopMilk_TS01_PV_L	TS01 Full Milk Terr

### Acknowledge alarms in a project based on the Tab Style templates

Alarms can be acknowledged during runtime on the following alarm pages:

- Active Alarms
- Hardware Alarms.

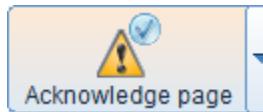
**To acknowledge selected alarms:**

1. On the Active Alarms or Hardware Alarms page, right-click the alarm(s) that you would like to acknowledge.
2. Select **Acknowledge** from the menu that appears.

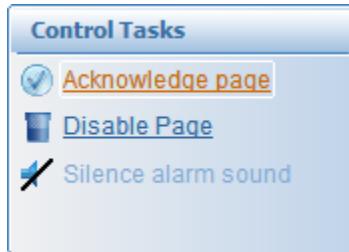
Date	Time	Tag	Name
2/5/2021	05:16:14 PM	TopMilk_TS02_PV_L	TS02 Skim Milk Temp
2/5/2021	05:01:58 PM	TopMilk_AC01_PV_L	AC01 Full Milk Level
2/5/2021	05:01:58 PM	TopMilk_AC01_PV_L	AC01 Full Milk Level
2/5/2021	05:01:58 PM	TopMilk_AC02_PV_L	AC02 Skim Milk Level
2/5/2021	05:01:58 PM	TopMilk_AC02_PV_L	AC02 Skim Milk Level
2/5/2021	05:15:26 PM	TopMilk_TS03_PV_L	TS03 Flavoured Milk Temp
2/5/2021	05:15:23 PM	TopMilk_TS03_PV_L	TS03 Flavoured Milk Temp
2/5/2021	05:15:20 PM	TopMilk_TS03_PV_L	TS03 Flavoured Milk Temp
2/5/2021	05:11:36 PM	TopMilk_TS03_PV_L	TS03 Flavoured Milk Temp
2/5/2021	05:06:55 PM	TopMilk_TS03_PV_L	TS03 Flavoured Milk Temp
2/5/2021	05:02:00 PM	TopMilk_TS03_PV_L	TS03 Flavoured Milk Temp

**To acknowledge all alarms on a page:**

1. Go to the **Action** tab and click on the **Acknowledge Page** button.



If the Action tab is not available, it means the page was created using an early version of the alarm page template. If this is the case, the **Acknowledge Page** command will available via the **Control Tasks** panel.

**Acknowledge alarms in a project based on a standard template**

Alarms can be acknowledged during runtime on the following alarm pages:

- Active Alarms

- Hardware Alarms.

**To acknowledge an alarm:**

1. Place the cursor over an alarm and press the **Enter** key.

**To acknowledge all alarms on a page:**

1. You can acknowledge every alarm on a page by clicking on the **Acknowledge All Alarms** button.



---

**Note:** In projects based on the SxW or Tab Style templates, the ability to acknowledge alarms may be restricted by the privilege level set in the parameter [Privilege]AckAlarms.

---

**Disable Alarms**

If an alarm does not appear to be operating as designed, or is determined to be unnecessary, an operator can disable it. A disabled alarms is ignored by the alarm system until it is returned to an enabled state (see [Enable a Disabled Alarm](#)).

---

**Note:** If the parameter [\[Alarm\]DisplayDisable](#) is set to 1, the system will continue to process state changes for alarms that are disabled. This means any state changes can continue to be displayed at runtime, for example, on the SOE and Disabled Alarms page. They will not display on the Active Alarms page.

---

If your project uses the SxW or Tab Style templates (accessible via a starter project), you can disable alarms during runtime from the Active Alarms page. Be aware that this may be restricted by the privilege level set in the parameter [\[Privilege\]DisableAlarms](#).

In a standard project, you need to define a command that uses the [AlarmDisable](#) function to disable alarms.

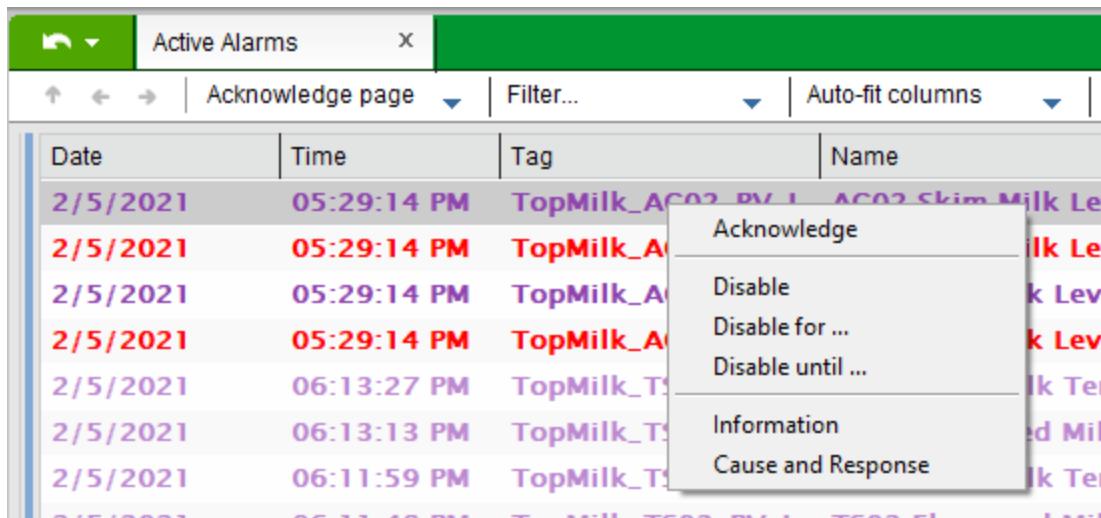
---

**Note:** You can also "shelve" an alarm, which will temporarily disable it for a specified period of time. In a Situational Awareness project, only shelving is supported. You are not able to permanently disable an alarm. See [Shelf Alarms](#).

---

**Disable alarms in a project based on the SxW templates****To disable selected alarms:**

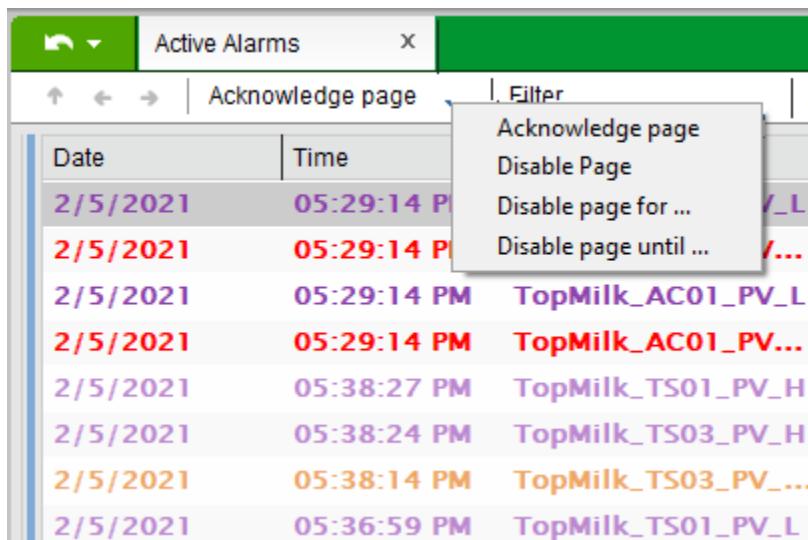
1. On the Active Alarms page, right-click on the alarm(s) that you would like to disable.
2. Select **Disable** from the menu that appears.



The selected alarm(s) will be removed from the Active Alarms page.

#### To disable all alarms on the Active Alarms page:

1. Display the drop-down menu for the **Acknowledge Page** Command Bar button.
2. Select **Disable Page** from the menu that appears.



#### Disable alarms in a project based on the Tab Style templates

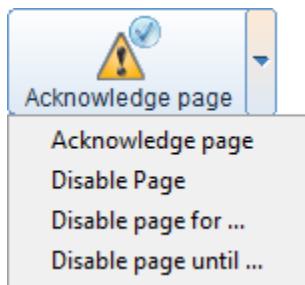
##### To disable selected alarms:

1. On the Active Alarms page, right-click the alarm(s) that you would like to disable.
2. Select **Disable** from the menu that appears.

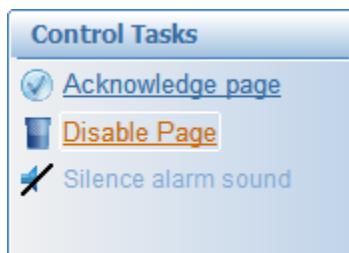
Date	Time	Tag	Name
2/5/2021	06:18:03 PM	TopMilk_TS02_PV_...	TS02 Skin
2/5/2021	06:17:55 PM	TopMilk_TS03_PV_H	TS03 Flav
2/5/2021	06:17:50 PM	To...	Skin
2/5/2021	06:17:44 PM	To...	Full
2/5/2021	05:29:14 PM	To...	Skin
2/5/2021	05:29:14 PM	To...	Skin
2/5/2021	05:29:14 PM	To...	Full
2/5/2021	05:29:14 PM	To...	Full
2/5/2021	06:16:59 PM	To...	Skim
2/5/2021	06:16:59 PM	TopMilk_TS03_PV_L	TS03 Flav
2/5/2021	06:16:49 PM	To...	TS01 F...

### To disable all alarms on a page:

1. Go to the Action tab and display the drop-down menu for the **Acknowledge Page** button.
2. Select **Disable Page** from the menu that appears.



If the Action tab is not available, it means the page was created using an early version of the alarm page template. If this is the case, the **Disable Page** command will available via the **Control Tasks** panel.



**Note:** If Plant SCADA is installed on an English operating system and you want to provide a comment about a disabled alarm using a Unicode language (such as Korean, Russian or Chinese), you will need to change the Windows™ Region setting for the runtime computer. To do this, go to Windows Control Panel and open the Region dialog box. On the Administrative tab, use the Change system locale button to select the required system locale. Be aware that you will have to restart your computer. When you launch Plant SCADA runtime, select the

matching language on the login form. Your runtime comments will be recorded using the specified language characters.

## Enable a Disabled Alarm

The Disabled Alarms page displays a list of the alarms that have been disabled by an operator (see [Disable Alarms](#)).

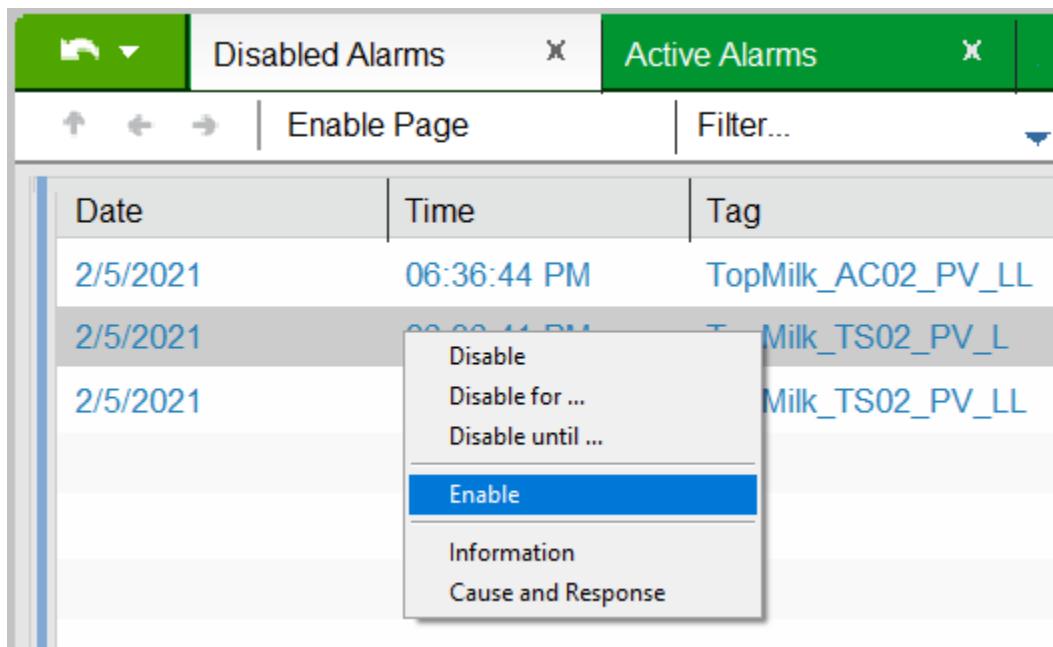
If your project uses the SxW or Tab Style templates (accessible via a starter project), you can re-enable an alarm using commands that are accessible via the Disabled Alarms page.

To enable an alarm in a standard project, you need to define a command that uses the [AlarmEnable](#) Cicode function.

## Enable alarms in a project based on SxW templates

### To enable an alarm:

1. On the Disabled Alarms page, right-click the alarm that you would like to enable.
2. Select **Enable** from the menu that appears.



The selected alarm will be removed from the Disabled Alarms page.

### To enable all alarms on the Disabled Alarms page:

1. In a project based on the SxW templates, select **Enable Page** on the page Command Bar.

## Enable alarms in a project based on the Tab Style templates

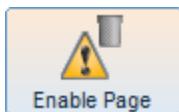
### To enable an alarm:

1. On the Disabled Alarms page, right-click the alarm that you would like to enable.
2. Select **Enable** from the menu that appears.

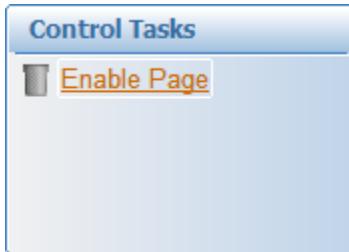
The selected alarm will be removed from the Disabled Alarms page.

### To enable all alarms on the Disabled Alarms page:

1. Go to the **Action** tab and select **Enable Page**.



If the Action tab is not available, it means the page was created using an early version of the disabled alarm page template. If this is the case, the **Enable Page** command will available via the **Control Tasks** panel.



**Note:** In projects based on the SxW or Tab Style templates, the ability to enable alarms may be restricted by the privilege level set in the parameter [\[Privilege\]DisableAlarms](#).

## Shelve Alarms

Alarm shelving allows an operator to temporarily disable nuisance or low value alarms that are causing an unnecessary distraction.

With Plant SCADA, you can shelve alarms by setting the *EndTime* argument for the following Cicode functions:

- [AlarmDisable](#)
- [AlarmDisableRec](#)
- [AlarmDisableTag](#)
- [AlmBrowseDisable](#)
- [AlmSummaryDisable](#)
- [AlmTagsDisable](#).

When the specified *EndTime* is reached, the alarm is automatically enabled and normal operation is resumed.

If you apply a new *EndTime* value to an alarm that is already shelved, the current setting will be overwritten; the values will not accumulate.

**Note:**

- 
- If the parameter [\[Alarm\]DisplayDisable](#) is set to 1, the system will continue to process state changes for alarms that are disabled or shelved. This means any state changes can continue to be displayed at runtime, for example, on the SOE and Disabled Alarms page.
  - If [\[Alarm\]DisplayDisable](#) is set to 1, the User Name and User Location recorded at the time an alarm was disabled will be maintained, despite any subsequent state changes; these properties will not be overwritten by the System User account.
  - If you would like an alarm list to display "Shelve End Time" in the header row instead of the default alarm field name "DisableEndTime", you can use the parameter [\[AlarmHeading\]AlarmField](#).
- 

You can also shelve alarms using the default runtime interface for the following projects.

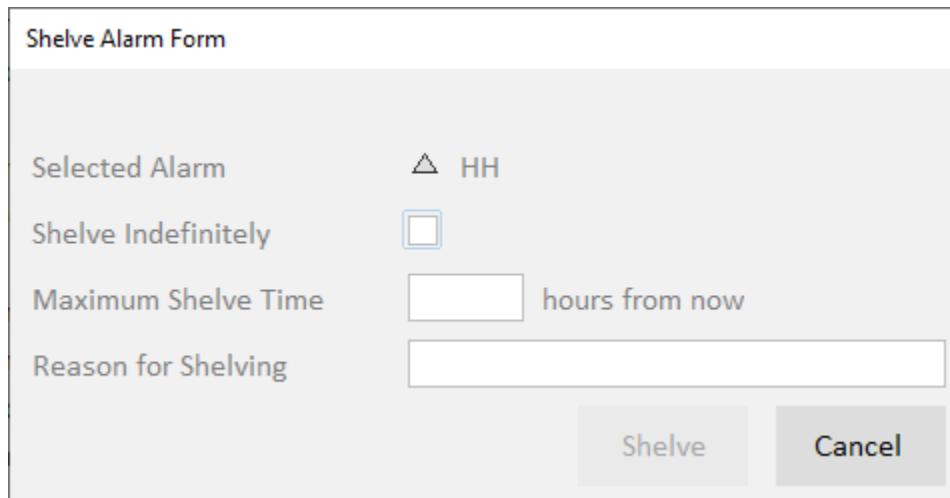
## Shelve alarms in a Situational Awareness project

Alarms can be shelved during runtime in the following locations:

- Active Alarms page (see [Default Alarm Pages](#))
- Shelved Alarms page (see [Default Alarm Pages](#))
- Active Alarms Zone (UHD4K projects only)
- Top 5 Active Alarm list on the Header Bar
- Information Zone **Alarms** tab.

### To shelve a selected alarm:

1. In an alarms list, right-click the alarm that you would like to shelve.
2. On the menu that appears, select one of the following options:
  - **Shelf for ...** — opens a dialog that allows you to **Shelf Indefinitely** or set a **Maximum Shelf Time**. You can also provide a **Reason For Shelving**.



- **Shelf until ...** — opens a dialog that allows you to set a **Shelf Until Date** and **Shelf Until Time**. You can also provide a **Reason for Shelving**.

Shelve Alarm Until Form

Selected Alarm	△ HH
Shelve Until Date	5 ▾ 2 ▾ 2021 ▾ DD/MM/YYYY
Shelve Until Time	HH:mm
Reason for Shelving	<input type="text"/>
Shelve Cancel	

**Note:** To extend the shelved period, repeat these steps on the Shelved Alarms page.

#### To restore a shelved alarm:

1. Go to the Shelved Alarms page.
2. Right-click on the alarm you want to restore.
3. On the menu that appears, select **Restore**.

### Shelve alarms in a project based on the SxW templates

If your project is based on the SxW templates, you can shelve alarms at runtime from the following alarm pages:

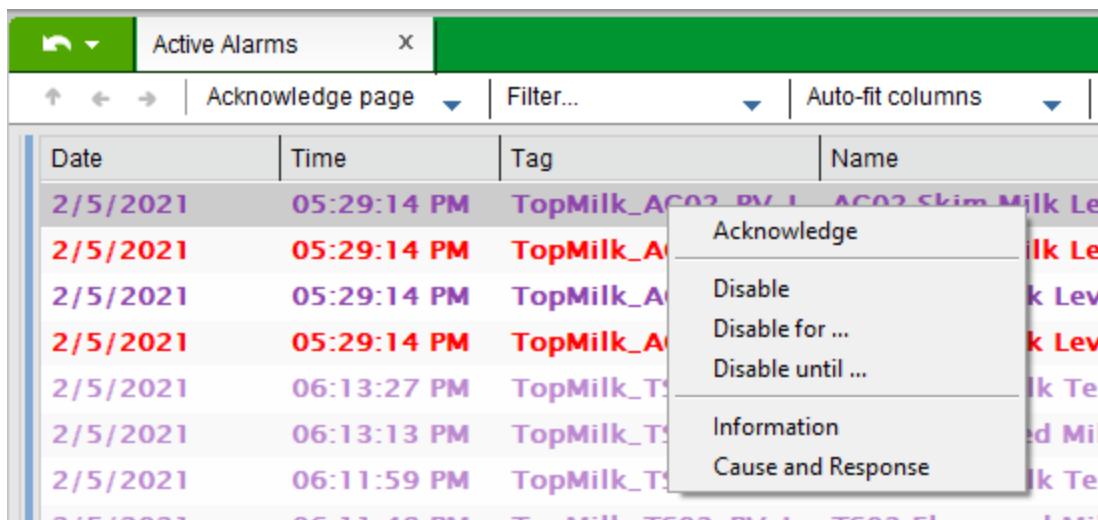
- Active Alarms
- Alarm Summary
- Disabled Alarms.

These pages provide commands that disable an alarm for a specified period of time, or until a specified time is reached.

**Note:** In projects based on the SxW templates, the ability to disable alarms may be restricted by the privilege level set in the parameter [\[Privilege\]DisableAlarms](#).

#### To shelve alarms:

1. On an alarms page, select the alarm(s) you would like to shelve.
2. Right-click on the selected alarms.



The screenshot shows a table titled "Active Alarms" with columns: Date, Time, Tag, and Name. A context menu is open over the last row of the table, listing options: Acknowledge, Disable, Disable for ..., Disable until ..., Information, and Cause and Response.

Date	Time	Tag	Name
2/5/2021	05:29:14 PM	TopMilk_AC02_PV_L	TopMilk_A...
2/5/2021	05:29:14 PM	TopMilk_A...	Acknowledge
2/5/2021	05:29:14 PM	TopMilk_A...	Disable
2/5/2021	05:29:14 PM	TopMilk_A...	Disable for ...
2/5/2021	06:13:27 PM	TopMilk_TS...	Disable until ...
2/5/2021	06:13:13 PM	TopMilk_TS...	Information
2/5/2021	06:11:59 PM	TopMilk_TS...	Cause and Response

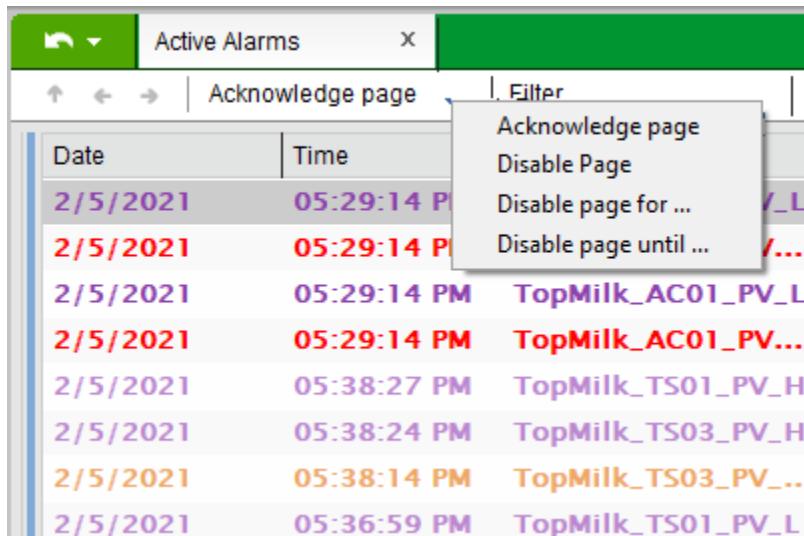
3. On the menu that appears, select one of the following options:

- **Disable for ...** — opens a dialog that allows you to disable an alarm for a specified number of hours.
- **Disable until ...** — opens a dialog that allows you to disable an alarm until a specified date and time.

In both cases, you can add a comment that describes why the alarm has been disabled.

#### To shelve all alarms on a page:

1. Display the drop-down menu for the **Acknowledge Page** Command Bar button.



The screenshot shows a table titled "Active Alarms" with columns: Date, Time, Tag, and Name. A context menu is open over the last row of the table, listing options: Filter, Acknowledge page, Disable Page, Disable page for ..., and Disable page until ...

Date	Time	Tag	Name
2/5/2021	05:29:14 P...	TopMilk_AC01_PV_L	TopMilk_AC01_PV...
2/5/2021	05:29:14 P...	TopMilk_AC01_PV_H	TopMilk_AC01_PV...
2/5/2021	05:38:27 PM	TopMilk_TS01_PV_H	TopMilk_TS01_PV...
2/5/2021	05:38:24 PM	TopMilk_TS03_PV_H	TopMilk_TS03_PV...
2/5/2021	05:38:14 PM	TopMilk_TS03_PV_L	TopMilk_TS03_PV...
2/5/2021	05:36:59 PM	TopMilk_TS01_PV_L	TopMilk_TS01_PV...

- Select **Disable page for ...** to disable all alarms for a specified number of hours.
- Select **Disable page until ...** to disable all alarms until a specified date and time.

#### Shelve alarms in a project based on the Tab Style templates

If your project is based on the Tab Style templates, you can shelve alarms at runtime from the following alarm pages:

- Active Alarms

- Alarm Summary
- Disabled Alarms.

These pages provide commands that disable an alarm for a specified period of time, or until a specified time is reached.

**Note:** In projects based on the Tab Style templates, the ability to disable alarms may be restricted by the privilege level set in the parameter [Privilege]DisableAlarms.

#### To shelve alarms:

1. On an alarms page, select the alarm(s) you would like to shelve.
2. Right-click on the selected alarms.

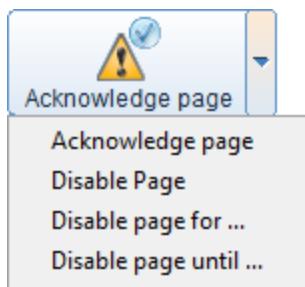
Date	Time	Tag	Name
2/5/2021	05:16:14 PM	TopMilk_TS02_PV_L	TS02 Skim Milk Temp
2/5/2021	05:01:58 PM	TopMilk_AC01_PV_L	AC01 Full Milk Level
2/5/2021	05:01:58 PM	TopMilk_AC01_PV_L	AC01 Full Milk Level
2/5/2021	05:01:58 PM	TopMilk_AC02_PV_L	AC02 Skim Milk Level
2/5/2021	05:01:58 PM	TopMilk_TS03_PV_L	TS03 Flavoured Milk Temp
2/5/2021	05:15:26 PM	TopMilk_TS03_PV_L	TS03 Flavoured Milk Temp
2/5/2021	05:15:23 PM	TopMilk_TS03_PV_L	TS03 Flavoured Milk Temp
2/5/2021	05:15:20 PM	TopMilk_TS03_PV_L	TS03 Flavoured Milk Temp
2/5/2021	05:11:36 PM	TopMilk_TS03_PV_L	TS03 Flavoured Milk Temp
2/5/2021	05:06:55 PM	TopMilk_TS03_PV_L	TS03 Flavoured Milk Temp
2/5/2021	05:02:00 PM	TopMilk_TS03_PV_L	TS03 Flavoured Milk Temp

3. On the menu that appears, select one of the following options:
  - **Disable for ...** — opens a dialog that allows you to disable an alarm for a specified number of hours.
  - **Disable until ...** — opens a dialog that allows you to disable an alarm until a specified date and time.

In both cases, you can add a comment that describes why the alarm has been disabled.

#### To shelve all alarms on a page:

1. Go to the **Action** tab, and display the drop-down menu for the **Acknowledge Page** button.



- Select **Disable page for ...** to disable all alarms for a specified number of hours.
- Select **Disable page until ...** to disable all alarms until a specified date and time.

---

**Note:** If Plant SCADA is installed on an English operating system and you want to add a comment or reason for shelving an alarm using a Unicode language (such as Korean, Russian or Chinese), you will need to change the Windows™ Region setting for the runtime computer. To do this, go to Windows **Control Panel** and open the **Region** dialog box. On the **Administrative** tab, use the **Change system locale** button to select the required system locale. Be aware that you will have to restart your computer. When you launch Plant SCADA runtime, select the matching language on the login form. Your runtime comments will be recorded using the specified language characters.

---

## See Also

[Disable Alarms](#)

[Enable a Disabled Alarm](#)

## Display Cause and Response Information

Plant SCADA allows you to associate cause and response information with an alarm tag (see [Provide Cause and Response Information to Operators](#)).

In a **Situational Awareness** project, cause and response information is presented on the Default Alarm Pages and the **Alarms** tab in the Information Zone.

If your project is based on the **SxW** or **Tab Style** templates, you can view this information at runtime via the Cause and Response dialog. If multiple alarm responses have been configured for an alarm, this dialog will allow you to step through all of them.

You can launch the Cause and Response dialog from the following locations:

- Alarm Banner
- Active Alarms page
- Disabled Alarms page.

---

**Note:** If alarm response information has not been specified for an alarm, the menu items described in the procedure below will not be available.

---

### To display alarm response information for an alarm:

1. Select the alarm for which you would like to view cause and response information.
2. Right-click and select **Cause and Response**.

The Cause and Response dialog will appear. It includes the following information:

- Possible Alarm Cause — a description of the cause of an alarm.
  - Response — a description of the appropriate response to an alarm.
  - Response Time — the period of time in which the specified response needs to be acted upon.
  - Consequence — a description of the likely outcome if the appropriate response is not acted upon within the specified response time.
3. If multiple alarm responses have been configured for the selected alarm, click the **Next** button to step through the available cause and response information. When you have reached the last alarm response, the Next button will no longer be available.
  4. **Close** the dialog.

## See Also

[Add Cause and Response Information to Alarms](#)

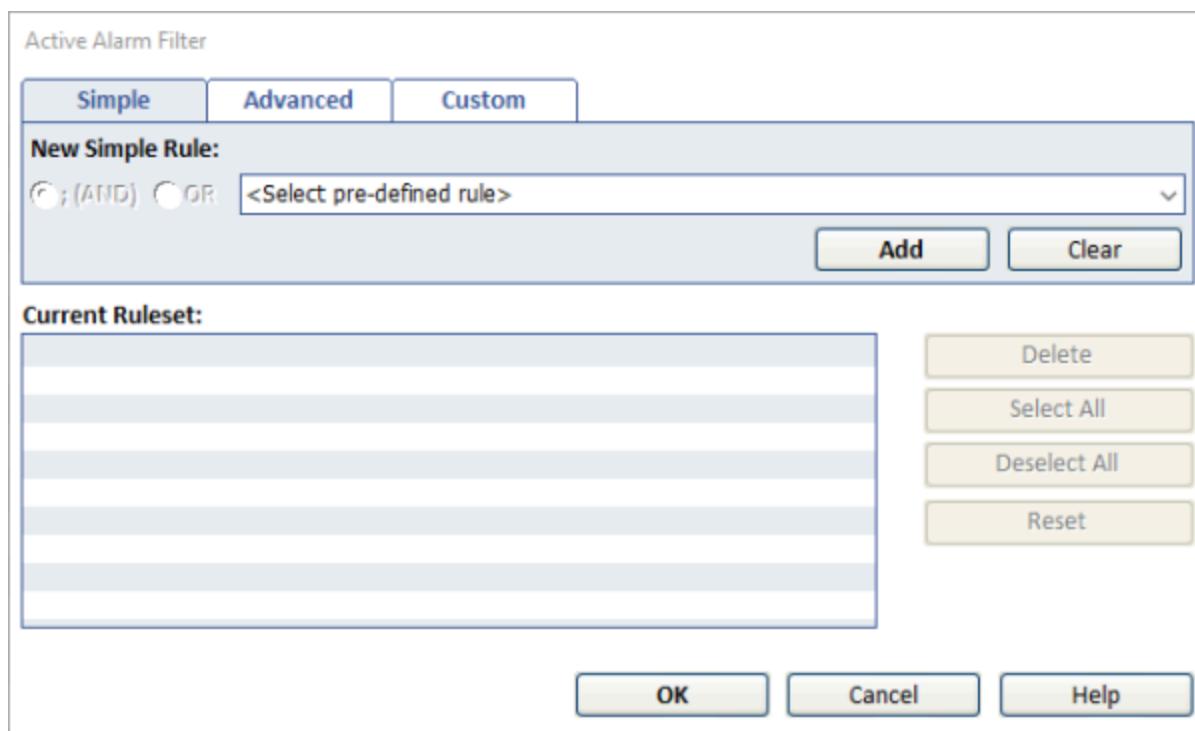
## Filter Alarms

In a **Situational Awareness** project, you can filter an alarms page at runtime using the following tools:

- the alarms list tree view
- the filters toolbar (directly above the alarms list)
- the Reference Equipment list.

These tools are available on the [Default Alarm Pages](#).

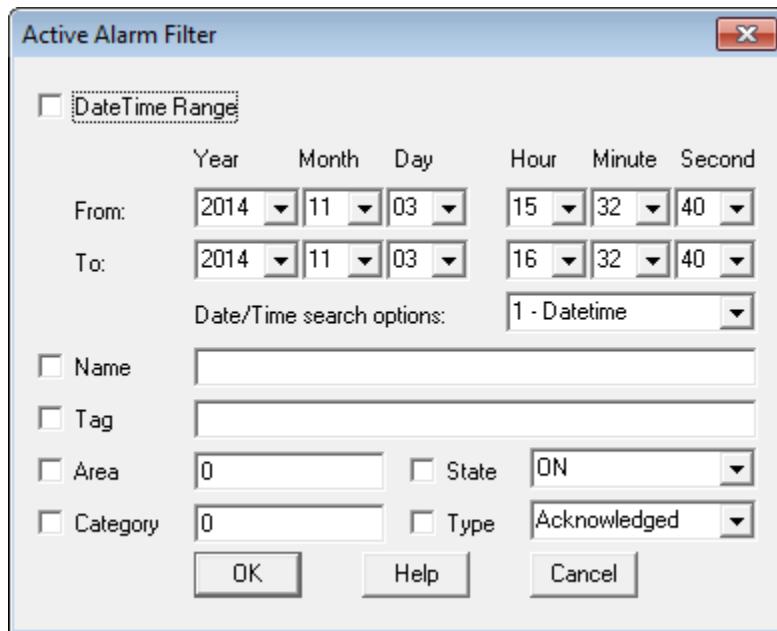
If your project uses the **SxW** or **Tab Style** templates (accessible via a starter project), you can use the Alarm Filter dialog to filter an alarms list.



This dialog is accessible from the following alarms pages:

- Active Alarms
- Disabled Alarms
- Sequence of Events
- Alarm Summary.

If your project uses standard page templates, you will have access to an earlier version of this dialog.



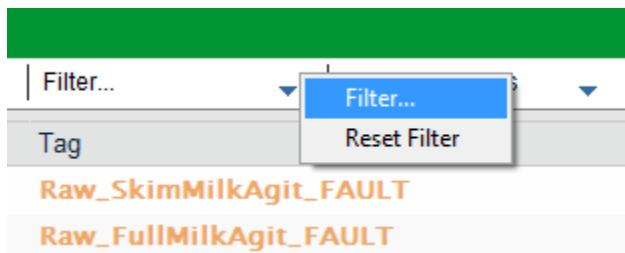
**Note:** Both versions of the Alarm Filter form are supported by their own documentation. For more information

on how to use a form to filter a list of alarms, refer to the documentation that is accessible from the **Help** button.

## View the alarm filter form in a project based on the SxW templates

### To view the alarm filter form:

- Select **Filter** on the Command Bar. You can click on the Command Bar button, or select **Filter** from the button's drop-down menu.



The Alarm Filter form will appear.

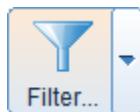
### To remove the filter that is applied to an alarms list:

- Select **Reset Filter** from the **Filter** drop-down menu.

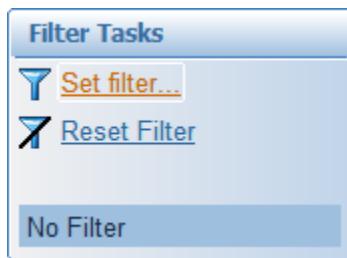
## View the alarm filter form in a project based on the Tab Style templates

### To view the alarm filter form:

- Go to the **Action** tab and select **Filter**. You can click on the button, or select **Filter** from the button's drop-down menu.



If the Action tab is not available, it means the page was created using an early version of the alarm page template. If this is the case, you can select **Set Filter** from the **Filter Tasks** panel.



### To remove the filter that is applied to an alarms list:

- Go to the **Action** tab and select **Reset Filter** from **Filter** button's drop-down menu.

If the Action tab is not available, you can select **Reset Filter** on the **Filter Tasks** panel.

## View the alarm filter form in a project based on standard templates

### To view the alarm filter form:

- You can view the Alarm Filter form by clicking on the **Display Alarm Filter Form** button.



The Alarm Filter form will appear.

### To remove the filter that is applied to an alarms list:

- Select the **Reset Alarm Filter** button.



## See Also

[Sort Alarms](#)

### Sort Alarms

If your project uses the SxW or Tab Style templates (accessible via a starter project), the alarm pages are presented using tables.

This means you can sort a list of alarms according to the values in a particular column by clicking in the header of the column.

Date	Time	Tag
31/10/2014	04:26:41	PM ALARM_1
31/10/2014	04:25:00	PM PLT_LINE1_PM800\MSSTA1\MaxW
31/10/2014	03:55:18	PM Raw_FullMilkAgit_FAULT
31/10/2014	03:23:09	PM Mixer_AgitatorFAULT

Click in the header row to sort an alarms list according to the

When you click a column header multiple times, the following will occur:

- The first click will sort the list in descending order
- The second click will sort the list in ascending order
- The third click will return the page to its default sort order.

A small blue arrow will appear in a selected column header to indicate if the current sort order is ascending or descending.

**Note:** You can only sort the Sequence of Events (SOE) page using the **Date**, **Time** and **Milliseconds** columns.

If you would like to adjust the default sort order for an alarm page, see the topic [Alarms List Default Sort Order](#).

## Add a Column to an Alarms List

If your project uses the **SxW** or **Tab Style** templates (accessible via a starter project), the following alarms pages are presented using tables:

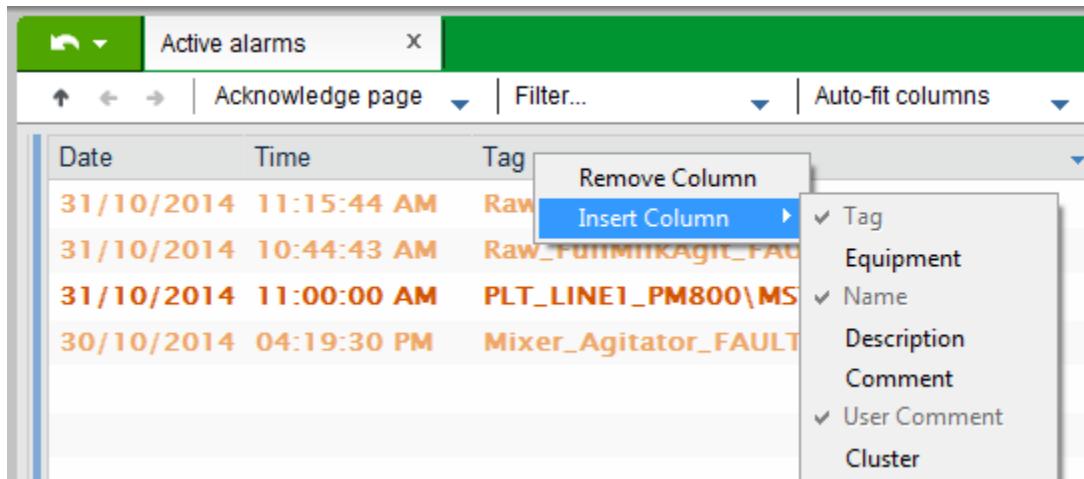
- Active Alarms
- Disabled Alarms
- Sequence of Events
- Alarm Summary.

The alarms list on each of these pages includes a default set of columns that are suited to the purpose of the page. If required, you can add columns to the lists to display additional information such as specific alarm property values or quality flags.

**Note:** The Hardware Alarms page includes a fixed set of columns and does not support any additional columns.

### To add a column to an alarms list:

1. While in runtime, open the alarm page to which you would like to add a column.
2. Right-click within the header row of the alarms list in the location where you would like to add the column.
3. In the menu that appears, hold the mouse over the **Insert Column** option. A further menu will appear displaying a list of column headings.



4. From this list, select the column you would like to add. (If a column has a check mark next to it, it means it is already displayed.)

The selected column will now appear in the list of alarms in the location where the right-click occurred. If required, you can click on the column header and drag it to a different location within the table.

**To remove a column from an alarms list:**

- Right-click within the header row of the column you would like to remove, and select **Remove Column** from the menu that appears.

The arrangement of columns is saved with your user profile and is restored each time you log in.

**See Also**

[Save and Restore an Alarms List View](#)

**Save and Restore an Alarms List View**

If your project uses the SxW or Tab Style templates (accessible via a starter project), the following alarms pages are presented using tables:

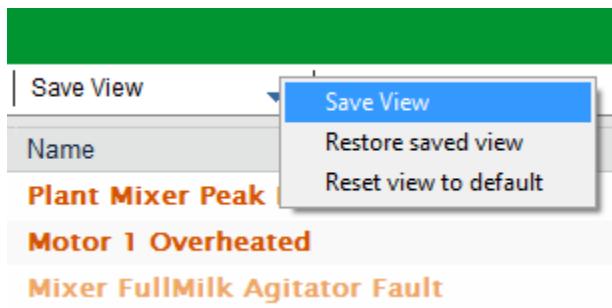
- Active Alarms
- Disabled Alarms
- Sequence of Events
- Alarm Summary.

This allows you to make simple runtime changes to an alarms list, as you can adjust, rearrange, add and remove columns.

If required, you can save the current arrangement of columns as a "view" that can be restored at a later point.

**Save an alarms list view in a project based on the SxW templates****To save the current arrangement of an alarms list as a view:**

- Select **Save View** on the Command Bar. You can click the Command Bar button, or select **Save View** from the button's drop-down menu.

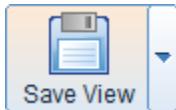
**To restore a saved view:**

- Select **Restored Saved View** from the **Save View** drop-down menu.

## Save an alarms list view in a project based on the Tab Style templates

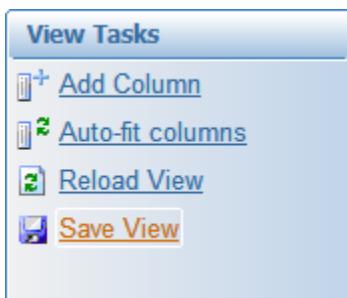
To save the current arrangement of an alarms list as a view:

- Go to the **Action** tab and select **Save View**.



You can just click on the button, or select **Save View** from the button's drop-down menu.

If the Action tab is not available, it means the page was created using an early version of the alarm page template. If this is the case, the **Save View** command will available via the **View Tasks** panel.



To restore a saved view:

- Go to the **Action** tab and select **Restored Saved View** from **Save View** drop-down menu.

If the Action tab is not available, you can select **Reload View** on the **View Tasks** panel.

---

**Note:** You can only save one view at a time. Each time you use the **Save View** command, the existing saved view will be replaced.

---

## Print an Alarms List

If your project uses the SxW or Tab Style templates (accessible via a starter project), you can print the list of alarms that is displayed on the following pages:

- Active Alarms
- Disabled Alarms
- Sequence of Events
- Alarm Summary.

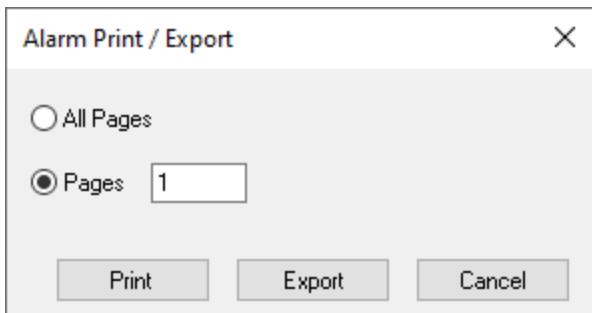
To print an alarms list:

- In a project based on the SxW templates, select **Print/Export** on the page Command Bar.

Or:

In a project based on the Tab Style templates, go to the **Action** tab and select **Print/Export**. If the Action tab is not available, you can select **Print/Export** on the **Page Tasks** panel.

The Alarm Print/Export dialog will appear.



2. Select **All Pages** to print the entire alarms list, or **Pages** to specify a subset of pages.
3. Click on the **Print** button to make the Print dialog appear.

## See Also

[Export an Alarms List](#)

### Export an Alarms List

If your project uses the SxW or Tab Style templates (accessible via a starter project), you can export a list of alarms as an HTML file. This feature is supported on the following pages:

- Active Alarms
- Disabled Alarms
- Sequence of Events
- Alarm Summary.

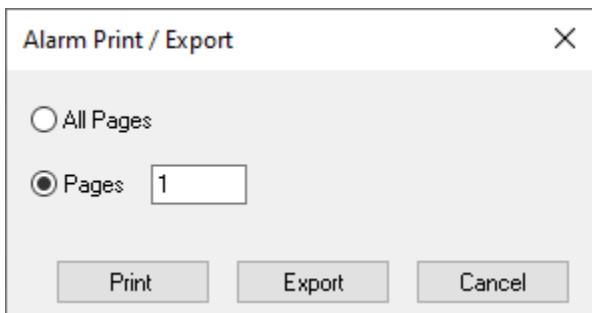
#### To export an alarms list:

1. In a project based on the SxW templates, select **Export All** on the page Command Bar.

Or:

In a project based on the Tab Style templates, go to the **Action** tab and select **Print/Export**. If the Action tab is not available, you can select **Print/Export** on the **Page Tasks** panel.

The Alarm Print/Export dialog will appear.



2. Select **All Pages** to export the entire alarms list, or **Pages** to specify a subset of pages.
3. Click on the **Export** button. The Alarm Export dialog will appear, allowing you to save the current alarms list as an HTML file.

## See Also

[Print an Alarms List](#)

### Add a Comment to the SOE Page

You can add a comment to an event listed on the Sequence of Events (SOE) page.

**Note:** You cannot modify or delete an existing comment for an event. You can only add a comment.

#### To add a comment to the SOE page:

1. Right-click on the relevant event and select **Comment**.

The Comment dialog will appear. The name of the associated alarm tag (if relevant to the selected event) displays in the dialog title bar (for example, "Comment [Alarm\_6]").

2. Enter your comment in the **Enter Comment** field.
3. Click **OK**.

The comment will display in the list on the Sequence of Events page next to the associated event.

You can also add comments to an event on the SOE page using the AlarmComment Cicode function. To add a message to the relevant event you can include the 'AN' as an additional parameter in the function.

For example:

```
AlarmComment ("Temperature discrepancies", 21)
```

To successfully add a comment to an event on the SOE page, the following rules need to be met:

- The event is not of "Comment" classification.
- A user should be logged on to the client process.
- If the event is associated with an alarm tag, the logged on user needs to have sufficient privileges for the alarm.
- If the event is not automatically generated by the system, the logged on user needs to match to the user associated with the event.
- If the event is automatically generated by the system, the logged on user needs to have full privileges (from 1 to 8).

**Note:** If you have installed Plant SCADA on an English operating system and want to add comments at runtime using a Unicode language (such as Korean, Russian or Chinese), you will need to change the Windows™ **Region** setting for the runtime computer. To do this, go to **Control Panel** and open the **Region** dialog box. On the **Administrative** tab, use the **Change system locale** button to select the required system locale. Be aware that you will have to restart your computer. When you launch runtime, select the matching **Language** on the login form. Your runtime comments will be recorded using the specified language characters.

## See Also

[Add an Event to the SOE Page](#)

## Add an Event to the SOE Page

You can add a new event to the Sequence of Events (SOE) page. If the new event is not associated with an alarm tag, it will be classified as a user event.

To add an event, you need to be logged on with a user account that has full privileges (that is, privileges from 1 to 8).

### To add an event to the SOE page:

1. Right-click on the relevant event and select **Add Event**.

The Message dialog will appear. The name of the associated alarm tag (if relevant to the selected event) displays in the dialog title bar (for example, "Message [Alarm\_6]").

2. Enter your message in the **Enter Message** field.
3. Click **OK**.

The new event will display in the list on the Sequence of Events page (see [Alarm Pages](#)).

You can also add events to the event journal page using the **SOEEventAdd** Cicode function.

---

**Note:** If you have installed Plant SCADA on an English operating system and want to add events messages at runtime using a Unicode language (such as Korean, Russian or Chinese), you will need to change the Windows™ Region setting for the runtime computer. To do this, go to **Control Panel** and open the **Region** dialog box. On the **Administrative** tab, use the **Change system locale** button to select the required system locale. Be aware that you will have to restart your computer. When you launch runtime, select the matching **Language** on the login form. Your event messages will be recorded using the specified language characters.

---

## See Also

[Add a Comment to the SOE Page](#)

## Archive the Alarms History

You can archive the alarms history stored on the alarm server to a variety of devices, including:

- Removable hard disk
- Removable flash memory
- CD-R and CD-RW
- DVD-R and DVD-RAM
- Network file share.

To archive events, you firstly need to confirm that the following parameters are configured correctly:

- [\[Alarm\]ArchiveAfter](#)
- [\[Alarm\]KeepOnlineFor](#)

For more information, see [Configure the Archiving Parameters](#).

To specify a destination for your archived events, you also need set the following alarm server properties:

- **Archive Path**
- **Archive Prefix** (optional)

For more information, see [Specify a Destination for Archived Events](#).

Once you have prepared your system, you can initiate archiving in the following ways:

- [Initiate Archiving with a Cicode Function](#)
- [Configure Automatic Archiving for an Alarm Server](#)
- [Using Scheduler to Trigger Archiving](#)

## See Also

[View Archived Event Data](#)

## Configure the Archiving Parameters

To archive events, you firstly need to confirm that the following parameters are configured correctly:

- [\[Alarm\]ArchiveAfter](#)

This parameter specifies how long event data is kept before it is marked for archiving. The default value is four weeks. After the specified period of time has passed, event data that is marked for archiving will remain in the database but will be read only. When the data becomes read only it is no longer writable. You are unable to make it writable.

- [\[Alarm\]KeepOnlineFor](#)

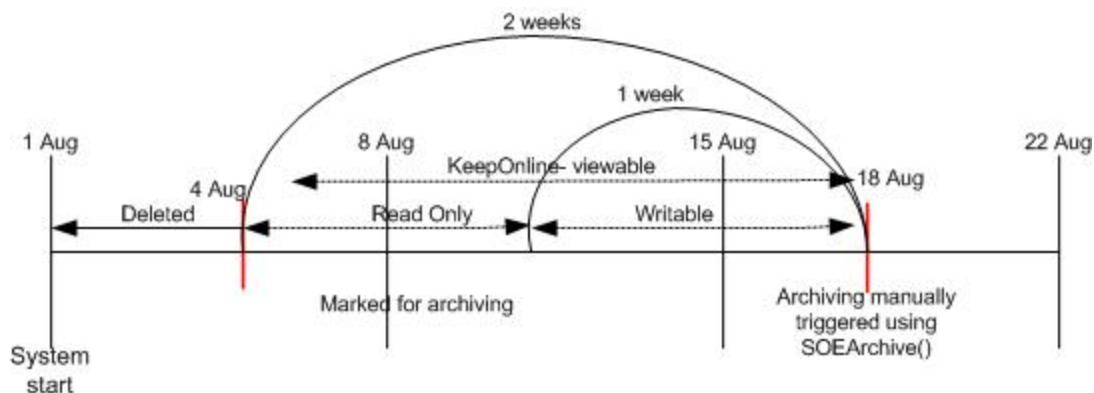
This parameter defines how long event data remains on the Alarm Summary or SOE pages. The default value is six weeks. After this time, the data will no longer appear.

The period of time set for the [Alarm]ArchiveAfter needs to be shorter than the time specified for [Alarm]KeepOnlineFor. If this is not the case, the value for ArchiveAfter will be automatically adjusted to equal the current value for KeepOnlineFor and an out-of-range hardware alarm will be raised.

## Example

The diagram below demonstrates the following scenario:

- "ArchiveAfter" is set to 1 week
- "KeepOnlineFor" is set to 2 weeks.



If archiving is manually triggered on Saturday 18 August 15:00:00, any data reported after midnight on Saturday 4 August will remain viewable via the Alarm Summary or SOE pages for two weeks. Any data reported before midnight on Saturday the 4th will be deleted.

Data that was reported between midnight on Saturday the 4th and midnight on Saturday the 11th is marked for archiving and is read only.

The data reported after midnight Saturday 11 August and the current time is unavailable for archiving as this data is still writable.

---

**Note:** The information available on your active and disabled alarm pages is classified as active data. For data integrity, [Alarm]ArchiveAfter needs to be configured to be longer than the lifetime of the active events.

---

## See Also

[Specify a Destination for Archived Events](#)

### Specify a Destination for Archived Events

To specify a destination for your archived events, you need to set the following properties on the alarm server process that is managing the events (see [Add an Alarm Server Process](#)).

#### 1. Archive Path

This property specifies where your archived event data will be stored. The location should be accessible from the alarm server when an archiving operation is executed. When the volume is in place, Plant SCADA will mount the volume automatically (if required) prior to writing data.

**Note:**

- The media where the archived file is to be saved should be labeled. If this is not the case, archiving will not succeed and a message will display on the SOE page.
  - If you choose the Windows system drive (typically the "C:\\" drive) as the archive path destination, you may need to adjust Windows' UAC (User Access Control) settings to enable archiving to occur.
- 

#### 2. Archive Prefix

When archiving occurs, sub folders are created in the archive volume's destination directory. These folders are named using a timestamp (YYYYMMDD). This optional setting allows you to apply a prefix to the name of the folders that are created.

**Example**

An alarm server could be configured to use the following property values:

- Archive Path = D:\Data
- Archive Prefix = Archive

If archiving is executed on this server on 21 January 2021, the archive volume path would be:  
D:\Data\Archive20210121.

**Note:** If a redundant alarm server is configured, archiving should be enabled on both servers.

---

## See Also

- [Initiate Archiving with a Cicode Function](#)  
[Configure Automatic Archiving for an Alarm Server](#)  
[Using Scheduler to Trigger Archiving](#)

### Initiate Archiving with a Cicode Function

You can manually trigger event data archiving with the Cicode function [SOEArchive\(\)](#).

**Note:** You need to be logged in as a user that can run Cicode to initiate archiving with the SOEArchive() function.

When you call the function, archiving will commence according to the values defined for the "ArchiveAfter" and "KeepOnlineFor" parameters (see [Configure the Archiving Parameters](#)).

If required, you can use the function's **Path** attribute to specify a destination for the archived data. If no path is specified, the data will be archived to the location defined in the **Archive Path** property on the associated alarm server (see [Specify a Destination for Archived Events](#)).

**Note:** If you use this method to archive your event data, you will need to regularly call the SOEArchive() function to avoid data becoming inaccessible. For example, if the "ArchiveAfter" parameter is set to 1 week, and KeepOnlineFor is set to 2 weeks, you will need to manually trigger archiving after every eight days to confirm that all data is captured.

---

## See Also

- [Configure Automatic Archiving for an Alarm Server](#)  
[View Archived Event Data](#)

### Configure Automatic Archiving for an Alarm Server

You can configure automatic event archiving for each alarm server in your system. This is achieved via the alarm server properties (see [Add an Alarm Server Process](#)).

**To configure automatic archiving for an alarm server:**

1. In the Topology activity, select **Edit**.
2. On the menu below the Command Bar, select **Alarm Servers**. All configured alarms servers are displayed.
3. Locate the record for the required alarm server.
4. In the **Archive Internal (Days)** field, enter the number of days after which archiving will automatically occur.
5. Confirm that **Archive Path** and **Archive Prefix** fields are correctly configured (see [Specify a Destination for Archived Events](#)).

6. Click **Save**.

**Note:** In order for archiving succeed, the "KeepOnlineFor" and "ArchiveAfter" parameters need to be configured correctly (see [Configure the Archiving Parameters](#)).

## See Also

[View Archived Event Data](#)

### Using Scheduler to Trigger Archiving

You can use Scheduler to trigger event archiving. However, the steps needed to complete this task will differ if the system you are running is in single process or multi-process mode.

If you attempt to trigger archiving in multi-process mode, Scheduler needs to invoke a custom Cicode function that includes the following:

```
Login (Username, Password);  
SOEArchive()
```

Follow the instructions outlined in [Configure Equipment for Scheduling](#) to define new Equipment. The Equipment name can be any name specified by the user.

Follow the instructions outlined in [Define Equipment States in Plant SCADA Studio](#) to define the 'Trigger' and 'OFF' states for the newly added equipment.

#### To trigger archiving in single process mode:

**Note:** For archiving to succeed in single process mode you need to log in to a display client in Runtime.

1. Open the 'Trigger' Equipment State form.
2. In the **Entry Action** field enter the function SOEArchive(). Click **OK** to save.
3. Start runtime and log in to the display client.
4. Open Scheduler. The Archive Equipment branch should be displayed in the equipment hierarchy.
5. Following the instructions outlined in [Add a Schedule Entry](#), schedule the state 'trigger' to occur.

When the time transitions for this scheduled task, SOEArchive() will be called and archiving will be triggered.

#### To trigger archiving in Multi Process Mode:

1. In the Cicode Editor create a custom Cicode function and save it. This Cicode function needs to include login information. For example:

```
FUNCTION  
ArchiveMyData()  
Login("ConfigUsers", "SCADA");  
SOEArchive();  
END
```
2. Open the 'Trigger' Equipment State form.
3. In the **Entry Action** field enter the name of the custom Cicode function. e.g. ArchiveMyData(). Click **OK** to save.
4. Open Scheduler. The Archive Equipment branch should be displayed in the equipment hierarchy.

5. Following the instructions outlined in [Add a Schedule Entry](#), schedule the state 'trigger' to occur.

## See Also

[Initiate Archiving with a Cicode Function](#)  
[Configure Automatic Archiving for an Alarm Server](#)

## View Archived Event Data

Archived data that is no longer available online for viewing can be accessed using the [SOEMount](#) Cicode function.

In mounting the volume, you are directing the server to where the archived data is stored. You are not restoring the data, merely making the data available for viewing via the SOE page. To view the data on the mounted volume you need to query the data using the filter options available on the SOE page. To view data, including data on the mounted volume, use 'page down' to query the data and bring it into the SOE page.

When mounting a volume, you should pass the volume path when calling SOEMount() Cicode function. For example, the path should be the following:

C:\Data\Archive20120121

rather than:

C:\Data\Archive20120121\EventJournal

If you need to copy or move archived data to another directory (for example, from "C:\Data\Archive20120121" to "D:\Data"), you will need to copy or move the entire "EventJournal" folder to the new directory. You can then invoke SOEMount("D:\Data") to restore the data from the new location.

If you no longer need to view the data on the mounted volume, you can call use the [SOEDismount](#) Cicode function.

**Note:** To call SOEMount() or SOEDismount(), you need to be logged on as a user that has full privileges.

## Using Custom Alarm Filters

You can define keywords for your alarm tags that you can then use to perform customized queries on your alarm data.

The Alarm Properties dialog allows you to define up to eight custom filters for each of the alarms in your system, allowing you to generate queries that filter your alarm data for specific information.

For example, you will want to prioritize any alarms that monitor equipment and conditions with the potential to cause a fire. You could set Custom Filter 1 for each relevant tag to "fire hazard". You could then call a Cicode function that requests alarms with a "custom filter 1" field equal to "fire hazard". The end result would be a list of alarms notifying an operator of any potentially flammable circumstances.

You could also set aside Custom Filter 2 to define the type of equipment the alarm is associated with, and label each alarm accordingly (for example "pump", "conveyor", and so on). Queries could then be created to list the alarms related to a particular type of machinery; for example, alarms associated with pumps.

## See Also

[Implementing Queries that Use Custom Alarm Filters](#)

[Named Filters](#)[Implementing Alarm Filters Using Cicode](#)[Converting Legacy AlarmSetQuery\(\) Functions](#)

## Implementing Queries that Use Custom Alarm Filters

To implement a query that creates a custom alarm filter, you need to use the following Cicode functions:

- [AlarmFilterEditOpen](#)
- [AlarmFilterEditAppend](#)
- [AlarmFilterEditSet](#)
- [AlarmFilterEditCommit](#)
- [AlarmFilterEditFirst](#)
- [AlarmFilterEditLast](#)
- [AlarmFilterEditNext](#)
- [AlarmFilterEditClose](#)

The alarms within any query function can also be filtered via [AlarmSetInfo](#) set to Type=12 and a named filter (see [Named Filters](#)).

---

**Note:** If a requested filter is too complex (for example, it contains too many conditions or too many nested brackets), the filter is cleared (no filter is used). If this occurs, the hardware alarm "Too many alarms in filters" is generated on the client components, and a tracelog error message is logged.

---

## Example

You may want to create a query that calls the current alarms for the conveyors in your plant. This could be drawn from the alarm tags you have identified as being associated with a conveyor by applying the keyword "conveyor" to the field **Custom Filter 1**.

The AlarmFilterEditSet (hdl, "Custom1=conveyor") followed by the AlarmFilterEditCommit(hndl) function will filter your alarm data for "conveyor" information.

The query function you would create would be as follows:

```
! This query function called FilterCustom1()
! filters alarms by the requested Custom filter 1.
! Only alarms with the requested Custom Filter 1 are displayed.
INT
FUNCTION
    FilterCustom1(STRING sValue)
        INT iResult=-1, INT iHndl=-1,
        STRING sFilter="";
        sFilter = "Custom1=" + "^" + sValue + "^"; iHndl= AlarmFilterEditOpen(21);
        IF iHndl <> -1 THEN
            iResult = AlarmFilterEditSet(iHndl,sFilter)
            IF iResult = 0 THEN
                iResult = AlarmFilterEditCommit(iHndl)
            END
            AlarmFilterEditClose(iHndl)
```

```
END
RETURN iResult;
END
```

Say you then want to create a button on a graphics page that lists current conveyor alarms. You would implement this by applying the custom function to the button.

```
FilterCustom1("conveyer");
```

You could also create a reset button that clears the current displayed list by canceling the query:

```
hSession = AlarmFilterEditOpen(21);
AlarmFilterEditSet(hSession, "");
AlarmFilterEditCommit(hSession);
AlarmFilterEditClose(hSession);
```

You have the choice of calling a specific Cicode function, for example, "Customfield1", with the argument "pumpcontrol", or using a more generic approach with the function "Checkfield" with argument "CUSTOM1=pumpcontrol". In this case, the Cicode function filters the passed string and checks the field specified.

## See Also

[Implementing Alarm Filters Using Cicode](#)

## Named Filters

You can create a named filter and use it across multiple alarm lists.

A named filter can be used to filter an alarm list or an alarm count only if that alarm list or alarm count is associated to the named filter. It is not used to directly filter alarms. Instead it is used to update the active filter of multiple alarm lists or alarm counts. Once you create the named filter you then edit it to add the filter criteria you want the named filter to query.

To associate an alarm list to a named filter, you need to call [AlarmSetInfo](#)(AN, 12, "Named Filter"), where AN is the animation number of the associated alarms list, 12 is the mode for associating the named filter to the alarms list, and "NamedFilter" is the name of the filter that was created by calling the function [AlarmFilterOpen](#).

To disassociate the named filter from an alarm list set "NamedFilter" to empty text, for example, [AlarmSetInfo](#)(AN, 12, ""). This will remove the association from the named filter, but the disassociated alarm list will not be refreshed, hence the filter query will still be in place until a new filter is applied.

In order to associate an alarm count to a named filter, you need to call [AlarmCount](#)(Type, "NamedFilter", KeepAliveSeconds, CacheMode), where Type, KeepAliveSeconds and CacheMode are documented in the [AlarmCount](#) function, and "NamedFilter" is the named filter that was previously opened by calling [AlarmFilterOpen](#).

In each of these cases, editing the filter criteria belonging to the named filter will update all associated alarm lists or alarm counts.

---

**Note:** If a requested filter is too complex (for example, it contains too many conditions or too many nested brackets), the filter is cleared (no filter is used). If this occurs, the hardware alarm "Too many alarms in filters" is generated on the client components, and a tracelog error message is logged.

### Example

The following examples show how to open, edit (add filter criteria), associate, disassociate and close a named filter called "MyFilter".

```
// The following examples show how to open, edit, associate, disassociate and close
// a named filter (e.g. "MyFilter").
// These are fictional examples for demonstration purposes only.
// The prompt message is displayed in prompt-line for increasing clarity.
// This function returns zero when all examples are successful,
// otherwise error is returned.
INT
FUNCTION
AlarmNamedFilterExamples()
    INT iError, iHndl;
    INT iAN=21; // Animation Number
    INT iOpenModeNew = 1;
    INT iCloseModeManual = 0;
    INT iAlarmCounted=0, IAlarmDisplayed=0;
    STRING sName, sEmptyString="", sNamedFilter, sLinkedFilter;
    ! Open the new filter named "MyFilter"
    sNamedFilter = "MyFilter";
    iError = AlarmFilterOpen(sNamedFilter, iOpenModeNew, iCloseModeManual);
    IF iError <> 0 THEN
        Prompt ("MyFilter open error!");
        RETURN iError;
    END;
    ! Edit this filter by selecting alarms of Category 1 and Area 0
    iHndl = AlarmFilterEditOpen(sNamedFilter);
    IF iHndl <> -1 THEN;
        iError = AlarmFilterEditSet(iHndl,"Category=1;Area=0;");
        IF iError = 0 THEN
            iError = AlarmFilterEditCommit(iHndl);
            IF iError = 0 THEN
                iError = AlarmFilterEditClose(iHndl);
            END
        END
    END
    IF iError <> 0 THEN
        Prompt ("MyFilter edit error!");
        RETURN iError;
    END
    ELSE
        Prompt ("MyFilter hndl-edit error!");
        RETURN iHndl;
    END;
    ! Associate this filter to the specified AN of Alarm page.
    ! Other alarm pages can be linked in similar way.
    PageDisplay("Alarm");
    iError = AlarmSetInfo(iAN, 12, sNamedFilter);
    IF iError <> 0 THEN
        Prompt ("MyFilter link error!");
        RETURN iError;
    END
    ! Retrieve associated filter name. Verify associated filter.
    sLinkedFilter = AlarmGetFilterName(iAN); // "MyFilter"
    IF sLinkedFilter <> sNamedFilter THEN
        Prompt ("MyFilter get-name error!");
        RETURN iError;
    END;
    ! Count active alarms of this filter and verify against displayed alarms
    iAlarmCounted = AlarmCount(0, sNamedFilter, 1,0);
```

```
iAlarmDisplayed = AlarmCountList(iAN);
IF iAlarmCounted <> iAlarmDisplayed THEN
    Prompt ("Count verification error!");
    RETURN iError;
END;
! disassociate "MyFilter"
iError = AlarmSetInfo(iAN, 12, "");
IF iError <> 0 THEN
    Prompt ("MyFilter unlink error!");
    RETURN iError;
END;
! Verify disassociate action.
sName = AlarmGetFilterName(iAN); // ""
IF sName <> sEmptyString THEN
    Prompt ("MyFilter unlink error!");
    RETURN iError;
END;
! Close this filter
iError = AlarmFilterClose(sNamedFilter);
IF iError <> 0 THEN
    Prompt ("MyFilter Close error!");
END;
RETURN iError;
END
```

## See Also

[Implementing Queries that Use Custom Alarm Filters](#)

[Implementing Alarm Filters Using Cicode](#)

## Implementing Alarm Filters Using Cicode

For historical event lists, you can define and apply a filter (the filter is not shared between lists). The filter is a combination of conditions on the different fields which define the set of rows to display. These conditions may become large, with Cicode only permitting a string of 254 characters. The set / append functions allows you to separate the filter into smaller pieces without verification of the syntax on the sub parts. The whole filter (e.g. concatenation of the pieces) is validated and applied later by the commit function.

Multiple Cicode tasks can access the same filter and modify it, with the filter browsing and modification part of a session. This follows the principles used in tag browsing.

## Filter Syntax

The filter expression may contain a field, comparison operator and a value e.g. Tag=A. The field can be any name listed in the Browse Field Reference, with comparison operators being < <= > >= <>, and value being either a String, Integer, Date Time, and Boolean. Tag Value pairs can be connected by logical operators AND OR Eqv NOT Xor.

A filter may contain multiple expressions separated by a semi colon.

The syntax of the filter can be verified against the following:

---

**Note:**

- 
- (1) The ";" between <filter\_expr> is interpreted as AND
  - (2) The parenthesis are optional but recommended to obtain the right evaluation order. If they are present they need to match.
  - (7) The "\*" is standard wild card. it can be used only once per <string\_value>
- 

1. <filter> ::= <filter\_expr> { ";" <filter\_expr> }
  2. <filter\_expr> ::= { "(" <filter\_expr> ")" } <logical\_binary\_op> { "(" <filter\_expr> ")" }
  3. | <comparison>
  4. | "NOT" "(" <filter\_expr> ")"
  5. <comparison> ::= <field> <comparison\_op> <value>
  6. <comparison\_op> ::= "=" | ">" | "<" | ">=" | "<="
  7. <logical\_binary\_op> ::= "AND" | "OR"
  8. <value> ::= <string\_value> | <boolean\_value> | <DateTime\_value> | <int\_value>
  9. <string\_value> ::= <singleword> ["\*"]
  10. | " " <singleword> { {"<space>"} <singleword> ["\*"] } " "
  11. <singleword> ::= {any character but space}
  12. <boolean\_value> ::= "0" | "1"
  13. <int\_value> ::= ["-"]{numeric char}["." numeric char]
  14. <DateTime\_value> ::= string of number of sec.  
as returned by IntToStr( TimestampToInt(..)) in UTC
  15. <field> ::= valid Plant SCADA field name (case insensitive).
- 

**Note:** If a requested filter is too complex (for example, it contains too many conditions or too many nested brackets), the filter is cleared (no filter is used). If this occurs, the hardware alarm "Too many alarms in filters" is generated on the client components, and a tracelog error message is logged.

---

## Valid filters examples:

- Tag=A
- (Tag=A) OR (TAG=B)
- (Tag=A) OR (TAG=B) ; Name=C
- Time > IntToStr(TimestampToInt(yesterday))
- Tag=Dig\*
- Message="quoted sentence with blanks"
- (((Tag=A) OR (TAG=B)) AND (category=1))

When using the alarm filter Cicode functions first open a session (AlmFilterOpen) and then use this session to browse the different parts (AlmFilterFirst/Next/Prev) of the filter and to update (AlmFilterSet / Append / Commit) the filter. The browse session will be available until closed (AlmFilterClose).

Multiple sessions can be used on the same list. For multiple sessions modify the same filter then the current applied filter is the result of the last commit.

## See Also

[Implementing Queries that Use Custom Alarm Filters](#)

### Converting Legacy AlarmSetQuery() Functions

With the release of Plant SCADA 7.30, the AlarmSetQuery() and Query() Cicode functions were replaced with a set of alarm filter functions. If your project uses AlarmSetQuery(), you will need to review and manually replace each instance with the alarm filter edit functions (see [Implementing Queries that Use Custom Alarm Filters](#)).

Below are examples of v7.20 Cicode and what the v7.30 Cicode may look like with the new alarm filter edit functions.

### Conversion examples from v7.20 to v7.30:

#### v7.20 code:

```
.....
INT resultAlarmsetQuery=0;
// e.g. fieldName = Category, fieldValue = 10
// or fieldName = Tag, fieldValue = DigitalAlarm1
resultAlarmsetQuery =
AlarmsetQuery(21,"AlarmFilter","^"+fieldName+"^",^"+filterValue+"^"); //Obsolete in
v7.30
.....
PUBLIC
INT
FUNCTION AlarmFilter
    (INT iRecId,
    INT iInst,
    STRING fieldName,
    STRING filterValue)
    INT resultFilter=0;
    STRING getValue="";
    getValue = AlarmGetFieldRec(iRecId, fieldName, iInst);
    IF filterValue = getValue THEN
        resultFilter = TRUE;
    ELSE
        // when "*" entered -> all values are accepted
        IF filterValue = "*" THEN
            resultFilter = TRUE;
        ELSE
            resultFilter = FALSE;
        END
    END;
    RETURN resultFilter;
END
```

#### v7.30 code:

```
.....
INT resultAlarmsetQuery=0;
//AlarmsetQuery(21,"AlarmFilter30","^"+fieldName+"^",^"+filterValue+"^");Obsolete in
7.30
//replace the above by direct call
```

```
resultAlarmSetQuery = AlarmFilter730(fieldName,filterValue);
.....
PUBLIC
INT
FUNCTION AlarmFilter730
    (STRING fieldName,
     STRING filterValue)
    INT hndl=-1, resultFilter=-1;
    TRING sFilter="";
    sFilter = fieldName + "=" + filterValue;
    hndl= AlarmFilterEditOpen(21);
    IF hndl <> -1 THEN
        resultFilter = AlarmFilterEditSet(hndl,sFilter)
    IF resultFilter = 0 THEN
        resultFilter = AlarmFilterEditCommit(hndl)
    END
    AlarmFilterEditClose(hndl)
END
RETURN resultFilter;
END
```

Another example of how this function could be returned:

```
PUBLIC
INT
FUNCTION AlarmFilter730
    (STRING fieldName,
     STRING filterValue)
    INT hndl;
    INT resultFilter=-1;
    hndl = AlarmFilterEditOpen(21);
    IF hndl <> -1 THEN
        resultFilter = AlarmFilterEditSet(hndl, fieldName)
        IF resultFilter = 0 THEN
            resultFilter = AlarmFilterEditAppend(hndl, "=^""")
        END
        IF resultFilter = 0 THEN
            resultFilter = AlarmFilterEditAppend(hndl, filterValue)
        END
        IF resultFilter = 0 THEN
            resultFilter = AlarmFilterEditAppend(hndl, "^^")
        END
        IF resultFilter = 0 THEN
            resultFilter = AlarmFilterEditCommit(hndl)
        END
        AlarmFilterEditClose(hndl)
    END
    RETURN resultFilter;
END
```

## Using Alarm Properties as Tags

Alarm properties can be used wherever variable tags can be used (except in alarm descriptions). For example, you can provide the operator with a visual indication when the alarm **CV110\_ERROR** is active:

- **CV110\_ERROR.On** will be TRUE when the alarm is active

- **CV110\_ERROR.On** will be FALSE when it is inactive.

This means **CV110\_ERROR.On** could be entered as the fill color expression in a graphics object. When the conveyor becomes inoperative, the graphics object will change color.

**Note:**

- Using alarm properties as tags is not supported by an OPC UA Server or an Industrial Graphics Server
- If you want to use time-stamped digital or time-stamped analog alarm properties as variable tags, you will need to verify that they are configured correctly with the necessary data being pushed to the relevant variables via the Cicode function [AlarmNotifyVarChange](#). See the topic [Alarms](#) for more details on how Time Stamped Digital Alarms and Time Stamped Analog Alarms operate.

## Alarm reference syntax

You can reference alarm tags in the following ways:

- Refer to an alarm property by tag name
- Refer to an alarm using an 'equipment.item' reference

### Refer to an alarm property by tag name

You can refer to an alarm's properties (and an optional tag extension) using the following syntax:

[Cluster.]Alarm.AlarmProperty[.ExtItem]

Where:

Cluster	The cluster name (optional).
Alarm	The alarm tag name.
AlarmProperty	The alarm property name. This part of the reference is not optional as it is for variable tags. There are different available properties for different categories of alarms. See <a href="#">Supported Alarm Properties</a> .
ExtItem	<p>The tag extension item name (optional). Available items are the same as for tags:</p> <ul style="list-style-type: none"><li>• Value = V</li><li>• Value Timestamp = VT</li><li>• Quality = Q</li><li>• Quality Timestamp = QT</li><li>• Overall Timestamp = T</li></ul> <p>See <a href="#">Tag Extensions</a>.</p> <p>If the item name is not specified, the value of the alarm property is referenced.</p>

For example, "Alarm1.On" will enable you to see and control alarm states. If you want to know the timestamp of the On state, you can use the tag extension "T" (for example, "Alarm1.On.T").

Such references can appear in Cicode and on graphic pages.

### Refer to an alarm using an 'equipment.item' reference

You can use "equipment.item" instead of an alarm tag name to refer to an alarm's properties. To do this, use the following syntax:

[Cluster.]Equipment.Item.AlarmProperty[ .ExtItem]

Where:

Cluster	The cluster name (optional).
Equipment.Item	A substitution for a tag name using an "equipment.item" reference. See <a href="#">Items</a> .
AlarmProperty	The alarm property name. This part of the reference is not optional as it is for variable tags. There are different available properties for different categories of alarms. See <a href="#">Supported Alarm Properties</a> .
ExtItem	The tag extension item name (optional). See <a href="#">Tag Extensions</a> . If the item name is not specified, the value of the alarm property is referenced.

This syntax means you can:

- Use equipment.item in place of an alarm tag name in expression fields in graphic pages.
- Use equipment.item to reference alarms in Genie and Super Genie substitutions.
- Use equipment.item in the value field to represent an alarm when associating Super Genie substitutions via metadata.
- Pass equipment.item to Cicode functions in which an alarm tag name is an argument.

## See Also

[Writing to Alarm Properties](#)

[Setting Up Alarm Properties](#)

## Supported Alarm Properties

The following properties can be used for every alarm type. Remember, the return value relates to the description. For example, for a digital, if 1 is returned, that means the description is TRUE, whereas 0 (zero) means it is FALSE.

Property	Description	Return Type
.On	Alarm active  The .On property for analog alarms is true if any alarms associated with the alarm tag are active.	Digital
.Ack	Alarm acknowledged	Digital
.AcqErr	Data acquisition error	Integer
.Category	Alarm category	Integer
.ComBreak	For Multi-Digital alarms, the property is set to 1 if the device cannot read data from the underlying tag at start-up for a time greater than [Alarm]ArgyleTagValueTimeout value. The property is set to 0 and re-alarms the corresponding alarm, when the alarm server receives valid data from the device.  For every Disabled and Display Disabled alarms (except Time Stamped Digital and Time Stamped Analog alarms) the property is set to 1, when they are being Enabled. The property is set to 0 and re-alarms the corresponding alarm, when the alarm server receives valid data from the device.	Digital
.Custom1 .Custom2 .Custom3 .Custom4 .Custom5 .Custom6 .Custom7 .Custom8	Custom Field.	String(64 bytes)
.Disabled	Alarm disabled (see note below)	Digital
.Historian	Alarm can be historized and published in CitectHistorian.	Digital
.Item	The name of the item with which the alarm is associated.	String

Property	Description	Return Type
.Millisec	The milliseconds part of the time the alarm was triggered	Long
.Name	Alarm name	String (80 bytes)
.OnTime	The time the alarm was triggered.	Long
.Paging	Alarm paged	Boolean
.PagingGroup	Paging group alarm belongs to.	String (80 bytes)
.Priority	Alarm priority	Integer
.State	<p>An alarm's state value. An alarm state value is a combination of state enumeration and action bit masks described below.</p> <p><b>State enumerations</b></p> <p>0 – Alarm OFF state or state 000 for Multi-Digital alarm</p> <p>1 – Alarm ON state or state 00A for Multi-Digital alarm</p> <p>2 – State 0B0 for Multi-Digital alarm</p> <p>3 – State 0BA for Multi-Digital alarm</p> <p>4 – State C00 for Multi-Digital alarm</p> <p>5 – State C0A for Multi-Digital alarm</p> <p>6 – State CB0 for Multi-Digital alarm</p> <p>7 – State CBA for Multi-Digital alarm</p> <p>8 – Analog deviation from set point High</p> <p>9 – Analog deviation from set point low</p> <p>10 – Analog rate of change alarm state</p> <p>11 – Analog low limit alarm state</p> <p>12 – Analog high limit alarm state</p> <p>13 – Analog low low limit alarm state</p> <p>14 – Analog High High limit alarm</p>	Short

Property	Description	Return Type
	<p>state</p> <p><b>Alarm action masks</b></p> <ul style="list-style-type: none"> <li>32 – Unable to get the status of underlying tag at start-up</li> <li>64 – Alarm cleared bit mask</li> <li>128 – Alarm acknowledged but held</li> <li>256 – Alarm unacknowledged bit mask</li> <li>512 – Alarm disabled bit mask</li> <li>1024 – Argyle type alarm bit mask</li> <li>2048 – Argyle type alarm ON bit mask</li> <li>4096 – Analog alarm threshold value changed</li> <li>8192 – User generated event</li> <li>16384 – Event already logged</li> <li>32768 – Disable the alarm display</li> </ul> <p><b>Example:</b></p> <p>A digital alarm called "AlmDigital1" that is ON and unacknowledged, the value of the state property will be:</p> <p>AlmDigital1.State = 1 (ON state) + 256 (Unacknowledged alarm) = 257</p>	
.Tag	Alarm tag	String (80 bytes)
.Time	<p>The time at which the alarm changed state (hh:mm:ss).</p> <p>You can use the following parameters to change this default behavior:</p> <ul style="list-style-type: none"> <li>• <a href="#">[Alarm]SetTimeOnAck</a> — sets the property to the time the alarm is acknowledged.</li> <li>• <a href="#">[Alarm]SetTimeOnOff</a> — sets the property to the time the alarm becomes inactive.</li> </ul>	Long

**Note:** Once an alarm is disabled, it cannot be re-enabled unless you use the function [AlarmEnable](#) or [AlarmEnableRec](#).

For digital alarms, time stamped alarms, time stamped digital alarms, multi digital alarms, advanced alarms and double point status alarms, the following property can also be used.

Property	Description	Return Type
.Desc	Alarm description	String (128 bytes)

**Note:** Desc exists for every alarm type but will not return meaningful data for analog or time-stamped analog alarms.

For digital alarms, time-stamped digital alarms, advanced alarms and double point status alarms, the following property can also be used.

Property	Description	Return Type
.Delay	Alarm delay	Long

For analog alarms and time-stamped analog alarms, the following properties can also be used.

Property	Description	Return Type
.DevDelay	Deviation delay	Long
.DeadBand	Deadband	Real
.Deviation	Deviation	Real
.HighHigh	High High	Real
.High	High	Real
.LowLow	Low Low	Real
.Low	Low	Real
.HHDelay	High High delay	Long
.HDelay	High delay	Long
.LDelay	Low delay	Long
.LLDelay	Low Low delay	Long
.Rate	Rate	Real
.Setpoint	Setpoint	Real
.Value	Alarm tag Value	Real

For the digital properties below, only one can be true at any point in time for each alarm. They are arranged in order of priority, from lowest to highest.

.DVL	Deviation alarm triggered (Low)	Digital
.DVH	Deviation alarm triggered (High)	Digital

.R	Rate of Change alarm triggered	Digital
.L	Low alarm triggered	Digital
.H	High alarm triggered	Digital
.LL	Low Low alarm triggered	Digital
.HH	High High alarm triggered	Digital

**Note:** DVL and DVH are only evaluated if Deviation > 0. R is only evaluated if Rate > 0.

Some alarm properties return configuration data. If the user has not defined this information, the following defaults are provided:

Property	Default
.Setpoint	0
.HighHigh	3.4e+38
.High	3.4e+38
.LowLow	-3.4e+38
.Low	-3.4e+38
.Rate	0
.Deviation	0
.Deadband	0
.Category	0
.Priority	0

## See Also

[Writing to Alarm Properties](#)

[Setting Up Alarm Properties](#)

## Writing to Alarm Properties

If you have the necessary access rights, you can write to the following alarm properties. (Remember, the value you write to the property relates to the description. For example, if you set a digital alarm property to 1, you are making the description TRUE. If you set it to 0 (zero), you are making it FALSE.)

Property	Description	Input Type
.Ack	Alarm acknowledged (once)	Digital

Property	Description	Input Type
	acknowledged cannot be "unacknowledged")	
.Category	Alarm category	Integer
.Deadband	Alarm deadband	Real
.Delay	Alarm delay	Long
.Deviation	Deviation from setpoint	Real
.Disabled	Alarm disabled	Digital
.HighHigh	High High	Real
.High	High	Real
.LowLow	Low Low	Real
.Low	Low	Real
.HHDelay	High High delay	Long
.HDelay	High delay	Long
.LDelay	Low delay	Long
.LLDelay	Low Low delay	Long
.DevDelay	Deviation delay	Long
.Custom1 .Custom2 .Custom3 .Custom4 .Custom5 .Custom6 .Custom7 .Custom8	Custom field	String
.Paging	Alarm paged	Boolean
.PagingGroup	The paging group the alarm belongs to.	String
.Rate	Rate of change	Real

Analog alarm thresholds can also be changed using the [AlarmSetThreshold](#) function.

Note the following:

- The alarm tag needs to be unique.
- The alarms databases in the included projects on the alarm servers and the control client computer need to be identical.

## See Also

[Supported Alarm Properties](#)  
[Setting Up Alarm Properties](#)

### Setting Up Alarm Properties

To use alarm properties, you need to enable them on the Alarm Server.

1. In the **Topology** activity, select **Edit | Alarm Servers**.
2. Locate the alarm server you wish to update.
3. Specify the port that the Alarm Server will listen on using the Grid Editor or the Property Grid.

**Note:** You are not required to specify the port if using the default settings for ports.

## See Also

[Add an Alarm Server Process](#)  
[Supported Alarm Properties](#)  
[Writing to Alarm Properties](#)

### Alarm Server Database Specification

List of available Alarm Server Database tables.

Sample SQL Query:

```
SELECT CIAlarmObject.AlarmSource, COUNT (CDBEventJournal.Id) as "Number of Alarms"
FROM (((CDBEventJournal LEFT OUTER JOIN CIAlarmObject ON
CDBEventJournal.Id=CIAlarmObject.Id) LEFT OUTER JOIN CIAnalogAlarm ON
CDBEventJournal.Id=CIAnalogAlarm.Id) LEFT OUTER JOIN CIMultiStateDigAlarm ON
CDBEventJournal.Id=CIMultiStateDigAlarm.Id)
GROUP BY CIAlarmObject.AlarmSource, CDBEventJournal.Id
ORDER BY 2 DESC
```

<a href="#">CDBAlarmSummary</a>
<a href="#">CDBEventJournal</a>
<a href="#">CiAdvancedAlarm</a>
<a href="#">CiAlarmObject</a>

CiAnalogAlarm
CiDigitalAlarm
CiMultiStateDigAlarm
CiTimestampedAlarm

## See Also

[Alarm Server Side Synchronization](#)

## CDBAlarmSummary

ColumnName	DataType	Attribute	NotNull	Comment
RecordId	VARCHAR(100)	Attr Indexed Unique ReadOnly RecordId	NN	
RecordTime	TIMESTAMP	Attr ReadOnly Constraint	NN	
FileId	VARCHAR(17)	Attr Indexed ReadOnly	NN	
FileOffset	INTEGER	Attr ReadOnly	NN	
SeqNo	ULONGLONG	Attr ReadOnly SequenceNo	NN	
Id	INTEGER	Attr Indexed ReadOnly Constraint	NN	
AckUserName	VARCHAR(65)	Attr ReadOnly	NN	
DisabledByValue	TINYINT	Attr ReadOnly	NN	
DisabledByDesc	VARCHAR(65)	Attr ReadOnly	NN	
ClientAddress	INTEGER	Attr ReadOnly	NN	
ClientAddressDesc	VARCHAR(16)	Attr ReadOnly	NN	
SeverityDesc	VARCHAR(32)	Attr ReadOnly	NN	
SeverityValue	INTEGER	Attr ReadOnly	NN	
State	INTEGER	Attr ReadOnly	NN	
StateDesc	VARCHAR(24)	Attr ReadOnly	NN	

ColumnName	DataType	Attribute	NotNull	Comment
Deleted	BIT	Attr ReadOnly	NN	
Cached	BIT	Attr ReadOnly Constraint	NN	
ActiveTime	TIMESTAMP	Attr ReadOnly	NN	
InactiveTime	TIMESTAMP	Attr ReadOnly	NN	
AckTime	TIMESTAMP	Attr ReadOnly	NN	
UnAckTime	TIMESTAMP	Attr ReadOnly	NN	
DisabledTime	TIMESTAMP	Attr ReadOnly	NN	
ReceiptTime	TIMESTAMP	Attr ReadOnly	NN	
DisabledEndTime	TIMESTAMP	Attr ReadOnly	NN	
VisibleTime	TIMESTAMP	Attr ReadOnly	NN	
Source	VARCHAR(256)	Attr ReadOnly Constraint	NN	
ConditionName	VARCHAR(65)	Attr ReadOnly	NN	
SubConditionName	TIMESTAMP	Attr ReadOnly	NN	
AggregateName	VARCHAR(63)	Attr ReadOnly	NN	
AckComment	VARCHAR(33)	Attr ReadOnly	NN	
Message	VARCHAR(257)	Attr ReadOnly	NN	
EncodedMessage	VARCHAR(254)	Attr ReadOnly	NN	
SourceMessage	VARCHAR(257)	Attr ReadOnly	NN	
EncodedSourceMessage	VARCHAR(254)	Attr ReadOnly	NN	
Location	VARCHAR(33)	Attr ReadOnly	NN	
CustomStringField1	VARCHAR(254)	Attr ReadOnly	NN	Full User Name
CustomStringField2	VARCHAR(254)	Attr ReadOnly	NN	Native_SumDesc
CustomStringField3	VARCHAR(254)	Attr ReadOnly	NN	Native_Comment
CustomStringField4	VARCHAR(254)	Attr ReadOnly	NN	UserLocation

ColumnName	DataType	Attribute	NotNull	Comment
CustomStringField5	VARCHAR(254)	Attr ReadOnly	NN	InvariantRecordId
CustomStringField7	VARCHAR(254)	Attr ReadOnly	NN	UserDesc
CustomStringField8	VARCHAR(254)	Attr ReadOnly	NN	<reserved field>
CustomNumericField1	TINYINT	Attr ReadOnly	NN	<reserved field>
CustomNumericField2	TINYINT	Attr ReadOnly	NN	MarkAsDeleted - used to identify the record has been deleted, e.g. By AlarmSumDelete()
CustomNumericField3	TINYINT	Attr ReadOnly	NN	Logged - used to identify the record has been exported to alarm summary log device
CustomNumericField4	UINTEGER	Attr ReadOnly	NN	<reserved field>
CustomNumericField5	UINTEGER	Attr ReadOnly	NN	<reserved field>
CustomNumericField6	UINTEGER	Attr ReadOnly	NN	<reserved field>
CustomNumericField7	INTEGER	Attr ReadOnly	NN	StateNumeric
CustomNumericField8	INTEGER	Attr ReadOnly	NN	<reserved field>
CustomNumericField9	INTEGER	Attr ReadOnly	NN	<reserved field>
CustomNumericField10	ULONGLONG	Attr ReadOnly	NN	<reserved field>
CustomNumericField11	ULONGLONG	Attr ReadOnly	NN	<reserved field>
CustomNumericField12	ULONGLONG	Attr ReadOnly	NN	<reserved field>
CustomNumericField	LONGLONG	Attr ReadOnly	NN	Duration of the

ColumnName	DataType	Attribute	NotNull	Comment
d13				alarm condition referred by the alarm summary record
CustomNumericField14	LONGLONG	Attr ReadOnly	NN	<reserved field>
CustomNumericField15	LONGLONG	Attr ReadOnly	NN	<reserved field>
CustomNumericField16	FLOAT	Attr ReadOnly	NN	Value
CustomNumericField17	FLOAT	Attr ReadOnly	NN	SetPoint
CustomNumericField18	FLOAT	Attr ReadOnly	NN	<reserved field>
CustomNumericField19	REAL	Attr ReadOnly	NN	<reserved field>
CustomNumericField20	REAL	Attr ReadOnly	NN	<reserved field>
CustomNumericField21	REAL	Attr ReadOnly	NN	<reserved field>
CustomNumericField22	TIMESTAMP	Attr ReadOnly	NN	<reserved field>
CustomNumericField23	TIMESTAMP	Attr ReadOnly	NN	<reserved field>
CustomNumericField24	TIMESTAMP	Attr ReadOnly	NN	<reserved field>
AreaOfInterestId	INTEGER	Attr ReadOnly AOI	NN	
AreaOfInterest	VARCHAR(255)	Attr ReadOnly AOI	NN	

IndexName	IndexType	Columns
pkCDBAlarmSummary	Index	RecordId RecordI
idxRecordId	Index	RecordId

IndexName	IndexType	Columns
idxFileId	Index	FileId
idxTime	Index	Time
idxRecordTime	Index	RecordTime
idxId	Index	Id
idxSource	Index	Source
idxDeleted	Index	Deleted
idxInvariantRecordId	Index	InvariantRecordId

## See Also

[Alarm Server Side Synchronization](#)

## CDBEventJournal

ColumnName	DataType	Primary Key	NotNull	Flags	Default Value	Comment	AutoInc
AlarmStateDesc	VARCHAR(31)		NN				
AreaOfInterest	VARCHAR(254)		NN				
Category	VARCHAR(31)		NN				
ClientAddress	INTEGER		NN				
ClientAddressDesc	VARCHAR(15)		NN				
ClientName	VARCHAR(15)		NN				
CommentNo	INTEGER		NN				
CustomNumericField1	TINYINT		NN				
CustomNu	SMALLINT		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
mericField2							
CustomNumericField3	INTEGER		NN				
CustomStringField	VARCHAR(253)		NN				
Deleted	BIT		NN				
EncodedMessage	VARCHAR(253)		NN				
FileId	VARCHAR(16)		NN				
FileOffset	INTEGER		NN				
Foreground	INTEGER		NN				
Id	INTEGER		NN				
InvariantRecordId	VARCHAR(40)		NN				
Location	VARCHAR(32)		NN				
Message	VARCHAR(253)		NN				
MessageLong	INTEGER		NN				
NamedRegion	VARCHAR(31)		NN				
ReceiptTime	TIMESTAMP		NN				
RecordId	VARCHAR(33)		NN				
RecordTime	TIMESTAMP		NN				
SeqNo	INTEGER		NN				
Severity	VARCHAR(31)		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
SeverityDesc	VARCHAR(31)		NN				
SeverityValue	INTEGER		NN				
Source	VARCHAR(254)		NN				
SuppressionType	VARCHAR(253)		NN				
Time	TIMESTAMP		NN				
User	VARCHAR(63)		NN				

IndexName	IndexType	Columns
pkCDBEventJournal	Index	RecordId RecordI
idxRecordId	Index	RecordId
idxFileId	Index	FileId
idxTime	Index	Time
idxRecordTime	Index	RecordTime
idxId	Index	Id
idxSource	Index	Source
idxDeleted	Index	Deleted
idxInvariantRecordId	Index	InvariantRecordId

## See Also

[Alarm Server Side Synchronization](#)

[CiAdvancedAlarm](#)

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
Abstract	BIT		NN				
AccessControlList	INTEGER		NN				
AckTime	TIMESTAMP		NN				
ACLASText	VARCHAR(253)		NN				
AcqError	SMALLINT		NN				
AlarmArea	INTEGER		NN				
AlarmCanDisable	BIT		NN				
AlarmCanLocate	BIT		NN				
AlarmCanRespond	BIT		NN				
AlarmCanUnaccept	BIT		NN				
AlarmCategory	SMALLINT		NN				
AlarmDesc	VARCHAR(31)		NN				
AlarmDisabled	BIT		NN				
AlarmEnableConfirm	TINYINT		NN				
AlarmHandlerMethod	INTEGER		NN				
AlarmHandlerObjectId	INTEGER		NN				
AlarmLastUpdateDateTime	TIMESTAMP		NN				
AlarmPrivilege	SMALLINT		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
AlarmRedirection	BIT		NN				
AlarmRemovable	BIT		NN				
AlarmSeverity	SMALLINT		NN				
AlarmSeverityDesc	VARCHAR(31)		NN				
AlarmSource	INTEGER		NN				
AlarmSourceValid	BIT		NN				
AlarmState	INTEGER		NN				
AlarmType	VARCHAR(23)		NN				
AlarmViewClass	VARCHAR(23)		NN				
AlarmViewId	INTEGER		NN				
AlarmViewLink	INTEGER		NN				
AlarmViewName	VARCHAR(254)		NN				
AlmUnspSeverity	SMALLINT		NN				
AlmUnspSeverityDesc	VARCHAR(31)		NN				
AlmUnspState	INTEGER		NN				
AlmUnspStateDesc	VARCHAR(31)		NN				
Background	INTEGER		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
Blink	BIT		NN				
CiAdvancedAlarmState_AcceptComment	VARCHAR(253)		NN				
CiAdvancedAlarmState_Accepted	BIT		NN				
CiAdvancedAlarmState_AcceptTime	TIMESTAMP		NN				
CiAdvancedAlarmState_AcceptUser	VARCHAR(63)		NN				
CiAdvancedAlarmState_ActiveSubCondition	VARCHAR(31)		NN				
CiAdvancedAlarmState_AreaOfInterest	VARCHAR(253)		NN				
CiAdvancedAlarmState_AreaOfInterestId	INTEGER		NN				
CiAdvancedAlarmState_CanAccept	BIT		NN				
CiAdvancedAlarmState_CanDisable	BIT		NN				
CiAdvancedAlarmState	BIT		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
_CanRemove							
CiAdvancedAlarmState _CanRespond	BIT		NN				
CiAdvancedAlarmState _CanUnaccept	BIT		NN				
CiAdvancedAlarmState _ConditionActiveTime	TIMESTAMP		NN				
CiAdvancedAlarmState _CondName	VARCHAR(23)		NN				
CiAdvancedAlarmState _Cookie	INTEGER		NN				
CiAdvancedAlarmState _CurrSuppressingAlarmId	INTEGER		NN				
CiAdvancedAlarmState _CurrSuppressingCond	VARCHAR(127)		NN				
CiAdvancedAlarmState _CurrSuppressingConsDesc	VARCHAR(63)		NN				
CiAdvancedAlarmState _CurrSuppr	TINYINT		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
essingConsT ype							
CiAdvanced AlarmState _Disabled	BIT		NN				
CiAdvanced AlarmState _DisabledBy	VARCHAR(2 53)		NN				
CiAdvanced AlarmState _DisabledE ndTime	TIMESTAMP		NN				
CiAdvanced AlarmState _DisableTim e	TIMESTAMP		NN				
CiAdvanced AlarmState _EvtSeqNu mberClear	INTEGER		NN				
CiAdvanced AlarmState _EvtSeqNu mberSet	INTEGER		NN				
CiAdvanced AlarmState _Id	INTEGER		NN				
CiAdvanced AlarmState _InactiveTi me	TIMESTAMP		NN				
CiAdvanced AlarmState _LastUpdat eTime	TIMESTAMP		NN				
CiAdvanced AlarmState	VARCHAR(2 53)		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
_Message							
CiAdvancedAlarmState _OrigSuppressingAlarmId	INTEGER		NN				
CiAdvancedAlarmState _OrigSuppressingCond	VARCHAR(127)		NN				
CiAdvancedAlarmState _OrigSuppressingConsDesc	VARCHAR(63)		NN				
CiAdvancedAlarmState _OrigSuppressingConstype	TINYINT		NN				
CiAdvancedAlarmState _ReceiptTime	TIMESTAMP		NN				
CiAdvancedAlarmState _Response	VARCHAR(253)		NN				
CiAdvancedAlarmState _ResponseTime	TIMESTAMP		NN				
CiAdvancedAlarmState _ResponseUser	VARCHAR(63)		NN				
CiAdvancedAlarmState _Responsible	VARCHAR(127)		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
eEntity							
CiAdvancedAlarmState_ResponsiblityEntityType	TINYINT		NN				
CiAdvancedAlarmState_Severity	INTEGER		NN				
CiAdvancedAlarmState_SeverityDesc	VARCHAR(31)		NN				
CiAdvancedAlarmState_State	INTEGER		NN				
CiAdvancedAlarmState_StateDesc	VARCHAR(31)		NN				
CiAdvancedAlarmState_SubConditionActiveTime	TIMESTAMP		NN				
CiAdvancedAlarmState_SuppressedAlarmCount	INTEGER		NN				
CiAdvancedAlarmState_VisibleTime	TIMESTAMP		NN				
ColourPalette	INTEGER		NN				
Comment	VARCHAR(254)		NN				
ConditionAc	TIMESTAMP		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
tiveTime							
ConfigTime	TIMESTAMP		NN				
ConfigUser	VARCHAR(63)		NN				
ConfigValid	BIT		NN				
ConfigVersion	INTEGER		NN				
ConsAlarmsType	TINYINT		NN				
ConsAlarmsTypeDesc	VARCHAR(63)		NN				
ConsParentAlarmCond	INTEGER		NN				
ConsParentAlarmObjId	INTEGER		NN				
CustomFilters	VARCHAR(253)		NN				
DataTimestamp	TIMESTAMP		NN				
Delay	INTEGER		NN				
DeltaTime	INTEGER		NN				
Description	VARCHAR(254)		NN				
DisableTime	TIMESTAMP		NN				
DisplayName	VARCHAR(79)		NN				
DisplayTime	TIMESTAMP		NN				
DocumentContent	INTEGER		NN				
Equipment	VARCHAR(254)		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
ExcludeFromExclCtrl	BIT		NN				
ExclusiveCtrIActive	BIT		NN				
ExclusiveCtrIUser	VARCHAR(254)		NN				
Foreground	INTEGER		NN				
FullName	VARCHAR(254)		NN				
HelpPage	VARCHAR(64)		NN				
HelpViewClass	VARCHAR(23)		NN				
HelpViewId	INTEGER		NN				
HelpViewLink	INTEGER		NN				
HelpViewName	VARCHAR(254)		NN				
Historian	TINYINT		NN				
Id	INTEGER		NN				
InvariantRecordId	VARCHAR(40)		NN				
Latched	BIT		NN				
MarkerColorBad	INTEGER		NN				
MarkerColorGood	INTEGER		NN				
MemoryUsage	INTEGER		NN				
MobileHelpViewId	INTEGER		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
Name	VARCHAR(63)		NN				
NoteText	VARCHAR(253)		NN				
NoteTextLong	INTEGER		NN				
NoteTime	TIMESTAMP		NN				
ObjectLink	INTEGER		NN				
OffTime	TIMESTAMP		NN				
OnTime	TIMESTAMP		NN				
PagingEnabled	BIT		NN				
PagingGroup	VARCHAR(80)		NN				
ParentGroup	INTEGER		NN				
ParentGroupName	VARCHAR(254)		NN				
Priority	SMALLINT		NN				
Quality	INTEGER		NN				
Rights	INTEGER		NN				
StateNumeric	SMALLINT		NN				
StoredDynMetadataFields	INTEGER		NN				
SummaryRecordId	INTEGER		NN				
SuppressingAlarmId	INTEGER		NN				
Suppressing	VARCHAR(1)		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
Condition	27)						
Suppressing ConsType	TINYINT		NN				
Suppressing ConsTypeDesc	VARCHAR(63)		NN				
TypeDesc	VARCHAR(63)		NN				
TypeName	VARCHAR(63)		NN				
TypeNum	SMALLINT		NN				
UserLocation	VARCHAR(20)		NN				
UserMethods	BIT		NN				
UsrAlmSeverity	SMALLINT		NN				
UsrAlmSeverityDesc	VARCHAR(31)		NN				
UsrAlmState	INTEGER		NN				
UsrAlmStateDesc	VARCHAR(31)		NN				
UsrAlmUns pSeverity	SMALLINT		NN				
UsrAlmUns pSeverityDesc	VARCHAR(31)		NN				
UsrAlmUns pState	INTEGER		NN				
UsrAlmUns pStateDesc	VARCHAR(31)		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
WSAlmSeverity	SMALLINT		NN				
WSAlmSeverityDesc	VARCHAR(31)		NN				
WSAlmState	INTEGER		NN				
WSAlmStateDesc	VARCHAR(31)		NN				
WSAlmUnsPSeverity	SMALLINT		NN				
WSAlmUnsPSeverityDesc	VARCHAR(31)		NN				
WSAlmUnsPState	INTEGER		NN				
WSAlmUnsPStateDesc	VARCHAR(31)		NN				

IndexName	IndexType	Columns
pkCiAdvancedAlarm	Index	Id Id
idxId	Index	Id
idxParentGroupId	Index	ParentGroupId
idxFullName	Index	FullName
idxConsParentAlarmObjId	Index	ConsParentAlarmObjId
idxCiAdvancedAlarmState_Id	Index	CiAdvancedAlarmState_Id

## See Also

[Alarm Server Side Synchronization](#)

**CiAlarmObject**

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
Abstract	BIT		NN				
AccessControlList	INTEGER		NN				
AckTime	TIMESTAMP		NN				
ACLASText	VARCHAR(253)		NN				
AcqError	SMALLINT		NN				
AlarmArea	INTEGER		NN				
AlarmCanDisable	BIT		NN				
AlarmCanLocate	BIT		NN				
AlarmCanRespond	BIT		NN				
AlarmCanUnaccept	BIT		NN				
AlarmCategory	SMALLINT		NN				
AlarmDesc	VARCHAR(31)		NN				
AlarmDisabled	BIT		NN				
AlarmEnableConfirm	TINYINT		NN				
AlarmHandlerMethod	INTEGER		NN				
AlarmHandlerObjectId	INTEGER		NN				
AlarmLastUpdateDateTime	TIMESTAMP		NN				
AlarmPrivilege	SMALLINT		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
AlarmRedirection	BIT		NN				
AlarmRemovable	BIT		NN				
AlarmSeverity	SMALLINT		NN				
AlarmSeverityDesc	VARCHAR(31)		NN				
AlarmSource	INTEGER		NN				
AlarmSourceValid	BIT		NN				
AlarmState	INTEGER		NN				
AlarmType	VARCHAR(23)		NN				
AlarmViewClass	VARCHAR(23)		NN				
AlarmViewId	INTEGER		NN				
AlarmViewLink	INTEGER		NN				
AlarmViewName	VARCHAR(254)		NN				
AlmUnspSeverity	SMALLINT		NN				
AlmUnspSeverityDesc	VARCHAR(31)		NN				
AlmUnspState	INTEGER		NN				
AlmUnspStateDesc	VARCHAR(31)		NN				
Background	INTEGER		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
Blink	BIT		NN				
ColourPallette	INTEGER		NN				
Comment	VARCHAR(254)		NN				
ConditionActiveTime	TIMESTAMP		NN				
ConfigTime	TIMESTAMP		NN				
ConfigUser	VARCHAR(63)		NN				
ConfigValid	BIT		NN				
ConfigVersion	INTEGER		NN				
ConsAlarmsType	TINYINT		NN				
ConsAlarmsTypeDesc	VARCHAR(63)		NN				
ConsParentAlarmCond	INTEGER		NN				
ConsParentAlarmObjId	INTEGER		NN				
CustomFilters	VARCHAR(253)		NN				
DataTimestamp	TIMESTAMP		NN				
Delay	INTEGER		NN				
DeltaTime	INTEGER		NN				
Description	VARCHAR(254)		NN				
DisableTime	TIMESTAMP		NN				
DisplayName	VARCHAR(7)		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
e	9)						
DisplayTime	TIMESTAMP		NN				
DocumentContent	INTEGER		NN				
Equipment	VARCHAR(254)		NN				
ExcludeFromExclCtrl	BIT		NN				
ExclusiveCtlActive	BIT		NN				
ExclusiveCtlUser	VARCHAR(254)		NN				
Foreground	INTEGER		NN				
FullName	VARCHAR(254)		NN				
HelpPage	VARCHAR(64)		NN				
HelpViewClass	VARCHAR(23)		NN				
HelpViewId	INTEGER		NN				
HelpViewLink	INTEGER		NN				
HelpViewName	VARCHAR(254)		NN				
Historian	TINYINT		NN				
Id	INTEGER		NN				
InvariantRecordId	VARCHAR(40)		NN				
Latched	BIT		NN				
MarkerColorBad	INTEGER		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
MarkerColorGood	INTEGER		NN				
MemoryUsage	INTEGER		NN				
MobileHelpViewId	INTEGER		NN				
Name	VARCHAR(63)		NN				
NoteText	VARCHAR(253)		NN				
NoteTextLong	INTEGER		NN				
NoteTime	TIMESTAMP		NN				
NoteUser	VARCHAR(63)		NN				
ObjectLink	INTEGER		NN				
OnTime	TIMESTAMP		NN				
PagingEnabled	BIT		NN				
PagingGroup	VARCHAR(80)		NN				
ParentGroup	INTEGER		NN				
ParentGroupName	VARCHAR(254)		NN				
Priority	SMALLINT		NN				
Quality	INTEGER		NN				
Rights	INTEGER		NN				
StateNumeric	SMALLINT		NN				
StoredDyn	INTEGER		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
MetadataFields							
SummaryRecordId	INTEGER		NN				
SuppressingAlarmId	INTEGER		NN				
SuppressingCondition	VARCHAR(127)		NN				
SuppressingConsType	TINYINT		NN				
SuppressingConsTypeDesc	VARCHAR(63)		NN				
TypeDesc	VARCHAR(63)		NN				
TypeName	VARCHAR(63)		NN				
TypeNum	SMALLINT		NN				
UserLocation	VARCHAR(20)		NN				
UserMethods	BIT		NN				
UsrAlmSeverity	SMALLINT		NN				
UsrAlmSeverityDesc	VARCHAR(31)		NN				
UsrAlmStat	INTEGER		NN				
UsrAlmStat	VARCHAR(31)		NN				
UsrAlmUns	SMALLINT		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
UsrAlmUnspSeverityDesc	VARCHAR(31)		NN				
UsrAlmUnspState	INTEGER		NN				
UsrAlmUnspStateDesc	VARCHAR(31)		NN				
WSAlmSeverity	SMALLINT		NN				
WSAlmSeverityDesc	VARCHAR(31)		NN				
WSAlmStat e	INTEGER		NN				
WSAlmStat eDesc	VARCHAR(31)		NN				
WSAlmUnspSeverity	SMALLINT		NN				
WSAlmUnspSeverityDesc	VARCHAR(31)		NN				
WSAlmUnspStateDesc	VARCHAR(31)		NN				

IndexName	IndexType	Columns
pkCAAlarmObject	Index	Id
idxId	Index	Id
idxParentGroupId	Index	ParentGroupId
idxFullName	Index	FullName
idxConsParentAlarmObjId	Index	ConsParentAlarmObjId

## See Also

[Alarm Server Side Synchronization](#)

**CiAnalogAlarm**

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
Abstract	BIT		NN				
AccessContr olList	INTEGER		NN				
AckTime	TIMESTAMP		NN				
ACLAsText	VARCHAR(2 53)		NN				
AcqError	SMALLINT		NN				
AlarmArea	INTEGER		NN				
AlarmCanDi sable	BIT		NN				
AlarmCanLo cate	BIT		NN				
AlarmCanRe spond	BIT		NN				
AlarmCanU naccept	BIT		NN				
AlarmCateg ory	SMALLINT		NN				
AlarmDesc	VARCHAR(3 1)		NN				
AlarmDisabl ed	BIT		NN				
AlarmEnabl eConfirm	TINYINT		NN				
AlarmHandl erMethod	INTEGER		NN				
AlarmHandl erObjectId	INTEGER		NN				
AlarmLastU pdateTime	TIMESTAMP		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
AlarmPrivilege	SMALLINT		NN				
AlarmRedirection	BIT		NN				
AlarmRemovable	BIT		NN				
AlarmSeverity	SMALLINT		NN				
AlarmSeverityDesc	VARCHAR(31)		NN				
AlarmSource	INTEGER		NN				
AlarmSourceValid	BIT		NN				
AlarmState	INTEGER		NN				
AlarmType	VARCHAR(23)		NN				
AlarmViewClass	VARCHAR(23)		NN				
AlarmViewId	INTEGER		NN				
AlarmViewLink	INTEGER		NN				
AlarmViewName	VARCHAR(254)		NN				
AlmUnspSeverity	SMALLINT		NN				
AlmUnspSeverityDesc	VARCHAR(31)		NN				
AlmUnspState	INTEGER		NN				
AlmUnspSta	VARCHAR(3)		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
teDesc	1)						
Background	INTEGER		NN				
Blink	BIT		NN				
CiAnalogAla rmState _AcceptCo mment	VARCHAR(2 53)		NN				
CiAnalogAla rmState _Accepted	BIT		NN				
CiAnalogAla rmState _AcceptTim e	TIMESTAMP		NN				
CiAnalogAla rmState _AcceptUse r	VARCHAR(6 3)		NN				
CiAnalogAla rmState _ActiveSub Condition	VARCHAR(3 1)		NN				
CiAnalogAla rmState _AreaOfInte rest	VARCHAR(2 53)		NN				
CiAnalogAla rmState _AreaOfInte restId	INTEGER		NN				
CiAnalogAla rmState _CanAccept	BIT		NN				
CiAnalogAla rmState _CanDisabl e	BIT		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
CiAnalogAla rmState _CanRemov e	BIT		NN				
CiAnalogAla rmState _CanRespo nd	BIT		NN				
CiAnalogAla rmState _CanUnacc ept	BIT		NN				
CiAnalogAla rmState _Condition ActiveTime	TIMESTAMP		NN				
CiAnalogAla rmState _CondNam e	VARCHAR(2 3)		NN				
CiAnalogAla rmState _Cookie	INTEGER		NN				
CiAnalogAla rmState _CurrSuppr essingAlarm Id	INTEGER		NN				
CiAnalogAla rmState _CurrSuppr essingCond	VARCHAR(1 27)		NN				
CiAnalogAla rmState_< _CurrSuppre ssingConsD esc	VARCHAR(6 3)		NN				
CiAnalogAla	TINYINT		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
rmState _CurrSuppressingConsType							
CiAnalogAlarmState _Disabled	BIT		NN				
CiAnalogAlarmState _DisabledBy	VARCHAR(253)		NN				
CiAnalogAlarmState _DisabledEndTime	TIMESTAMP		NN				
CiAnalogAlarmState _DisableTime	TIMESTAMP		NN				
CiAnalogAlarmState _EvtSeqNumberClear	INTEGER		NN				
CiAnalogAlarmState _EvtSeqNumberSet	INTEGER		NN				
CiAnalogAlarmState_Id	INTEGER		NN				
CiAnalogAlarmState _InactiveTime	TIMESTAMP		NN				
CiAnalogAlarmState _LastUpdateTime	TIMESTAMP		NN				
CiAnalogAla	VARCHAR(2		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
rmState_Message	53)						
CiAnalogAlarmsState_OrigSuppressingAlarmId	INTEGER		NN				
CiAnalogAlarmsState_Ori gSuppressingCond	VARCHAR(127)		NN				
CiAnalogAlarmsState_OrigSuppressingConsDesc	VARCHAR(63)		NN				
CiAnalogAlarmsState_OrigSuppressingConstype	TINYINT		NN				
CiAnalogAlarmsState_ReceiptTime	TIMESTAMP		NN				
CiAnalogAlarmsState_Response	VARCHAR(253)		NN				
CiAnalogAlarmsState_ResponseTime	TIMESTAMP		NN				
CiAnalogAlarmsState_ResponseUser	VARCHAR(63)		NN				
CiAnalogAlarmsState	VARCHAR(127)		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
_ResponsibleEntity							
CiAnalogAlarmState	TINYINT		NN				
_ResponsibleEntityType							
CiAnalogAlarmState	INTEGER		NN				
_Severity							
CiAnalogAlarmState	VARCHAR(31)		NN				
_SeverityDesc							
CiAnalogAlarmState	INTEGER		NN				
_State							
CiAnalogAlarmState	VARCHAR(31)		NN				
_StateDesc							
CiAnalogAlarmState	TIMESTAMP		NN				
_SubConditionActiveTime							
CiAnalogAlarmState	INTEGER		NN				
_SuppressedAlarmCount							
CiAnalogAlarmState	TIMESTAMP		NN				
_VisibleTime							
ColourPalette	INTEGER		NN				
Comment	VARCHAR(254)		NN				
ConditionAc	TIMESTAMP		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
tiveTime							
ConfigTime	TIMESTAMP		NN				
ConfigUser	VARCHAR(63)		NN				
ConfigValid	BIT		NN				
ConfigVersion	INTEGER		NN				
ConsAlarmsType	TINYINT		NN				
ConsAlarmsTypeDesc	VARCHAR(63)		NN				
ConsParentAlarmCond	INTEGER		NN				
ConsParentAlarmObjId	INTEGER		NN				
CustomFilters	VARCHAR(253)		NN				
DataTimestamp	TIMESTAMP		NN				
Deadband	REAL		NN				
Delay	INTEGER		NN				
DeltaTime	INTEGER		NN				
Description	VARCHAR(254)		NN				
Deviation	REAL		NN				
DeviationDelay	INTEGER		NN				
DisableTime	TIMESTAMP		NN				
DisplayName	VARCHAR(79)		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
DisplayTime	TIMESTAMP		NN				
DocumentContent	INTEGER		NN				
Equipment	VARCHAR(254)		NN				
ExcludeFromExclCtrl	BIT		NN				
ExclusiveCtrlActive	BIT		NN				
ExclusiveCtrlUser	VARCHAR(254)		NN				
Foreground	INTEGER		NN				
Format	INTEGER		NN				
FullName	VARCHAR(254)		NN				
HelpPage	VARCHAR(64)		NN				
HelpViewClass	VARCHAR(23)		NN				
HelpViewId	INTEGER		NN				
HelpViewLink	INTEGER		NN				
HelpViewName	VARCHAR(254)		NN				
High	REAL		NN				
HighDelay	INTEGER		NN				
HighHigh	REAL		NN				
HighHighDelay	INTEGER		NN				
Historian	TINYINT		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
Id	INTEGER		NN				
InvariantRecordId	VARCHAR(40)		NN				
Latched	BIT		NN				
Low	REAL		NN				
LowDelay	INTEGER		NN				
LowLow	REAL		NN				
LowLowDelay	INTEGER		NN				
ManuallyNotified	BIT		NN				
MarkerColorBad	INTEGER		NN				
MarkerColorGood	INTEGER		NN				
MemoryUsage	INTEGER		NN				
MobileHelpViewId	INTEGER		NN				
Name	VARCHAR(63)		NN				
NoteText	VARCHAR(253)		NN				
NoteTextLong	INTEGER		NN				
NoteTime	TIMESTAMP		NN				
NoteUser	VARCHAR(63)		NN				
ObjectLink	INTEGER		NN				
OffTime	TIMESTAMP		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
OnTime	TIMESTAMP		NN				
PagingEnabled	BIT		NN				
PagingGroup	VARCHAR(80)		NN				
ParentGroup	INTEGER		NN				
ParentGroupName	VARCHAR(254)		NN				
Priority	SMALLINT		NN				
Quality	INTEGER		NN				
Rate	REAL		NN				
Rights	INTEGER		NN				
Setpoint	REAL		NN				
SetpointConfigured	BIT		NN				
StateNumeric	SMALLINT		NN				
StoredDynMetadataFields	INTEGER		NN				
SummaryRecordId	INTEGER		NN				
SuppressingAlarmId	INTEGER		NN				
SuppressingCondition	VARCHAR(127)		NN				
SuppressingConsType	TINYINT		NN				
SuppressingConsTypeDesc	VARCHAR(63)		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
TypeDesc	VARCHAR(63)		NN				
TypeName	VARCHAR(63)		NN				
TypeNum	SMALLINT		NN				
UserLocation	VARCHAR(20)		NN				
UserMethods	BIT		NN				
UsrAlmSeverity	SMALLINT		NN				
UsrAlmSeverityDesc	VARCHAR(31)		NN				
UsrAlmStat e	INTEGER		NN				
UsrAlmStat eDesc	VARCHAR(31)		NN				
UsrAlmUns pSeverity	SMALLINT		NN				
UsrAlmUns pSeverityDe sc	VARCHAR(31)		NN				
UsrAlmUns pState	INTEGER		NN				
UsrAlmUns pStateDesc	VARCHAR(31)		NN				
Value	REAL		NN				
WSAlmSeverity	SMALLINT		NN				
WSAlmSeverityDesc	VARCHAR(31)		NN				
WSAlmStat	INTEGER		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
e							
WSAlmStateDesc	VARCHAR(31)		NN				
WSAlmUns pSeverity	SMALLINT		NN				
WSAlmUns pSeverityDesc	VARCHAR(31)		NN				
WSAlmUns pState	INTEGER		NN				
WSAlmUns pStateDesc	VARCHAR(31)		NN				

IndexName	IndexType	Columns
pkCiAnalogAlarm	Index	Id
idxId	Index	Id
idxParentGroupId	Index	ParentGroupId
idxFullName	Index	FullName
idxConsParentAlarmObjId	Index	ConsParentAlarmObjId
idxCiAnalogAlarmState_Id	Index	CiAnalogAlarmState_Id

## See Also

[Alarm Server Side Synchronization](#)

## CiDigitalAlarm

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
Abstract	BIT		NN				
AccessControlList	INTEGER		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
AckTime	TIMESTAMP		NN				
ACLAsText	VARCHAR(253)		NN				
AcqError	SMALLINT		NN				
AlarmArea	INTEGER		NN				
AlarmCanDisable	BIT		NN				
AlarmCanLocate	BIT		NN				
AlarmCanRespond	BIT		NN				
AlarmCanUnaccept	BIT		NN				
AlarmCategory	SMALLINT		NN				
AlarmDesc	VARCHAR(31)		NN				
AlarmDisabled	BIT		NN				
AlarmEnableConfirm	TINYINT		NN				
AlarmHandlerMethod	INTEGER		NN				
AlarmHandlerObjectId	INTEGER		NN				
AlarmLastUpdateDateTime	TIMESTAMP		NN				
AlarmPrivilege	SMALLINT		NN				
AlarmRedirection	BIT		NN				
AlarmRemo	BIT		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
vable							
AlarmSeverity	SMALLINT		NN				
AlarmSeverityDesc	VARCHAR(31)		NN				
AlarmSource	INTEGER		NN				
AlarmSourceValid	BIT		NN				
AlarmState	INTEGER		NN				
AlarmType	VARCHAR(23)		NN				
AlarmViewClass	VARCHAR(23)		NN				
AlarmViewId	INTEGER		NN				
AlarmViewLink	INTEGER		NN				
AlarmViewName	VARCHAR(254)		NN				
AlmUnspSeverity	SMALLINT		NN				
AlmUnspSeverityDesc	VARCHAR(31)		NN				
AlmUnspState	INTEGER		NN				
AlmUnspStateDesc	VARCHAR(31)		NN				
Background	INTEGER		NN				
Blink	BIT		NN				
CiDigitalAlarmState_	VARCHAR(253)		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
AcceptComment							
CiDigitalAlarmState_Accepted	BIT		NN				
CiDigitalAlarmState_AcceptTime	TIMESTAMP		NN				
CiDigitalAlarmState_AcceptUser	VARCHAR(63)		NN				
CiDigitalAlarmState_ActiveSubCondition	VARCHAR(31)		NN				
CiDigitalAlarmState_AreaOfInterest	VARCHAR(253)		NN				
CiDigitalAlarmState_AreaOfInterestId	INTEGER		NN				
CiDigitalAlarmState_CanAccept	BIT		NN				
CiDigitalAlarmState_CanDisable	BIT		NN				
CiDigitalAlarmState_CanRemove	BIT		NN				
CiDigitalAlarmState_CanRespond	BIT		NN				
CiDigitalAlarm	BIT		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
mState_CanUnaccep							
CiDigitalAlar mState_ConditionActiveTime	TIMESTAMP		NN				
CiDigitalAlar mState_CondName	VARCHAR(23)		NN				
CiDigitalAlar mState_Cookie	INTEGER		NN				
CiDigitalAlar mState_CurrSupressingAlarmId	INTEGER		NN				
CiDigitalAlar mState_CurrSupressingCond	VARCHAR(127)		NN				
CiDigitalAlar mState_CurrSupressingConsDesc	VARCHAR(63)		NN				
CiDigitalAlar mState_CurrSupressingConsType	TINYINT		NN				
CiDigitalAlar mState_Disabled	BIT		NN				
CiDigitalAlar mState_DisabledBy	VARCHAR(253)		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
CiDigitalAlar mState_ DisabledEn dTime	TIMESTAMP		NN				
CiDigitalAlar mState_ DisableTime	TIMESTAMP		NN				
CiDigitalAlar mState_ EvtSeqNum berClear	INTEGER		NN				
CiDigitalAlar mState_ EvtSeqNum berSet	INTEGER		NN				
CiDigitalAlar mState_ Id	INTEGER		NN				
CiDigitalAlar mState_ InactiveTim e	TIMESTAMP		NN				
CiDigitalAlar mState_ LastUpdateTime	TIMESTAMP		NN				
CiDigitalAlar mState_ Message	VARCHAR(2 53)		NN				
CiDigitalAlar mState_ OrigSuppres singAlarmId	INTEGER		NN				
CiDigitalAlar mState_ OrigSuppres singCond	VARCHAR(1 27)		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
CiDigitalAlar mState_ OrigSuppres singConsDe sc	VARCHAR(6 3)		NN				
CiDigitalAlar mState_ OrigSuppres singConsTy pe	TINYINT		NN				
CiDigitalAlar mState_ ReceiptTim e	TIMESTAMP		NN				
CiDigitalAlar mState_ Response	VARCHAR(2 53)		NN				
CiDigitalAlar mState_ ResponseTi me	TIMESTAMP		NN				
CiDigitalAlar mState_ ResponseUs er	VARCHAR(6 3)		NN				
CiDigitalAlar mState_ Responsible Entity	VARCHAR(1 27)		NN				
CiDigitalAlar mState_ Responsible EntityType	TINYINT		NN				
CiDigitalAlar mState_ Severity	INTEGER		NN				
CiDigitalAlar mState_	VARCHAR(3 1)		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
SeverityDesc							
CiDigitalAlarmState_State	INTEGER		NN				
CiDigitalAlarmState_StateDesc	VARCHAR(31)		NN				
CiDigitalAlarmState_SubConditionActiveTime	TIMESTAMP		NN				
CiDigitalAlarmState_SuppressedAlarmCount	INTEGER		NN				
CiDigitalAlarmState_VisibleTime	TIMESTAMP		NN				
ColourPallette	INTEGER		NN				
Comment	VARCHAR(254)		NN				
ConditionActiveTime	TIMESTAMP		NN				
ConfigTime	TIMESTAMP		NN				
ConfigUser	VARCHAR(63)		NN				
ConfigValid	BIT		NN				
ConfigVersion	INTEGER		NN				
ConsAlarmsType	TINYINT		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
ConsAlarmsTypeDesc	VARCHAR(63)		NN				
ConsParentAlarmCond	INTEGER		NN				
ConsParentAlarmObjId	INTEGER		NN				
CustomFilters	VARCHAR(253)		NN				
TimeStamp	TIMESTAMP		NN				
Delay	INTEGER		NN				
DeltaTime	INTEGER		NN				
Description	VARCHAR(254)		NN				
DisableTime	TIMESTAMP		NN				
DisplayName	VARCHAR(79)		NN				
DisplayTime	TIMESTAMP		NN				
DocumentContent	INTEGER		NN				
Equipment	VARCHAR(254)		NN				
ExcludeFromExclCtrl	BIT		NN				
ExclusiveCtlActive	BIT		NN				
ExclusiveCtlUser	VARCHAR(254)		NN				
Foreground	INTEGER		NN				
FullName	VARCHAR(254)		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
HelpPage	VARCHAR(64)		NN				
HelpViewClass	VARCHAR(23)		NN				
HelpViewId	INTEGER		NN				
HelpViewLink	INTEGER		NN				
HelpViewName	VARCHAR(254)		NN				
Historian	TINYINT		NN				
Id	INTEGER		NN				
InvariantRecordId	VARCHAR(40)		NN				
Latched	BIT		NN				
ManuallyNotified	BIT		NN				
MarkerColorBad	INTEGER		NN				
MarkerColorGood	INTEGER		NN				
MemoryUsage	INTEGER		NN				
MobileHelpViewId	INTEGER		NN				
Name	VARCHAR(63)		NN				
NoteText	VARCHAR(253)		NN				
NoteTextLong	INTEGER		NN				
NoteTime	TIMESTAMP		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
NoteUser	VARCHAR(63)		NN				
ObjectLink	INTEGER		NN				
OffTime	TIMESTAMP		NN				
OnTime	TIMESTAMP		NN				
PagingEnabled	BIT		NN				
PagingGroup	VARCHAR(80)		NN				
ParentGroupId	INTEGER		NN				
ParentGroupName	VARCHAR(254)		NN				
Priority	SMALLINT		NN				
Quality	INTEGER		NN				
Rights	INTEGER		NN				
StateNumeric	SMALLINT		NN				
StoredDynMetadataFields	INTEGER		NN				
SummaryRecordId	INTEGER		NN				
SuppressingAlarmId	INTEGER		NN				
SuppressingCondition	VARCHAR(127)		NN				
SuppressingConsType	TINYINT		NN				
SuppressingConsTypeDesc	VARCHAR(63)		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
TypeDesc	VARCHAR(63)		NN				
TypeName	VARCHAR(63)		NN				
TypeNum	SMALLINT		NN				
UserLocation	VARCHAR(20)		NN				
UserMethods	BIT		NN				
UsrAlmSeverity	SMALLINT		NN				
UsrAlmSeverityDesc	VARCHAR(31)		NN				
UsrAlmState	INTEGER		NN				
UsrAlmStateDesc	VARCHAR(31)		NN				
UsrAlmUnsPSeverity	SMALLINT		NN				
UsrAlmUnsPSeverityDesc	VARCHAR(31)		NN				
UsrAlmUnsPState	INTEGER		NN				
UsrAlmUnsPStateDesc	VARCHAR(31)		NN				
WSAlmSeverity	SMALLINT		NN				
WSAlmSeverityDesc	VARCHAR(31)		NN				
WSAlmState	INTEGER		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
WSAlmStateDesc	VARCHAR(31)		NN				
WSAlmUnspSeverity	SMALLINT		NN				
WSAlmUnspSeverityDesc	VARCHAR(31)		NN				
WSAlmUnspState	INTEGER		NN				
WSAlmUnspStateDesc	VARCHAR(31)		NN				

IndexName	IndexType	Columns
pkCiDigitalAlarm	Index	Id
idxId	Index	Id
idxParentGroupId	Index	ParentGroupId
idxFullName	Index	FullName
idxConsParentAlarmObjId	Index	ConsParentAlarmObjId
idxCiDigitalAlarmState_Id	Index	CiDigitalAlarmState_Id

## See Also

[Alarm Server Side Synchronization](#)

## CiMultiStateDigAlarm

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment
Abstract	BIT		NN			
AccessControlList	INTEGER		NN			
AckTime	TIMESTAMP		NN			

ColumnName	DataType	Primary Key	NotNull	Flags	Default Value	Comment
ACLAsText	VARCHAR(253 )		NN			
AcqError	SMALLINT		NN			
AlarmArea	INTEGER		NN			
AlarmCanDisable	BIT		NN			
AlarmCanLocate	BIT		NN			
AlarmCanRespond	BIT		NN			
AlarmCanUnaccept	BIT		NN			
AlarmCategory	SMALLINT		NN			
AlarmDesc	VARCHAR(31)		NN			
AlarmDisabled	BIT		NN			
AlarmEnableConfirm	TINYINT		NN			
AlarmHandlerMethod	INTEGER		NN			
AlarmHandlerObjectId	INTEGER		NN			
AlarmLastUpdateTime	TIMESTAMP		NN			
AlarmPrivilege	SMALLINT		NN			
AlarmRedirection	BIT		NN			
AlarmRemovable	BIT		NN			
AlarmSeverity	SMALLINT		NN			

ColumnName	DataType	Primary Key	NotNull	Flags	Default Value	Comment
AlarmSeverityDesc	VARCHAR(31)		NN			
AlarmSource	INTEGER		NN			
AlarmSourceValid	BIT		NN			
AlarmState	INTEGER		NN			
AlarmType	VARCHAR(23)		NN			
AlarmViewClass	VARCHAR(23)		NN			
AlarmViewId	INTEGER		NN			
AlarmViewLink	INTEGER		NN			
AlarmViewName	VARCHAR(254)		NN			
AlmUnspSeverity	SMALLINT		NN			
AlmUnspSeverityDesc	VARCHAR(31)		NN			
AlmUnspState	INTEGER		NN			
AlmUnspStateDesc	VARCHAR(31)		NN			
Background	INTEGER		NN			
Blink	BIT		NN			
CiMultiDigAlarmState_AcceptComment	VARCHAR(253)		NN			
CiMultiDigAlarmState_Accepted	BIT		NN			
CiMultiDigAlarmState_	TIMESTAMP		NN			

ColumnName	DataType	Primary Key	NotNull	Flags	Default Value	Comment
AcceptTime						
CiMultiDigAla rmState_AcceptUser	VARCHAR(63)		NN			
CiMultiDigAla rmState_ActiveSubCon dition	VARCHAR(31)		NN			
CiMultiDigAla rmState_AreaOfInteres t	VARCHAR(253 )		NN			
CiMultiDigAla rmState_AreaOfInteres tId	INTEGER		NN			
CiMultiDigAla rmState_CanAccept	BIT		NN			
CiMultiDigAla rmState_CanDisable	BIT		NN			
CiMultiDigAla rmState_CanRemove	BIT		NN			
CiMultiDigAla rmState_CanRespond	BIT		NN			
CiMultiDigAla rmState_CanUnaccept	BIT		NN			
CiMultiDigAla rmState_ConditionActi veTime	TIMESTAMP		NN			
CiMultiDigAla rmState_	VARCHAR(23)		NN			

ColumnName	DataType	Primary Key	NotNull	Flags	Default Value	Comment
CondName						
CiMultiDigAlarmState_Cookie	INTEGER		NN			
CiMultiDigAlarmState_CurrSuppressingAlarmId	INTEGER		NN			
CiMultiDigAlarmState_CurrSuppressingCond	VARCHAR(127)		NN			
CiMultiDigAlarmState_CurrSuppressingConsDesc	VARCHAR(63)		NN			
CiMultiDigAlarmState_CurrSuppressingConsType	TINYINT		NN			
CiMultiDigAlarmState_Disabled	BIT		NN			
CiMultiDigAlarmState_DisabledBy	VARCHAR(253)		NN			
CiMultiDigAlarmState_DisabledEndTime	TIMESTAMP		NN			
CiMultiDigAlarmState_DisableTime	TIMESTAMP		NN			
CiMultiDigAlarmState_EvtSeqNumberClear	INTEGER		NN			

ColumnName	DataType	Primary Key	NotNull	Flags	Default Value	Comment
CiMultiDigAla rmState_ EvtSeqNumbe rSet	INTEGER		NN			
CiMultiDigAla rmState_Id	INTEGER		NN			
CiMultiDigAla rmState_ InactiveTime	TIMESTAMP		NN			
CiMultiDigAla rmState_ LastUpdateTime	TIMESTAMP		NN			
CiMultiDigAla rmState_ Message	VARCHAR(253 )		NN			
CiMultiDigAla rmState_ OrigSuppressi ngAlarmId	INTEGER		NN			
CiMultiDigAla rmState_ OrigSuppressi ngCond	VARCHAR(127 )		NN			
CiMultiDigAla rmState_ OrigSuppressi ngConsDesc	VARCHAR(63)		NN			
CiMultiDigAla rmState_ OrigSuppressi ngConsType	TINYINT		NN			
CiMultiDigAla rmState_ ReceiptTime	TIMESTAMP		NN			
CiMultiDigAla rmState_	VARCHAR(253 )		NN			

ColumnName	DataType	Primary Key	NotNull	Flags	Default Value	Comment
Response						
CiMultiDigAlarmState_ResponseTime	TIMESTAMP		NN			
CiMultiDigAlarmState_ResponseUser	VARCHAR(63)		NN			
CiMultiDigAlarmState_ResponsibleEntity	VARCHAR(127)		NN			
CiMultiDigAlarmState_ResponsibleEntityType	TINYINT		NN			
CiMultiDigAlarmState_Severity	INTEGER		NN			
CiMultiDigAlarmState_SeverityDesc	VARCHAR(31)		NN			
CiMultiDigAlarmState_State	INTEGER		NN			
CiMultiDigAlarmState_StateDesc	VARCHAR(31)		NN			
CiMultiDigAlarmState_SubConditionActiveTime	TIMESTAMP		NN			
CiMultiDigAlarmState_SuppressedAlarmCount	INTEGER		NN			
CiMultiDigAlarmState_	TIMESTAMP		NN			

ColumnName	DataType	Primary Key	NotNull	Flags	Default Value	Comment
VisibleTime						
ColourPalette	INTEGER		NN			
Comment	VARCHAR(254 )		NN			
ConditionActiveTime	TIMESTAMP		NN			
ConfigTime	TIMESTAMP		NN			
ConfigUser	VARCHAR(63)		NN			
ConfigValid	BIT		NN			
ConfigVersion	INTEGER		NN			
ConsAlarmsType	TINYINT		NN			
ConsAlarmsTypeDesc	VARCHAR(63)		NN			
ConsParentAIarmCond	INTEGER		NN			
ConsParentAIarmObjId	INTEGER		NN			
CurrentStateDesc	VARCHAR(15)		NN			
CurrentStateIndex	SMALLINT		NN			
CustomFilters	VARCHAR(253 )		NN			
DataTimestamp	TIMESTAMP		NN			
Delay	INTEGER		NN			
DeltaTime	INTEGER		NN			
Description	VARCHAR(254 )		NN			

ColumnName	DataType	Primary Key	NotNull	Flags	Default Value	Comment
DisableTime	TIMESTAMP		NN			
DisplayName	VARCHAR(79)		NN			
DisplayTime	TIMESTAMP		NN			
DocumentContent	INTEGER		NN			
Equipment	VARCHAR(254 )		NN			
ExcludeFromExclCtrl	BIT		NN			
ExclusiveCtrlActive	BIT		NN			
ExclusiveCtrlUser	VARCHAR(254 )		NN			
Foreground	INTEGER		NN			
FullName	VARCHAR(254 )		NN			
HelpPage	VARCHAR(64)		NN			
HelpViewClass	VARCHAR(23)		NN			
HelpViewId	INTEGER		NN			
HelpViewLink	INTEGER		NN			
HelpViewName	VARCHAR(254 )		NN			
Historian	TINYINT		NN			
Id	INTEGER		NN			
InvariantRecordId	VARCHAR(40)		NN			
Latched	BIT		NN			
MarkerColourBad	INTEGER		NN			

ColumnName	DataType	Primary Key	NotNull	Flags	Default Value	Comment
MarkerColourGood	INTEGER		NN			
MemoryUsage	INTEGER		NN			
MobileHelpViewId	INTEGER		NN			
Name	VARCHAR(63)		NN			
NoteText	VARCHAR(253 )		NN			
NoteTextLong	INTEGER		NN			
NoteTime	TIMESTAMP		NN			
NoteUser	VARCHAR(63)		NN			
ObjectLink	INTEGER		NN			
OffTime	TIMESTAMP		NN			
OldStateDesc	VARCHAR(15)		NN			
OldStateIndex	SMALLINT		NN			
OnTime	TIMESTAMP		NN			
PagingEnabled	BIT		NN			
PagingGroup	VARCHAR(80)		NN			
ParentGroupId	INTEGER		NN			
ParentGroupName	VARCHAR(254 )		NN			
Priority	SMALLINT		NN			
Quality	INTEGER		NN			
ReActivateEnabled	BIT		NN			
Rights	INTEGER		NN			

ColumnName	DataType	Primary Key	NotNull	Flags	Default Value	Comment
StateDescriptions	VARCHAR(253 )		NN			
StateNumeric	SMALLINT		NN			
States	VARCHAR(253 )		NN			
StoredDynMetadataFields	INTEGER		NN			
SummaryRecordId	INTEGER		NN			
SuppressingAlarmId	INTEGER		NN			
SuppressingCondition	VARCHAR(127 )		NN			
SuppressingConditionsType	TINYINT		NN			
SuppressingConditionsTypeDesc	VARCHAR(63)		NN			
SuppressionGroup	SMALLINT		NN			
SuppressionLevel	SMALLINT		NN			
TypeDesc	VARCHAR(63)		NN			
TypeName	VARCHAR(63)		NN			
TypeNum	SMALLINT		NN			
UserLocation	VARCHAR(20)		NN			
UserMethods	BIT		NN			
UsrAlmSeverity	SMALLINT		NN			
UsrAlmSeverityDesc	VARCHAR(31)		NN			
UsrAlmState	INTEGER		NN			

ColumnName	DataType	Primary Key	NotNull	Flags	Default Value	Comment
UsrAlmStateDesc	VARCHAR(31)		NN			
UsrAlmUnspSeverity	SMALLINT		NN			
UsrAlmUnspSeverityDesc	VARCHAR(31)		NN			
UsrAlmUnspState	INTEGER		NN			
UsrAlmUnspStateDesc	VARCHAR(31)		NN			
WSAlmSeverity	SMALLINT		NN			
WSAlmSeverityDesc	VARCHAR(31)		NN			
WSAlmState	INTEGER		NN			
WSAlmStateDesc	VARCHAR(31)		NN			
WSAlmUnspSeverity	SMALLINT		NN			
WSAlmUnspSeverityDesc	VARCHAR(31)		NN			
WSAlmUnspState	INTEGER		NN			
WSAlmUnspStateDesc	VARCHAR(31)		NN			

IndexName	IndexName	Columns
pkCiMultiStateDigAlarm	Index	Id
idxId	Index	Id
idxParentGroupId	Index	ParentGroupId
idxFullName	Index	FullName

IndexName	IndexName	Columns
idxConsParentAlarmObjId	Index	ConsParentAlarmObjId
idxCiMultiDigAlarmState_Id	Index	CiMultiDigAlarmState_Id

**See Also**

[Alarm Server Side Synchronization](#)

**CiTimestampedAlarm**

ColumnName	DataType	Primary Key	NotNull	Flags	Default Value	Comment	AutoInc
ACLAsText	VARCHAR(253)		NN				
AccessControlList	INTEGER		NN				
AcqError	SMALLINT		NN				
AlarmCanDisable	BIT		NN				
AlarmCanRespond	BIT		NN				
AlarmCategory	SMALLINT		NN				
AlarmDisabled	BIT		NN				
AlarmHandlerMethod	INTEGER		NN				
AlarmLastUpdateTime	TIMESTAMP		NN				
AlarmRedirection	BIT		NN				
AlarmSeverity	SMALLINT		NN				
AlarmSource	INTEGER		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
AlarmState	INTEGER		NN				
AlarmViewClass	VARCHAR(23)		NN				
AlarmViewLink	INTEGER		NN				
AlmUnspSeverity	SMALLINT		NN				
AlmUnspState	INTEGER		NN				
Background	INTEGER		NN				
CiTimestampedAlarmState_AcceptComment	VARCHAR(253)		NN				
CiTimestampedAlarmState_AcceptUser	VARCHAR(63)		NN				
CiTimestampedAlarmState_ActiveSubCondition	VARCHAR(31)		NN				
CiTimestampedAlarmState_AreaOfInterestId	INTEGER		NN				
CiTimestampedAlarmState_CanDisable	BIT		NN				
CiTimestampedAlarmState_CanRespon	BIT		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
d							
CiTimestam pedAlarmSt ate_CondName	VARCHAR(23)		NN				
CiTimestam pedAlarmSt ate_Cookie	INTEGER		NN				
CiTimestam pedAlarmSt ate_CurrSuppre ssingCond	VARCHAR(127)		NN				
CiTimestam pedAlarmSt ate_CurrSuppre ssingConstype	TINYINT		NN				
CiTimestam pedAlarmSt ate_Disabled	BIT		NN				
CiTimestam pedAlarmSt ate_DisabledEndTime	TIMESTAMP		NN				
CiTimestam pedAlarmSt ate_EvtSeqNumberSet	INTEGER		NN				
CiTimestam pedAlarmSt ate_InactiveTime	TIMESTAMP		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
CiTimestam pedAlarmSt ate_ Message	VARCHAR(2 53)		NN				
CiTimestam pedAlarmSt ate_ OrigSuppres singCond	VARCHAR(1 27)		NN				
CiTimestam pedAlarmSt ate_ OrigSuppres singConsTy pe	TINYINT		NN				
CiTimestam pedAlarmSt ate_ Response	VARCHAR(2 53)		NN				
CiTimestam pedAlarmSt ate_ ResponseUs er	VARCHAR(6 3)		NN				
CiTimestam pedAlarmSt ate_ Responsible EntityType	TINYINT		NN				
CiTimestam pedAlarmSt ate_ SeverityDes c	VARCHAR(3 1)		NN				
CiTimestam pedAlarmSt ate_ StateDesc	VARCHAR(3 1)		NN				
CiTimestam	INTEGER		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
pedAlarmState_SuppressedAlarmCount							
ColourPalette	INTEGER		NN				
ConditionActiveTime	TIMESTAMP		NN				
ConfigUser	VARCHAR(63)		NN				
ConfigVersion	INTEGER		NN				
ConsAlarmsTypeDesc	VARCHAR(63)		NN				
ConsParentAlarmObjId	INTEGER		NN				
DataTimestamp	TIMESTAMP		NN				
DeltaTime	INTEGER		NN				
DisableTime	TIMESTAMP		NN				
DisplayTime	TIMESTAMP		NN				
Equipment	VARCHAR(254)		NN				
ExclusiveCtrIActive	BIT		NN				
Foreground	INTEGER		NN				
HelpPage	VARCHAR(64)		NN				
HelpViewId	INTEGER		NN				
HelpViewName	VARCHAR(254)		NN				
Id	INTEGER		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
Latched	BIT		NN				
MarkerColorGood	INTEGER		NN				
MobileHelpViewId	INTEGER		NN				
NoteText	VARCHAR(253)		NN				
NoteTime	TIMESTAMP		NN				
ObjectLink	INTEGER		NN				
OnTime	TIMESTAMP		NN				
PagingGroup	VARCHAR(80)		NN				
ParentGroup	VARCHAR(254)		NN				
Quality	INTEGER		NN				
StateNumeric	SMALLINT		NN				
SummaryRecordId	INTEGER		NN				
SuppressingCondition	VARCHAR(127)		NN				
SuppressingConstTypeDesc	VARCHAR(63)		NN				
TypeName	VARCHAR(63)		NN				
UserLocation	VARCHAR(20)		NN				
UsrAlmSeverity	SMALLINT		NN				
UsrAlmStat	INTEGER		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
UsrAlmUnspSeverity	SMALLINT		NN				
UsrAlmUnspState	INTEGER		NN				
WSAlmSeverity	SMALLINT		NN				
WSAlmStat e	INTEGER		NN				
WSAlmUnspSeverity	SMALLINT		NN				
WSAlmUnspState	INTEGER		NN				
Abstract	BIT		NN				
AckTime	TIMESTAMP		NN				
AlarmArea	INTEGER		NN				
AlarmCanLocate	BIT		NN				
AlarmCanU naccept	BIT		NN				
AlarmDesc	VARCHAR(31)		NN				
AlarmEnabl eConfirm	TINYINT		NN				
AlarmHandl erObjectId	INTEGER		NN				
AlarmPrivile ge	SMALLINT		NN				
AlarmRemo vable	BIT		NN				
AlarmSeveri tyDesc	VARCHAR(31)		NN				
AlarmSourc	BIT		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
eValid							
AlarmType	VARCHAR(23)		NN				
AlarmViewId	INTEGER		NN				
AlarmViewName	VARCHAR(254)		NN				
AlmUnspSeverityDesc	VARCHAR(31)		NN				
AlmUnspStateDesc	VARCHAR(31)		NN				
Blink	BIT		NN				
CiTimestampedAlarmState_AcceptTime	TIMESTAMP		NN				
CiTimestampedAlarmState_Accepted	BIT		NN				
CiTimestampedAlarmState_AreaOfInterest	VARCHAR(253)		NN				
CiTimestampedAlarmState_CanAccept	BIT		NN				
CiTimestampedAlarmState_CanRemove	BIT		NN				
CiTimestampedAlarmState_	BIT		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
CanUnacceppt							
CiTimestampedAlarmState_ConditionActiveTime	TIMESTAMP		NN				
CiTimestampedAlarmState_CurrSupressingAlarmId	INTEGER		NN				
CiTimestampedAlarmState_CurrSupressingConsDesc	VARCHAR(63)		NN				
CiTimestampedAlarmState_DisableTime	TIMESTAMP		NN				
CiTimestampedAlarmState_DisabledBy	VARCHAR(253)		NN				
CiTimestampedAlarmState_EvtSeqNumberClear	INTEGER		NN				
CiTimestampedAlarmState_Id	INTEGER		NN				
CiTimestampedAlarmState_	TIMESTAMP		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
LastUpdateTime							
CiTimestampedAlarmState_OrigSuppressingAlarmId	INTEGER		NN				
CiTimestampedAlarmState_OrigSuppressingConsDesc	VARCHAR(63)		NN				
CiTimestampedAlarmState_ReceiptTime	TIMESTAMP		NN				
CiTimestampedAlarmState_ResponseTime	TIMESTAMP		NN				
CiTimestampedAlarmState_ResponsibleEntity	VARCHAR(127)		NN				
CiTimestampedAlarmState_Severity	INTEGER		NN				
CiTimestampedAlarmState_State	INTEGER		NN				
CiTimestampedAlarmState_	TIMESTAMP		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
SubConditionActiveTime							
CiTimestampedAlarmState_VisibleTime	TIMESTAMP		NN				
Comment	VARCHAR(254)		NN				
ConfigTime	TIMESTAMP		NN				
ConfigValid	BIT		NN				
ConsAlarmsType	TINYINT		NN				
ConsParentAlarmCond	INTEGER		NN				
CustomFilters	VARCHAR(253)		NN				
Delay	INTEGER		NN				
Description	VARCHAR(254)		NN				
DisplayName	VARCHAR(79)		NN				
DocumentContent	INTEGER		NN				
ExcludeFromExclCtrl	BIT		NN				
ExclusiveCtlUser	VARCHAR(254)		NN				
FullName	VARCHAR(254)		NN				
HelpViewClass	VARCHAR(23)		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
HelpViewLink	INTEGER		NN				
Historian	TINYINT		NN				
InvariantRecordId	VARCHAR(40)		NN				
MarkerColorBad	INTEGER		NN				
MemoryUsage	INTEGER		NN				
Name	VARCHAR(63)		NN				
NoteTextLong	INTEGER		NN				
NoteUser	VARCHAR(63)		NN				
OffTime	TIMESTAMP		NN				
PagingEnabled	BIT		NN				
ParentGroup	INTEGER		NN				
Priority	SMALLINT		NN				
Rights	INTEGER		NN				
StoredDynamicMetadataFields	INTEGER		NN				
SuppressingAlarmId	INTEGER		NN				
SuppressingConstType	TINYINT		NN				
TypeDesc	VARCHAR(63)		NN				
TypeNum	SMALLINT		NN				

ColumnName	DataType	Primary Key	NotNull	Flags	DefaultValue	Comment	AutoInc
UserMethods	BIT		NN				
UsrAlmSeverityDesc	VARCHAR(31)		NN				
UsrAlmStat eDesc	VARCHAR(31)		NN				
UsrAlmUns pSeverityDe sc	VARCHAR(31)		NN				
UsrAlmUns pStateDesc	VARCHAR(31)		NN				
WSAlmSeverityDesc	VARCHAR(31)		NN				
WSAlmStat eDesc	VARCHAR(31)		NN				
WSAlmUns pSeverityDe sc	VARCHAR(31)		NN				
WSAlmUns pStateDesc	VARCHAR(31)		NN				

IndexName	IndexType	Columns
pkCiTimestampedAlarm	Index	Id
idxId	Index	Id
idxParentGroupId	Index	ParentGroupId
idxFullName	Index	FullName
idxConsParentAlarmObjId	Index	ConsParentAlarmObjId
idxCiTimestampedAlarmState_Id	Index	CiTimestampedAlarmState_Id

## See Also

[Alarm Server Side Synchronization](#)

## Trends

The trend system can help you better understand your plant and equipment's performance. Trending can be used for dynamic visual analysis (trend and SPC graphs), production records, or for regularly recording the status of equipment for efficiency and preventive maintenance.

Using trend tags, you can specify the data you want to collect from your I/O device variables. This information can be logged at regular intervals (periodic trend), or only when an event occurs (event trend). Event trends are used for trending data that is not time-based, for example, for a product as it comes off an assembly line. Trend data is usually saved on disk for analysis or displayed on a trend graph or in [Process Analyst](#).

The trend system is based on real-time samples. The trend system expects a return of one data point each time it samples the data. Although gaps in the data can be filled, you will want to verify that your field device can return data values at the rate you specify (especially if you are using sample periods of less than 100 ms).

Any amount of data can be collected and stored. The only restriction on the amount of data that you can store is the size of the hard disk on your computer (an efficient data storage method is used to optimize the use of storage space on your computer's hard disk).

By default, Plant SCADA stores trend files in the [DATA] directory (defined by the parameter [\[CtEdit\]Data](#)) on the hard disk where installation occurred. However, you can use the **File Name** property to specify a different directory for a particular trend file.

For example, if your system includes a large number of trend tags (in excess of 1000), Plant SCADA will perform more efficiently if you distribute the associated trend files across a number of directories. To enable this, you could set the **File Name** property for a trend tag to the following:

[DATA]:Area1\TANK131

This would store the trend file in a subdirectory (named "Area1") within the [DATA] directory.

For long term storage, you can archive the data to disk or a removable drive (without disrupting your runtime system).

---

**Note:** If you are trending data across a network, it is recommended that you enable time synchronization using the [Time Synchronization](#) configuration application.

---

This section covers the following information:

- [View Trends](#)
- [Configure Trends](#)
- [Debugging Trending](#)

## View Trends

Trend data is presented in trend graphs that visually represent past and current activity of plant-floor data, building a picture over time of how a variable (such as product output, level, temperature, and so on) changes or how a device or process is performing. You can monitor current activity as it happens and scroll back through time to view trend history.

---

**Note:** The Process Analyst (a built-in trend visualization tool) superseded the functionality of trend graphs. However, trend graphs are still supported. Be aware that SPC is not supported by the Process Analyst. To view SPC trends you need to use trend graphs. For details, see [Process Analyst](#)

---

## Trend Graphs

Trend graphs visually represent past and current activity of plant-floor data. As the values of variables change over time or as events happen, the graph moves across the page. The latest values are displayed by default. You can scroll back through historical data to display past values of the variable (or process).

You can trend any single variable or Cicode expression. You can display any number of trends on the screen simultaneously, even if they have different sample periods. You can also display up to eight trend tags (pens) in any trend window.

A trend graph can only communicate with one cluster, therefore you cannot mix trends from multiple clusters on a single trend graph. To graph trends from multiple clusters you will need to use multiple trend graphs, or, use the Process Analyst which has no such restrictions.

Historical data collection continues even when the display is not active. You can switch between pages without affecting trend graphs. Trend data acquisition and storage of data (in trend history files) continues even when the display is not active.

You can use the following standard trends:

- A single full page trend, where one trend window displays on a graphics page.
- A double full page trend, where two trend windows display on a graphics page.
- A zoom trend with two trend windows and added functionality for zooming.
- A pop-up trend that you can 'pop up' anywhere (in a separate window) on your computer screen.
- User-defined trends that you can position anywhere on any graphics page.

---

**Note:** Variable tags can also be visually trended using an SPC Control Chart. Statistical Process Control (SPC) is a facility that enables you to control the quality of materials, manufactured products, services, etc. This quality control is achieved by collecting, arranging, analyzing, and testing sampled data in a manner that detects lack of uniformity or quality.

---

## See Also

[Configure Trends](#)

[Create Pages for Trend Graphs](#)

## Configure Trends

Before you can view trends, the following configuration tasks need to be completed:

- [Add a Trend Tag](#)
- [Create Pages for Trend Graphs](#)
- [Define Trend Interpolation](#).

---

**Note:** If a variable tag is associated with a driver that supports the Driver Runtime Interface (see [Retrieving Time-stamped Data from I/O Devices](#)), you can define a trend tag as a pure-event trend. See [Configure Event Trends for a DRI Driver](#).

---

## Add a Trend Tag

Trend tags allow you to collect data from your I/O device variables for analysis and/or trend graphing.

### To add a trend tag:

1. In the **System Model** activity, select **Trends**.
2. Add a row to the Grid Editor.
3. Type the required information in each column, or in the fields in the Property Grid.  
For a description of the properties, see below.
4. Click **Save**.

## Trend Tag Properties

**Note:** If a trend tag was generated by the Equipment Editor, a number of fields on the properties form will be shaded. To configure these fields, you will need to use [Equipment Editor](#).

### Equipment Properties

Property	Description
<b>Equipment</b>	The name of the equipment associated with the trend tag. Select a name from the drop-down list of existing equipment definitions, or enter a name. There is a limit of 254 characters across the <b>Equipment</b> and <b>Item Name</b> fields, including any separating periods (.).
<b>Item Name</b>	The name of the item with which the trend tag is associated. Items form part of an equipment hierarchy. They can be used to associate tags, alarms and trends with a particular attribute of a physical piece of equipment (see <a href="#">Items</a> ). There is a limit of 254 characters across the <b>Equipment</b> and Item Name, including any separating periods (.). <b>Note:</b> When defining an item name, avoid using the <a href="#">Reserved Words</a> . If you use any of these, an error message will display when you compile your project.

### General Properties



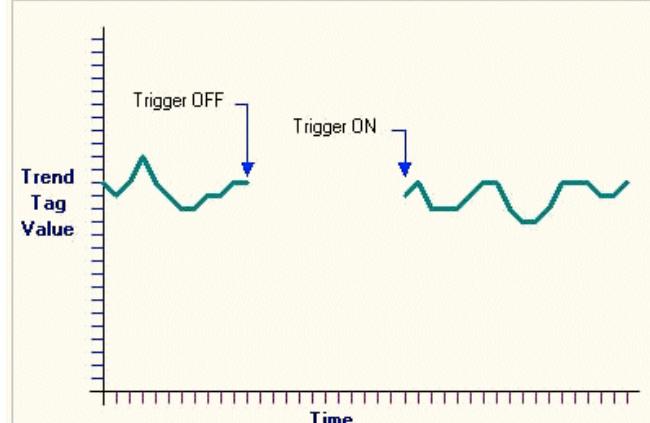
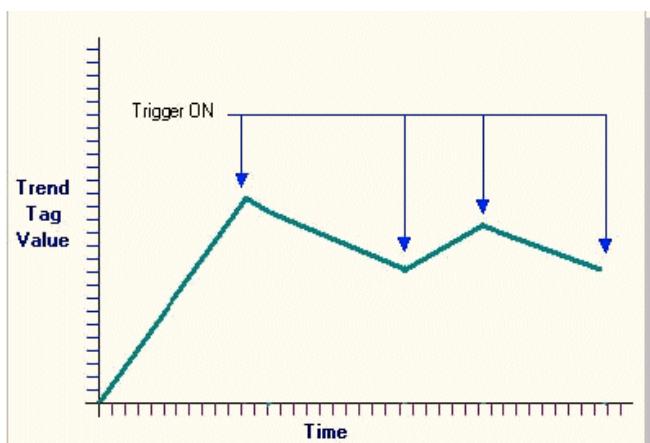
#### UNINTENDED EQUIPMENT OPERATION

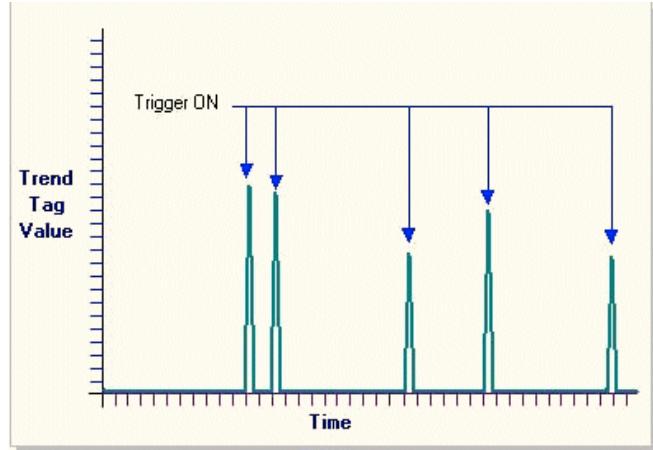
If using the **File Name** property, ensure that each trend tag uses a unique file name. Two tags accessing the

same file can result in system errors which may include lost or corrupted trend/SPC data.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Field	Description
<b>Tag Name</b>	<p>The name assigned to the trend data (79 characters maximum). If the trend tag is logging a particular variable, use a 16-character name that resembles the 32-character name of the related variable tag. This will mean an association between the two is easily recognizable. The name needs to be unique to the cluster. Trend Tag names need to adhere to the <a href="#">Tag Name Syntax</a>. If the name is not unique or is not syntactically correct it may not be recognized. If you have many tags, use a naming convention (see <a href="#">Structured Tag Names</a>). This makes it easier to find and debug your tags.</p> <p><b>Note:</b> Where Cluster Name is left blank, the name needs to be unique to every defined cluster.</p> <p><b>Note:</b> Trend tag names have to be unique and not identical to any SPC tag names within the cluster(s) that run this trend. Two tags accessing the same file can result in system errors which may include lost or corrupted trend/SPC data.</p>
<b>Cluster Name</b>	<p>The name of the cluster that runs the trend tag. This field needs to be defined if your project has more than one cluster.</p> <p>You can leave this field blank in a multi-cluster system if cluster replication is enabled (see the parameter <a href="#">[General]ClusterReplication</a>).</p>
<b>Type</b>	<p>The type of trend (32 characters maximum):</p> <p><b>TRN_PERIODIC</b> - A trend that is sampled continuously at a specified period. You can also define a trigger to stop and start the trend (when a specified condition occurs in the plant).</p>

Field	Description
	 <p>The graph illustrates a trend tag value plotted against time. Two specific points on the curve are highlighted with vertical arrows and labeled "Trigger OFF" and "Trigger ON". The curve shows a continuous trend between these discrete trigger events.</p>
	<p><b>TRN_EVENT</b> - A trend that is sampled once each time the value of the trigger changes from FALSE to TRUE. Interpolation occurs between each data point to create a continuous graph.</p>  <p>The graph illustrates a trend tag value plotted against time. Three distinct vertical spikes are shown, each labeled "Trigger ON". The trend line connects these spikes at specific intervals, indicating no interpolation between the data points.</p> <p><b>Note:</b> If a variable tag is associated with a driver that supports the Driver Runtime Interface, you can define a trend tag as a pure-event trend. See <a href="#">Configure Event Trends for a DRI Driver</a>.</p> <p><b>TRN_PERIODIC_EVENT</b> - A trend that is sampled once each time the value of the trigger changes from FALSE to TRUE. No interpolation occurs between these points, thus providing a graph which spikes at each data point.</p>

Field	Description
	
<b>Expression</b>	<p>The logged value of the trend tag (254 characters maximum). You can log individual variables by using a variable tag. For example:</p> <p>Expression: LT131</p> <p>Comment: Logs the Variable Tag LT131</p> <p>The value of the process variable LT131 is logged.</p> <p>You can also log any Cicode expression or function, for example:</p> <p>Expression: LT131/COUNTER</p> <p>Comment: Logs Variable Tag LT131 divided by the Variable Tag COUNTER</p> <p><b>Note:</b> When a variable tag is used in the expression field of a trend tag property, the <b>Eng Zero Scale</b> and <b>Eng Full Scale</b> fields of that variable tag needs to be set appropriately, or data may be lost because the trend logs negative values are invalid.</p> <p>If a variable tag is associated with a driver that supports the Driver Runtime Interface, you can enter a variable tag in this field. See <a href="#">Configure Event Trends for a DRI Driver</a>.</p>
<b>Trigger</b>	<p>The Cicode expression (or variable tag) that triggers data logging (254 characters maximum). For example:</p> <p>Trigger: LT131&lt;50</p> <p>In this example, logging occurs when the value of the variable tag (LT131) falls below 50.</p> <p>For a periodic trend, data is logged only while the value of the trigger is TRUE. (The trend graph will still scroll, but will display &lt;GATED&gt; where the trigger is FALSE.) In the above example, data is logged</p>

Field	Description
	<p>continuously while the value of LT131 remains less than 50. Logging ceases when the value rises to (or above) 50. Logging does not occur again until the value of LT131 falls below 50.</p> <p>You do not have to specify a trigger for a periodic trend. If you do not specify a trigger for a periodic trend, logging occurs continuously.</p> <p>For an event trend, data is logged once when the value of the trigger changes from FALSE to TRUE. In the above example, one sample is logged when the value of LT131 first becomes less than 50. Another sample is not logged until the value of LT131 rises to (or above 50) and again falls below 50.</p> <p><b>Note:</b> If a variable tag is associated with a driver that supports the Driver Runtime Interface, you should leave this field blank as the trend sample updates are controlled by the driver. See <a href="#">Configure Event Trends for a DRI Driver</a>.</p>
<b>Sample Period</b>	<p>The sampling period of the data. You can either enter a period of your own, or choose one from the menu.</p> <p>Enter sampling periods of greater than one second in hh:mm:ss (hours:minutes:seconds) format. If you enter a single digit, without the colon (:), it will be considered a second. For example, if you enter 2, it will be interpreted as 2 seconds.</p> <p>Sampling periods of less than one second needs to be entered as decimals. For example, to enter a period of 200 milliseconds, you would enter 0.2.</p> <p>If the sample period is less than one second, then one second needs to be divisible by the period (to give an integer). For example, a sample period of 0.05 is valid, because <math>1/0.05 = 20</math>, whereas a sample period of 0.3 is not valid because <math>1/0.3 = 3.333\dots</math>.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>Your I/O Device needs to be capable of providing data at the specified rate, otherwise gaps will appear in the trend data and/or the hardware alarm <b>Trend has missed samples</b> will be evoked. You can fill gaps in the file and graph using the <a href="#">[Trend]GapFillTime</a> parameter. Gaps in the graph only can be filled using the <a href="#">TrnSetDisplayMode()</a> function.</li> <li>If trends with a sample period of less than a second are shared by several clients across a network (distributed processing), enable time synchronization</li> </ul>

Field	Description
	<p>using the <a href="#">Time Synchronization</a> configuration application. This verifies that trends are synchronized with each other.</p> <p>The Trigger is checked each sample period. If the Trigger is TRUE (or has just changed from FALSE to TRUE, in the case of event trends), the value of the Expression is logged.</p> <p><b>Examples</b></p> <ul style="list-style-type: none"> <li>Sample Period: 30 - Logs data every 30 seconds</li> <li>Sample Period: 10:00 - Logs data every 10 minutes</li> <li>Sample Period: 10:00:00 - Logs data every 10 hours</li> <li>Sample Period: 2:30:00 - Logs data every 2 and a half hours</li> </ul> <p>The sampling period of the fastest trend on the page is taken as the default value for the display period of the page.</p> <p>This property is optional. If you do not specify a sample period, the sampling period defaults to 10 seconds.</p> <p><b>Note:</b> If you edit this property in an existing project, delete the associated trend files before running the new runtime system.</p>
<b>Eng Units</b>	The engineering units of the variable/expression being logged (8 characters maximum). The engineering units are used by the trend scales and trend cursor displays.
<b>Format</b>	<p>The format of the variable/expression being logged (11 characters maximum). The format is used by the trend scales and trend cursor displays.</p> <p>This property is optional. If you do not specify a format, the format defined for the associated variable will be used. If the associated variable does not have a format defined, it will default to #####.##.</p> <p><b>Note:</b> If multiple variable tags are defined in the trend expression and a trend format is not specified, the format will be indeterminate. Under these circumstances, it is recommended that you use this field to specify a trend format.</p>
<b>Deadband</b>	A deadband allows the value of a variable tag to fluctuate within a defined threshold without updates being sent through to the trend tag. This may be useful if a tag produces many small, insignificant value

Field	Description
	<p>changes. The threshold is represented as a percentage of the tag's engineering range. The default value is 0 (zero), which captures every value change.</p> <p>For example, if a tag has an engineering range of zero to 10000, a deadband of 1 would mean a change in value would have to be greater than 100 (or 1 percent of the range) to be recognized. If the current value was 5600, the tag in the PLC would have to change to a value greater than 5700 or less than 5500 before an update would be sent to the trend tag.</p> <p>If an associated variable tag has a deadband setting, it will not automatically apply to a related trend tag. The trend tag deadband should be configured independently.</p> <p><b>Note:</b> If a variable tag is associated with a driver that supports the Driver Runtime Interface, you can define a trend tag as a pure-event trend (see <a href="#">Configure Event Trends for a DRI Driver</a>). Under this configuration, the trend sample updates are controlled by the driver and the <b>Deadband</b> setting will be based on what is defined in the variable tag instead of the trend tag.</p>
Comment	Any useful comment (254 characters maximum).

### Storage Properties



#### UNINTENDED EQUIPMENT OPERATION

If using the **File Name** property, ensure that each trend tag uses a unique file name. Two tags accessing the same file can result in system errors which may include lost or corrupted trend/SPC data.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Field	Description
<b>File Name</b>	<p>The file where the data is to be stored (253 characters maximum). Specify the complete path or use a path substitution (see <a href="#">Use a Path Substitution</a>).</p> <p>When data is collected from your plant floor, it is stored in a file on the hard disk of your computer and used to display a trend or SPC graph (a separate file is used for each trend tag).</p> <p>The File Name property is optional. If you leave this field blank, the trend file is stored in the [DATA] directory (defined by the parameter <a href="#">[CtEdit]Data</a>) on</p>

Field	Description
	<p>the hard disk where installation occurred. The default name of the file is the trend tag name.</p> <p>If required, you can use this field to specify an alternate file name like the following:</p> <p>File Name: [DATA] : TANK131</p> <p>In this case, [DATA] specifies the disk and path for the data (as defined by [CtEdit]Data) and "TANK131" is the name specified for the trend file.</p> <p>To specify an alternate directory for the trend file, use a file name like the following:</p> <p>File Name: [DATA] : Area1\TANK311</p> <p>This would store the trend file in a subdirectory (named "Area1") within the [DATA] directory.</p> <p>Be aware that you will need to manually configure the access control list for a new folder location so that it matches the permissions that were applied to the default location during installation. See <a href="#">Configure Directory Security for Modified Folder Locations</a>.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"><li>• If you use the File Name property, confirm that no other SPC tags or trend tags use the same file name. A compile error message (E2172) will be raised if duplicate file names are detected.</li><li>• Duplicated file names can occur if your path substitutions are not configured correctly, for example, if [TREND1] and [TREND2] incorrectly point to the same location. If this occurs, an error will be logged to the syslog.dat file, and a hardware error will be raised.</li><li>• If a path with invalid characters (such as '*' or '?') is detected at runtime, an error will be logged to the syslog.dat file, and a hardware error will be raised.</li><li>• The trend system will buffer the acquired data before saving it to a file. The [Trend]CacheSize parameter determines the buffer sizes for returned data.</li><li>• A file extension should not be used when specifying a file name. If you edit this property (change the file name or path) in an existing project, existing trend data is ignored.</li></ul>

Field	Description
	<ul style="list-style-type: none"> <li>You can no longer store trend files in the 'Bin', 'Runtime', 'Backup' or 'User' directories, or any subdirectories of these. If this is attempted, an error will be logged to the syslog.dat file, and a hardware error will be raised. If you have existing Version 3.xx or 4.xx projects that use these directories to store trend files, the path for these will have to be changed to the 'Data' directory.</li> </ul>
<b>No. Files</b>	<p>The number of history files stored on your hard disk (for this tag) (4 characters maximum). The maximum number of files you can specify per trend tag is 999. Performance and storage will be severely impacted by having a large number of history files per trend.</p> <p>By default, 2 history files are stored on your hard disk.</p>
<b>Period</b>	<p>The period of the history file, in hh:mm:ss (32 characters maximum). Alternatively, you can:</p> <ul style="list-style-type: none"> <li>Specify a weekly period by entering the day of the week on which to start the history file, for example Monday, Tuesday, Wednesday, etc.</li> <li>Specify a monthly period by entering the day of the month on which to start the history file, for example 1st, 2nd, 3rd, 4th, 5th, etc.</li> <li>Specify a yearly period by entering the day and the month on which to start the history file, for example 1st January, 25th February, etc. The day and month needs to be separated by a space.</li> </ul> <p>If you do not specify a period, the period defaults to Sunday (weekly).</p> <p>When deciding on a period setting, be aware that the performance of a trend viewer (be it the existing Plant SCADA or Process Analyst) may be impacted by the size of a trend file. This is particularly true when displaying event-based trend data.</p> <p><b>Note:</b> If you edit this property in an existing project, delete or move the associated trend files before you run the new runtime system. (For location of the trend files, see the <b>File Name</b> property.)</p>
<b>Time</b>	<p>The time of day to synchronize the beginning of the</p>

Field	Description
	<p>history file, in hh:mm:ss (32 characters maximum). If you do not specify a time, the file is synchronized at 0:00:00 (i.e. midnight). The time needs to be specified in Greenwich Mean Time, not the local time zone.</p> <p><b>Note:</b> If you edit this property in an existing project, you will need to delete the associated trend files before you run the new runtime system. Otherwise, the trend history files will continue to roll over at the previously specified time.</p> <p>For the location of the trend files, see the <b>File Name</b> property.</p>
<b>Storage Method</b>	<p>Select <b>Scaled</b> or <b>Floating Point</b> (64 characters).</p> <p><b>Scaled</b> is a 2-byte data storage method; floating point uses 8 bytes.</p> <p>Trend records need to explicitly define their trend storage method. The compiler will raise an error if not defined.</p> <p><b>Floating point</b> storage has an expanded data range in comparison to scaled storage, allowing values to have far greater resolution. However, you need to consider that it also uses a lot more disk space.</p> <p>If you use scaled trends, then a loss of precision may occur when reading trend data via CtAPI applications. This occurs when the engineering range does not evenly divide into 32000. In this case, it is recommended that you use floating point (8-byte) trends.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>• Trend records need to explicitly define their trend storage method. The compiler will raise an error if not defined. In earlier versions, the default when not defined was "Scaled (2 byte)". When upgrading, customers should set the trend storage method for blank entries to match the default from the previous version.</li> <li>• If you edit this property in an existing project, you need to delete the associated trend files before you run the new runtime system. (For the location of the trend files, see the <b>File Name</b> property.)</li> </ul>

## Security Properties

Property	Description
Area	The area to which the trend data belongs.
Privilege	The privilege necessary by an operator to display the trend data on a trend.

**Historian Properties**

Property	Description
Historize	This field enables you to automatically historize and publish the specified trend tag in CitectHistorian. If you set this field to "TRUE", the variable will be included in an automated configuration process within the Historian environment. If you set the field to "FALSE" (or leave it blank), the variable will not be included.  See <a href="#">Integration with Historian</a> .

**Project Properties**

Property	Description
Project	The project in which the trend tag is configured.

**See Also**

[Create Pages for Trend Graphs](#)

**Configure Event Trends for a DRI Driver**

If a variable tag is associated with a driver that supports the Driver Runtime Interface (see [Retrieving Time-stamped Data from I/O Devices](#)), you can define a trend tag as pure-event trend.

This means each sample value and timestamp is automatically populated to the trend tag by the driver whenever the I/O value changes.

**To define a pure-event trend:**

1. Follow the procedure described in [Add a Trend Tag](#).
2. Set the trend **Type** to "TRN\_EVENT".
3. Set the **Expression** to a variable tag from an I/O device that uses a DRI driver.
4. Set the **Trigger** to blank.

Under this configuration, trend sample updates are controlled by the driver. The **Deadband** setting will be based on what is defined in the variable tag instead of the trend tag.

## See Also

[Trend Interpolation](#)

[Create Pages for Trend Graphs](#)

### Create Pages for Trend Graphs

You can use any of the predefined trend templates for your trend pages, or use a predefined template to produce your own trend templates. You can draw a trend background (such as grid lines) on your trends.

#### To configure a trend page:

1. Open Graphics Builder.
2. Click the **New Page** button on the Command Bar, or choose **File | New**.
3. On the New dialog box, select **Type: Page**.
4. Choose the **Resolution** (size) of the trend page.
5. Choose a trend **Template** for the trend page:
  - **Singletrend** - One trend on the page
  - **Doubletrend** - Two trends on the page
  - **Eventtrend** - One event trend on the page
  - **Zoomtrend** - Two trends on the page (one window for zooming)
  - **Poptrend** - A single trend on the page (for display in a pop-up window)
6. Click **OK**.

To create multiple trend pages, you can:

- Create a trend page for each set of trends to display in the runtime system.
- Create a single trend page and use the [PageTrend\(\)](#) function to display trends as necessary. With this function, you can display the trends in the system with a single trend page
- Create the trend page with the Graphics Builder, and set the pen names to blank. You then display that page by calling this function and passing the necessary trend tags (up to 8). Call the PageTrend() function from a menu of trend pages.

## See Also

[Trend Interpolation](#)

### Trend Interpolation

Trend interpolation is used to define the appearance of a trend graph when the incoming samples fall out of synchronization with the display period or when samples are missed.

For example, a particular trend might be sampled five times between each update of the trend graph. As only one value can be displayed for each update, a single value needs to be used that appropriately represents the five samples; and that could be the highest value, the lowest value, or an average.

To define how Plant SCADA calculates the value to use, you set a particular trend interpolator display method.

The following table shows the available interpolator display methods, grouped into *condense methods* (where the display period is longer than the sample period) and *stretch methods* (where the display period is less than or equal to the sample period).

Condense methods	Stretch methods
<b>Average</b> ( <i>default</i> ) - this displays the average of the samples within the previous display period	<b>Step</b> ( <i>default</i> ) - This method simply displays the value of the most recent sample.
<b>Minimum</b> - This displays the lowest value that occurred during the previous display period.	<b>Ratio</b> - This method uses the ratio of sample times and values immediately before and after the requested time to interpolate a "straight line" value.
<b>Maximum</b> - This displays the highest value that occurred during the previous display period.	<b>Raw Data</b> - This method displays the actual raw values.

The interpolation display method is set via `TrnSetDisplayMode()` function. You can also use the `[Trend]GapFillMode` parameter, but it will interpolate values within the actual trend file as well as on the trend graph.

## Print Trend Data

You can print trend data using the following functions:

Function	Purpose
<code>TrnPrint</code>	Prints a trend that is displayed on the screen.
<code>TrnPlot</code>	Prints a plot of one or more trend tags.
<code>TrnComparePlot</code>	Prints two trends (one overlaid on the other), each of up to four trend tags.
<code>WinPrint</code>	Prints the active window

The standard trend templates have buttons that call these functions to print data.

When you print using the `TrnPrint` function, the Plot Setup dialog box appears. Use this dialog box to:

- Specify the title of the trend.
- Add a comment which is displayed beneath the title.
- Specify whether the trend is going to print in black and white, or in color. The selection that you make here will become the setting for the `[General]PrinterColourMode` parameter.
- Define your printer setup. The printer that you select here will be set as the default printer at the `[General]TrnPrinter` parameter.
- Specify whether or not the form displays the next time the function is used. This check box sets the `[General]DisablePlotSetupForm` parameter.

## See Also

[Export Trend Data](#)

## Export Trend Data

You can export trend data to reports and databases with the following functions:

Function	Purpose
TrnGetTable	Retrieves trend information and stores it in a Cicode array
TrnExportClip	Copies trend data to the clipboard
TrnExportCSV	Copies trend data to a CSV file
TrnExportDBF	Copies trend data to a DBF file

The standard trend templates have buttons that call these functions to export data.

---

**Note:** You can also select part of your trend graph (click and drag) and copy the underlying values to the Windows clipboard. You can then paste them into an Excel spreadsheet. (If you are pasting millisecond values, you will need to create a custom format for the TIME column to display these values correctly. To do this, select the column and select **Format | Cells**. In the **Number** tab, select **Custom** for Category, and type **h:mm:ss.000 AM/PM.**)

---

## See Also

[Use Trend History Files](#)

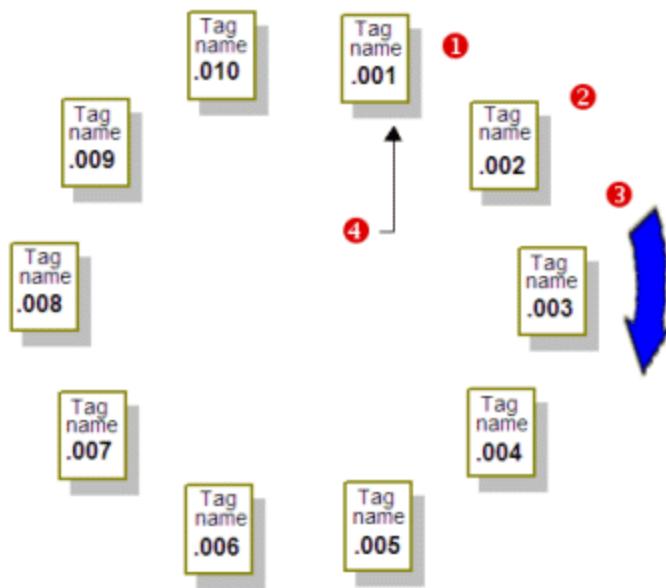
## Use Trend History Files

When Plant SCADA starts up for the first time, it creates the trend files necessary for each trend tag in the runtime system. You can change this default using the [\[Trend\]AllFiles](#) parameter.

A system of rotational history files is used to store the trend data. Data is stored in several files rather than in a single large file.

By default, there are 2 files for each trend tag. If required, you can specify the number of files to use via a trend tag's **No. Files** property (see [Add a Trend Tag](#)). For example:

No. Files	Comment
10	Use ten files for the data, as in the diagram below.



1. When Plant SCADA begins logging, data is written to the first file.
2. At midnight the following Sunday, Plant SCADA writes to the second file.
3. At midnight the following Sunday, Plant SCADA writes to the third file, and so on.
4. After week 10, the first file is overwritten with new data.

The maximum number of files you can specify per trend tag is 999.

You can also specify the period between files, i.e., when a new history file is used, for example:

Period	Comment
1:00:00	Use a new file each hour
6:00:00	Use a new file every six hours
72:00:00	Use a new file every three days
Monday	Use a new file each week beginning on Monday
15th	Use a new file every month beginning on the 15th of each month
25th June	Use a new file every year beginning on the 25th of June

You can also specify the time of day to synchronize the start of the history file; for example:

Time	Comment
6:00:00	Synchronize the file at 6:00 am
12:00:00	Synchronize the file at 12:00 midday

Time	Comment
18:30:00	Synchronize the file at 6:30 pm

## See Also

[Storage Method](#)

### Storage Method

You can select the storage method to use for trend tags and SPC tags. You are given a choice of either **Scaled** or **Floating Point**. Scaled represents a 2-byte data storage method; floating point uses 8 bytes.

Floating point storage has a dramatically expanded data range in comparison to scaled storage, allowing values to be more precise, but it also uses more disk space. Use scaled where compatibility with pre-V5.31 trend history files is necessary.

You can set the necessary storage method via the Trend Tag or SPC Tag properties. The storage method is set to Scaled by default.

#### Note:

- Trend records need to explicitly define their trend storage method. The compiler will raise an error if not defined. In previous versions, the default when not defined was "Scaled (2 byte)"8 byte. When upgrading, customers should set the trend storage method for blank entries to match the default from the previous version.
- If you edit this property in an existing project, you need to delete the associated trend files before you run the new runtime system. (For location of the trend files, see the **File Name** property for the trend tag.)

## See Also

[Calculating Disk Storage](#)

### Calculating Disk Storage

Equations allow you to calculate the total disk space necessary to store a trend across a specified period of time.

The storage method used for a trend (**Scaled** or **Floating Point**) affects the number of bytes necessary for each sample, so in order to calculate the disk space necessary correctly it is important to base your calculations on the appropriate formula. By default, the Scaled storage method is used.

### Floating point

Each data sample requires eight bytes of storage. This alters the equation to:

$$\text{Bytes necessary for each trend} = 704 \times \text{No. Files} + 160 + \left( \frac{\text{File Period (secs)} \times (\text{No. Files})}{\text{Sample Period (secs)}} \times 8 \right)$$

The number of bytes necessary then becomes:

$$\text{Bytes necessary} = 704 \times 5 + 160 + \left( \frac{(7 \times 24 \times 60 \times 60) \times 5 \times 8}{10} \right)$$

Bytes necessary = 2,422,976 bytes

## Scaled

Each data sample requires two bytes of storage. You can therefore calculate the total disk storage necessary for each trend by using the following formula:

$$\text{Bytes necessary for each trend} = 464 \times \text{No. Files} + 176 + \left( \frac{\text{File Period (secs)} \times (\text{No. Files}) \times 2}{\text{Sample Period (secs)}} \right)$$

For example, if a trend record produces one sample every ten seconds for one week, and you are using five data files (five weeks), the number of bytes necessary is:

$$\text{Bytes necessary} = 464 \times 5 + 176 + \left( \frac{(7 \times 24 \times 60 \times 60) \times 5 \times 2}{10} \right)$$

Bytes necessary = 607,296 bytes

---

**Note:** The calculations above do not take into account the space necessary to store the history file for each trend. This is because these files remain at a set size and therefore do not significantly impact the amount of disk space necessary.

---

**Note:** For efficient trends storage, use Windows file compression. By using this method you often can reduce your files to 10% of their original size; the actual amount of compression varies depending on the rate of change of the data.

---

## See Also

[Reconfiguring History Files](#)

### Reconfiguring History Files

If you change the configuration of your trend history files (in an existing project), or you change the configuration of a trend tag that affects the number, time, or period of the trend files, you need to delete the existing trend files - before you run the new system.

If you change the path of your trend history files (in an existing project), existing trend data is ignored.

---

**Note:** You need to not delete history files (that Plant SCADA creates) from your hard disk while your system is running, as the trend server will attempt to recreate the files and this may cause system performance issues.

---

## Use a Path Substitution

Instead of specifying the full path to data files in your system, you can use a path substitution.

With path substitution, you use a name that is a substitution for a full directory path. You can then use the substitution name in the following format:

File Name	[SUBSTITUTION]:<filename>
-----------	---------------------------

For example, if you decide to store a trend data file in the default data directory, you could specify the full path to the file:

File Name	%PROGRAMDATA%\AVEVA Plant SCADA 2023\Data\MYFILE
-----------	--

Or you could use a default path substitution and specify the following:

File Name	[DATA]:MYFILE
-----------	---------------

Path substitution provides greater control of data storage. You can change the location of data files by changing the definition of the data path - instead of locating and changing each occurrence of the data path.

A set of default path names are listed below. You can also define your own path substitutions using the parameter [\[Path\]<PathName>](#).

## Default Path Substitutions

Plant SCADA has the following pre-defined path substitutions:

Path Name	Default Directory
[Bin]	%PROGRAMFILES(x86)%\AVEVA Plant SCADA\Bin
[User]	%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\User
[Data]	%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Data
[Run]	The current project directory.
[Copy]	The current copy project directory.
[Back]	The current backup project directory.

**Note:** You will need to manually configure the access control list for a folder specified in a path substitution so that it matches the permissions that were applied to the default location during installation. See [Configure Directory Security for Modified Folder Locations](#).

## See Also

[Add a Trend Tag](#)

## Debugging Trending

The [\[Trend\]TrendDebug](#) Citect.ini parameter is provided to help you while logging and trending data.

### Example:

```
[Trend]  
TrendDebug=n
```

Where *n* can be the combination of the following debug options:

- 1 - Logs the client, server, and redundancy message types and also the samples being written in the Trends Server from normal acquisition.
- 2 - Logs detailed information about the currently active backfill process, including the redundant samples written to the archive.
- 4 - Logs detailed information for the TrendSetTable functions.
- 7 - Logs trend activities.
- 8 - Logs the summary information only for the currently active backfill process.
- 16 - Logs to syslog.dat the client trend data.

These settings can be added together to have combinations of logging levels. For example:

```
[Trend]  
TrendDebug=6
```

logs the detailed backfill process and TrendSetTable functions.

These settings are read dynamically, meaning that you can change these settings while Plant SCADA is running and the changes will take effect from that point onwards.

## See Also

[Trends - Frequently Asked Questions](#)

## Trends - Frequently Asked Questions

### Q: How do I get trend data into a dBASE database?

A: Display the trend on screen and select the File Save/As tool. This tool displays a Save/As dialog box for you to enter the name of the file, and calls the TrnExportDBF function to save the data. (For more complex procedures, call this function directly.)

### Q: How do I get trend data into Excel?

A: You can get data into Excel in three ways:

- Display the trend on the screen, select the Clipboard tool to copy the data, and use the Excel paste command to paste the data into Excel.
- Display the trend on the screen and select the File Save/As tool. Save the data in CSV format, and open the CSV file in Excel.
- Display the trend on the screen and select the File Save/As tool. Save the data in DBF format and open the DBF file in Excel.

(For other procedures, call the TrnExportCSV or TrnExportDBF function.)

**Q: How do I get trend data into MS Access?**

**A:** You can get data into Access in three ways:

- Display the trend on the screen, select the Clipboard tool to copy the data, and use the Access Paste command to paste the data into Access.
- Display the trend on the screen and select the File Save/As tool. Save the data in CSV format, and open the CSV file in Access.
- Display the trend on the screen and select the File Save/As tool. Save the data in DBF format and open the DBF file in Access.

(For other procedures, call the TrnExportCSV or TrnExportDBF function.)

**Q: How do I update trend data?**

**A:** Use the TrnSetTable function to write data back to the trend system.

**Q: How do I get trend data into a report?**

**A:** Use the TrnGetTable function.

## Accumulators

Accumulators track incremental runtime data, such as motor run hours, power consumption, and downtime. You set a trigger (for example, motor on) to increment three counters:

- The number of times the accumulator is triggered (for example the number of starts for the motor).
- The run time, in steps of 1 second.
- A totalizer value, by an increment you define.

The accumulated data is stored as variable tags in an I/O device. This allows easy access to these variables and, because they are saved in the I/O device, their values are retained if the project is shutdown.

Variable tags are read at startup and updated regularly while the trigger is active. You can monitor and display accumulated data by animating, trending, or logging the variable tags.

You can increase system performance by storing these variables in a disk I/O device. If you store these variables in an external I/O devices, communications bandwidth will be consumed updating the variables. If using a network, you can use a redundant disk I/O device to secure these variables.

---

**Note:** The accumulator server runs as a part of the report server. You need to add a report server to your project to make accumulators work. See [Add a Report Server Process](#).

---

If you have a redundant reports server, you need to use a primary/standby configuration to stop the accumulators running on both reports servers. Use the Setup Wizard to define the reports servers.

You can control (re-read or reset) any accumulator at runtime by using the [AccControl](#) Cicode function.

## See Also

[Add an Accumulator](#)

## Add an Accumulator

[Accumulators](#) track incremental runtime data.

### To add an accumulator:

1. In the **System Model** activity, select **Accumulators**.
2. Add a row to the Grid Editor.
3. Type the required information in each column, or in the fields in the Property Grid.  
For a description of the properties, see below.
4. Click **Save**.

## Accumulators Properties

### Equipment Properties

Field	Description
<b>Equipment</b>	The name of the equipment associated with the accumulator. Select a name from the drop-down list of existing equipment definitions, or enter a name. There is a limit of 254 characters across the <b>Equipment</b> and <b>Item Name</b> fields, including any separating periods (.).
<b>Item Name</b>	The name of the item with which the accumulator is associated. Items form part of an equipment hierarchy. They can be used to associate accumulators, tags, alarms and trends with a particular attribute of a physical piece of equipment (see <a href="#">Items</a> ). There is a limit of 254 characters across the <b>Equipment</b> and Item Name, including any separating periods (.). <b>Note:</b> When defining an item name, avoid using the <a href="#">Reserved Words</a> . If you use any of these, an error message will display when you compile your project.

### General Properties

Field	Description
<b>Name</b>	The name of the accumulator. Enter a value of 79 characters or less.
<b>Cluster Name</b>	The name of the cluster that runs the accumulator. This field needs to be defined if your project has more than one cluster.

Field	Description
	You can leave this field blank in a multi-cluster system if cluster replication is enabled (see the parameter <a href="#">[General]ClusterReplication</a> ).
<b>Trigger</b>	<p>The Cicode expression (or variable tag) to trigger the accumulator. If the result of the expression (in this field) is TRUE, accumulation starts. If the result of the expression (in this field) becomes FALSE, accumulation stops. Enter a value of 254 characters or less.</p> <p>The frequency with which the trigger is checked is controlled by the <a href="#">[Accumulator]WatchTime</a> parameter.</p>
<b>Totalizer Inc</b>	Any Cicode expression (or variable tag) to add to (increment) the <b>Totalizer</b> variable while the <b>Trigger</b> condition is TRUE. Enter a value of 254 characters or less.
<b>Totalizer</b>	<p>The variable (tag) that contains the totalized value. Enter a value of 254 characters or less.</p> <p>On startup, this value is read. Each time the trigger is checked and while the trigger is TRUE, the value in the <b>Totalizer Inc</b> field is added to the local copy of this <b>Totalizer</b> variable. The new <b>Totalizer</b> variable is written back to the I/O Device at a frequency determined by the <a href="#">[Accumulator]UpdateTime</a> parameter.</p> <p>For example, if you configure an accumulator for a motor, and <b>Totalizer Inc</b> is the current (amperage) used by the motor, then <b>Run Time</b> will contain the time (in seconds) that the motor has run, <b>No. of Starts</b> will contain the number of times the motor was started, and <b>Totalizer</b> will contain the total current used by the motor. The average current used by the motor is <b>Totalizer/Run Time</b>.</p>
<b>Run Time</b>	The variable (tag) that contains the run time (in seconds). Enter a value of 254 characters or less. On startup, this value is read. Then the local copy of this variable is incremented (while <b>Trigger</b> is TRUE) and the variable is written back to the I/O Device at a frequency determined by the <a href="#">[Accumulator]UpdateTime</a> parameter.
<b>No. of Starts</b>	The variable (tag) that contains the number of starts (that is, the number of times the trigger changes from FALSE to TRUE). Enter a value of 254 characters or less. On startup, this value is read. Then the local copy

Field	Description
	of this variable is incremented and the variable is written back to the I/O Device at a frequency determined by the [Accumulator]UpdateTime parameter.
<b>Comment</b>	Any useful comment. Enter a value of 48 characters or less.

**Security Properties**

Field	Description
<b>Area</b>	The area to which the accumulator belongs. Only users with access to this area (and any required privileges) will be able to perform operations on the accumulator. For example, if you enter Area 1 here, operators need to have access to Area 1 (plus any required privileges) to perform operations on the accumulator. Enter a value of 16 characters or less.
<b>Privilege</b>	The privilege needed by an operator to perform operations on the accumulator (by using accumulator functions). Enter a value of 16 characters or less.

**Project Properties**

Property	Description
<b>Project</b>	The project in which the accumulator is configured.

**See Also**

[Accumulators](#)

## Statistical Process Control (SPC)

Statistical Process Control (SPC) is the primary tool of statistical quality control and encompasses the collection, arrangement, and interpretation of process variables associated with a product, in regard to uniformity of quality.

**About Statistical Process Control**

Statistical quality control (SQC) helps track and improve product or service quality. Statistical process control (SPC) is the primary tool of SQC and encompasses the collection, arrangement, and interpretation of process variables associated with a product, in regard to uniformity of quality.

Every process is subject to variation and therefore there is always room to improve process consistency and quality. To respond to this, adopt a continuous improvement strategy. By repeating the following cycle, a strategy of prevention can be implemented. This is the key to using SPC effectively. Consider the following steps:

- Analyze the process
- What do we know about the variability of this process?
- Is this process statistically controllable (predictable)?
- Is the process capable of meeting the set requirements?
- What considerations are most important to meeting process quality criteria?
- Maintain (control) the process
- Improve the process

SPC encompasses the following concepts:

- Process Variation
- Statistical Control
- Process Capability

It typically uses the tools [XRS Control Charts](#), [Configure a Capability Chart](#), and [Pareto Charts](#).

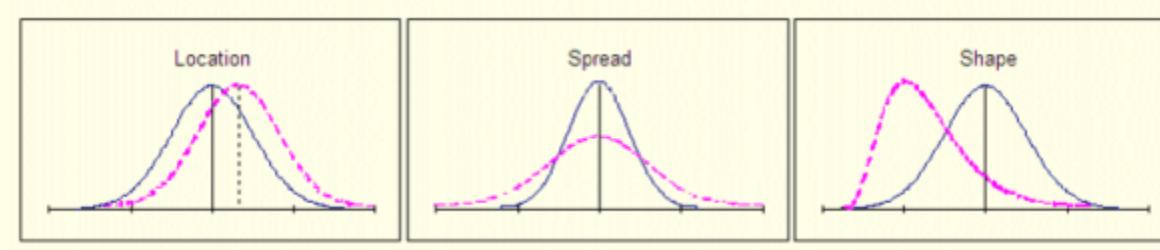
## See Also

[Using Statistical Process Control \(SPC\)](#)

## Process Variation

To use SPC effectively, understand the concept of *variation*. When a product characteristic is measured repeatedly, each measurement is likely to differ from the last. This is because the process contains sources of variability.

When the data is grouped into a frequency histogram, it will tend to form a pattern. The pattern is referred to as a probability distribution and is characterized in three ways:



---

**Note:** Most SPC techniques assume that the collected data has a normal distribution.

Variation is generally categorized into one of two types:

- **Common:** refers to variation that is predictable and repeatable over time. The distribution characteristics will be stable. Common variation could be due to consistent process inaccuracy or similar.  
Statistics indicate that common variations account for about 85% of departures from process quality requirements. Usually these departures require solution at the management level.
- **Special:** refers to variation that is not consistently present. When special variation occurs it will tend to change the distribution characteristics. The distribution is not stable over time.

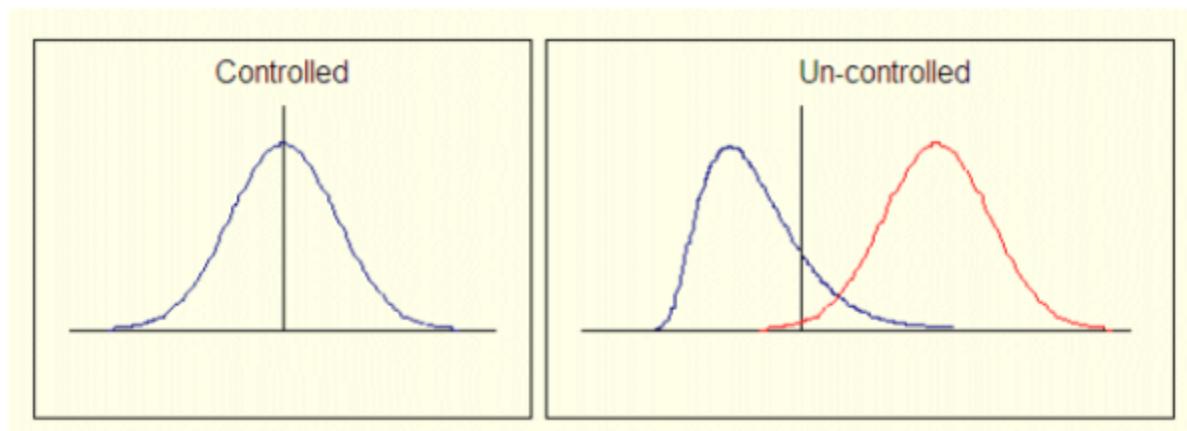
Statistics indicate that special variations account for about 15% of departures from process quality requirements. Typically these departures require local action (equipment repair and so on) for solution.

## See Also

[Process Capability](#)

## Statistical Control

A process is said to be in statistical control when the only sources of variation are from common causes. A statistically controllable process is desirable because it is predictable, while a statistically uncontrollable process will yield unpredictable distributions.



Even though a process might not be statistically controllable, it might still meet requirements. Conversely, a process which is controllable might not meet requirements. This issue is clarified by considering Process Capability.

## See Also

[Process Variation](#)

## Process Capability

A process is said to be capable when the percentage of samples outside the specification limits is less than a predefined value. The following assumptions need to also be true:

- The process is statistically stable (only common causes of variation exist)
- The individual measurements conform to a normal distribution
- Measurement variation (due to the measuring instrument) is small

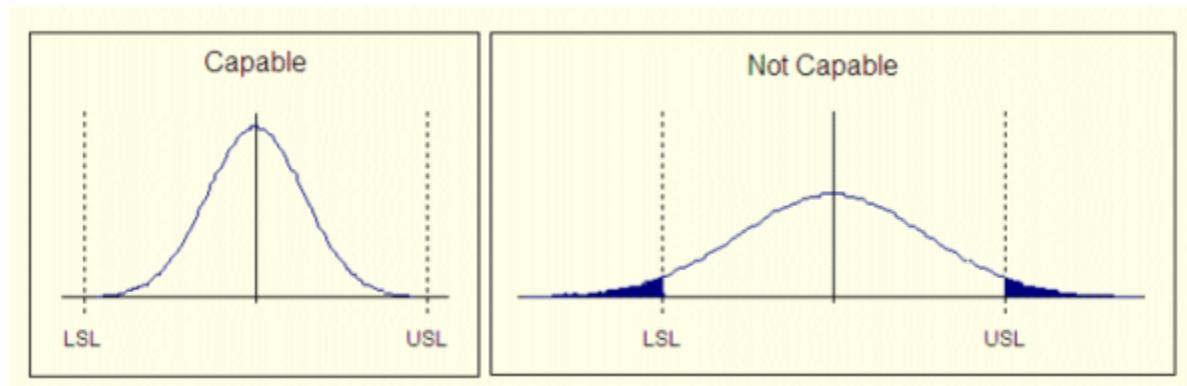
The specification limits are a reflection of the customers requirements and are selectable. The percentage of samples that need to lie within the specification limits is calculated from the standard deviation ("sigma") "3-sigmas" on either side of the mean.

**Note:** The "3-sigma" term refers to the boundaries which are located 3 standard deviations on either side of the

---

center. For a normal distribution 99.74% of the samples are expected to fall within this boundary.

Ultimately capability determines whether the process is statistically able to meet the specification or not. Reducing the effects of common variation will make your process more capable, as shown here:

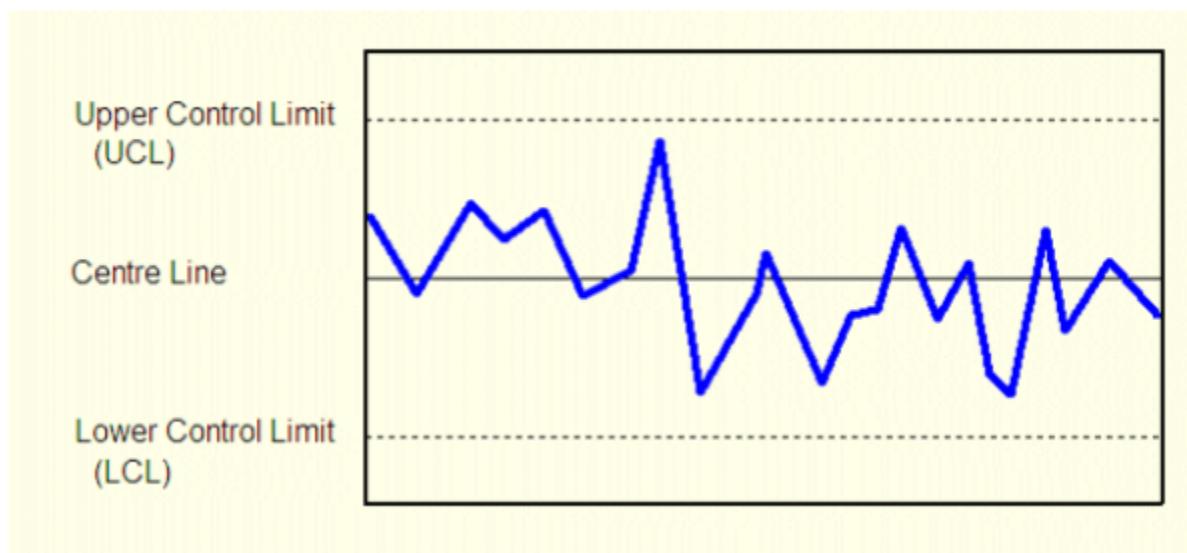


## See Also

[XRS Control Charts](#)

## XRS Control Charts

XRS charts are the primary tool of SPC and convey information about variation and control-ability. The charts are trend graphs that individually show mean ( $\bar{X}$  bar), range ( $R$ ), and standard deviation ( $s$  or  $\sigma$ ). Each point on the graph represents sub-grouped data, not individual samples.



Refer to the following for more information:

## CL, UCL, LCL

Each graph has center, upper control limit (UCL), and lower control limit (LCL) lines drawn. The control limits are estimates of where the "3-sigma" limits would lie for an approximately normal distribution. In practice the limits

are calculated from the measured X double bar, R bar, and s bar, and table values which are based on the size of the subgroup used.

These lines are used as a guide for analysis as they are based on the natural variability of the process. These limits are not specification limits.

## Interpreting the Chart

Control charts can provide valuable information about process variation. By watching for particular patterns and events, conclusions can be drawn as to how the process is controlling.

A (normal) process that is statistically under control will tend to distribute data according to a normal distribution. The data will appear randomly, but centered around the center line. Data that appears near the control limits may be infrequent but expected. Presence of data that appears outside of the control limits indicates the process is statistically uncontrolled and action should be taken to address the cause. Similarly, runs of constant data or patterns indicate instability.

The following list of patterns and events are considered to be cause for alarm:

- Points beyond the control limits
- Several consecutive points on only one side of the average
- Several consecutive points that are consistently increasing or decreasing
- Substantially more than 2/3 of plotted points lie close to the average
- Substantially less than 2/3 of plotted points lie close to the average

The presence of these types of patterns and events has different meanings, depending on which type of chart is being analyzed. By using each of the three control charts together (X bar, R and s) together, a better understanding of readings is gained.

## See Also

[Configure an XRS Control Chart](#)

## Capability Charts

The process capability chart is the frequency histogram and distribution of the process mean data and is used to analyze process capability. This chart is drawn with center line, lower specification limit and upper specification limit indicators in place. Interpreting capability charts is typically made with the Cp, Cpk, kurtosis and skewness indices.

Refer to the following for more information:

## Upper Specification Limit and Lower Specification Limit

Upper specification limit (USL) and lower specification limit (LSL) specify the requirements of the product, and so are customer driven. Make the target value the midpoint between USL and LSL.

## Cp Index

The inherent process capability ( $C_p$ ) is the ratio of tolerance to 6-sigma. Essentially this index indicates whether the distribution would fit inside the USL and LSL limits. Its meaning is defined as follows:

- $C_p > 1.0$  - Indicates the process variation will fit within the specified limits (USL and LSL) and therefore, is capable.
- $C_p < 1.0$  - Indicates the process is not capable.

## Cpk Index

The process capability based on the worst case ( $C_{pk}$ ) is similar  $C_p$ . This index, however, indicates where the mean lies in relation to the USL and LSL limits. It is used to mathematically clarify  $C_p$ . Its meaning is defined as follows:

- $C_{pk} < 0$  - Indicates the process mean is outside the specified limits (USL and LSL)
- $C_{pk} = 0$  - Indicates the process mean is equal to one of the specified limits.
- $C_{pk} > 0$  - Indicates the process mean is within the specified limits.
- $C_{pk} = 1.0$  - Indicates that one side of the 6-sigma limits falls on a specification limit.
- $C_{pk} > 1.0$  - Indicates that the 6-sigma limits fall completely within the specified limits.

## Pareto Charts

A Pareto chart is a tool which is used to analyze the relative frequency of deviations from specifications or success criteria. The Pareto chart is a frequency histogram which is ordered from highest to lowest. Unlike control and capability charts, this chart uses no statistical guides.

Pareto charts emphasize Pareto's empirical law that any assortment of events consists of a few major and many minor elements. In the context of SQC, it is important to select the few vital major opportunities for improvement from the many trivial minor ones. The Pareto chart is particularly useful for Cost-to-fix versus Deviation-frequency analysis.

In addition to the histogram, typically a cumulative percentage is also given. From top to bottom, the percentage represents the ratio of the sum of every value to this point, to the sum of every value in the chart.

## See Also

[Configure a Pareto Chart](#)

## Using Statistical Process Control (SPC)

To implement Statistical Process Control (SPC) in a Plant SCADA system, there are two procedures to follow. They are:

- [Add an SPC Tag](#)
- [Configure an SPC Chart](#).

## See Also

- [SPC Alarms](#)
- [SPC Formulas and Constants](#)
- [Control Chart Line Constants](#)

## Add an SPC Tag

Statistical Process Control (SPC) tags specify data that is to be collected for use in SPC operations. Once data is defined it can be dynamically analyzed (as SPC graphs and alarms) at runtime.

SPC tags are similar to trend tags. Like trends, Plant SCADA can collect and store any amount of SPC data. The only restriction on the amount of data that you can store is the size of the hard disk on your computer. (Plant SCADA uses an efficient data storage method to optimize the use of storage space on your computer's hard disk.) For long-term storage, you can archive the data to disk or tape (without disrupting your runtime system). You can also log data at regular intervals (periodic trend), or only when an event occurs (event trend), in the same manner as Trend Tags.

---

**Note:** SPC does not support Periodic-Event trends, which is a combination of the properties of Periodic and Event trends. In addition, if you use event-based SPC tags, your display screen refresh rate may be slower than desired or practical for your application.

---

When you define an SPC tag you will want to be sure to fill in the upper specification limit (USL) and lower specification limit (LSL) if you intend to perform a capability analysis. These values have to accurately represent the users requirements, and the target value lie midway between the two. If these fields are left blank the capability analysis will be meaningless.

### To add an SPC tag:

1. On the **System Model** activity, select **SPC**.
2. Add a row to the Grid Editor.
3. Type the required information in each column, or in the fields in the Property Grid.  
For a description of the properties, see below.
4. Click **Save**.

## Statistical Process Control (SPC) Properties

---

**Note:** If an SPC tag was generated by the Equipment Editor, a number of fields on the properties form will be shaded. To configure these fields, you will need to use [Equipment Editor](#).

---

### Equipment Properties

Field	Description
<b>Equipment</b>	The name of the equipment associated with the SPC tag. Select a name from the drop-down list of existing equipment definitions, or enter a name. There is a limit of 254 characters across the <b>Equipment</b> and <b>Item Name</b> fields, including any

Field	Description
	separating periods (.).
<b>Item Name</b>	<p>The name of the item with which the SPC tag is associated.</p> <p>Items form part of an equipment hierarchy. They can be used to associate tags, alarms and trends with a particular attribute of a physical piece of equipment (see Items).</p> <p>There is a limit of 254 characters across the <b>Equipment</b> and Item Name, including any separating periods (.).</p> <p><b>Note:</b> When defining an item name, avoid using the reserved words. If you use any of these, an error message will display when you compile your project.</p>

**General Properties****⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

If using the File Name property, ensure that the SPC tag uses a unique name. Two tags accessing the same file can result in system errors which may include lost or corrupted trend/SPC data.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Field	Description
<b>SPC Tag Name</b>	The name assigned to the SPC data. If you are logging a variable, use the same name for the SPC tag that you used for the variable tag.
<b>Cluster Name</b>	<p>The name of the cluster that runs the SPC tag. This field needs to be defined if your project has more than one cluster.</p> <p>You can leave this field blank in a multi-cluster system if cluster replication is enabled (see the parameter <a href="#">[General]ClusterReplication</a>).</p>
<b>Type</b>	<p>The type of SPC trend:</p> <ul style="list-style-type: none"> <li>• TRN_PERIODIC</li> <li>• TRN_EVENT</li> </ul> <p><b>Note:</b> SPC does not support Periodic-Event trends, which is a combination of the properties of Periodic and Event trends. In addition, if you use</p>

Field	Description
	event-based SPC tags, your display screen refresh rate may be slower than desired or practical for your application.
<b>Expression</b>	<p>The logged value of the SPC tag. Enter a value of 254 characters or less. You can log individual variables by using a Variable Tag, for example:</p> <p>Expression: PT104</p> <p>Comment: Logs the Variable Tag PT104</p> <p>The value of the process variable PT104 is logged. Variable PT104 has to be defined as a variable tag. You can also log any Cicode expression or function, for example:</p> <p>Expression: PT104/COUNTER</p> <p>Comment: Logs Variable Tag PT104 divided by the Variable Tag COUNTER</p>
<b>Trigger</b>	<p>The Cicode expression (or variable tag) that triggers data logging. Enter a value of 254 characters or less. For example:</p> <p>Trigger: PT104 &lt; 500</p> <p>In this example, logging occurs when the value of the variable tag (PT104) falls below 500.</p> <p>For a periodic SPC trend, data is logged only while the value of the trigger is TRUE. In the above example, data is logged continuously while the value of PT104 remains less than 500. Logging ceases when the value rises to (or above) 500. Logging does not occur again until the value of PT104 falls below 500.</p> <p>You do not have to specify a trigger for a periodic SPC trend. If you do not specify a trigger for a periodic SPC trend, then logging will occur continuously.</p> <p>For an event SPC trend, data is logged once when the value of the trigger changes from FALSE to TRUE. In the above example, one sample is logged when the value of PT104 first becomes less than 500. Another sample is not logged until the value of PT104 rises to (or above 500) and again falls below 500.</p>
<b>Subgroup Size</b>	The size of each subgroup. The default value for this value is 5. Valid values are 1 - 25 inclusive.
<b>Sample Period</b>	The sampling period of the data, in hh:mm:ss

Field	Description
	<p>(hours:minutes:seconds). Plant SCADA checks the <b>Trigger</b> each sample period. If the Trigger is TRUE (or has just changed from FALSE to TRUE, in the case of event SPC trends), Plant SCADA will log the value of the <b>Expression</b>.</p> <p>Examples</p> <p>Sample Period: 30 - Logs data every 30 seconds</p> <p>Sample Period: 10:00 - Logs data every 10 minutes</p> <p>Sample Period: 10:00:00 - Logs data every 10 hours</p> <p>Sample Period: 2:30:00 - Logs data every 2 and a half hours</p> <p>This property is optional. If you do not specify a sample period, the sampling period will default to 10 seconds.</p> <p><b>Note:</b> If you edit this property in an existing project, delete or move or move the associated trend files before you run the new runtime system.</p>
<b>Eng Units</b>	The engineering units of the variable/expression being logged. The engineering units are used by the SPC trend scales and SPC trend cursor displays.
<b>Format</b>	<p>The format of the variable/expression being logged. The format is used by the SPC trend scales and SPC trend cursor displays.</p> <p>This property is optional. If you do not specify a format, the format defaults to #####.#.</p>
<b>Deadband</b>	The value that <b>SPC Tag</b> needs to return to before the SPC data becomes inactive.
<b>Lower Spec Limit</b>	<p>The Lower Specification Limit (LSL). This value is used as the lower limit to determine process capability. When used in conjunction with the USL it provides a tolerance for your process.</p> <p>If you are unfamiliar with process capability and capability indices, ask for expert opinion. Rather than leave this blank (at least) attempt an estimate. Enter a value that you think is the lowest acceptable value of this tag. If you leave this field blank only your capability analysis will be affected.</p>
<b>Upper Spec Limit</b>	The Upper Specification Limit (USL). This value is used as the upper limit to determine process capability. When used in conjunction with the LSL it provides a

Field	Description
	<p>tolerance for your process.</p> <p>If you are unfamiliar with Process Capability and capability indices, ask for expert opinion. Rather than leave this blank (at least) attempt an estimate - Enter a value that you think is the highest acceptable value of this tag. If you do leave this field blank only your capability analysis will be affected.</p>
<b>x Double Bar</b>	<p>The calculation override for process mean (X double bar). If a value is specified here it will be used in every SPC calculation, instead of the value calculated by Plant SCADA. This will affect the calculation of control limits which are normally a function of the collected samples of data.</p> <p>You should not use this field unless you are experienced in SPC.</p>
<b>Range</b>	<p>The calculation override for process range (R bar). If a value is specified here it will be used in every SPC calculation, instead of the value calculated by Plant SCADA. This will affect the calculation of control limits which are normally a function of the collected samples of data.</p> <p>You should not use this field unless you are experienced in SPC.</p>
<b>St Deviation</b>	<p>The calculation override for process standard deviation (s bar). If a value is specified here it will be used in every SPC calculation, instead of the value calculated by Plant SCADA. This will affect the calculation of control limits which are normally a function of the collected samples of data.</p> <p>Do not use this field unless you are experienced in SPC.</p>
<b>Comment</b>	Any useful comment (254 characters).

#### Storage Properties



##### UNINTENDED EQUIPMENT OPERATION

If using the File Name property, ensure that each trend tag uses a unique file name. Two tags accessing the same file can result in system errors which may include lost or corrupted trend/SPC data.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

If you edit one of the following properties in an existing project, delete associated SPC trend files before placing the system back in service.

- No Files
- Period
- Time
- Storage Method

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Field	Description
File Name	<p>The file where the data is to be stored. You need to specify the complete path or use path substitution (see <a href="#">Use a Path Substitution</a>).</p> <p>When Plant SCADA collects data from your plant floor, it stores the data in a file on the hard disk of your computer. When it subsequently uses the data to display an SPC trend, it reads the data from this file. (Plant SCADA uses a separate file for each SPC tag.)</p> <p>The File Name property is optional. If you do not specify a file name, the file name defaults to [DATA]:&lt;Name&gt; on the hard disk where you installed Plant SCADA. Where &lt;Name&gt; is the SPC Tag Name.</p> <p>You can also specify an alternate file name like this:</p> <p>File Name: [DATA]:TANK131</p> <p>Where [DATA] specifies the disk and path for the data. Use path substitution to make your project more 'portable'.</p> <p>Be aware that you will need to manually configure the access control list for a new folder location so that it matches the permissions that were applied to the default location during installation. See <a href="#">Configure Directory Security for Modified Folder Locations</a>.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"><li>• If you use the File Name property, confirm that no other SPC tags or trend tags use the same file name. A compile error message (E2172) will be raised if duplicate file names are detected.</li><li>• Duplicated file names can occur if your path</li></ul>

Field	Description
	<p>substitutions are not configured correctly, for example, if [TREND1] and [TREND2] incorrectly point to the same location. If this occurs, an error will be logged to the syslog.dat file, and a hardware error will be raised.</p> <ul style="list-style-type: none"> <li>If a path with invalid characters (such as '*' or '?') is detected at runtime, an error will be logged to the syslog.dat file, and a hardware error will be raised.</li> <li>A file extension should not be used when specifying a file name. If you edit this property (change the file name or path) in an existing project, existing trend data is ignored.</li> <li>You can no longer store trend files in the 'Bin', 'Runtime', 'Backup' or 'User' directories, or any sub-directories of these. If this is attempted, an error will be logged to the syslog.dat file, and a hardware error will be raised. If you have existing Version 3.xx or 4.xx projects that use these directories to store trend files, the path for these will have to be changed to the 'Data' directory.</li> </ul>
<b>No. Files</b>	<p><b>Note:</b> If you edit this property in an existing project, delete associated SPC trend files before placing the system back in service.</p> <p>The number of history files stored on your hard disk (for this tag).</p> <p>By default, two history 2 history files are stored on your hard disk. The maximum number of files you can specify is 270.</p>
<b>Period</b>	<p><b>Note:</b> If you edit this property in an existing project, delete associated SPC trend files before placing the system back in service.</p> <p>The period of the history file, in hh:mm:ss (hours:minutes:seconds). Alternatively, you can:</p> <ul style="list-style-type: none"> <li>Specify a weekly period by entering the day of the week on which to start the history file, for example Monday, Tuesday, Wednesday, etc.</li> <li>Specify a monthly period by entering the day of the month on which to start the history file, for example 1st, 2nd, 3rd, 4th, 5th, etc.</li> </ul>

Field	Description
	<ul style="list-style-type: none"> <li>Specify a yearly period by entering the day and the month on which to start the history file, for example 1st January, 25th February, etc. The day and month needs to be separated by a space.</li> </ul> <p>If you do not specify a period, the period defaults to Sunday (weekly).</p>
<b>Time</b>	<p><b>Note:</b> If you edit this property in an existing project, delete associated SPC trend files before placing the system back in service.</p> <p>The time of day to synchronize the beginning of the history file, in hh:mm:ss (hours:minutes:seconds). If you do not specify a time, the file is synchronized at 0:00:00 (i.e. midnight).</p>
<b>Storage Method</b>	<p><b>Note:</b> If you edit this property in an existing project, delete associated SPC trend files before placing the system back in service.</p> <p>Select either <b>Scaled</b> or <b>Floating Point</b> as the storage method for the SPC data (64 characters). The key difference between these two options is that Scaled is a two-byte data storage method, whereas Floating Point uses eight bytes.</p> <p>Floating Point storage has a dramatically expanded data range in comparison to Scaled storage, allowing values to have far greater resolution. However, you need to consider that it also uses a lot more disk space. Scaled should be used where compatibility with pre-V5.31 trend history files is necessary.</p> <p>If you do not specify a storage method, it is set to <b>Scaled</b> by default.</p>

### Security Properties

Field	Description
<b>Area</b>	<p>The area to which the SPC data belongs. Only users with access to this area (and any necessary privileges) will be able to display the SPC data on an SPC page. For example, if you enter Area 1 here, operators need to have access to Area 1 (plus any necessary privileges) to display the SPC data.</p>
<b>Privilege</b>	<p>The privilege necessary by an operator to display the SPC data on an SPC page.</p>

## Historian Properties

Field	Description
Historize	This field enables you to automatically historize and publish the specified SPC tag in CitectHistorian. If you set this field to "TRUE", the variable will be included in an automated configuration process within the Historian environment. If you set the field to "FALSE" (or leave it blank), the variable will not be included.

## Project Properties

Property	Description
Project	The project in which the SPC tag is configured.

## See Also

[Configure an SPC Chart](#)

## Configure an SPC Chart

Plant SCADA displays Statistical Process Control information on three types of charts:

- XRS Control Chart
- Process Capability Chart
- Pareto Chart.

### To configure an SPC chart:

1. Open Graphics Builder.
  2. Click the **New Page** button on the Command Bar, or choose **File | New**.
  3. On the New dialog, select **Page**.
  4. Choose the **Resolution** (size) of the SPC page.
  5. Choose an SPC **Template** for the SPC page:
    - **SPCXRSChart** - An XRS control chart
    - **SPCCpk** - A capability (Cpk) chart combined with an Mean chart.
- 
- Note:** Pareto charts are configured slightly differently and hence, are not included here (see [Configure a Pareto Chart](#)).
6. Click **OK**.
  7. Double-click the graph display.
  8. Enter the variable tag in the Genie pop-up.
  9. Click **OK**.

10. Save the page.

## See Also

[SPC Control Charts](#)

### SPC Control Charts

Genies simplify the task of adding a new SPC page.

You can use Plant SCADA to configure the following types of control charts:

- [Configure an XRS Control Chart](#)
- [Configure a Capability Chart](#)
- [Configure a Pareto Chart.](#)

## See Also

[Control Chart Line Constants](#)

### Configure an XRS Control Chart

The XRS charts display trends of subgroup means (X bar), ranges (R) and standard deviations (s). The XRS chart operates similarly to a standard trend, but with additional SPC extra features. Each subgroup displays as a single node on the graph and consecutive nodes are linked by a line.

Each control chart has a central line and two control limits-upper and lower (UCL and LCL). Plant SCADA automatically calculates these SPC values at run-time. If you want to override the UCL and LCL you can do so by entering the Process Mean, Range, and Standard Deviation fields in SPC Tags.

#### To configure an XRS Control Chart:

1. Define the SPC Tags.
2. Create the page using an XRS template.

---

**Note:** If you want to develop your own XRS template, the method is to copy and modify an existing template.

---

## See Also

[Configure a Capability Chart](#)

### Configure a Capability Chart

The process capability chart is a frequency histogram and distribution of the sample data currently displayed (on the Mean chart). Plant SCADA automatically takes the data being trended, builds a distribution, adds the LSL and USL. It also calculates the Cp, Cpk, kurtosis, and skewness indices.

The process capability is defined in relation to the upper and lower specification limits (USL and LSL) for a given

SPC Tag. These values are defined in SPC Tags and accurately represent the users requirements.

#### To configure a capability chart:

1. Define the SPC Tags and specify the LSL and USL.
2. Create the page using a Capability (Cpk) template.

#### Configure a Pareto Chart

Pareto analysis is a technique used to identify the relative importance of problems and conditions. The Pareto chart is a frequency histogram ordered from highest to lowest (Plant SCADA automatically orders the bars at run-time). The data for each bar in the histogram represents one Plant SCADA variable - as defined in Variable Tags. Do not use SPC tags.

---

**Note:** Typically the frequency in a histogram is of integer type, though you can use floating point types if you want. Negative values are not valid.

---

#### To configure a Pareto Chart:

1. Define the Variable Tags (Pareto charts do not use SPC Tags).
2. Create the page using a Pareto template.

#### To configure a page using a Pareto chart:

1. In Graphics Builder click **New Page**, or choose **File | New**.
2. Select **Type: Page**.
3. Choose the **Resolution** (size) of the SPC page.
4. Choose the **SPCParetoTemplate** for the SPC chart.
5. Click **OK**.
6. Double-click the display (where prompted by the template).
7. Enter the variable tags in the **Tag Name** fields. (Use Variable tags here, not necessarily SPC tags.)
8. Enter the variable descriptions in the **Tag Description** boxes.
9. Click **OK**.
10. Save the page.

#### SPC Alarms

Plant SCADA automatically monitors several special kinds of conditions that are specific to SPC data. When specific patterns or events occur to an SPC tag, it will set the appropriate alarm. Typically these alarms are related to, and used in conjunction with, the XRS control charts.

SPC alarms are configured differently to standard digital alarms to provide for this extra functionality. SPC alarms needs to be configured using the Advanced Alarms form. You use the SPCAlarms Cicode function to check for the condition of the alarms:

---

**Note:** An SPC alarm can only be defined on an Alarm Server in the same cluster as the Trends Server that contains the SPC tag (though the variable tag referenced in the SPC tag can be on a different IOServer cluster).

---

Complete the **Advanced Alarm** settings as shown here:

Alarm Tag	Feed_Above_UCL
Alarm Desc	Un-controlled variation
Expression	SPCALarms("Feed_SPC", XAboveUCL)
Comment	Several samples are above UCL

The SPC (Trends) Server checks for any specified alarm conditions. When one is detected, it informs the Alarm Server that an alarm has occurred. Be aware of the number of subgroups displayed on your SPC charts, and the number used in SPC alarm calculations (as set by the [\[SPC\]AlarmBufferSize](#) parameter). If these two values differ, SPC alarms might not correlate with your SPC charts.

The following list shows the alarms types that are valid:

Name	Description
XFreak	Single point greatly differs (2 sigma) from the center line.
XOutsideCL	Process mean outside either of the control limits (UCL or LCL).
XAboveUCL	Process mean above the upper control limit (UCL).
XBelowLCL	Process mean below the lower control limit (LCL).
XOutsideWL	Process mean outside the alert limits which are 67% of the UCL and LCL.
XGradualUp	Process mean is gradually drifting up to a new level indicated by several consecutive points above the mean.
XGradualDown	Process mean is gradually drifting down to a new level indicated by several consecutive points below the mean.
XUpTrend	Several points continuously increasing in value.
XDownTrend	Several points continuously decreasing in value.
XErratic	Large fluctuations that are greater than the control limits.
XStratification	Artificial constancy. Several consecutive points are close to (within 1 sigma of) the center line.
XMixture	Several consecutive points are far from (outside 1 sigma of) the center line.
ROutsideCL	Process range outside either of the control limits (UCL or LCL).

Name	Description
RAboveUCL	Process range above the upper control limit (UCL).
RBelowLCL	Process range below the lower control limit (LCL).

**Note:** The above alarms rely on  $n$  number of consecutive points to generate the alarm. The value of  $n$  can be set for each type of alarm through [SPC Parameters](#).

## See Also

[SPC Formulas and Constants](#)

### SPC Formulas and Constants

The SPC calculations are based on the samples collected in subgroups. Each subgroup will have the same number of samples, typically 4. The subgroup size for each SPC tag is set at the **SPC Tags** properties form.

The number of samples in each subgroup can range from 1 to 25 inclusive.

### When the number of samples in each subgroup is 1

#### Subgroup Mean ( $\bar{X}_i$ ):

is the value of the single sample in the group, and is defined by:

$$\bar{X}_i = X_i$$

Where  $X_i$  is the single sample value in the subgroup.

#### Moving Range (MR):

is the difference between successive sample values, and is defined by:

$$MR_i = X_i - X_{i-1} \quad i \geq 2$$

Where  $X_i$  is the current sample value and  $X_{i-1}$  is the previous sample value. The number of moving ranges in the process is always one less than the number of subgroups.

#### Subgroup Standard Deviation (s):

is a measure of absolute variation or dispersion. It describes how much the sample values differ from their mean, and is estimated by:

$$s_i = \frac{MR_i}{D2} \quad i \geq 2$$

The number of subgroup standard deviations in the process is always one less than the number of subgroups.

#### Process Average ( $\bar{\bar{X}}$ ):

$$\bar{\bar{X}} = \frac{\bar{X}_1 + \bar{X}_2 + \dots + \bar{X}_m}{m}$$

Where  $\bar{X}_1$ ,  $\bar{X}_2$ , and  $\bar{X}_m$  are the subgroup means, and m is the total number of subgroups in the process.

**Process Range ( $\bar{R}$ ):**

$$\bar{R} = \frac{MR_2 + MR_3 + \dots + MR_m}{m - 1}$$

Where MR2, MR3, and MRm are the subgroup moving ranges, and m is the total number of subgroups in the process.

**Process Standard Deviation ( $s$ ):**

$$\bar{s} = \frac{\bar{R}}{D2}$$

## When the number of samples in each subgroup is greater than 1

**Subgroup Mean ( $\bar{X}$ ):**

is the average (not median or center) of the samples in the group, and is defined by:

$$\bar{X} = \frac{X_1 + X_2 + \dots + X_n}{n}$$

Where X1, X2, and Xn are the sample values in the subgroup, and n is the total number of samples in the subgroup.

**Subgroup Range (R):**

is the difference between the highest and lowest samples in the group, and is defined by:

$$R = X_{\max} - X_{\min}$$

Where Xmax is the maximum sample value and Xmin is the minimum sample value in the group.

**Subgroup Standard Deviation (s):**

is a measure of absolute variation or dispersion. It describes how much the sample values differ from their mean, and is defined by:

$$s = \sqrt{\frac{\sum (X_i - \bar{X})^2}{n - 1}}$$

Where X's are the sample values in the group,  $\bar{X}$  is the group average, and n is the number of samples in the group.

**Process Average ( $\bar{\bar{X}}$ ):**

$$\bar{\bar{X}} = \frac{\bar{X}_1 + \bar{X}_2 + \dots + \bar{X}_m}{m}$$

Where  $\bar{X}_1$ ,  $\bar{X}_2$ , and  $\bar{X}_m$  are the subgroup averages, and m is the total number of subgroups in the process.

**Process Range ( $\bar{R}$ ):**

$$\bar{R} = \frac{R_1 + R_2 + \dots + R_m}{m}$$

Where  $R_1$ ,  $R_2$ , and  $R_m$  are the subgroup ranges, and  $m$  is the total number of subgroups in the process.

**Process Standard Deviation ( $\bar{s}$ ):**

$$\bar{s} = \frac{s_1 + s_2 + \dots + s_m}{m}$$

Where  $s_1$ ,  $s_2$  and  $s_m$  are the group standard deviations, and  $m$  is the total number of groups in the process.

**Average Control Limits (UCL<sub>x</sub> and LCL<sub>x</sub>):**

Specify the approximated 3-sigma boundaries. For a normal distribution 99.74% of the samples will fall within this boundary.

$$UCL_x = \bar{\bar{X}} + A_2 * \bar{R}$$

$$LCL_x = \bar{\bar{X}} - A_2 * \bar{R}$$

Where  $\bar{R}$  is the Process Range and  $A_2$  is a constant (given in the Control Chart Line Constants table).

**Range Control Limits (UCLR and LCLR):**

Specify the approximated 3-sigma boundaries. For a normal distribution 99.74% of the samples will fall within this boundary.

$$UCL_R = D_4 * \bar{R}$$

$$LCL_R = D_3 * \bar{R}$$

Where  $\bar{R}$  is the Process Range and  $D_3$  and  $D_4$  are constants (given in the Control Chart Line Constants table).

**Standard Deviation Control Limits (UCLs and LCLs):**

Specify the approximated 3-sigma boundaries. For a normal distribution 99.74% of the samples will fall within this boundary.

$$UCL_s = B_4 * \bar{s}$$

$$LCL_s = B_3 * \bar{s}$$

Where  $\bar{s}$  is the Process Standard Deviation and  $B_3$  and  $B_4$  are constants given in the Control Chart Line Constants table).

**Process Capability (Cp):**

Is the capability of a process to meet a specific tolerance. A process is considered capable when the percentage of samples of a variable for that process that fall within the upper and lower specification limits is greater than a specified value.

The inherent process capability is defined as:

$$C_p = \frac{(USL - LSL)}{6s}$$

- $C_p > 1.0$  - Indicates the process variation is within the specified limits (USL and LSL) and therefore, is capable.
- $C_p < 1.0$  - Indicates the process is not capable.

The process capability based on worst case data is defined as:

$$C_{pk} = \frac{\min((USL - \bar{\bar{X}}, (\bar{\bar{X}} - LSL))}{3s}$$

- $C_{pk} < 0$  - Indicates the process mean is outside the specified limits (USL and LSL)
- $C_{pk} = 0$  - Indicates the process mean is equal to one of the specified limits.
- $C_{pk} > 0$  - Indicates the process mean is within the specified limits.
- $C_{pk} = 1.0$  - Indicates that one side of the 6-sigma limits falls on a specification limit.
- $C_{pk} > 1.0$  - Indicates that the 6-sigma limits fall completely within the specified limits.

#### **Skewness (Sk):**

Is the degree of asymmetry of a frequency distribution (usually in relation to a normal distribution).

$$Sk = \frac{\sum(X_i - \bar{\bar{X}})^3}{Ns}$$

where N is the number of samples for the entire process (i.e. Subgroup Size \* number of Subgroups).

- Skewness  $> 0$  - Indicates that the histogram's mean (and tail) is pushed to the right.
- Skewness  $< 0$  - Indicates that the histogram's mean (and tail) is pushed to the left.

#### **Kurtosis (Ku):**

Is the degree of peakedness of a frequency distribution (usually in relation to a normal distribution).

$$Ku = \frac{\sum(X_i - \bar{\bar{X}})^4}{Ns}$$

where N is the number of samples for the entire process (i.e. Subgroup Size \* number of Subgroups).

- Kurtosis  $< 3$  - Indicates a thin distribution with a relatively high peak.
- Kurtosis  $> 3$  - Indicates a distribution that is wide and flat topped.

## **Control Chart Line Constants**

The table below shows the control chart line constants:

	Averages	Ranges			Standard Deviations	
Samples in Group	A2	D2*	D3	D4	B3	B4
1	2.660	1.128	0	3.267	0	3.267
2	1.880		0	3.267	0	3.267
3	1.023		0	2.574	0	2.568
4	0.729		0	2.282	0	2.266
5	0.577		0	2.114	0	2.089
6	0.483		0	2.004	0.030	1.970
7	0.419		0.076	1.924	0.118	1.882
8	0.373		0.136	1.864	0.185	1.815
9	0.337		0.184	1.816	0.239	1.761
10	0.308		0.223	1.777	0.284	1.716
10	0.308		0.223	1.777	0.284	1.716
11	0.285		0.256	1.744	0.321	1.679
12	0.266		0.283	1.717	0.354	1.646
13	0.249		0.307	1.693	0.382	1.618
14	0.235		0.328	1.672	0.406	1.594
15	0.223		0.347	1.653	0.428	1.572
16	0.212		0.363	1.637	0.448	1.552
17	0.203		0.378	1.622	0.466	1.534
18	0.194		0.391	1.608	0.482	1.518
19	0.187		0.403	1.597	0.497	1.503
20	0.180		0.415	1.585	0.510	1.490
21	0.173		0.425	1.575	0.523	1.477
22	0.167		0.434	1.566	0.534	1.466
23	0.162		0.443	1.557	0.545	1.455

	Averages	Ranges			Standard Deviations	
24	0.157		0.451	1.548	0.555	1.445
25	0.153		0.459	1.541	0.565	1.435

\* D2 is only used for estimating standard deviation when there is one sample per subgroup.

Reference ANSI Z1.1-1985, Z1.2-1985 & Z1.3-1985: American National Standard, *Guide for Quality Control Charts, Control Chart Method of Analyzing Data, Control Chart Method of Controlling Quality During Production*.

#### Hints

Double-click the chart area on an SPC page to display the SPC Genie and change the SPC variables.

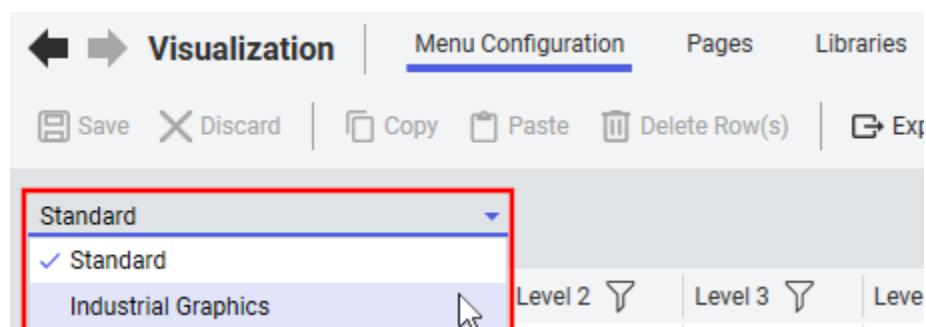
The tools and menu items in these procedures automatically open the Plant SCADA form for you. Move the cursor till it changes to a hand to find these tools or options.

## Visualization

The **Visualization** activity is used to manage the content that comprises the interface an operator uses to monitor and control a runtime system. It includes the following views:

- [Menu Configuration](#)
- [Pages](#)
- [Libraries](#)
- [Content Types](#)
- [Keyboard Commands](#)
- [Reports](#).

In some cases, a view will feature a drop-down menu below the command bar that allows you to choose the graphics between "Standard" and "Industrial Graphics."



- Standard graphics are designed to run on Plant SCADA display clients, typically running on network computers. They are created and edited using [Graphics Builder](#).
- Industrial Graphics are designed for delivery to web-based clients via an AVEVA™ Industrial Graphics Server. They are created and edited using the AVEVA Industrial Graphics Editor.

## See Also

[AVEVA Industrial Graphics](#)

## Menu Configuration

The **Menu Configuration** view in the **Visualization** activity allows you to define hierarchical menu items that can be used at runtime to provide navigational assistance to an operator.

Plant SCADA supports two types of menus:

- **Standard Menus**

Standard menus are used to navigate projects that are run on Plant SCADA display clients.

The way a standard menu is implemented is determined by the Starter Project upon which a project was based.

For example, in a project based on the SxW Starter Project, the menu is incorporated as a common feature across every page. In a project based on the Situational Awareness Starter Project, the menu you configure determines the layout of the Navigation Zone on the operator dashboard.

For more information, see [Configure a Standard Plant SCADA Menu](#).

- **Industrial Graphics Menus**

Industrial Graphics menu is designed to arrange the pages that are delivered to web-based clients via the AVEVA Web Server.

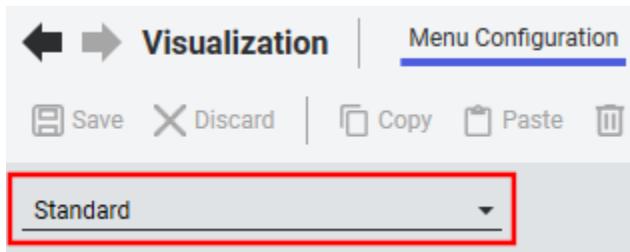
For more information, see [Configure a Menu for Industrial Graphics Pages](#).

## See Also

[Menu Functions](#)

## Configure a Standard Plant SCADA Menu

You can use the **Menu Configuration** view in Plant SCADA Studio's **Visualization** activity to configure a standard menu for your project's runtime system. To do this, firstly select **Standard** from the drop-down menu below the command bar.



**Note:** If you want to configure a menu for Industrial Graphics pages displayed on a Web Server, see [Configure a Menu for Industrial Graphics Pages](#).

The method you use to configure a standard menu is determined by the Starter Project that was used to create your project. For an appropriate description of the Menu Configuration Properties, select from the following

Starter Projects.

### Situational Awareness Starter Project

If your project is created from a Situational Awareness Starter Project, the menu configuration is used to build the following:

- The layout of buttons and tabs on the Navigation Zone on the operator dashboard (see [Prepare the Navigation Menu](#)).
- The tabs that appear across the Information Zone on the operator dashboard.
- The content of the workspace header bar.
- A set of custom tabs for the [Tab Bar Genie](#).

A Menu Configuration has the following properties:

#### General Properties

Property	Description
<b>Page</b>	<p>The Page property is used to give a name to the menu that will include the item.</p> <p>In a Situational Awareness project, the system expects the following menu names:</p> <ul style="list-style-type: none"><li>• <b>InformationZone</b> – used to define the tabs that appear across the Information Zone on the operator dashboard.</li><li>• <b>Headerbar</b> – used to define the content of the workspace header bar.</li><li>• <b>Navigation</b> – used to define the layout of buttons and tabs on the Navigation Zone on the operator dashboard.</li></ul> <p>A project based on the Situational Awareness Starter Project will include a set of default entries for the <b>InformationZone</b> and <b>Headerbar</b> menus.</p> <p>There are no default entries for the <b>Navigation</b> menu, as you will need to manually configure these to reflect the pages included in your project. See <a href="#">Prepare the Navigation Menu</a>.</p>
<b>Level 1-6</b>	<p>In a Situational Awareness project, the <b>Level 1-6</b> properties are used in the following ways, based on the three expected menus:</p> <ul style="list-style-type: none"><li>• <b>InformationZone</b> – this menu defines the tabs that appear across the Information Zone. Only a single sets of tabs is supported, which means only the <b>Level 1</b> field is used.</li></ul>

Property	Description
	<ul style="list-style-type: none"> <li>• <b>Headerbar</b> – this menu defines the content of the workspace header bar. The included buttons are specified by the <b>Level 1</b> field. The default "Login Menu" uses <b>Level 2</b> to add a "Log Out" option as a drop-down menu.</li> <li>• <b>Navigation</b> – this menu defines the layout of buttons and tabs on the Navigation Zone on the operator dashboard. <b>Level 1</b> of the menu sets the overall context for the workspace. The tabs along the navigation section header represent the menu's <b>Level 2</b> entries. The buttons on each tab represent up to 20 lower-level menu entries.</li> </ul> <p>In each case, the arrangement of menu entries is determined by the <b>Order</b> property.</p> <p><b>Note:</b> The greater-than character ( &gt; ) cannot be used within a menu name.</p>
<b>Menu Command</b>	<p>The Cicode expression to be executed when the menu item is selected. String of up to 254 characters.</p> <p>For Situational Awareness projects, use the "Navigation_ShowTargetPage" command to call a page.</p> <p>If you have the same page in HD1080 and UHD4K resolutions (to support a range of client screens), the variation of the page that displays at runtime will be determined by the context of the current workspace. This means you do not need to add the resolution suffix ("_HD1080" or "_UHD4K") to the end of the page name specified in the <b>Target Page</b> field.</p>
<b>Target Page</b>	<p>The page that is displayed when the menu item is executed. See the <b>Menu Command</b> property.</p>
<b>Comment</b>	<p>An optional comment (of up to 128 characters) about the menu entry.</p>
<b>Order</b>	<p>Defines the position of a menu entry in relation to the other menu entries in the same hierarchy level. The <b>Order</b> property functions in the following ways, based on the three expected menus:</p> <ul style="list-style-type: none"> <li>• <b>InformationZone</b> – the order defines how the tabs will appear across the top of Information Zone (from left to right).</li> </ul>

Property	Description
	<ul style="list-style-type: none"> <li>• <b>Headerbar</b> – the order defines how the command buttons will appear across the workspace header bar (from left to right).</li> <li>• <b>Navigation</b> – For <b>Level 1</b> entries, the order defines how the tabs will appear across the top of Navigation Zone (from left to right). For lower level entries, the order determines the layout of up to 20 buttons on each tab (four rows of five buttons, from left to right).</li> </ul> <p>If this field is left blank, it will take the default value of zero (0). If more than one entry of the same order exists, they are displayed in the same order as they are defined in the database. In the case of Navigation menu, the entry will fill the first available space in the layout of buttons.</p> <p>This property affects the order that the menu entries are returned through menu Cicode functions.</p>
<b>Hidden when</b>	A Cicode expression that determines when the menu entry is hidden on the page.
<b>Disabled when</b>	A Cicode expression that determine when the menu entry is disabled on the page.
<b>Disabled style</b>	<p>A number to indicate how the disabled entry is displayed in the menu. It is up to the user to decide what the numbers mean.</p> <p><b>Note:</b> By default, this property is not used by a Situational Awareness project.</p>
<b>Symbol</b>	If the menu entry is associated with a tab, this field allows you to display an icon on the tab. Enter the name of the graphic file you would like to display. You can use the <b>Custom 2</b> property (see below) to hide the symbol if required.
<b>Privilege</b>	The user privilege (0–8) that is necessary to select the menu entry.
<b>Area</b>	The user area (0–255) that is necessary to select the menu entry.
<b>Checked</b>	If the menu entry is associated with a tab, this field allows you to specify if the tab is pinned by default. Leaving the field blank will mean the tab is not pinned

Property	Description
	by default. Any other value will mean the tab is pinned by default.  This setting will only work if pinning is enabled for the tab bar. This is determined by the <b>Can Tabs Be Pinned?</b> parameter in the <a href="#">Tab Bar Genie</a> .
Width	The width of the menu entry on the page.  <b>Note:</b> By default, this property is not used by a Situational Awareness project.

**Equipment Properties**

Property	Description
Cluster	The name of the cluster associated with the menu item. The specified cluster should be configured otherwise a compile will not be successful.
Equipment	Name of equipment associated with the menu item. Select a name from the drop down list, or type a name. String limit of 254 characters, including separating periods (.)

**Custom Properties**

Property	Description
Custom 1	If the menu entry is associated with a tab, this property determines if the <b>Close</b> button will appear on the tab. If you leave this field blank, the Close button will appear. Any other value will hide the Close button.
Custom 2	If the menu entry is associated with a tab, this property determines if the icon identified in the <b>Symbol</b> field will appear on the tab. If you leave this field blank, the icon will appear. Any other value will hide the icon. .
Custom 3	If the menu entry is associated with a tab, this property determines if a label will appear on the tab. If you leave this field blank, the label will appear. Any other value will hide the label.
Custom 4-8	These custom properties are reserved for system use.

**Project Properties**

Property	Description
<b>Project</b>	The project in which the menu will be created.

### SxW Starter Project

If your project is created from a SxW starter project, some common menu items are already pre-configured in the menu configuration database (as part of the starter project). They will be displayed on the page automatically at runtime. To view additional pages in your project at runtime you will need to manually configure the menu items.

You can assign individual menu items to a particular page, or make them available to every page. Menu items configured in included projects will be merged with the ones configured in the main project, into a single menu tree structure when you compile your main project. This allows extra menu items to be automatically added to your project depending on what projects are included.

A Menu Configuration has the following properties:

#### General Properties

Property	Description
<b>Page</b>	The page on which this entry will exist. If blank, the entry exists on every page.
<b>Level 1-6</b>	Creates the menu hierarchy path of the entry. String length can be up to 64 characters.  Up to 6 levels of menu hierarchy are supported. Menu entries that have the same upper levels are considered to be under the same hierarchical branch.  Menu entries are not case sensitive.  <b>Note:</b> The greater-than character ( > ) cannot be used within a menu name.
<b>Menu Command</b>	The Cicode expression to be executed when the menu item is selected. String of up to 254 characters.
<b>Comment</b>	A comment about the menu entry. String of up to 128 characters.
<b>Order</b>	The position of a menu entry in relation to the other menu entries in the same hierarchy level.  This property affects the order that the menu entries are returned through the menu Cicode functions. If this field is left blank, it will take the default value of zero (0). The relevant menu entries will be returned at the start as a result. If more than one entry of the same order exists, they are displayed in the same order as they are defined in the database.
<b>Symbol</b>	The symbol to associate with the menu entry. The string length can be up to 128 characters. A symbol

Property	Description
	needs to be already defined in the project / included project, and specified in the format of <library name>.<symbol name>.
<b>Hidden when</b>	Cicode expression to determine when the menu entry is hidden on the page. String of up to 254 characters.
<b>Disabled when</b>	Cicode expression to determine when the menu entry is disabled on the page. String of up to 254 characters.
<b>Disabled style</b>	<p>By default, this property is not used by an SxW project. However, if you choose to implement your own menu system, you can retrieve the value of this field (or any other fields) via the Menu family of Cicode functions at runtime.</p> <p>For example, if you set a numeric value for the <b>Disable Style</b> field in a menu configuration, you can then use the Menu family of Cicode functions to retrieve the value:</p> <pre data-bbox="871 958 1519 1262">INT hMenuNode = MenuGetPageNode("Headerbar");  INT hItemNode = MenuGetNodeByPath(hMenuNode, "Login Menu.@(Log Out)");  INT nDisabledStyle = StrToInt(MenuNodeGetProperty(hItemNode, 4));  INT bIsDisabled = MenuNodIsDisabled(hItemNode);</pre> <p>Once you have retrieved the disabled style and disabled state, you can expose them as the return value of Cicode functions. You can then call the functions in page animations such as color fill, visibility, and so on.</p> <p><b>Note:</b> This property has no direct relation to the Disable Style property on the Disable tab for a graphic object. The latter is defined as part of the graphic object and cannot be altered via Cicode.</p>
<b>Width</b>	The width of the menu entry on the page. It is up to the user to decide what units to be used.
<b>Checked</b>	Determines if the menu entry is initialized with a checked status.
<b>Privilege</b>	The user privilege of (0-8) necessary to select the menu entry.

Property	Description
Area	<p>The user area (0-255) necessary to select the menu entry.</p> <p><b>Note:</b> At runtime, you can use the Menu family of Cicode functions to access the information entered in the Menu Configuration dialog.</p>

**Equipment Properties**

Property	Description
Cluster	The name of the cluster associated with the menu item. The specified cluster should be configured otherwise a compile will not be successful.
Equipment	Name of equipment associated with the menu item. Select a name from the drop down list, or type a name. String limit of 254 characters, including separating periods (.).

**Custom Properties**

Property	Description
Custom 1-8	User-defined strings that can be accessed by Cicode Functions <a href="#">MenuNodeGetProperty</a> and <a href="#">MenuNodeSetProperty</a> . Max length is 254 characters.

**Project Properties**

Property	Description
Project	The project in which the menu will be created.

To configure menus for SxW templates, see [Creating Custom Menus for SxW Templates](#).

**Tab Style Starter Project**

A set of page templates known as [Tab Style Templates](#) is provided with the product. These templates already contain a tabbed menu system.

If your pages are based on these templates, the menu items configured in the menu configuration database will be displayed on the page automatically at runtime. If no menu is defined in the project, a default menu will be created at runtime to provide access to pages in the project. The default menu functionality can be changed using [\[Page\]AddDefaultMenu](#).

A Menu Configuration has the following properties:

**General Properties**

Property	Description
Page	The page on which this entry will exist. If blank, the entry exists every page.

Property	Description
	The page name of "Template" is a keyword used in the Tab Style templates for defining custom pop-up menus.
<b>Level 1-6</b>	<p>Creates the menu hierarchy path of the entry. String length can be up to 64 characters.</p> <p>Up to 6 levels of menu hierarchy are supported. Menu entries that have the same upper levels are considered to be under the same hierarchical branch.</p> <p>Menu entries are not case sensitive.</p> <p><b>Note:</b> The greater-than character ( &gt; ) cannot be used within a menu name.</p> <p>In a Tab Style Project:</p> <ul style="list-style-type: none"> <li>• Level 1 represents the tabs on the tab bar.</li> <li>• Level 2 represents the buttons displayed under a particular tab.</li> <li>• Level 3 represents the menu items of the drop-down menu next to a particular button.</li> <li>• Level 4 represents the sub-menu items of a particular drop-down menu item.</li> </ul>
<b>Menu Command</b>	The Cicode expression to be executed when the menu item is selected. String of up to 254 characters.
<b>Comment</b>	A comment about the menu entry. String of up to 128 characters.
<b>Order</b>	<p>The relative display position of the tabs/buttons or menu items among themselves.</p> <p>For the tabs and buttons, it determines the display order from left to right. For the drop-down menu items, it determines the display order from the top to the bottom.</p> <p>If this field is left blank, it will take the default value of 0. The relevant entries will be displayed at the start as a result.</p> <p>If more than one entry of the same order exists, they are displayed in the same order as they are defined in the database.</p>
<b>Symbol</b>	The symbol to associate with the menu entry. The string length can be up to 128 characters. The format to use is <library name>.<symbol name>.

Property	Description
	<p>The template expects the symbol to be a certain size when it is configured against different menu types.</p> <ul style="list-style-type: none"> <li>For tabs, symbols are 16 x 16 pixels.</li> <li>For buttons, symbols are 32 x 32 pixels.</li> </ul> <p>Some common symbols can be found in the symbol libraries "icons_16x16" and "icons_32x32".</p> <p>The field is not applicable for menu and sub-menu items.</p>
<b>Hidden when</b>	<p>The tab menu bar on the template does not support visibility. When the expression defined in this field is true, the relevant menu entry is disabled instead.</p>
<b>Disabled when</b>	<p>Cicode expression to determine when the menu entry is disabled on the page. String of up to 254 characters.</p>
<b>Disabled style</b>	<p>By default, this property is not used by a Tab Style project. However, if you choose to implement your own menu system, you can retrieve the value of this field (or any other fields) via the Menu family of Cicode functions at runtime.</p> <p>For example, if you set a numeric value for the <b>Disable Style</b> field in a menu configuration, you can then use the Menu family of Cicode functions to retrieve the value:</p> <pre> INT hMenuNode = MenuGetPageNode("Headerbar");  INT hItemNode = MenugetNodeByPath(hMenuNode, "Login Menu.@(Log Out)");  INT nDisabledStyle = StrToInt(MenuNodeGetProperty(hItemNode, 4));  INT bIsDisabled = MenuNodIsDisabled(hItemNode); </pre> <p>Once you have retrieved the disabled style and disabled state, you can expose them as the return value of Cicode functions. You can then call the functions in page animations such as color fill, visibility, and so on.</p> <p><b>Note:</b> This property has no direct relation to the Disable Style property on the Disable tab for a graphic object. The latter is defined as part of the graphic object and cannot be altered via Cicode.</p>

Property	Description
<b>Width</b>	<p>The width (measured in pixels) of the item on the page. If this is not specified, the item is automatically sized to show the most content. This field only applies to the tabs and buttons.</p> <p>The objects used in the template are subjected to minimum widths:</p> <ul style="list-style-type: none"> <li>• Tab (Level 1) = 63 pixels.</li> <li>• Button(Level 2) = 61 pixels.</li> </ul>
<b>Checked</b>	<p>Determines if the menu entry is initialized with a checked status. This field only applies to the menu items and sub-menu items.</p>
<b>Privilege</b>	<p>The user privilege of (0-8) necessary to select the menu entry.</p>
<b>Area</b>	<p>The user area (0-255) necessary to select the menu entry.</p> <p><b>Note:</b> At runtime, you can use the Menu family of Cicode functions to access the information entered in the Menu Configuration dialog.</p>

### Equipment Properties

Property	Description
<b>Cluster</b>	<p>The name of the cluster associated with the menu item. The specified cluster should be configured otherwise a compile will not be successful.</p>
<b>Equipment</b>	<p>Name of equipment associated with the menu item. Select a name from the drop down list, or type a name. String limit of 254 characters, including separating periods (.).</p>

### Custom Properties

Property	Description
<b>Custom 1-8</b>	<p>User-defined strings that can be accessed by Cicode Functions <a href="#">MenuNodeGetProperty</a> and <a href="#">MenuNodeSetProperty</a>. Max length is 254 characters.</p>

### Project Properties

Property	Description
Project	The project in which the menu will be created.

To configure menus for Tab Style templates, see [Create Custom Tab Style Menus](#).

You can also create a custom menu system for your project through the combination of Cicode and graphical objects.

At runtime, three types of menu trees are created.

1. The first type of menu tree allows you to access the configured menu items for generic (all) pages or a specific page. You can access this menu tree using the Cicode function [MenuGetGenericNode](#) to get the page node for the generic pages, and [MenuGetPageNode](#) to get the menu items specific to a page.
2. The second menu tree automatically merges the menu items for the generic pages and the items specified for the current page. You can access this menu tree using the Cicode function [MenuGetWindowNode](#) to get to the root node.
3. The third type of menu tree exists as a virtual menu that is not associated with the generic pages. You can access this menu tree using the Cicode function [MenuGetWindowNode](#) to get to the root node.

**Note:** For existing users of the CSV\_Include project, the menu was defined in a file called Menu.dbf in your project folder. You can migrate the menu definition in this file to the built-in menu configuration database using the [Project Migration Tool](#). This step is useful if you plan to migrate your project from the CSV\_Include project to a newer set of templates.

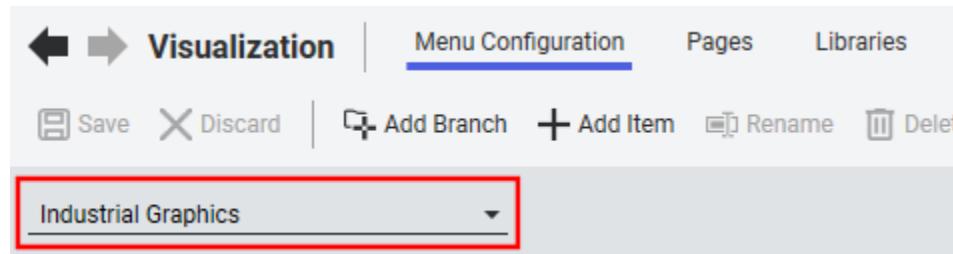
## See Also

[Menu Functions](#)

## Configure a Menu for Industrial Graphics Pages

You can use the **Menu Configuration** view in Plant SCADA Studio's **Visualization** activity to configure a navigation menu for the Industrial Graphics pages delivered to web-based clients via the Web Server.

To do this, firstly select **Industrial Graphics** from the drop-down menu below the command bar.



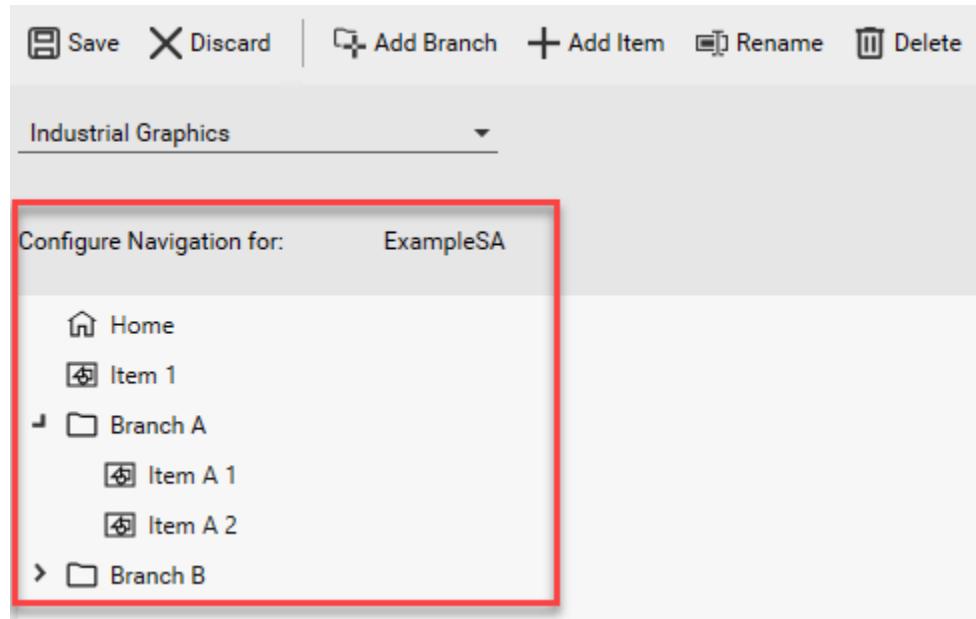
**Note:** If you want to configure a menu for standard project, see [Configure a Standard Plant SCADA Menu](#).

You can create one Industrial Graphics menu for each Plant SCADA project. At runtime, the menu configured in the primary active project will be applied to the Web Server; any menus configured in an included project will not be used.

A menu can include the following:

 Home	Home represents the top-level of the menu tree. It cannot be moved, renamed or deleted.  You can link to a page at this level of the tree. The page you choose will display when the Web Client is launched at runtime.
 Branch	A branch creates the entry point for a new level within the menu tree. You can: <ul style="list-style-type: none"><li>• Add items to a branch</li><li>• Add additional branches within a branch (up to 10 levels).</li></ul> You cannot link a branch to a page.
 Item	Items provide the links to the Industrial Graphics pages within your project. The name you give to an item should represent the page that will display when it is selected at runtime.

To build a menu, use the tree to the left of the Menu Configuration view.



If a menu item has a page linked to it, the page will display in the panel to the right of the tree when the item is selected.

---

**Note:** If the name of an item appears as red italic text, it means the linked page cannot be found. Confirm that the page location can still be accessed, and that the page has not been moved, renamed or deleted.

---

To build a menu, perform the following tasks.

#### Add a Branch

1. Select the location in the tree where you would like to add a new branch.

This can be within the top level of the tree, or within an existing branch.

2. Select the **Add Branch** button on the command bar.



3. Enter a name for the branch.

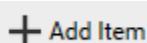
**Note:** You can only add up to 10 branch levels within a menu. If you reach this limit, the **Add Branch** button will be disabled.

## Add an Item

1. Select the location in the tree where you would like to add a new item.

This can be within the top level of the tree, or within a branch.

2. Select the **Add Item** button on the command bar.



3. Enter a name for the item.

## Link to a Page

1. Select the item you would like to link to a page.

Or:

Select **Home** to link to the page that you want to display when runtime is launched.

2. Click the **Select Page...** button.



Or:

Click the **Select Page...** button in the Properties panel.



The Select a Page dialog will appear.

3. Use the drop-down menu to select a Industrial Graphics page from the active project.
4. Click **OK**.

The selected page will display to the right of the menu tree.

## Rename a Branch or Item

1. Select the branch or item you want to rename.
2. Select the **Rename** button on the command bar.

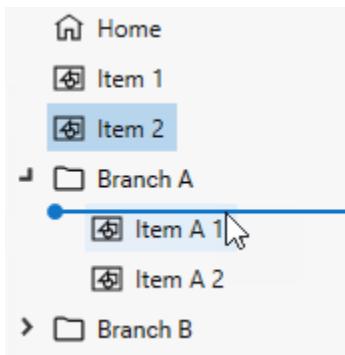


3. Enter a new name.
4. Press the **Delete** key.

### Move a Branch or Item

1. Select the branch or item you want to move to a different location in the tree.
2. Drag the branch or item to the new location.

A blue line will indicate the currently selected destination.



3. Release the mouse key.

### Delete a Branch or Item

1. Select the branch or item you want to delete.  
If you select a branch, be aware that everything within the branch will also be deleted.
2. Select the **Delete** button on the command bar.



Or:

Press the **Delete** key.

## See Also

[Browse Industrial Graphics Pages in Plant SCADA Studio](#)

[Add an Industrial Graphics Page in Plant SCADA Studio](#)

## Pages

Graphics pages create the visual interface an operator uses to monitor and control a runtime system. They are typically designed to reflect a particular area or process within a facility.

Plant SCADA supports **Standard Pages** and **Industrial Graphics Pages**.

### Standard Pages

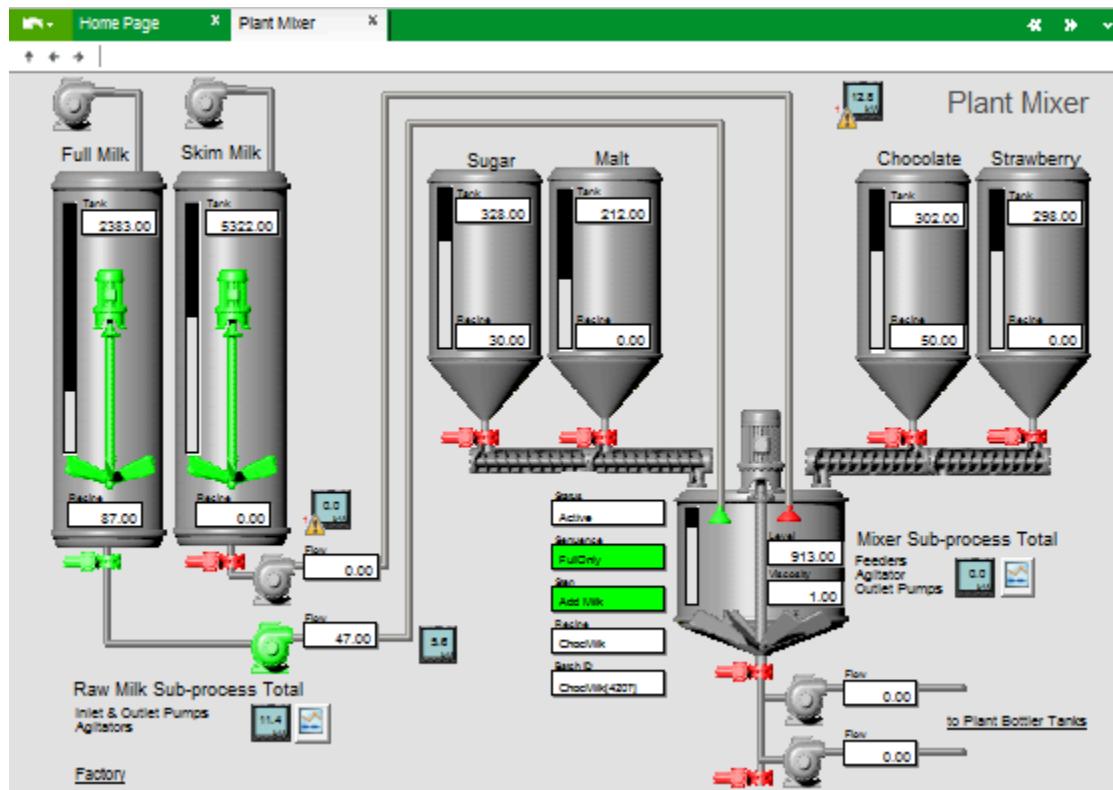
Standard pages are designed to run on Plant SCADA display clients, typically running on network computers.

To browse the standard pages associated with the active project, use the **Pages** view in the **Visualization** activity (see [Browse Pages in Plant SCADA Studio](#)). You can also use the Property Grid to view and edit some of the properties for any selected pages (see [View and Edit Page Properties in Plant SCADA Studio](#)).

## Examples

- **StruxureWare (SxW) Templates Project**

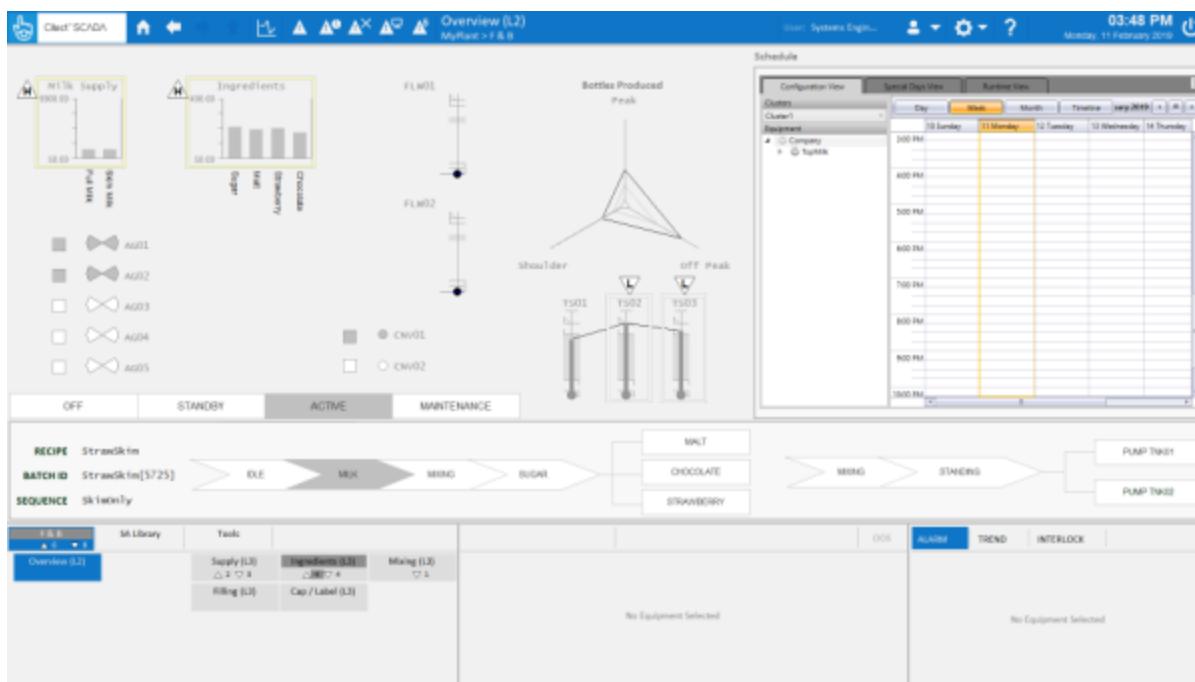
The image below shows a graphics page that represents a plant mixer based on an SxW template. This page is available to view in the Plant SCADA "Example" project.



- **Situational Awareness Starter Project**

If a project is based on the Situational Awareness Starter Project, its pages will be designed to operate within an architecture that enables the runtime environment to respond to contextual changes. A project will include master pages, and pages designed to operate within the panes of a master page (see [Key Components of a Situational Awareness Project](#)).

For example, the image below shows an overview of a production line, constructed from a set pages hosted within the panes of a master page. This page is available to view in the Plant SCADA "ExampleSA" project.



To facilitate contextual updates, pages in a Situational Awareness project may be assigned a [Content Types](#).

To configure standard graphics pages, you need to use [Graphics Builder](#). This is the tool required to create a page, and add the content that makes it reflect the area or process it will be used to monitor.

## Industrial Graphics Pages

Industrial Graphics pages are designed for delivery to web-based clients via the AVEVA Web Client. They are created and edited using AVEVA Industrial Graphics Editor.

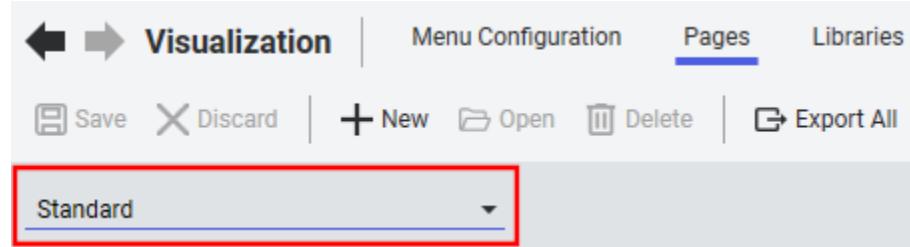
To browse the Industrial Graphics pages associated with the active project, use the **Pages** view in the **Visualization** activity (see [Browse Industrial Graphics Pages in Plant SCADA Studio](#)).

## See Also

[Menu Configuration](#)

## Browse Pages in Plant SCADA Studio

Use the **Pages** view in Plant SCADA Studio's **Visualization** activity to browse the pages associated with the active project and its included projects. To do this, select **Standard** from the drop-down menu below the command bar.

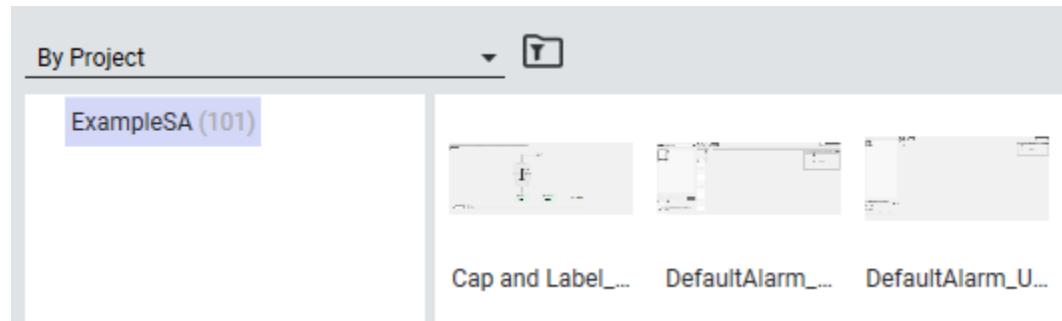


**Note:** If you want to browse Industrial Graphics pages, see [Browse Industrial Graphics Pages in Plant SCADA](#)

Studio.

You can also use the Property Grid to view and edit some of the properties for any selected pages (see [View and Edit Page Properties in Plant SCADA Studio](#)).

The tree to the left of the view shows the projects that are currently available to browse. By default, the active project is selected. When you select a project, the pages it includes are displayed in the pages list to the right.



The number to the right of a project name indicates how many pages are included in the project.

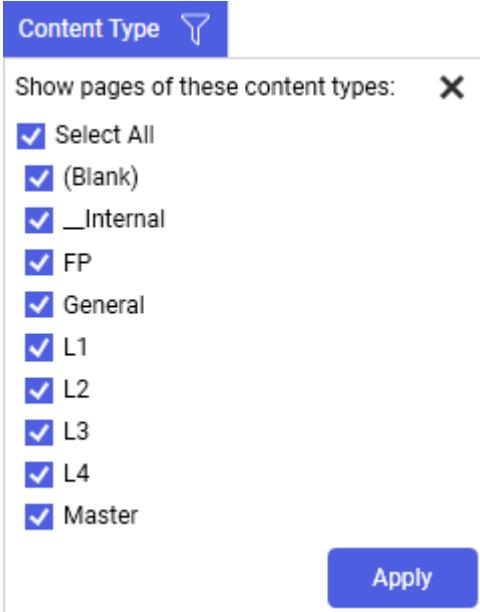
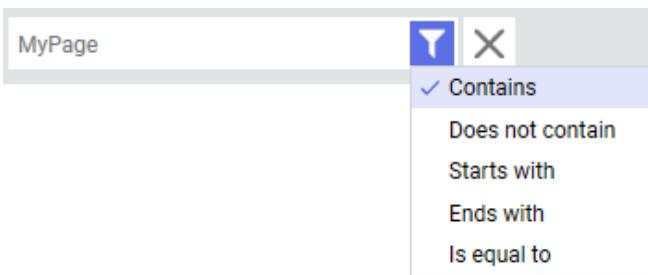
The drop-down menu above the tree allows you to select **All Items**. This option displays the pages from all the projects that are currently available to browse, and hides the project tree.

To view the pages in any included system projects, select the **Show or Hide System Projects** button.

Button	Description
	<p><b>Show or Hide System Projects</b></p> <p>When the <b>Show or Hide System Projects</b> button is selected, any system projects that are included in the active project are added to the Pages view.</p> <p>If the <b>By Project</b> option is selected for the project tree, you can individually select a system project and view a list of the pages included within it.</p> <p>If the <b>All Items</b> option is selected, the system project pages are merged into the list of pages and displayed according to the current sort order.</p>

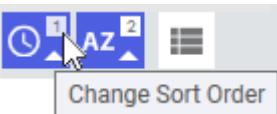
You can use the following filters to manipulate the pages list.

Tool	Description
	<p><b>Content Type Filter</b></p> <p>Allows you to filter the current list of pages by <a href="#">Content Types</a>.</p>

Tool	Description
	<p>The drop-down menu includes a list of the content types that are configured in the current project and its included projects. It also includes a "(Blank)" option for pages that do not have a content type assigned.</p>  <p>Select the content types you want to see represented in the list of pages, and click <b>APPLY</b>. At least one content type needs to be selected to apply a filter.</p> <p>Click on the <b>Select All</b> check box to select every content type.</p> <p>If every content type is currently selected, click <b>Select All</b> to clear the selections.</p>
<input type="text" value="Search items"/>	<p><b>Search Items</b></p> <p>Use the <b>Search Items</b> field to locate specific pages within the current list based on the page names.</p> <p>Click on the filter icon to the right of the field to select a comparison option.</p>  <p>To clear the current search and any active filters, click on the <b>Clear Filters</b> button (see below).</p>

Tool	Description
	<p><b>Clear Filters</b></p> <p>The <b>Clear Filters</b> button removes all the active filters that are currently applied to the list of pages.</p>
	<p><b>Display Mode</b></p> <p>The Display Mode button changes the layout of the pages list. Four options are available. The appearance of the button indicates which mode is currently selected.</p> <ul style="list-style-type: none"> <li> Small tiles (default)</li> <li> Medium tiles</li> <li> Large tiles</li> <li> Details</li> </ul> <p>The <b>Details</b> mode presents the list of pages in a tabular format with the following columns:</p> <ul style="list-style-type: none"> <li>• Name</li> <li>• Title</li> <li>• Content Type</li> <li>• Template</li> <li>• Parent Page</li> <li>• Previous Page</li> <li>• Next Page</li> <li>• Width</li> <li>• Height</li> <li>• Modified Time</li> <li>• Project</li> </ul> <p>For a description of these properties, see <a href="#">View and Edit Page Properties in Plant SCADA Studio</a>.</p>
	<p><b>Sort by Modified Time</b></p> <p>The <b>Sort by Modified Time</b> button sorts the page list according to the time each page was last modified. When this option is selected, an arrow will appear next to the clock icon to indicate if the order of the list</p>

Tool	Description
	<p>is ascending or descending.</p>  Ascending  Descending (default) <p>Clicking on the button while it is selected will toggle between these two options.</p> <p>When the <b>Sort By Modified Time</b> button is clicked, the <b>Sort By Name</b> option is turned off by default. However, you can combine the two together (see <b>Combined Sort Mode</b> below).</p>
	<p><b>Sort by Name</b></p> <p>The <b>Sort by Name</b> button sorts the page list according to page names. When this sort option is selected, an arrow will appear next to the AZ icon to indicate if the order of the list is ascending or descending</p>  Ascending (default)  Descending <p>Clicking on the button while it is selected will toggle between these two options.</p> <p>When the <b>Sort By Name</b> button is clicked, the <b>Sort By Modified Time</b> option is turned off by default. However, you can combine the two together (see <b>Combined Sort Mode</b> below).</p>
	<p><b>Combined Sort Mode</b></p> <p>You can combine the <b>Sort By Modified Time</b> and <b>Sort by Name</b> options. This sorts a list according to the primary option, then by the secondary option.</p> <p>To create a combined sort:</p> <ol style="list-style-type: none"> <li>1. Select the primary sort option, for example, <b>Sort By Modified Time</b>.  </li> <li>2. Then hold down the <b>Shift</b> key and select the <b>Sort By Name</b> button.  </li> </ol> <p>Both buttons will be highlighted to indicate that</p>

Tool	Description
	<p>they are active.</p>  <p>The numbers in the top corner of each button indicates which sort is primary (1) and which is secondary (2).</p> <p>To change the sort order, click on either number.</p>  <p>You can also change the sort direction of a particular button by clicking on it as normal.</p> <p>To remove a sort option, hold down the <b>Shift</b> key and click on the button you want to remove.</p>

If you hold the mouse over a thumbnail in the pages list or tile view, a popup will appear showing the large tile version of the thumbnail, the page name and its source project.



You can use the Pages view to perform the following tasks.

### Open pages

You can use the **Pages** view to open up to 20 selected pages in Graphics Builder. To do this:

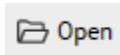
1. Locate the page you want to open in the pages list.
2. Double-click on the page.

Or:

Select up to 20 pages, then press the Enter key.

Or:

Select up to 20 pages, then click **Open** on the Command Bar.

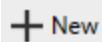


The selected page(s) will open for editing in Graphics Builder.

### Create a new page

You can use the **Pages** view to open a new page in Graphics Builder. To do this:

1. On the Command Bar, click **New**.



Graphics Builder will launch with **Use Template** dialog displayed.

For more information, see [Create a Graphics Page](#).

## Delete pages

You can use the **Pages** view to delete up to 500 selected pages. To do this:

1. Select the page(s) you want to delete in the pages list.
2. Press the **Delete** key.  
Or:
3. On the Command Bar, click **Delete**.



## Export data from the current view

You can export the data for the current view as a CSV file. To do this, click **Export All** on the Command Bar.



For more information, see [Export to Microsoft™ Excel](#).

## Update pages

You can use the **Pages** view to update the pages in a project. This will update all pages and library graphics in the active project and its included projects. For more information, see [Update Pages](#).

To update pages, click **Update Pages** on the Command Bar.



This option is not available if an Update Pages operation is already in progress.

## See Also

[Configure Graphics Pages](#)

## View and Edit Page Properties in Plant SCADA Studio

The **Pages** view in Plant SCADA Studio's **Visualization** activity displays the properties for selected pages in the Property Grid. See below for a description of the included properties.

You can also use the Property Grid to edit the following properties:

- Window Title
- Description
- Content Type
- Parent Page
- Previous Page

- Next Page
- Area
- Log Device.

Any other properties need to be edited directly in Graphics Builder (see [Edit the Properties for a Page](#)).

**To edit the properties for a page in Plant SCADA Studio's Property Grid:**

1. Go to the **Pages** view in the **Visualization** activity.
2. Select the required page(s). See [Browse Pages in Plant SCADA Studio](#).
3. In the Property Grid, make the required changes to the editable properties.  
If multiple pages are selected, your changes will be applied to every selected page.
4. On the Command Bar, click **Save**.

## Page Properties

### General Properties

Field	Description
<b>Name</b>	The name of the graphics page.
<b>Window Title</b>	The title that is used for the page at runtime. You can edit the title (using up to 64 characters). It can include plain text, Super Genie syntax or both (for example, "Pump ?AREA? status").
<b>Description</b>	A description of the page. You can edit the description (using up to 250 characters). The description is intended for configuration comments only. It is not used at runtime.
<b>Content Type</b>	Identifies a page as a particular type of content. For example, you can indicate if a page is a level 1 page (L1), a level 2 page (L2), or a faceplate (FP). This determines which panes will display the page at runtime when an associated piece of equipment comes in to context (see <a href="#">Content Types</a> ). Select the required content type from the drop-down menu to the right of the field. The available content types are configured in the <b>Visualization</b> activity (see <a href="#">Configure Content Types</a> ).
<b>Parent Page</b>	The name of the page that is one level up in the project's page hierarchy. See <a href="#">Page Levels</a> . You can edit the page name (using up to 64 characters). This property is optional.

Field	Description
<b>Previous Page</b>	<p>The name of the page that precedes the current page in the runtime <a href="#">Use a Browse Sequence</a>. You can edit the page name (using up to 64 characters).</p> <p>This property is optional. If not specified, the Page Previous command will be inoperative at runtime while the page is displayed.</p>
<b>Next Page</b>	<p>The name of the page that follows the current page in the runtime <a href="#">Use a Browse Sequence</a>. You can edit the page name (using up to 64 characters).</p> <p>This property is optional. If not specified, the Page Next command will be inoperative at runtime while the page is displayed.</p>
<b>Area</b>	<p>The area to which the page belongs (1–255). Only users with access to the specified area will be able to view the page. If an area is not specified, the page is accessible to every user.</p> <p>You can edit the number. If you enter an invalid value, a compile error will be generated.</p>
<b>Page Scan Time</b>	<p>Determines how often the page will be updated at runtime. It also determines the rate of execution for the <b>While Page Shown</b> command.</p> <p>If the field shows a value, it indicates the scan rate (in milliseconds).</p> <p>If the field is blank, it indicates the page will use the default scan rate as specified by the <a href="#">[Page]ScanTime</a> parameter.</p> <p><b>Note:</b> You can set the default page scan time using the <a href="#">General Options Setup</a> page of the Setup Wizard.</p>
<b>Log Device</b>	<p>Specifies the device to which messages are logged for the page's keyboard commands. You can select a device from the drop-down menu to the right of the field, or type a device name.</p> <p><b>Note:</b> You need to include the <i>MsgLog</i> field in the format of the log device for the message to be sent.</p>
<b>Inherit Cluster</b>	<p>If this field is set to "TRUE", the page will inherit cluster context from the page or Cicode task that opened the page (for example, the <a href="#">PageDisplay</a> or <a href="#">WinNewAt</a> Cicode functions).</p> <p>If this field is blank, cluster context for the page will be determined by the <b>Default Cluster</b> field (see below).</p>

Field	Description
<b>Default Cluster</b>	The default cluster context for the page. If this field is blank it indicates that <b>Inherit Cluster</b> is set to "TRUE", or no default cluster has been specified.
<b>Modified Time</b>	Indicates when the page was last modified.

**Template Properties**

Field	Description
<b>Style</b>	The style (appearance) of template that was used to create the graphics page. For a description of the available styles, see the topic <a href="#">Page Templates</a> .
<b>Resolution</b>	The default screen resolution of the template on which the page is based: <ul style="list-style-type: none"> <li>• VGA — 640 x 480 pixels (4:3)</li> <li>• SVGA — 800 x 600 pixels (4:3)</li> <li>• XGA — 1024 x 768 pixels (4:3)</li> <li>• SXGA — 1280 x 1024 pixels (5:4)</li> <li>• HD768 — 1366 x 768 pixels (16:9)</li> <li>• HD1080 — 1920 x 1080 pixels (16:9)</li> <li>• WUXGA — 1920 x 1200 pixels (16:10)</li> <li>• UHD4K — 3840 x 2160 pixels (16:9)</li> <li>• User Defined.</li> </ul>
<b>Name</b>	The name of the template on which the page is based.
<b>Title Bar</b>	If this field is selected, it indicates the page was intended to display the Windows title bar at runtime. If this is the case, the page size will allow space to accommodate the title bar.

**Appearance Properties**

Property	Description
<b>Width</b>	The width (in pixels) of the area that the operator can view at runtime.
<b>Height</b>	The height (in pixels) of the area that the operator can view at runtime.

**Event Properties**

Field	Description
<b>On Page Entry</b>	The command that is executed when the page is initializing.
<b>On Page Exit</b>	The command that is executed when the operator exits the page.
<b>While Page Shown</b>	The command that is executed continually for the entire time that the page is open.
<b>On Page Shown</b>	The command that is executed when a page is first displayed and the objects on it have been initialized.
<b>On Activate</b>	The command that is executed when the window is activated at runtime.
<b>On Deactivate</b>	The command that is executed when the window is deactivated.

For more information, see [Page Properties - Events](#).

#### Project Properties

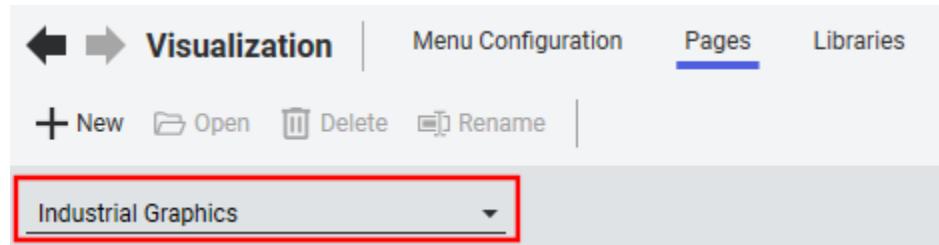
Property	Description
<b>Project</b>	The project in which the page is included.

#### See Also

[Content Types](#)

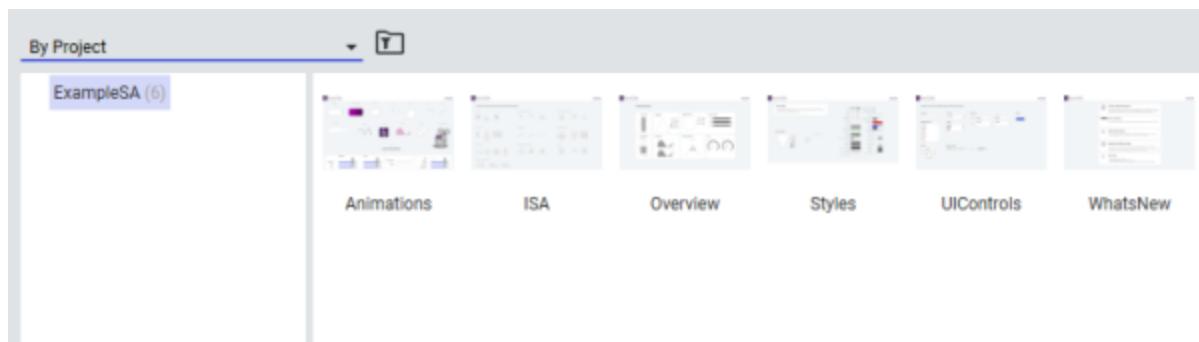
### Browse Industrial Graphics Pages in Plant SCADA Studio

You can use the **Pages** view in Plant SCADA Studio's **Visualization** activity to browse the Industrial Graphics pages associated with the active project and its included projects. To do this, select **Industrial Graphics** from the drop-down menu below the command bar.



**Note:** If you want to browse standard Plant SCADA pages, see [Browse Pages in Plant SCADA Studio](#).

The tree to the left of the view shows the projects that are currently available to browse. By default, the active project is selected. When you select a project, the Industrial Graphics pages it includes are displayed in the pages list to the right.



The number to the right of a project name indicates how many pages are included in the project.

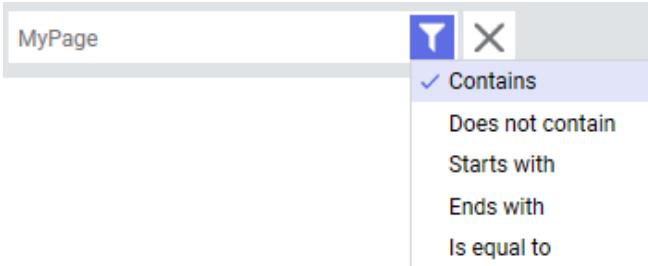
The drop-down menu above the tree allows you to select **All Items**. This option displays the pages from all the projects that are currently available to browse, and hides the project tree.

To view the pages in any included system projects, select the **Show or Hide System Projects** button.

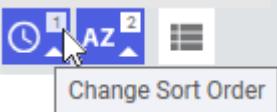
Button	Description
	<p><b>Show or Hide System Projects</b></p> <p>When the <b>Show or Hide System Projects</b> button is selected, any system projects that are included in the active project are added to the Pages view.</p> <p>If the <b>By Project</b> option is selected for the project tree, you can individually select a system project and view a list of the pages included within it.</p> <p>If the <b>All Items</b> option is selected, the Industrial Graphics pages in the system projects are merged into the list and displayed according to the current sort order.</p>

You can use the following to manipulate the pages list.

Tool	Description
<input type="text" value="Search items"/>	<p><b>Search Items</b></p> <p>Use the <b>Search Items</b> field to locate specific pages within the current list based on the page names.</p> <p>Click on the filter icon to the right of the field to select a comparison option.</p>

Tool	Description
	 <p>To clear the current search and any active filters, click on the <b>Clear Filters</b> button (see below).</p>
	<p><b>Clear Filters</b></p> <p>The <b>Clear Filters</b> button removes all the active filters that are currently applied to the list of pages.</p>
	<p><b>Display Mode</b></p> <p>The Display Mode button changes the layout of the pages list. Four options are available. The appearance of the button indicates which mode is currently selected.</p> <ul style="list-style-type: none"> <li> Small tiles (default)</li> <li> Medium tiles</li> <li> Large tiles</li> <li> Details</li> </ul> <p>The <b>Details</b> mode presents the list of pages in a tabular format.</p>
	<p><b>Sort by Modified Time</b></p> <p>The <b>Sort by Modified Time</b> button sorts the page list according to the time each page was last modified. When this option is selected, an arrow will appear next to the clock icon to indicate if the order of the list is ascending or descending.</p> <ul style="list-style-type: none"> <li> Ascending</li> <li> Descending (default)</li> </ul> <p>Clicking on the button while it is selected will toggle</p>

Tool	Description
	<p>between these two options.</p> <p>When the <b>Sort By Modified Time</b> button is clicked, the <b>Sort By Name</b> option is turned off by default. However, you can combine the two together (see <b>Combined Sort Mode</b> below).</p>
	<p><b>Sort by Name</b></p> <p>The <b>Sort by Name</b> button sorts the page list according to page names. When this sort option is selected, an arrow will appear next to the AZ icon to indicate if the order of the list is ascending or descending</p> <p> Ascending (default)</p> <p> Descending</p> <p>Clicking on the button while it is selected will toggle between these two options.</p> <p>When the <b>Sort by Name</b> button is clicked, the <b>Sort By Modified Time</b> option is turned off by default. However, you can combine the two together (see <b>Combined Sort Mode</b> below).</p>
	<p><b>Combined Sort Mode</b></p> <p>You can combine the <b>Sort By Modified Time</b> and <b>Sort by Name</b> options. This sorts a list according to the primary option, then by the secondary option.</p> <p>To create a combined sort:</p> <ol style="list-style-type: none"> <li>1. Select the primary sort option, for example, <b>Sort By Modified Time</b>. </li> <li>2. Then hold down the <b>Shift</b> key and select the <b>Sort By Name</b> button. </li> </ol> <p>Both buttons will be highlighted to indicate that they are active.</p> <p></p> <p>The numbers in the top corner of each button indicates which sort is primary (1) and which is secondary (2).</p>

Tool	Description
	<p>To change the sort order, click on either number.</p>  <p>You can also change the sort direction of a particular button by clicking on it as normal.</p> <p>To remove a sort option, hold down the <b>Shift</b> key and click on the button you want to remove.</p>

If you hold the mouse over a thumbnail in the pages list or tile view, a pop up will appear showing the large tile version of the thumbnail, the page name and its source project.



You can use the Industrial Graphics view to perform the following tasks.

### Open Industrial Graphics pages

You can select a maximum of 20 Industrial Graphics pages each time you use the open command. Each item will open in its own instance of the AVEVA Industrial Graphics Editor. To do this:

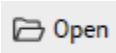
1. Locate the page you want to open in the pages list.
2. Double-click on the page.

Or:

Select up to 20 pages, then press the Enter key.

Or:

Select up to 20 pages, then click **Open** on the Command Bar.



The selected page(s) will open for editing in Industrial Graphics Editor.

### Create a new Industrial Graphics page

You can create a new page in the AVEVA Industrial Graphics Editor. To do this:

1. On the Command Bar, click **New**.



For further instructions, see [Add an Industrial Graphics Page in Plant SCADA Studio](#).

### Delete Industrial Graphics pages

You can use the **Pages** view to delete up to 500 Industrial Graphics pages. To do this:

1. Select the page(s) you want to delete in the pages list.
2. Press the **Delete** key.

Or:

On the Command Bar, click **Delete**.



The **Delete** dialog appears.

3. Click **DELETE**. If you do not want to delete the page(s), click **CANCEL**.

### Rename an Industrial Graphics page

You can use the **Pages** view to rename an Industrial Graphics page. To do this:

1. Select the page you want to rename from the pages item list.
2. On the Command Bar, click **Rename**.



The **Rename Page** dialog appears.

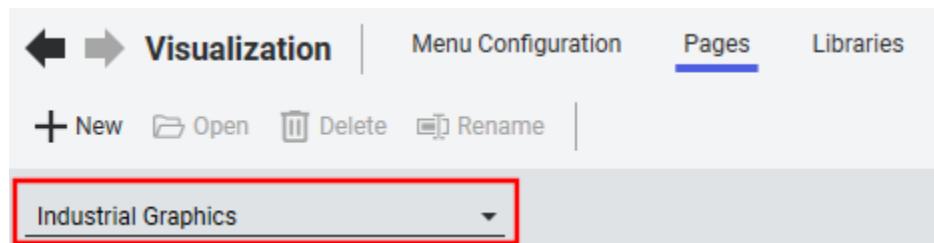
3. Use the page naming rules to update the page name (see [Add an Industrial Graphics Page in Plant SCADA Studio](#)).
4. Click **Rename**. If you do not want to rename the page, click **CANCEL**.

### Add an Industrial Graphics Page in Plant SCADA Studio

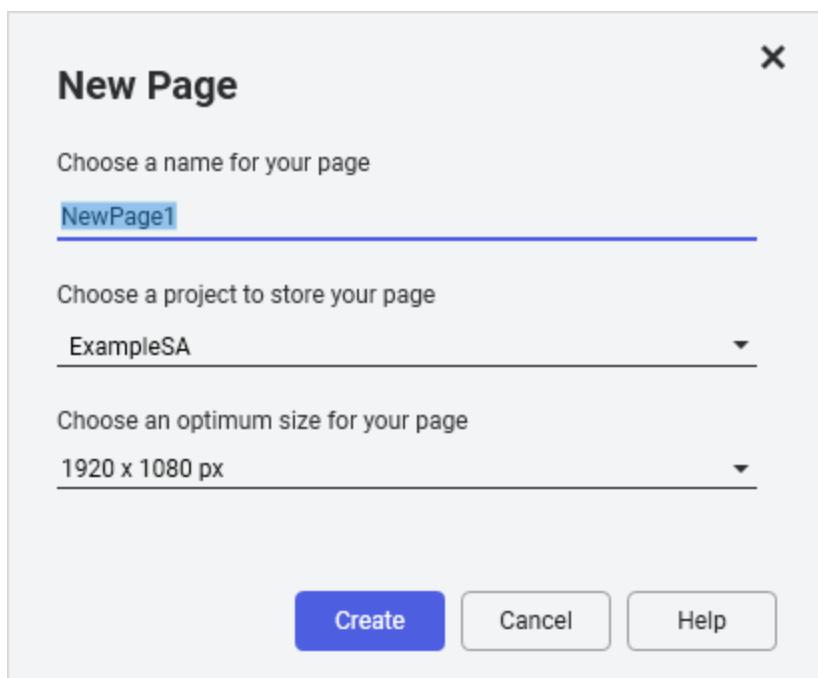
You can use the **Pages** view in Plant SCADA Studio's **Visualization** activity to add new Industrial Graphics pages to the active project.

#### To add a new Industrial Graphics page:

1. Go to the **Pages** view in the **Visualization** activity.
2. Select **Industrial Graphics** from the drop-down menu below the command bar.



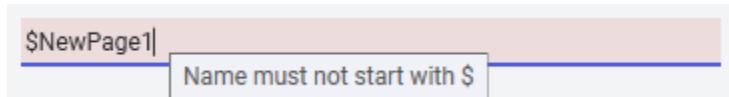
3. On the command bar, select **New**. The **New Page** dialog will appear.



4. Choose a name for the page.

The name you use needs to be unique amongst all Industrial Graphics pages and symbols in the active project and its included projects. It also needs to adhere to a set of rules (see the *Page Naming Rules* below). Based on these rules, you can only enter valid characters.

If the background of the field is highlighted, it indicates that the current name does not comply with the naming rules. Place the cursor over the field to display a tool tip with a description of the required changes.



5. Choose a project for the page from the drop-down menu. The list will include the active project, its included projects, and their respective system projects.

- The selected project from the project tree is the default selection.
- If you have not selected any project, the active project is the default selection.
- It is not recommended to select a system project, because system projects are overwritten when Plant SCADA is upgraded or reinstalled.

6. Choose an optimum size for the page from the drop-down menu, or type the width and height of the desired size in the field.

- The resolution of the selected project is used as the default selection for the page size.
- For customizing the page size, the following rules apply:
  - Min Width, Height = 0, 0
  - Max Width, Height = 7360, 4320
- The last selected / entered size will be retained as the default selection for subsequent new pages.

7. Click **CREATE**.

The page thumbnail will appear in the **Pages** view, and the page will open for editing in the Industrial Graphics Editor.

If you do not want to create the page, click **CANCEL**.

## Page Naming Rules

A page name needs to adhere to the following rules:

- Needs to be between 1 to 32 characters in length.
- Can contain any letter characters supported by Unicode.
- Can contain any decimal digit (0-9).
- Can only contain the punctuation characters # \$ and \_.
- Cannot start with a \$.
- Needs to contain at least one letter.
- Cannot be one of the following reserved words:

and	as	boolean	catch	citect
dim	discrete	do	double	each
elapsedtime	else	elseif	endif	endtry
endwhile	exit	float	for	if
in	indirect	integer	Me	message
mod	MyArea	MyContainer	MyEngine	MyHost
MyPlatform	MyViewApp	new	next	not
object	or	real	scada	shl
shr	step	string	System	then
time	to	try	while	

## See Also

[Browse Industrial Graphics Pages in Plant SCADA Studio](#)

## Libraries

You can use libraries to store items that you regularly reuse when creating graphics pages.

Plant SCADA supports two types of libraries:

- **Standard Libraries**

Standard libraries are designed to support projects that run on Plant SCADA display clients.

The items in a standard library can include:

- Symbols
- Genies
- Super Genies
- Page Templates.

The content of a library can be distributed across a project and its included projects. If you use a Starter Project to create a new project, a set of libraries are available for use in the included projects.

You can add an item to a library when it is created and saved in [Graphics Builder](#).

To browse the content of the libraries associated with the active project, use the **Libraries** view in the **Visualization** activity (see [Browse Libraries in Plant SCADA Studio](#)).

- **Industrial Graphics Libraries**

Industrial Graphics symbols are designed for delivery to web-based clients via the AVEVA Web Server. They are created and edited using AVEVA Industrial Graphics Editor.

To browse the Industrial Graphics symbols associated with the active project, use the **Libraries** view in the **Visualization** activity (see [Browse Industrial Graphics Libraries in Plant SCADA Studio](#)).

---

**Note:** When you delete an item from a library, it is only marked for deletion. To permanently delete marked items you need to "pack" your libraries. For more information, see [Pack Libraries](#).

---

## See Also

[Pages](#)

## Browse Libraries in Plant SCADA Studio

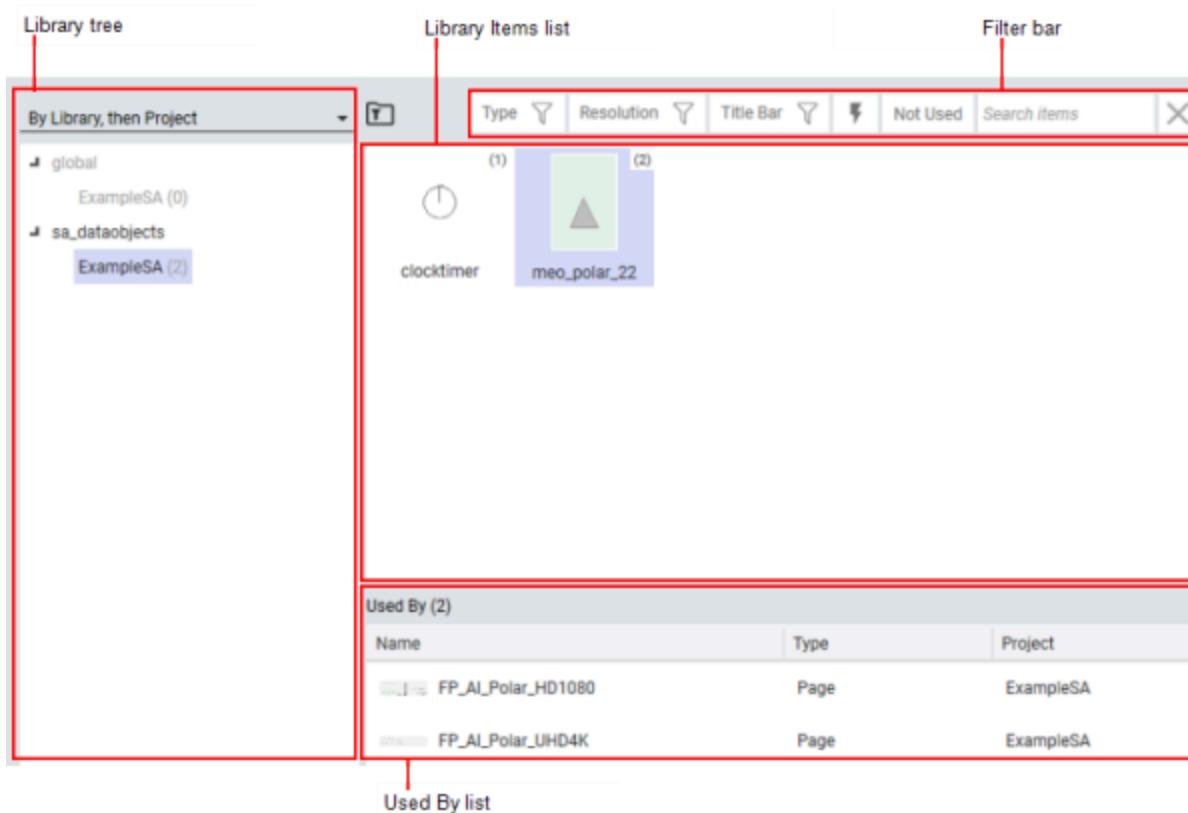
You can use the **Libraries** view in Plant SCADA Studio's **Visualization** activity to browse the content of the libraries associated with the active project and its included projects.

---

**Note:** If you want to browse Industrial Graphics libraries, see [Browse Industrial Graphics Libraries in Plant SCADA Studio](#).

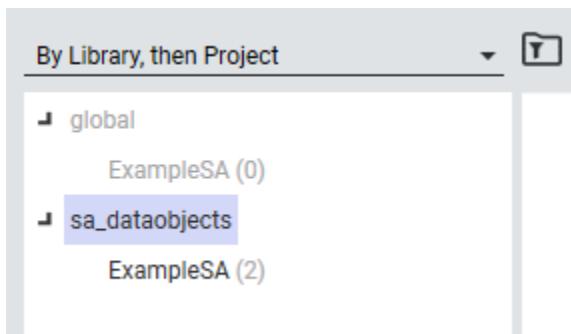
---

The Libraries view includes the following sections:



## Library Tree

The tree to the left of the view shows the projects and/or libraries that are currently available to browse.



When you select a project or library, the library items it includes are displayed in the library item list to the right. By default, the active project is selected.

The number that appears to the right of a branch indicates how many library items it includes.

By default, the tree is organized according to the setting **By Library, then Project**. The drop-down menu above the tree allows you to change this arrangement based on the following options:

- **By Library, then Project:** Libraries appear at the top level of the tree. The projects that host the library items are branched beneath. Selecting a library at the top level displays the library items from all the branched projects.
- **By Project, then Library:** Projects appear at the top level of the tree. The libraries included in each project are branched beneath. Selecting a project at the top level displays the library items from all the branched libraries.

- **By Library:** Only libraries appear in the tree. Selecting a library displays the items included in the library across all projects.
- **By Project:** Only projects appear in the tree. Selecting a project shows items from all libraries included in the project.
- **All Items:** Displays library items from all the projects that are currently available to browse. This option hides the project/library tree.

To view the libraries in any included system projects, select the **Show or Hide System Projects** button.

Button	Description
	<p><b>Show or Hide System Projects</b></p> <p>When the <b>Show or Hide System Projects</b> button is selected, any system projects that are included in the active project are added to the Libraries view.</p>  <p>If the <b>All Items</b> option is selected for the tree, the system project items are merged into the list of library items and displayed according to the current sort order.</p>

### Library Items List

This section displays the items that are included in the project or library that is currently selected in the **Library Tree**. This can include:

- Genies
- Page Templates
- Super Genies
- Symbols.

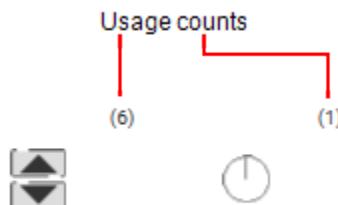
The content of the list will be dependent upon any filters that are currently applied using the Filter Bar.

The layout of the list is determined by the current selection for the **Display Mode** button (located on the Filter Bar).

Tool	Description
	<p><b>Display Mode</b></p> <p>The Display Mode button changes the layout of the item list. Four options are available. The appearance of the button indicates which mode is currently selected.</p>

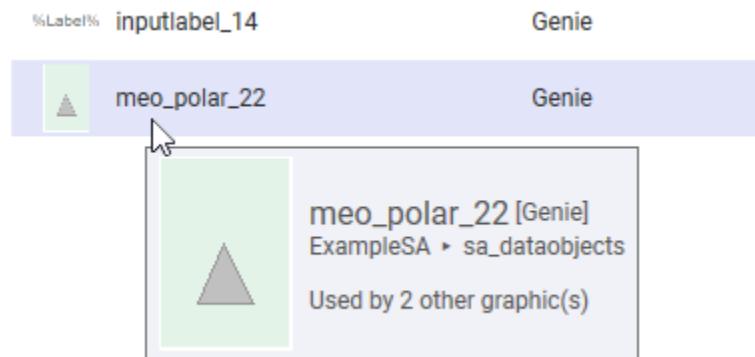
Tool	Description
	 Small tiles (default)  Medium tiles  Large tiles  Details <p>The <b>Details</b> mode present the list of library items in a tabular format with the following columns:</p> <ul style="list-style-type: none"> <li>• <b>Name</b> - Name of the library item.</li> <li>• <b>Type</b> - Genie, Super Genie, page template or symbol.</li> <li>• <b>Resolution</b> - Resolution of a page template (for example, UHD4K).</li> <li>• <b>Titlebar</b> - Indicates if a page template includes a title bar.</li> <li>• <b>Use with DspSym</b> - A check mark indicates if a Genie has DspSym enabled.</li> <li>• <b>Width</b> - Width of the item.</li> <li>• <b>Height</b> - Height of the item.</li> <li>• <b>Library</b> - Name of the library to which the item belongs.</li> <li>• <b>Modified Time</b> - The time when the last update occurred.</li> <li>• <b>Project</b> - Name of the project that hosts the item.</li> </ul>

When the library items are displayed as tiles, the number to the right of each thumbnail provides a usage count for the item. This indicates the how many pages, templates, Genies, Super Genes, or Composite Genies make use of the item.



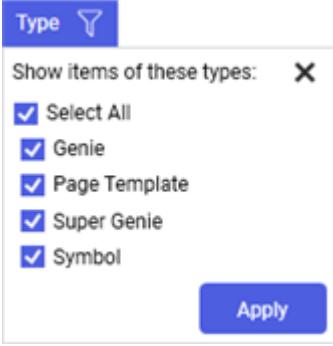
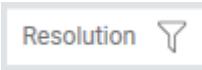
You can view details about the items a usage count represents in the **Used By** list.

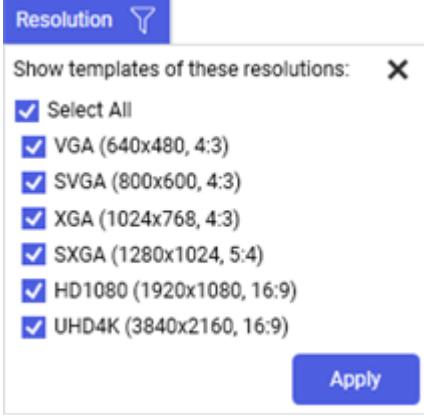
If you hold the mouse over a thumbnail in the list or tile view, a popup will appear showing the large tile version of the thumbnail, the item's name, its type, and the library and project that host it.

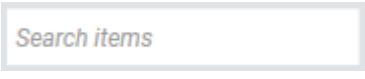
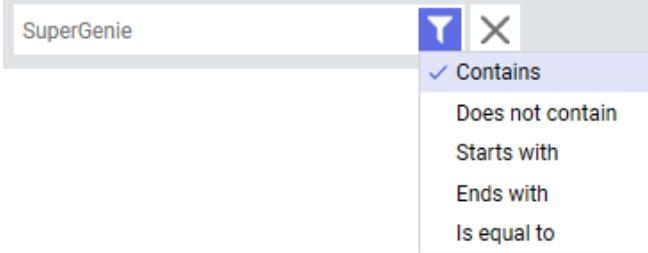


#### Filter Bar

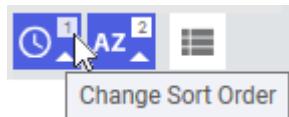
You can use the following filters to manipulate the current list of library items.

Tool	Description
	<p><b>Type Filter</b></p> <p>Allows you to filter the current list of library items according to type. The drop-down menu includes the following options:</p> <ul style="list-style-type: none"><li>• Genies</li><li>• Page Templates</li><li>• Super Genies</li><li>• Symbols.</li></ul>  <p>Show items of these types: <input type="button" value="X"/></p> <p><input checked="" type="checkbox"/> Select All <input checked="" type="checkbox"/> Genie <input checked="" type="checkbox"/> Page Template <input checked="" type="checkbox"/> Super Genie <input checked="" type="checkbox"/> Symbol</p> <p><input type="button" value="Apply"/></p> <p>Select the types you want to include in the list of library items, and click <b>APPLY</b>. At least one type needs to be selected to apply a filter. Click on the <b>Select All</b> check box to select every type. If every type is currently selected, click <b>Select All</b> to clear the selections.</p>
	<p><b>Resolution Filter</b></p> <p>Allows you to filter any page templates in the current list of library items according to resolution.</p> <p><b>Note:</b> The Resolution filter will not be available if the <b>Type</b> filter does not have <b>Page Template</b> currently selected.</p> <p>The drop-down menu includes the list of resolutions for the page templates detected in the current project and its included projects.</p>

Tool	Description
	 <p>Select the resolutions for the page templates you would like to view, and click <b>APPLY</b>. At least one resolution needs to be selected to apply a filter.</p> <p>Click on the <b>Select All</b> check box to select every resolution.</p> <p>If every resolution is currently selected, click <b>Select All</b> to clear the selections.</p>
	<p><b>Title Bar Filter</b></p> <p>Allows you to filter any page templates in the current list of library items based on whether or not they are designed to show a title bar.</p> <p><b>Note:</b> The Title Bar filter will not be available if the <b>Type</b> filter does not have <b>Page Template</b> currently selected.</p> <p>Select the type of page templates you would like to view, and click <b>APPLY</b>. At least one option needs to be selected to apply a filter.</p> <p>Click on the <b>Select All</b> check box to select all options.</p> <p>If every option is currently selected, click <b>Select All</b> to clear the selections.</p>

Tool	Description
	<p><b>DspSym Filter</b></p> <p>Allows you to filter the Genies in the current list of library items based on whether or not they are configured to work with the DspSym Cicode function (see Dynamically Instantiate a Genie at Runtime).</p> <p><b>Note:</b> The DspSym filter will not be available if the <b>Type</b> filter does not have <b>Genie</b> currently selected.</p> <p>A colored background indicates that the DspSym filter is currently selected.</p>  <p>While active, only Genies that are configured for use with DspSym are displayed in the list of library items.</p>
	<p><b>Not Used Filter</b></p> <p>Select this filter to display a list of all the items in the current project/library that have a usage count of zero. This means they are not used by any other graphic item.</p> <p>This is useful if you want to remove unused library items from your project.</p> <p><b>Note:</b> Items that have a usage count of zero may still be used indirectly through circumstances such as a menu configuration or Cicode (for example, via the DspSym function). Take care not to delete such items.</p>
	<p><b>Search Items</b></p> <p>Use the <b>Search Items</b> field to locate specific items within the current list based on the file names.</p> <p>Click on the filter icon to the right of the field to select a comparison option.</p>  <p>To clear the current search, click on the <b>Clear Filter</b> button (see below).</p>
	<p><b>Clear Filters</b></p> <p>The <b>Clear Filters</b> button removes all the active filters</p>

Tool	Description
	that are currently applied to the list of library items.
	<p><b>Sort by Modified Time</b></p> <p>The <b>Sort by Modified Time</b> button sorts the list according to the time each item was last modified. When this option is selected, an arrow will appear next to the clock icon to indicate if the order of the list is ascending or descending.</p> <p> Ascending</p> <p> Descending (default)</p> <p>Clicking on the button while it is selected will toggle between these two options.</p> <p>When the <b>Sort By Modified Time</b> button is clicked, the <b>Sort By Name</b> option is turned off by default. However, you can combine the two together (see <b>Combined Sort Mode</b> below).</p>
	<p><b>Sort by Name</b></p> <p>The <b>Sort by Name</b> button sorts the list according to item names. When this sort option is selected, an arrow will appear next to the AZ icon to indicate if the order of the list is ascending or descending.</p> <p> Ascending (default)</p> <p> Descending</p> <p>Clicking on the button while it is selected will toggle between these two options.</p> <p>When the <b>Sort By Name</b> button is clicked, the <b>Sort By Modified Time</b> option is turned off by default. However, you can combine the two together (see <b>Combined Sort Mode</b> below).</p>
	<p><b>Combined Sort Mode</b></p> <p>You can combine the <b>Sort By Modified Time</b> and <b>Sort by Name</b> options. This sorts a list according to a selected the primary option, then by the secondary option.</p> <p>To create a combined sort:</p> <ol style="list-style-type: none"> <li>1. Select the primary sort option, for example, <b>Sort By Modified Time</b>.</li> </ol>

Tool	Description
	 2. Then hold down the <b>Shift</b> key and select the <b>Sort By Name</b> button.  Both buttons will be highlighted to indicate that they are active.  The numbers in the top corner of each button indicates which sort is primary (1) and which is secondary (2). To change the sort order, click on either number.  You can also change the sort direction of a particular button by clicking on it as normal. To remove a sort option, hold down the <b>Shift</b> key and click on the button you want to remove.

### Used By List

The Used By list displays a list of items that use the item that is currently selected in the Library Items list.

Used By (2)				
Name	Type	Project	Library	Used Directly
FP_AI_Polar_HD1080	Page	ExampleSA	-	<input checked="" type="checkbox"/>
FP_AI_Polar_UHD4K	Page	ExampleSA	-	<input checked="" type="checkbox"/>

The list includes the following columns:

- **Name** — the name of the item that uses the selected library item.
- **Type** — the type of object (page, template, Genie, Super Genie or Composite Genie).
- **Project** — the project that includes the item.
- **Library** — the library that includes the item.
- **Used Directly** — indicates if the item is directly associated with the selected library item. If an item is not used directly, it means it is embedded as part of another item. For example, it may be a Genie within a Genie pasted onto a page.

You can open an item included in the Used By list by:

1. Doubling clicking within the item's row.  
Or:
2. Selecting an item and pressing the **Enter** key.

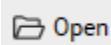
The selected item will open for editing in Graphics Builder. If the usage is a Composite Genie, the template file will open in the default XML editor.

You can use the Libraries view to perform the following tasks.

### Open the selected item(s)

You can use the **Libraries** view to open up to 20 selected items in Graphics Builder. To do this:

1. Locate an item you want to open in the items list.
2. Double-click on the item.  
Or:  
Select up to 20 items, then hit the Enter key.  
Or:  
Select up to 20 items, then click **Open** on the Command Bar.

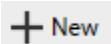


The selected item(s) will open for editing in Graphics Builder.

### Create a new item

You can use the **Libraries** view to create a new item in Graphics Builder. To do this:

1. On the Command Bar, click **New**.



2. From the drop-down menu that appears, select the type of item you want to create. The options are:
  - Genie
  - Symbol
  - Super Genie
  - Page Template

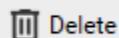
Graphics Builder will launch with a new library item of the specified type open and ready for editing. If Page Template is selected, the Use Template dialog will display so you can choose the new template's base template.

### Delete items

You can use the **Libraries** view to delete up to 500 selected items. To do this:

1. Select the item(s) you want to delete.
2. Hit the **Delete** key.  
Or:

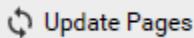
On the Command Bar, click **Delete**.



## Update pages

You can use the **Libraries** view to update the pages in a project. This will update all pages and library graphics in the active project and its included projects. For more information, see [Update Pages](#).

To update pages, click **Update Pages** on the Command Bar.



This option is not available if an Update Pages operation is already in progress.

## Pack Libraries

You can use the **Libraries** view to pack the libraries in a project and its included projects. This will delete all items marked for deletion from the various database files and re-index these files. For more information, see [Pack Libraries](#).

To pack libraries:

1. On the Command Bar, click **Pack Libraries**.



2. From the drop-down menu that appears, select which project(s) you want to include. The options are:
  - Active Project
  - Active Project and Included Projects

Graphics Builder will launch, and a notification will appear about the packing process.

3. Click **OK** to proceed.

This option will not be available if a packing operation is already in progress.

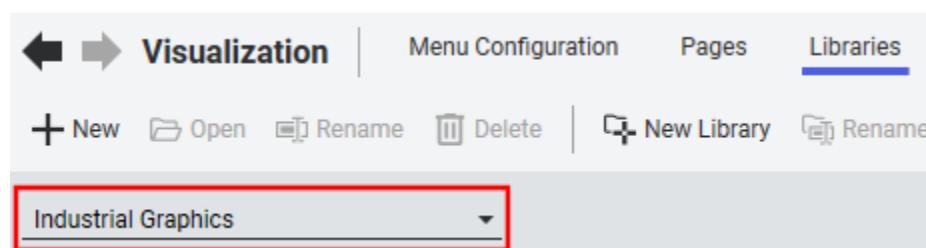
## See Also

[Libraries](#)

## Browse Industrial Graphics Libraries in Plant SCADA Studio

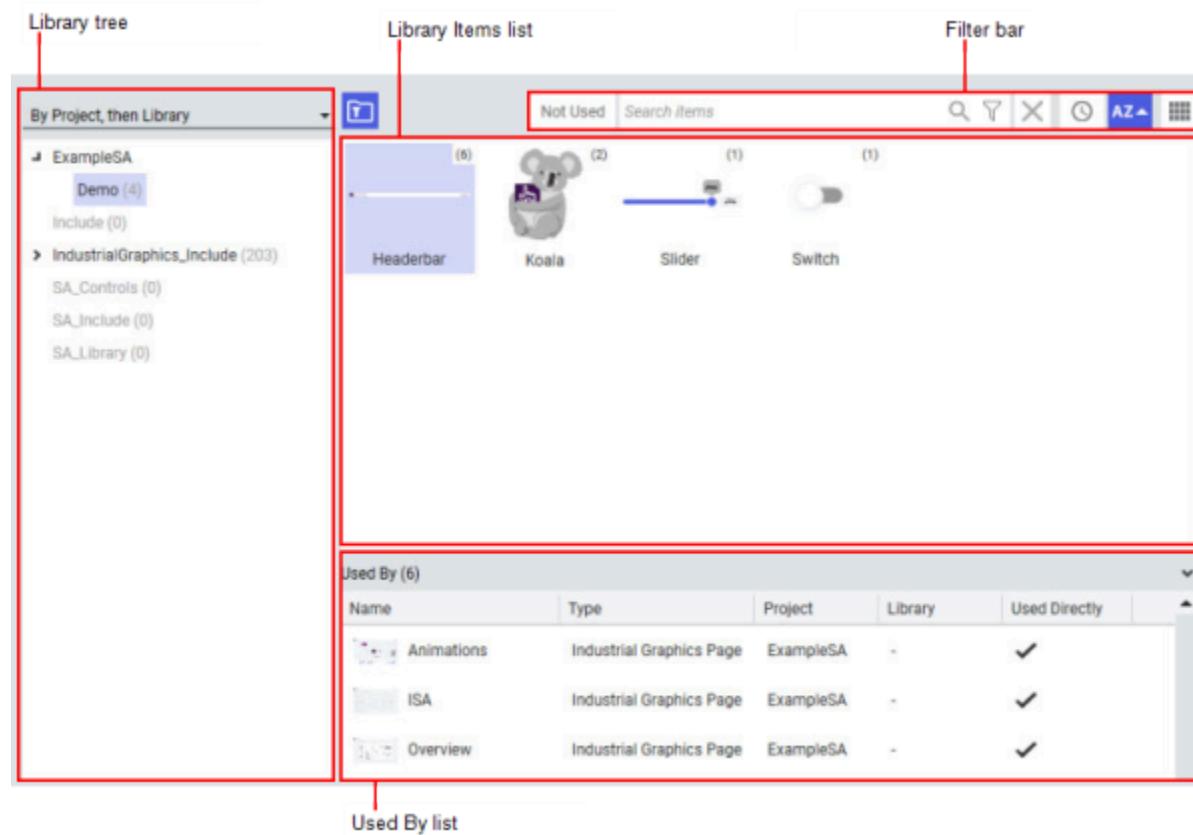
You can use the **Libraries** view in Plant SCADA Studio's **Visualization** activity to browse the Industrial Graphics symbols included in the active project and its included projects.

To do this, you need to select **Industrial Graphics** from the drop-down menu below the command bar.



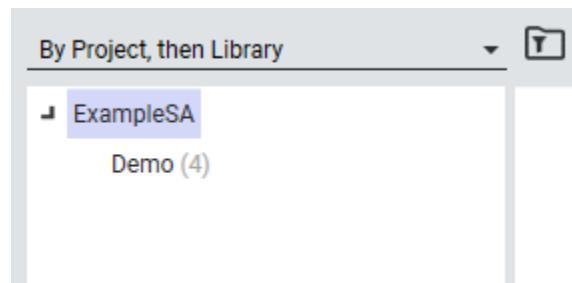
**Note:** If you want to browse standard Plant SCADA libraries, see [Browse Libraries in Plant SCADA Studio](#).

The Industrial Graphics Libraries view includes the following sections:



### Library Tree

The tree to the left of the view shows the projects and/or libraries that are currently available to browse.



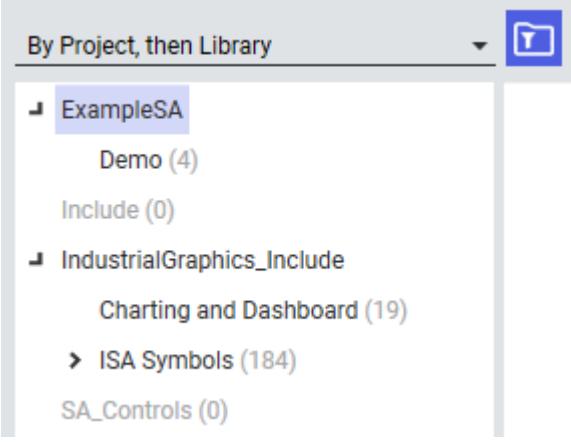
When you select a project or library, the Industrial Graphics symbols it includes are displayed in the main list. The number that appears to the right of a branch indicates how many symbols it includes.

By default, the tree is organized according to the setting **By Project, then Library**. The drop-down menu above the tree allows you to change this arrangement based on the following options:

- **By Project, then Library:** Projects appear at the top level of the tree. The libraries included in each project are branched beneath. Selecting a project at the top level displays the library items from all the branched libraries.
- **By Library:** Only libraries appear in the tree. Selecting a library displays the items included in the library across all projects.

- **By Project:** Only projects appear in the tree. Selecting a project shows items from all libraries included in the project.
- **All Items:** Displays library items from all the projects that are currently available to browse. This option hides the project/library tree.

To view the Industrial Graphics symbols in any included system projects, select the **Show or Hide System Projects** button.

Button	Description
	<p><b>Show or Hide System Projects</b></p> <p>When the <b>Show or Hide System Projects</b> button is selected, any system projects that are included in the active project are added to the tree.</p>  <p>If the <b>All Items</b> option is selected for the tree, the system project items are merged into the list of library items and displayed according to the current sort order.</p>

### Library Items List

This section displays the Industrial Graphics symbols that are included in the project or library that is currently selected in the **Library Tree**.

The content of the list will be dependent upon any filters that are currently applied using the Filter Bar.

The layout of the symbols list is determined by the current selection for the **Display Mode** button.

Button	Description
	<p><b>Display Mode</b></p> <p>The Display Mode button changes the layout of the item list. Four options are available. The appearance of the button indicates which mode is currently selected.</p> <p> Small tiles (default)</p>

Button	Description
	 Medium tiles  Large tiles  Details <p>The <b>Details</b> mode present the list of library items in a tabular format with the following columns:</p> <ul style="list-style-type: none"> <li>• <b>Name</b> - Name of the library item.</li> <li>• <b>Library</b> - Name of the library to which the item belongs.</li> <li>• <b>Project</b> - Name of the project that hosts the item.</li> </ul>

When the symbols are displayed as tiles, the number to the right of each thumbnail provides a usage count for the item. This indicates the how many pages or symbols make use of the symbol.

You can view details about the items a usage count represents in the Used By list.

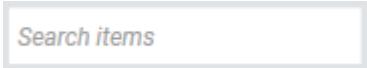
If you hold the mouse over a thumbnail in the list or tile view, a pop up will appear showing the large tile version of the thumbnail, the symbol's name, and the library and project that host it.

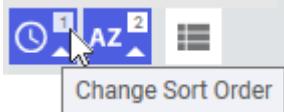


### Filter Bar

You also can use the following tools to search and sort the current list of symbols.

Tool	Description
 Not Used	<b>Not Used Filter</b> Select this filter to display a list of all the items in the current project/library that have a usage count of zero. This means they are not used by any other graphic item. This is useful if you want to remove unused library items from your project. <b>Note:</b> Symbols that have a usage count of zero may

Tool	Description
	still be used indirectly through circumstances such as a scripting.
	<p><b>Search Items</b></p> <p>Use the <b>Search Items</b> field to locate specific items within the current list based on the file names.</p> <p>Click on the filter icon to the right of the field to select a comparison option.</p>  <p>To clear the current search, click on the <b>Clear Filter</b> button (see below).</p>
	<p><b>Clear Filters</b></p> <p>The <b>Clear Filters</b> button removes any active filters that are currently applied to the list of symbols.</p>
	<p><b>Sort by Modified Time</b></p> <p>The <b>Sort by Modified Time</b> button sorts the list according to the time each item was last modified. When this option is selected, an arrow will appear next to the clock icon to indicate if the order of the list is ascending or descending.</p> <p> Ascending</p> <p> Descending (default)</p> <p>Clicking on the button while it is selected will toggle between these two options.</p> <p>When the <b>Sort By Modified Time</b> button is clicked, the <b>Sort By Name</b> option is turned off by default. However, you can combine the two together (see <b>Combined Sort Mode</b> below).</p>
	<p><b>Sort by Name</b></p> <p>The <b>Sort by Name</b> button sorts the list according to item names. When this sort option is selected, an arrow will appear next to the AZ icon to indicate if the order of the list is ascending or descending.</p>

Tool	Description
	 Ascending (default)  Descending Clicking on the button while it is selected will toggle between these two options. When the <b>Sort By Name</b> button is clicked, the <b>Sort By Modified Time</b> option is turned off by default. However, you can combine the two together (see <b>Combined Sort Mode</b> below).
 	<p><b>Combined Sort Mode</b></p> <p>You can combine the <b>Sort By Modified Time</b> and <b>Sort by Name</b> options. This sorts a list according to a selected the primary option, then by the secondary option.</p> <p>To create a combined sort:</p> <ol style="list-style-type: none"> <li>1. Select the primary sort option, for example, <b>Sort By Modified Time</b>.  </li> <li>2. Then hold down the <b>Shift</b> key and select the <b>Sort By Name</b> button.  </li> </ol> <p>Both buttons will be highlighted to indicate that they are active.</p> <p> </p> <p>The numbers in the top corner of each button indicates which sort is primary (1) and which is secondary (2).</p> <p>To change the sort order, click on either number.</p>  <p>You can also change the sort direction of a particular button by clicking on it as normal.</p> <p>To remove a sort option, hold down the <b>Shift</b> key and click on the button you want to remove.</p>

### Used By List

The Used By list displays a list of items that use the item that is currently selected in the Library Items list.

Used By (1)				
Name	Type	Project	Library	Used Directly
ISA	Industrial Graphics Page	ExampleSA	-	<input checked="" type="checkbox"/>

The list includes the following columns:

- **Name** — the name of the item that uses the selected library item.
- **Type** — the type of object.
- **Project** — the project that includes the item.
- **Library** — the library that includes the item.
- **Used Directly** — indicates if the item is directly associated with the selected library item. If an item is not used directly, it means it is embedded as part of another item. For example, it may be a symbol embedded within another symbol and pasted onto a page.

You can open an item included in the Used By list by:

1. Doubling clicking within the item's row.  
Or:
2. Selecting an item and pressing the **Enter** key.

The selected item will open for editing in AVEVA Industrial Graphics Editor.

You can use the Industrial Graphics Libraries view to perform the following tasks.

### Open the selected item(s)

You can select a maximum of 20 Industrial Graphics symbols each time you use the open command. Each item will open in its own instance of the AVEVA Industrial Graphics Editor. To do this:

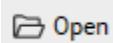
1. Locate an item you want to open in the items list.
2. Double-click on the item.

Or:

Select up to 20 items, then hit the Enter key.

Or:

Select up to 20 items, then click **Open** on the Command Bar.



The selected item(s) will open for editing in the AVEVA Industrial Graphics Editor.

### Create a new Industrial Graphics symbol

You can create a new item in the AVEVA Industrial Graphics Editor. To do this:

1. On the Command Bar, click **New**.



For further instructions, see [Add an Industrial Graphics Symbol in Plant SCADA Studio](#).

### Create a new Industrial Graphics library

You can create a new library to store your Industrial Graphics symbols. To do this:

1. On the Command Bar, click **New Library**.



For further instructions, see [Add an Industrial Graphics Library in Plant SCADA Studio](#).

### Delete Industrial Graphics symbols

You can use the **Libraries** view to delete up to 500 Industrial Graphics symbols. To do this:

1. Select the symbol(s) you want to delete from the libraries item list.
2. Press the **Delete** key.

Or:

On the Command Bar, click **Delete**.



The **Delete** dialog appears.

3. Click **DELETE**. If you do not want to delete the symbol(s), click **CANCEL**.

---

**Note:** If you delete a symbol that is open in the graphics editor, it will be closed automatically and the symbol will be deleted.

---

**Note:** If you delete a symbol that is embedded in another symbol or page, a "Graphic not Available" label appears in place of the deleted symbol.

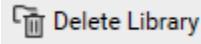
### Delete Industrial Graphics Library

You can use the **Libraries** view to delete a Industrial Graphics library. To do this:

1. Set the library tree arrangement as **By Project, then Library** to enable the **Delete Library** button.
2. Select a library you want to delete from the tree. You can select only one library at a time.
3. Press the **Delete** key.

Or:

On the Command Bar, click **Delete Library**.



The **Delete Library** dialog appears, showing the symbols available in the library, that would be deleted.

4. Click **DELETE**. If you do not want to delete the library, click **CANCEL**.

---

**Note:** If you delete a parent library, all the child libraries within the parent library will also be deleted.

## Rename an Industrial Graphics Library

You can use the **Libraries** view to rename an Industrial Graphics library. To do this:

1. Set the library tree arrangement as **By Project, then Library** to enable the **Rename Library** button.
2. Select a library you want to rename from the tree. You can select only one library at a time.
3. On the Command Bar, click **Rename Library**.



The **Rename Library** dialog appears.

4. Use the library naming rules to update the library name.
5. Click **Rename**. If you do not want to rename the library, click **CANCEL**.

## Rename Industrial Graphics Symbol

You can use the **Libraries** view to rename an Industrial Graphics symbol. To do this:

1. Select the symbol you want to rename from the libraries item list.
2. On the Command Bar, click **Rename**.



The **Rename Symbol** dialog appears.

3. Use the symbol naming rules to update the symbol name.
4. Click **Rename**. If you do not want to rename the symbol, click **CANCEL**.

## See Also

[Libraries](#)

[Add an Industrial Graphics Library in Plant SCADA Studio](#)

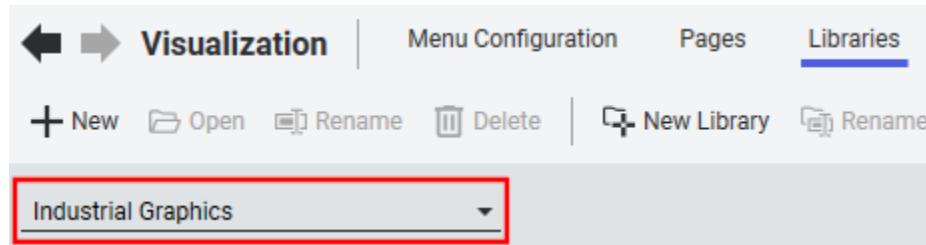
[Add an Industrial Graphics Symbol in Plant SCADA Studio](#)

## Add an Industrial Graphics Library in Plant SCADA Studio

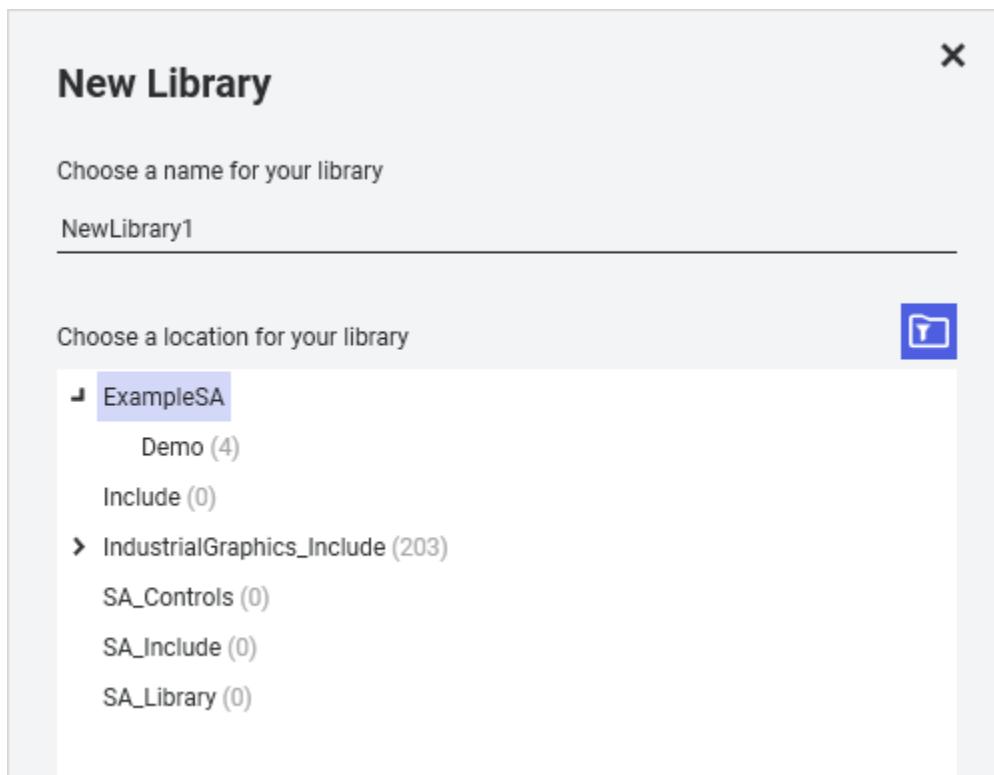
You can use the **Libraries** view in Plant SCADA Studio's **Visualization** activity to add a new Industrial Graphics library to the active project.

### To add a new Industrial Graphics library:

1. Go to the **Libraries** view in the **Visualization** activity.
2. Select **Industrial Graphics** from the drop-down menu below the command bar.



3. On the command bar, select **New Library**. The New Library dialog will appear.



**Note:** You can also launch the New Library dialog from the New Symbol dialog (see [Add an Industrial Graphics Symbol in Plant SCADA Studio](#)).

4. Choose a name for the library.

The name you use needs to be unique within the current branch of the tree. It also needs to adhere to a set of rules (see **Library Naming Rules** below). Based on these rules, you can only enter valid characters.

5. Choose a location for the library. If you select an existing library, the new library will be created as a sub-directory within it.

**Note:** You can create up to 10 library levels within the tree. If you select a location that goes beyond this limit, the Create button will be disabled.

6. Click **Create**.

The library will now appear in the tree in the **Libraries | Industrial Graphics** view.

## Library Naming Rules

A library name needs to adhere to the following rules:

- Needs to be between 1 to 64 characters in length.
- Any printable character (Alphanumeric, punctuation, space etc, but no tab, backspace, bell etc.)
- Cannot contain \$ and \.

## See Also

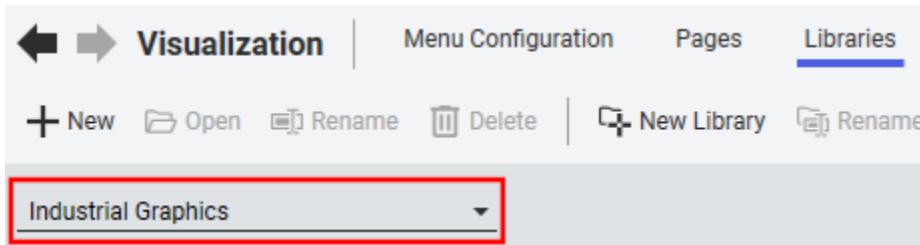
[Browse Industrial Graphics Libraries in Plant SCADA Studio](#)

## Add an Industrial Graphics Symbol in Plant SCADA Studio

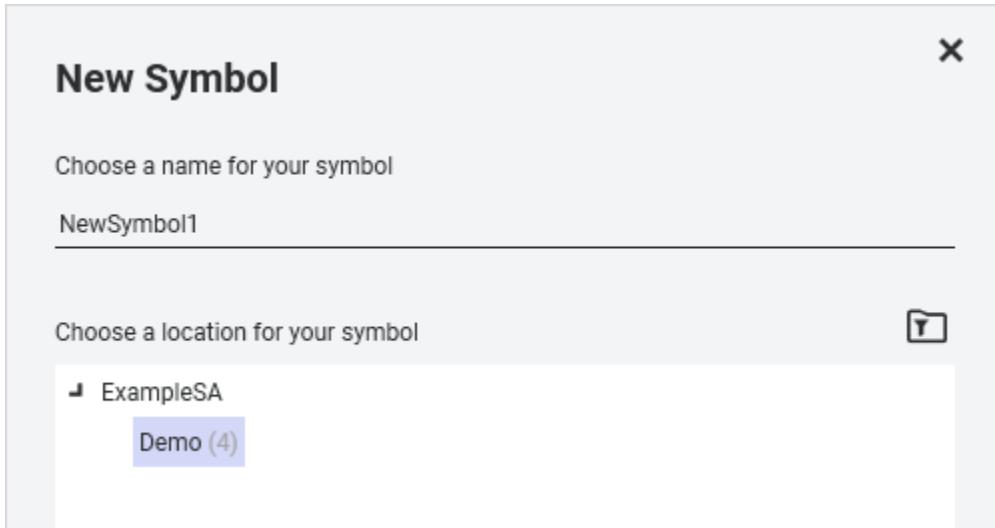
You can use the **Libraries** view in Plant SCADA Studio's **Visualization** activity to add a new Industrial Graphics symbol to the active project.

### To add a new Industrial Graphics symbol:

1. Go to the **Libraries** view in the **Visualization** activity.
2. Select **Industrial Graphics** from the drop-down menu below the command bar.



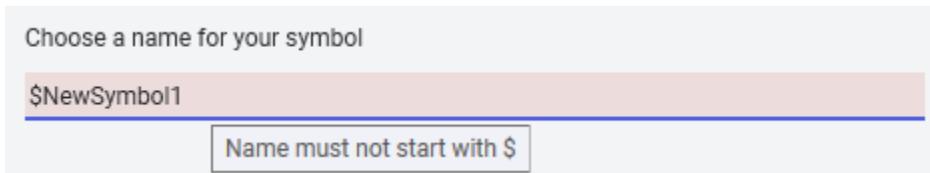
3. On the command bar, select **New**. The New Symbol dialog will appear.



4. Choose a name for the symbol.

The name you use needs to be unique within the current branch of the tree. It also needs to adhere to a set of rules (see **Symbol Naming Rules** below). Based on these rules, you can only enter valid characters.

If the background of the field is highlighted, it indicates that the current name does not comply with the naming rules. Place the cursor over the field to display a tool tip with a description of the required changes.



5. In the library tree, choose a location for the symbol.

**Note:** You cannot create a symbol in the root of a project. You need to select a library.

If required, select **New Library** (see [Add an Industrial Graphics Library in Plant SCADA Studio](#)).

6. Click **Create**.

The new symbol will appear in the tree in the specified location.

## Symbol Naming Rules

A symbol name needs to adhere to the following rules:

- Needs to be between 1 to 32 characters in length.
- Can contain any letter characters supported by Unicode.
- Can contain any decimal digit (0-9).
- Can only contain the punctuation characters # \$ and \_.
- Cannot start with a \$.
- Needs to contain at least one letter.
- Cannot be one of the following reserved words:

and	as	boolean	catch	citect
dim	discrete	do	double	each
elapsedtime	else	elseif	endif	endtry
endwhile	exit	float	for	if
in	indirect	integer	Me	message
mod	MyArea	MyContainer	MyEngine	MyHost
MyPlatform	MyViewApp	new	next	not
object	or	real	scada	shl
shr	step	string	System	then
time	to	try	while	

## See Also

[Browse Industrial Graphics Libraries in Plant SCADA Studio](#)

## Embed an Industrial Graphics Symbol

You can use AVEVA™ Industrial Graphics Editor to embed an Industrial Graphics symbol on a page or within another Industrial Graphics symbol. This allows you to create re-useable graphical components that share a common source.

### To embed a Industrial Graphics symbol:

1. In the Industrial Graphics Editor, open the Industrial Graphics page or symbol that will host an embedded symbol.

2. From the **Edit** menu, select **Embed Graphic**.

Or:

Click on the Embed Graphic icon on the command toolbar.



The Embed Graphic dialog will appear.

3. In the project/library tree to the left of the dialog, select the source location for the item you would like to embed.

The drop-down menu above the tree will change its current arrangement based on the following options:

- **By Project, then Library:** Projects appear at the top level of the tree. The libraries included in each project are branched beneath. Selecting a project at the top level displays the library items from all the branched libraries.
- **By Library:** Only libraries appear in the tree. Selecting a library displays the items included in the library across all projects.
- **By Project:** Only projects appear in the tree. Selecting a project shows items from all libraries included in the project.
- **All Items:** This option hides the project/library tree and displays all symbols that are currently available to browse.

You can also display system projects in the tree by selecting the following button.



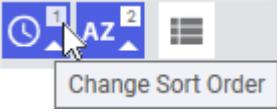
4. In the symbols list to the right of the dialog, select the item you would like to embed.

5. When you have made your selection, click on the **EMBED** button.

A cursor will appear in the Industrial Graphics Editor interface allowing you to position the embedded symbol.

The following tools may help you navigate the list of available symbols. .

Tool	Description
	<p><b>Search Items</b></p> <p>Use the <b>Search Items</b> field to locate specific items within the current list based on the file names.</p> <p>Click on the filter icon to the right of the field to select a comparison option.</p>  <p>To clear the current search, click on the <b>Clear Filter</b> button (see below).</p>
	<p><b>Clear Filters</b></p> <p>The <b>Clear Filters</b> button removes any active filters that are currently applied to the list of symbols.</p>
	<p><b>Sort by Modified Time</b></p> <p>The <b>Sort by Modified Time</b> button sorts the list according to the time each item was last modified. When this option is selected, an arrow will appear next to the clock icon to indicate if the order of the list is ascending or descending.</p> <p> Ascending</p> <p> Descending (default)</p> <p>Clicking on the button while it is selected will toggle between these two options.</p> <p>When the <b>Sort By Modified Time</b> button is clicked, the <b>Sort By Name</b> option is turned off by default. However, you can combine the two together (see <b>Combined Sort Mode</b> below).</p>
	<p><b>Sort by Name</b></p> <p>The <b>Sort by Name</b> button sorts the list according to item names. When this sort option is selected, an arrow will appear next to the AZ icon to indicate if the order of the list is ascending or descending.</p> <p> Ascending (default)</p>

Tool	Description
	 Descending Clicking on the button while it is selected will toggle between these two options. When the <b>Sort By Name</b> button is clicked, the <b>Sort By Modified Time</b> option is turned off by default. However, you can combine the two together (see <b>Combined Sort Mode</b> below).
	<b>Combined Sort Mode</b> You can combine the <b>Sort By Modified Time</b> and <b>Sort by Name</b> options. This sorts a list according to a selected the primary option, then by the secondary option. To create a combined sort: 1. Select the primary sort option, for example, <b>Sort By Modified Time</b> .  2. Then hold down the <b>Shift</b> key and select the <b>Sort By Name</b> button.  Both buttons will be highlighted to indicate that they are active.  The numbers in the top corner of each button indicates which sort is primary (1) and which is secondary (2). To change the sort order, click on either number. 
	<b>Display Mode</b> The Display Mode button changes the layout of the

Tool	Description
	<p>item list. Four options are available. The appearance of the button indicates which mode is currently selected.</p>  <ul style="list-style-type: none"><li><b>Small tiles (default)</b></li><li><b>Medium tiles</b></li><li><b>Large tiles</b></li><li><b>Details</b></li></ul> <p>The <b>Details</b> mode present the list of library items in a tabular format with the following columns:</p> <ul style="list-style-type: none"><li>• <b>Name</b> - Name of the library item.</li><li>• <b>Library</b> - Name of the library to which the item belongs.</li><li>• <b>Project</b> - Name of the project that hosts the item.</li></ul>

## See Also

[Browse Industrial Graphics Libraries in Plant SCADA Studio](#)

## Content Types

Content types allow you to identify a particular type of graphical content (such as pages or faceplates). You can use them to specify what a pane will display when a piece of equipment comes in to context. This enables the [Autofill](#) functionality in a Situational Awareness project.

For information on how to add content types to a project, see [Configure Content Types](#).

Once you have added a set of content types to a project, you can use them for the following:

- Pages — the **Content Type** property identifies a page as a particular type of page (see [Assign a Content Type to a Page](#)).
- Panes — the **ContentType** property specifies the type of content a workspace pane will display (see [Configure Panes on a Master Page](#)).

If a project was created using the Situational Awareness Starter project, the following content types will already be available via the SA\_Include project:

- L1 — Level 1 pages

- L2 — Level 2 pages
- L3 — Level 3 pages
- L4 — Level 4 pages
- FP — Faceplate pages
- General — General pages (for example, alarm or trend pages)
- \_Internal — Internal pages (any pages that are excluded by the autofill functionality and default menu).

## Configure Content Types

[Content Types](#) allow you to identify a particular type of graphical content (such as pages or faceplates). This allows you to specify what a pane will display when a piece of equipment comes into context.

### To add a content type:

1. In the **Visualization** activity, select **Content Types**.
2. Add a row to the Grid Editor.
3. Type the required information in each column, or in the fields in the Property Grid.  
For a description of the properties, see below.
4. Click **Save**.

**Note:** If your project was created using the Situational Awareness Starter project, a set of existing content types will already be available via the SA\_Include project.

## Content Types Properties

### General Properties

Property	Description
<b>Name</b>	The name of the content type. This is the name that will be used when configuring pages and panes.
<b>Comment</b>	Any useful comment, such as a description of the content type. Enter a value of 48 characters or less.

### Project Properties

Property	Description
<b>Project</b>	The project in which the report is included.

## See Also

[Autofill](#)

## Keyboard Commands

Keyboard commands allow your operators to interact with the runtime system by typing instructions on the keyboard.

You can assign privileges to commands and controls, and send a message to the command log each time an operator issues a command.

You can also define a meaningful name for any key on a keyboard, providing a label that you can use to configure your keyboard commands. See [Keyboard Keys](#).

**Note:** If you want to configure commands that an operator can issue by clicking specific objects on a graphics page, see [Touch Commands](#).

### See Also

[Define System Keyboard Commands](#)

[Define Key Sequences for Commands](#)

[Keyboards](#)

## Define System Keyboard Commands

System keyboard commands specify a key sequence that an operator can enter on a keyboard to execute a particular instruction (or series of instructions).

For information on keyboard commands that only operate for a particular page or selected object, see [Keyboard Commands](#).

To add a system keyboard command:

1. In the **Visualization** activity, select **Keyboard Commands**.
2. Add a row to the Grid Editor.
3. Type the required information in each column, or in the Property Grid (see below for a description of the properties).
4. Click **Save**.

## System Keyboard Command Properties

### General Properties

Property	Description
<b>Key Sequence</b>	The key sequence for the command (32 characters).
<b>Command</b>	The commands (set of instructions) to execute when an operator enters the key sequence (253 characters).
<b>Privilege</b>	The privilege necessary by an operator to issue the command (16 characters).

Property	Description
<b>Area</b>	The area to which the command belongs (16 characters). Only operators who belong to this area will be able to issue this command. For example, if you enter Area 1 here, operators need to have access to Area 1 (plus any necessary privileges) to issue this command.
<b>Message Log</b>	<p>A text message sent to the <i>MsgLog</i> field of the log device when the selected action is performed by the operator at runtime (32 characters).</p> <p>The log device to which the message is sent is specified in the <b>Log Device</b> field below.</p> <p>The message needs to be plain text. If you want to include field data as part of a logged message, you need to insert the field name as part of the device format when you configure the device. For instance, in the <b>Format</b> field of the <b>Devices</b> form, you could enter {MsgLog,20} {FullName,15}. This would accommodate the logging of messages such as "P2 started by John Smith".</p>
<b>Log Device</b>	The device to which the <b>Message Log</b> is sent when the command is issued (16 characters).
<b>Comment</b>	Any useful comment (48 characters).

#### Project Properties

Property	Description
<b>Project</b>	The project in which the specified keyboard command is included.

#### See Also

[Keyboard Commands](#)

[Define Key Sequences for Commands](#)

## Define Key Sequences for Commands

To define a command, you need to specify the key sequence that the operator types to issue the command. You can specify a single key for the key sequence, for example, the function key F2:

Key Sequence	F2
--------------	----

Alternatively, you can specify several keys that needs to be typed in sequence, for example, the function key F2

followed by the Enter key:

Key Sequence	F2 Enter
--------------	----------

**Note:** If you use more than one key for the sequence, you need to separate each key with a space.

You will want to avoid the ambiguity in keyboard commands that can occur if you define separate commands that use a common key. For example, if you define a key sequence for one command as F3, and the key sequence for a second command as F3 F4, then when F3 is pressed, the first command would execute immediately; the second command could not execute. To avoid overlapping keyboard commands, add a **delimiter** to common keyboard commands, for example:

Key Sequence	F3 Enter
Command	SP1 = 50;
Key Sequence	F3 F4 Enter
Command	SP1 = 100;

These commands will not execute until the operator types the delimiter (the Enter key).

## See Also

[Using a Hot Key](#)

[Using Variable Data Input](#)

[Passing Multiple Arguments](#)

[Passing Keyboard Arguments to Functions](#)

## Keyboards

You can use non-standard keyboards (as well as multiple keyboards) to control Plant SCADA. You can also define key names.

### Using non-standard keyboards

You can use many types of keyboards with your runtime system. The majority of keyboards are IBM compatible keyboards; this type of keyboard is used as a default. Some industrial keyboards may not conform with this standard; if you use a non-standard keyboard, you need to define each of the keyboards in the database.

### Using multiple keyboards

Sometimes you might need to use keyboards of different types; for example, an IBM compatible keyboard in your control room and a sealed-membrane keyboard on the plant floor. If you use more than one type of keyboard, you need to define every key for each keyboard and assign each keyboard type to the respective computer.

You need to set the keyboard type for each computer, with the [Keyboard]Type parameter.

**Note:** If you define commands that use mouse buttons, test that a double-click command cannot be mistaken for

---

a single-click command. A double-click is an extension of a single click; a single click message will be received before a double-click message.

---

## Defining key names

You can refer to a keyboard key by a meaningful name, rather than by the key itself. For example, you can refer to the F1 key as the "Help" key and the F2 key as the "Login" key. When you use the key in a command, you can use the name you have defined.

You can assign a key name to any key on the keyboard (including the alphanumeric keys A - Z, a -z, and 0 - 9). However, after a key is defined as a command key it can only be used as a command key. If you do assign a definition to an alphanumeric key (for example the character A), that key can not be used as a data key. Each time it is pressed, the definition for the key is recognized and not the keyboard character itself. Keyboard key definitions are usually only used with non-alphanumeric characters (for example the function keys).

## See Also

[Keyboard Commands](#)

[Define Key Sequences for Commands](#)

## Using a Hot Key

A command defined with a 'hot' key executes immediately the key is pressed. You can only define a single key in the key sequence, and only use a 'hot' key to define commands that act on the current keyboard buffer (for example the Backspace and Delete keys). To define a 'hot' key, prefix the key sequence with an asterisk (\*) as in the following example:

Key Sequence	*Backspace
Command	KeyBs()

At run time, this command operates in exactly the same way as the Backspace key on a standard computer keyboard. (Each time the Backspace key is pressed, the last key in the command line is removed.)

---

**Note:** You can only define a 'hot' key command as a system (global) command.

---

## See Also

[Using Variable Data Input](#)

## Using Variable Data Input

You can configure a keyboard command to accept variable data at run time. When the system is running, an operator can enter a value with the command, and the value is passed as an argument (or arguments) into the Cicode command. You can therefore define a single keyboard command that your operators use with different values. For example, you could define the following command to set the variable SP1 at run time:

Key Sequence	F3 ### Enter
Command	SP1 = Arg1;

In this example, an operator can set the variable SP1 to a new value by first pressing the F3 function key, entering the new value, and pressing the Enter key, for example:



This sets the value of SP1 to 10.

Each `#' character (in the Key Sequence) represents one keyboard character that an operator can enter in the command. In the above example, the operator can enter up to three keyboard characters when issuing the command. (The number of # characters determines the maximum number of characters that an operator can enter for the argument; if the operator enters more than three characters, an "Invalid Command" alert message displays.)

The command in the above example could be issued as follows:



or



or



When the command is issued (the operator presses the Enter key), the value is passed to the command at Arg1.

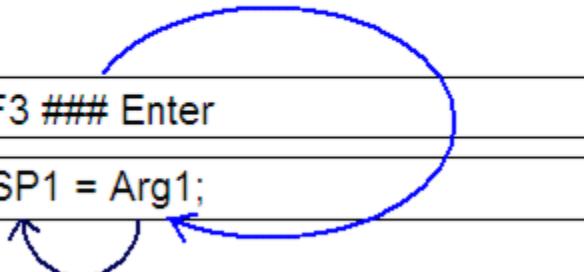
When an operator issues  
the command, the value  
is passed to Arg1 . . .

Key Sequence

F3 ### Enter

Command

SP1 = Arg1;



. . . and SP1 is set to that value.

**Note:** If an operator does not enter any data, the value of Arg1 is zero, and the variable is set to zero.

For example, the following key sequence may be used:



To avoid this, use the **ArgValue1** label, for example:

```
Command SP1 = ArgValue1;
```

The **ArgValue1** label checks for invalid input; if the input is invalid, the value of the variable is not changed and the command halts. Any instructions following an invalid ArgValue1 will not execute. You can also use the `StrToValue()` function.

## See Also

[Passing Multiple Arguments](#)

### Passing Multiple Arguments

You can pass multiple arguments into a Cicode command by separating each argument with a comma (,). Each argument is passed into the command in sequence, and is referred to in the command field as **Arg1**, **Arg2**, **Arg3**, and so on. For example:

Key Sequence	F3 ###, ### Enter
Command	SP1 = Arg1; SP2 = Arg2;

In this case, an operator can set two variables with the same command, for example:



sets the variables SP1 to 20 and SP2 to 35.

You can use up to eight arguments. However, avoid passing many arguments.

**Note:** There is no way to check if the input for each argument is valid (see example below).

## Example

An operator may not enter any data for one of the arguments. For example, the key sequence below is used.



This would mean the value of Arg2 is zero, and the second variable is set to zero. If you use multiple arguments in a command, invalid inputs might generate undesired behavior in the project or in the larger system.

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use keystroke sequences to pass multiple arguments to Cicode commands unless possible input combinations have been tested and determined to be safe.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**See Also**

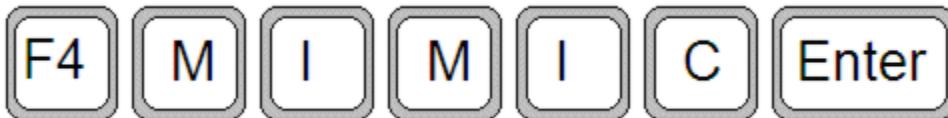
[Passing Keyboard Arguments to Functions](#)

**Passing Keyboard Arguments to Functions**

You can also pass arguments directly to functions at run time, as in the following example:

Key Sequence	F4 ##### Enter
Command	PageDisplay(Arg1);

In this example, an operator can select any graphics page (defined in the project) with a single command, for example:



This selects the "Mimic" page.

**Note:** Keyboard arguments are passed as string values. If the command (or function) requires a numeric value, Cicode converts the string to a numeric value before it is used.

If you use variable data, the operator can only enter alphanumeric characters (A - Z, a-z, and 0 - 9) for the data. Avoid using variable data input as the last item in a key sequence, as ambiguity could occur; use a delimiter.

**Reports**

You can request regular reports on the status of your plant, and reports that provide information about special conditions in your plant. Reports can be run on a request basis, at specified times, or when certain events occur (such as a change of state in a bit address). Output from a report is controlled by a device. A report can be printed when it runs or saved to disk for printing later. You can use a text editor or word processor to view, edit, or print the report, or you can display it in Plant SCADA as part of a page.

Reports can also include Cicode statements that execute when the report runs.

Reports are configured in two stages:

- Report properties
- Report format file

If report data is associated with an I/O Device that does not initialize properly at startup or goes offline while Plant SCADA is running, the associated data is not written to the report (because the values would be invalid). An error code is written instead.

## See Also

- [Configure a Report](#)
- [Format a Report](#)
- [Run a Report](#)
- [Handling Communication Errors in Reports](#)

## Configure a Report

Before you add a report to Plant SCADA, you should configure a device for output of the report. For example, if you want to save a report to a file when it is run, set up an ASCII\_DEV device.

### To add a report:

1. In the **Visualization** activity, select **Reports**.
2. Add a row to the Grid Editor.
3. Type the required information in each column, or in the fields in the Property Grid.  
For a description of the properties, see below.
4. Click **Save**.

You will then need to edit the Report Format File (see [Format a Report](#)). Remember that Edit the report format file. Remember that for an RTF report, the report format file needs to be saved in RTF format (with an .RTF file extension).

You will also need to define your PC as a report server using the Setup Wizard (see [Reports Configuration](#)).

## Reports Properties

### General Properties

Property	Description
<b>Name</b>	The name of the report. The name can be a maximum of 79 characters. It can consist of any character other than the semi-colon (;) or single quote ('). The name needs to be unique to the cluster. <b>Note:</b> Where Cluster Name is left blank, the name needs to be unique to every defined cluster.
<b>Time</b>	The time of day to synchronize the report, in hh:mm:ss (hours:minutes:seconds). If you do not specify a time, the report is synchronized at 0:00:00 (i.e., midnight). Enter a value of 32 characters or less.

Property	Description
<b>Period</b>	<p>The period of the report, in hh:mm:ss (hours:minutes:seconds). Enter a value of 32 characters or less. Alternatively you can:</p> <ul style="list-style-type: none"> <li>• Specify a weekly period by entering the day of the week when the report is to start, for example, Monday, Tuesday, Wednesday, etc.</li> <li>• Specify a monthly period by entering the day of the month when the report is to start, for example, 1st, 2nd, 3rd, 4th, 5th, etc.</li> <li>• Specify a yearly period by entering the day and the month when the report is to start, for example, 1st January, 25th February, etc. The day and month needs to be separated by a space.</li> </ul> <p>If you do not specify a period, the report is run daily.</p>
<b>Trigger</b>	<p>Any Cicode expression (or Variable tag) to trigger the report. Enter a value of 254 characters or less. If the result of the expression (in this field) is TRUE, and the Time and <b>Period</b> fields are blank, the report is run. The report is only run when the expression becomes TRUE, and it needs to become FALSE then TRUE again before the report is re-run.</p>
<b>Report Format File</b>	<p>The name of the report format file. Enter a value of 253 characters or less. If you do not specify a file extension, it defaults to .RPT. The file name of your report format file can be up to 64 characters long, or 253 characters including the path. It can consist of any characters other than the single quote ('), and the semi-colon (;). You cannot use a Path Substitution in this field.</p> <p>If you specify a file name without a path, the file saves into the directory predefined as Run. The report is assumed (by the Plant SCADA compiler) to be ASCII unless an RTF extension is used.</p> <p><b>Note:</b> You will need to manually configure the access control list for a folder location specified in a path so that it matches the permissions that were applied to the default Run location during installation. See <a href="#">Configure Directory Security for Modified Folder Locations</a>.</p>
<b>Privilege</b>	<p>The privilege necessary by an operator to run this report if the report is a command-driven report. Enter</p>

Property	Description
	<p>a value of 16 characters or less.</p> <p>If the report is time-driven or event-driven, this property is ignored.</p> <p><b>Note:</b> If you assign an acknowledgment privilege to a report, do not assign a privilege to the command(s) that run the report. If you do assign a different privilege to the commands, an operator needs to have both privileges to run the report.</p>
<b>Area</b>	<p>The area to which this report belongs. Enter a value of 16 characters or less. Only users with access to this area (and any necessary privileges) will be able to run this report. For example, if you enter Area 1 here, operators need to have access to Area 1 (plus any necessary privileges) to run this report.</p>
<b>Output Device</b>	<p>The device where the report will be sent. Enter a value of 16 characters or less.</p> <p>For RTF reports that are to be saved as a file, select a device of type ASCII_DEV here. Due to the differing natures of their content; however, it is not recommended that the same ASCII device be used for logging both RTF and non-RTF reports.</p> <p><b>Note:</b> If two or more reports are running at the same time and are sending their output to the same printer, the output of each report can become mixed. You need to use semaphores to control the access to the printer in each report. See the <a href="#">SemOpen</a> Cicode function. If the report only contains Cicode statements (and has no output data), this property is optional.</p>
<b>Comment</b>	<p>Any useful comment. Enter a value of 48 characters or less.</p>
<b>Cluster Name</b>	<p>The name of the cluster that runs this report. If the Cluster Name is not set, then Plant SCADA considers this report to run on every defined cluster.</p>

#### Project Properties

Property	Description
<b>Project</b>	<p>The project in which the report is included.</p>

## See Also

[Format a Report](#)

[Run a Report](#)

## Format a Report

**Note:** If the report format file exists, it is loaded into the property grid for you to edit. If the file does not exist, Plant SCADA creates a new file.

### To change the report format file:

1. In the **Setup** activity, select **Reports**.
2. Select the Report you want to change from the Property Grid.
3. Edit the Report Format File field with the new format.
4. Click **Save**.

The report format file specifies how data is formatted in a report. You can use fixed text, Cicode expressions, and database variables in any report.

You use a text editor that is supported by Windows to create (and modify) the report format file. If your report format file is in RTF (Rich Text Format), use Microsoft Wordpad.

#### Including fixed text

You can include fixed text in the report, specifying the text exactly as you want it to appear (for example, Name of Report, Description, and so on).

#### Including OLE (RTF files only)

Objects can be linked to or embedded within an RTF report format file; however, such objects will not be displayed or printed from Plant SCADA.

#### Using fonts (ASCII format only)

If your format file is in ASCII format, you can use any text font supported by Windows in the report. To specify a font, use the PrintFont() function. RTF format files do not require this function, as they use the formatting features of the host word processor.

#### Including Cicode expressions and variables

You can include Cicode expressions and variables by enclosing them (and optional format specifications) in braces {} - for example:

The size of each field (number of characters) is determined by either the format specification, or by the number of characters between the braces. In the above example, the variable PV12 is formatted with four characters before the decimal point and two characters after.

You do not have to include the format, for example:

{PV12}

Here, the variable is formatted using only four characters (the number of characters between the braces).

The following rules apply when logging a report to a **database device**:

- The format (for the report field) do not specify a field size greater than the size of the relevant field specified in the device.

- No spaces are allowed between each field specification, for example:

```
{TIME(1) }{PV12:####.##}{PV12:4.2}
```

### Including blocks of Cicode

You can include a block of Cicode, using the following format:

```
{CICODE}  
Statements;  
{END}
```

The block of Cicode is delimited by the commands {CICODE} and {END}. After the {END} command, the report switches back into WYSIWYG mode. If the entire report is Cicode or the last section is Cicode, the {END} command is not necessary.

A block of Cicode does not send any output to the device unless you use either the Print() or PrintLn() functions. If you use one of these functions, the argument is printed to the device.

### Cicode variables (ASCII format only)

You can also declare variables for use within your Cicode block. This is not available in RTF or HTML format files. Declare any variables at the beginning of the file (i.e. before any report format or Cicode). Add a {CICODE} block first; for example:

```
{CICODE}  
INT nVar1;  
STRING sVar2;  
Statements;  
{END}
```

Remainder of report

### Including comments

You can include comments by using the comment character `!' enclosed in braces - for example:

```
{!This is a Comment}
```

A comment in the body of a report differs from a comment in a Cicode block: a comment in a Cicode block does not require braces.

### Including other report elements

The following table describes other elements you can include in reports.

To...	Do this...
Issue a form feed	Use a form feed specifier: {FF}
Include a plot	Use the Plot functions.
Include trend data	Use the TrnGetTable() function.
Include trend graphs	Use the TrnPlot() function.

## Report Example

The following is an example of a report format file (for a printer or ASCII file device):

```
-----  
SHIFT REPORT  
-----  
{Time(1) } {Date(2) }  
Shift Production {Shift_Prod:###.##} tons  
Total Production {Total_Prod} tons  
{! The following Cicode displays "Shift Report Complete" on the  
screen}  
{CICODE}  
Print("End of Report")  
Shift_Prod = 0; ! Reset the Shift production tonnage  
  
Prompt("Shift Report Complete");  
{END}  
{FF}
```

This report produces the following output to the device and displays "Shift Report Complete" on the graphics page.

```
-----  
SHIFT REPORT  
-----  
6:00am 12/3/92  
Shift Production 352.45 tons  
Total Production 15728 tons  
End of Report
```

## See Also

[Run a Report](#)

## Run a Report

To run a report, use one (or a combination of) the methods listed below:

### Automatically run when Plant SCADA starts up

You can run a report on startup. Plant SCADA searches for a report called "Startup" when it starts up, and if it locates this report, it is run automatically. You can change the name of the default report with the Setup Wizard.

### Automatically run at a specified time and period

The period determines when the report is run. You can specify the period in hh:mm:ss (hours:minutes:seconds), for example:

Period	Comment
1:00:00	Run the report every hour
6:00:00	Run the report every six hours
72:00:00	Run the report every three days
Monday	Run the report each Monday

Period	Comment
15th	Run the report on the 15th of each month
25th June	Run the report on the 25th of June

You can also specify the time of day to synchronize the report, for example:

Time	Comment
6:00:00	Synchronize the report at 6:00 am
12:00:00	Synchronize the report at 12:00 midday

The time synchronizes the time of day to run the report and, with the Period, determines when the report is run, for example:

Time	Period
6:00:00	1:00:00

In this example, the report is run every hour, on the hour. If you start your runtime system at 7:25am, your report is run at 8:00am, and then every hour after that.

### Automatically run when an event is triggered

You can use any Cicode expression (or variable tag) as a trigger for a report. If the result of the expression (in the **Trigger** field) becomes TRUE, and if the **Time** and **Period** fields are blank, the report is run. For example:

Time	
Period	
Trigger	RCC1_SPEED<10 AND RCC1_MC

This report is only run when the expression (Trigger) becomes TRUE, i.e., when the digital tag RCC1\_MC is ON and the analog tag RCC1\_SPEED is less than 10. The expression needs to become FALSE and then TRUE again before the report is run again.

If you use the **Time** and/or **Period** fields, the trigger is checked at the time and/or period specified, for example:

This report is run each hour, but only if the expression (Trigger) is TRUE (i.e., if the digital tag RCC1\_MC is ON and the analog tag RCC1\_SPEED is less than 10).

Time	6:00:00
Period	1:00:00
Trigger	RCC1_SPEED<10 AND RCC1_MC

This report is run each hour, but only if the expression (Trigger) is TRUE (i.e., if the digital tag RCC1\_MC is ON and the analog tag RCC1\_SPEED is less than 10).

### Run using a command

If the **Time**, **Period**, and **Trigger** fields are blank, the report can only be run by a command that calls the **Report()** Cicode function.

## See Also

[Handling Communication Errors in Reports](#)

## Handling Communication Errors in Reports

You can handle errors in communication with I/O devices (for example, an I/O device does not initialize properly at startup or goes offline while Plant SCADA is running) in two ways:

### **Write communication errors and invalid data to the report as error codes.**

If a communication error is detected (with an I/O Device) or if the data is invalid, one of the following alert messages are written to the report (instead of the value):

Error	Meaning
#ASS	The value is incorrectly associated (with a substitution string or Genie).
#COM	Communication with the I/O Device has been lost
#DIV/0	An attempt was made to divide a number by 0 (zero)
#ERR	An uncommon error has occurred. (Use the IsError function to find the occurrence.)
#MEM	Out of memory or more than 64 kb bytes of memory requested.
#PEND	Data from this device is pending an initial update to display a value.
#RANGE	The value returned is out of range
#STACK	The value has caused a stack overflow
#WAIT	Data from this scheduled device is not available as it has not reached its scheduled interval and has no cache value.

For example:

#### **Report Format:**

If the above report is run when the value of PV\_1 is out of range (for example 101.5), SP\_1 is 42.35 and OP\_1 is 60.0, the output of the report is:

#### **Report Output:**

#RANGE 42.35 60.0

When reports are written to a database device, you might sometimes want to disable the alert messages and write the values to the report (even if the values are invalid). Use the ERR\_FORMAT\_OFF command to disable

alert messages and write data as values.

For example:

**Report Format:**

```
{ERR_FORMAT_OFF}  
{PV_1} {SP_1} {OP_1}  
If the above report is run when the value of PV_1 is out of range (for example 101.5)  
, SP_1 is 42.35 and OP_1 is 60.0, the output of the report is:
```

**Report Output:**

42.35 60.0

To re-enable the alert messages, use the ERR\_FORMAT\_ON specifier.

---

**Note:** If an I/O Device goes offline and you have disabled communication errors, the value printed into the report is either 0 (zero) or the last value read from the I/O Device when the report was last run. In either case, the value is invalid.

---

## Suppress Reports

You can suppress reports that are triggered from I/O Devices if a communication error is detected by using the [Report]ComBreak parameter. For example, you might configure a report to be run every hour when a bit is on. The I/O Device associated with that bit goes offline. If the [Report]ComBreak parameter is 0, the report does not run. If the parameter is 1, and if the latest valid value that was read from that bit was 1, the report is run.

This parameter only applies to the trigger of the report, not to the data in the report.

## Security

Plant SCADA security involves three key areas:

- **Encrypted communications** - a System Management Server is used to manage certificates that allow encrypted communications between computers in a Plant SCADA system (see [Encrypted Communications](#)).
- **Security Roles** - security-related operations within Plant SCADA's engineering and runtime environments can only be performed by users assigned to a role with relevant permissions (see [Security Roles](#)).
- **Runtime System Security** - settings within a Plant SCADA project that control permissions within the runtime environment, including **Roles** and **Users** (see [Runtime System Security](#)).

## Encrypted Communications

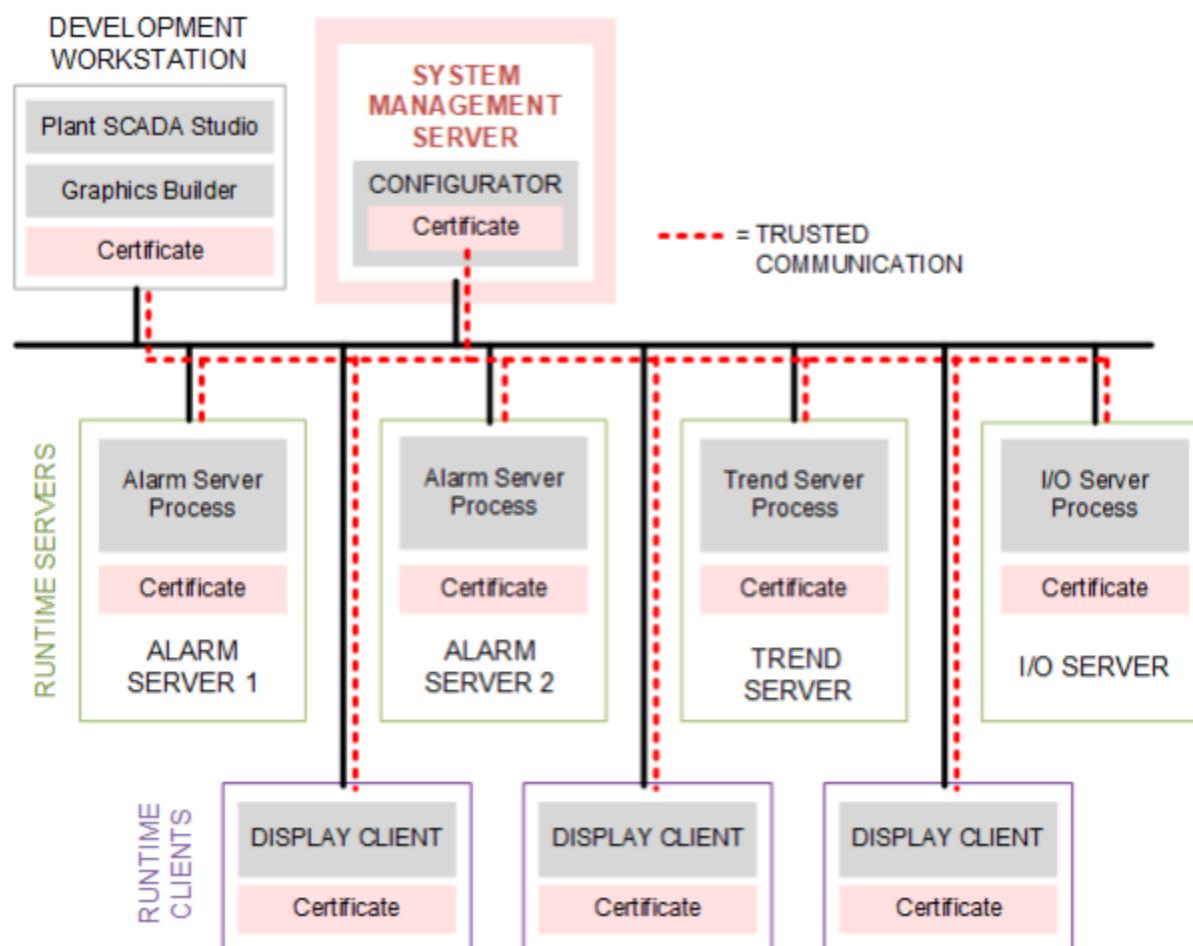
Communications between Plant SCADA processes, computers and CtAPI can be encrypted. To use encryption, a **System Management Server** is required to manage the certificates that enable trusted communications.

The components required to enable encryption are included with every installation of Plant SCADA.

Only one of the computers in a Plant SCADA system needs to be configured as the System Management Server. You use Configurator to configure the computer that will perform this role (see [Configure a System Management Server](#)).

The certificates required to establish trust can be generated automatically on the System Management Server or provided by your IT department.

Computers running AVEVA products can then use a certificate to connect to the System Management Server across an encrypted connection (see [Connect a Computer to a System Management Server](#)).



Encryption is a requirement for some Plant SCADA components, such as a Deployment Server, an Industrial Graphics Server or an OPC UA Server. To enable encryption, you need to set Runtime Manager to run as a service on any computers that host a server process. See [Enable Encryption](#).

In a typical Plant SCADA system that is also using deployment, it is recommended that the System Management Server is configured on the same computer as the Deployment Server.

---

**Note:** A System Management Server can also be installed in a large, multi-site environment running multiple AVEVA products. In such systems, the location of the System Management Server may be governed by one or more products. However, all AVEVA products should be able to connect to the System Management Server at all times so that certificates can be renewed when it is required.

## See Also

[Use Externally Provided Certificates for Encryption](#)

[Use SMS Certificates with Web Applications](#)

[Troubleshooting Certificate Error Messages](#)

## Configure a System Management Server

Plant SCADA needs post-installation configuration in order to use encrypted communications via a **System Management Server**. Encryption is a requirement for components such as a Deployment Server, an Industrial Graphics Server or an OPC UA Server.

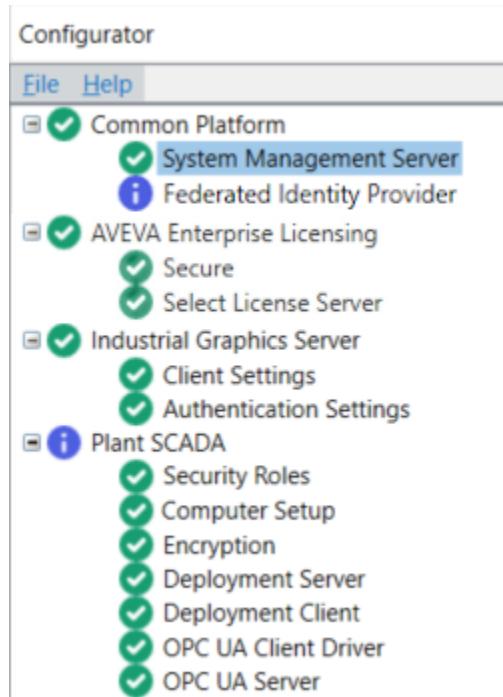
Only one of the computers in the network can be configured as a System Management Server. This is achieved using [Configurator](#).

You can then use Configurator to establish a trust relationship between one or more computers running Plant SCADA and its services. This is achieved by configuring a certificate on the System Management Server that can then be distributed to other computers to enable encrypted communications.

Certificates may be generated automatically on the System Management Server, or provided by a system administrator or IT department (see [Use Externally Provided Certificates for Encryption](#) in the Plant SCADA documentation).

### To configure the System Management Server:

1. Start the **Configurator**.
2. In the left pane, select **Common Platform | System Management Server**.



The following page is displayed:

Machines running AVEVA software must be configured to trust each other so that encrypted communications can be utilized. This is done by connecting them to a System Management Server.

- Connect to an existing System Management Server.

- This machine is the System Management Server.

There should only be one System Management Server in your topology for all AVEVA products. All other machines should be configured to connect to this System Management Server.

When configuring other machines, you should validate that the security code shown in the Configurator matches:

[Details...](#)

- No System Management Server configured. (NOT RECOMMENDED)

This option also allows you to remove any existing certificates that were managed by the System Management Server.

You can connect to an existing System Management Server or configure a new System Management Server by selecting one of the first two options, respectively. When you click Configure, a certificate and the web ports to use for communication are configured. To modify these configurations, click Advanced.

[Advanced](#)

3. Select **This machine is the System Management Server**. Review the notes on the screen before you start the configuration.

**Note:** The machine on which you want to install the System Management Server needs to be configured to have a static IP address.

4. Click **Configure**.

**Note:** If you need to change the port used by a System Management Server see the topic *Advanced Configuration for a System Management Server* in the Plant SCADA documentation.

5. On successful configuration, the message "Device configuration completed" is displayed. The security code is displayed in the **Configurator** as shown below. To view more information about the certificate, click **Details**.

#### Configuration Messages

Certificates are configured.  
Connecting to the System Management Server 'https://<IP>.com:443'.  
Registering the device.  
The device is registered.  
Device configuration completed.

If the configuration is unsuccessful, check the [Log Viewer](#).

You can access the Log Viewer by selecting **AVEVA | Operations Control Management Console** in the Windows **Start** menu.

Alternatively, view the error messages in the System Management Console.

6. Click **Close** to exit the **Configurator**.

---

**Note:** After you have configured a System Management Server, it is recommended that you either sign out or reboot the computer to ensure all settings are applied correctly.

---

## See Also

[Connect a Computer to a System Management Server](#)

## Connect a Computer to a System Management Server

Computers running Plant SCADA need to connect to the **System Management Server** to use encrypted communication enabled by a certificate.

If you do not have a System Management Server configured, refer to the section [Configure a System Management Server](#) for instructions. AVEVA applications need to be configured to trust each other to use encrypted communications.

To connect a computer to the System Management Server, you use the System Management Server page in **Configurator**. You can launch **Configurator** at the completion of the installation procedure, or from the Windows™ Start menu.

To establish a connection, the current user on the remote computer needs to be a member of either the "aaAdministrators" or the "Administrators" group on the machine where the System Management Server is installed. This is necessary to establish a trust relationship, it is not a runtime requirement.

---

**Note:** For a Deployment Server, you need to authorize the connection to a System Management Server from Configurator's Deployment Server page. See [Configure a Deployment Server](#).

---

### To connect a computer to the System Management Server:

1. In the panel on the left side of the **Configurator** select **Common Platform**, **System Management Server**.
2. Select the option **Connect to an existing System Management Server**.
3. In the field below, select the name of the System Management Server you want to use from the drop-down list.

The Configurator uses SSDP (Simple Service Discovery Protocol) to detect a System Management Server on a remote computer. If SSDP has been blocked on your network, nothing will appear in the drop-down list.

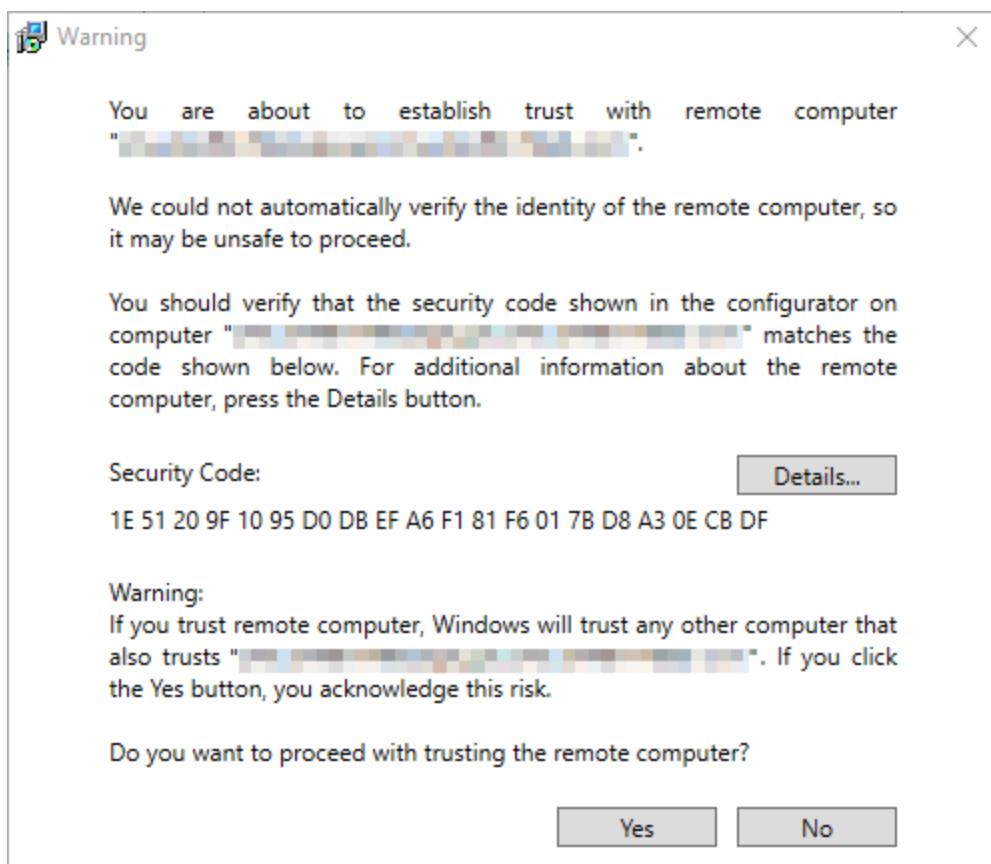
If the required computer name does not appear in the list, you can manually enter a hostname or Fully Qualified Domain Name (FQDN) for the computer on which the System Management Server is installed. Connecting via an FQDN will require a connection to a domain server.

---

**Note:** The Configurator interface refers to "redundant SSO servers". This feature is not supported by Plant SCADA, you cannot configure a computer as a redundant SSO server.

---

4. Click **Configure**. The root certificate is downloaded, and the following message is displayed.



5. Review the message carefully before you click Yes. If you select No, the configuration process will be canceled.

A dialog box may appear prompting you to log on to the System Management Server. This will occur if the current user is not a member of either the "aaAdministrators" or the "Administrators" group on the machine where the System Management Server is installed. If this happens, enter the credentials for a member of one of these groups and click OK.

The Configuration Messages area displays the steps in the configuration process and the progress.

If the configuration is unsuccessful, view details of the error messages in the System Management Console. See the topic [Troubleshooting Certificate Error Messages](#).

Alternatively, check the [Log Viewer](#).

You can access the Log Viewer by selecting **AVEVA | Operations Control Management Console** in the Windows **Start** menu.

6. Click **Close** to exit the **Configurator**.

**Note:** After you have connected a computer to a System Management Server, it is recommended that you either sign out or reboot the computer to ensure all settings are applied correctly.

Once you have connected to a System Management Server, you can modify the configuration settings by selecting **Advanced**. Refer to the topic [Advanced Configuration for a System Management Server](#) in the Plant SCADA documentation for more information.

**Note:** If you need to connect a client computer that is not on the same domain as the System Management Server (for example, it is part of a Workgroup), you need to configure the **Primary DNS Suffix** on the client to identify the domain on which the System Management Server resides. Refer to the Windows documentation for

---

instructions on how to specify a Primary DNS Suffix for a computer.

---

## Enable Encryption

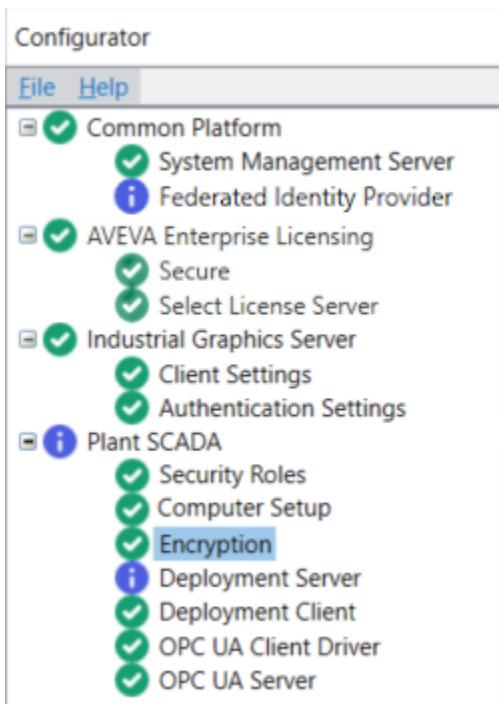
Communications between Plant SCADA processes, computers and CtAPI can be encrypted in Plant SCADA.

To use encrypted communications, you need to have a System Management Server configured. In addition, the Runtime Manager should be running as a service on any computers hosting a server process (see ).

If these prerequisites are not met, warning messages will be displayed.

### To enable encryption:

1. Launch the **Configurator** after the installation, or from the Windows™ Start menu.
2. In the left pane, select **Plant SCADA | Encryption**.



The following page is displayed.

[Help](#)

Communications between Plant SCADA processes, computers and CtAPI applications can be encrypted

Enable Encryption

Accept encrypted and non-encrypted connections (mixed mode)

*This mode is recommended only for online upgrades and/or supporting legacy CtAPI based applications*

**Encryption Prerequisites** [Refresh](#)

This computer must be registered with the System Management Server 

Runtime Manager must be configured to run as a service (not required for clients). 

3. Firstly, check that the **Encryption Prerequisites** have been met. If a red cross appears next to an item, you will need to make the required configuration changes before you can enable encryption.
  - To register the computer with the System Management Server, see [Connect a Computer to a System Management Server](#).
  - To configure Runtime Manager to run as a service, see [Configure a Runtime Computer for Encryption](#).
4. Select **Enable Encryption**.
5. If you are performing an online upgrade of Plant SCADA or running an older version of Citect SCADA, select **Accept encrypted and non-encrypted connections (mixed mode)**. Note that you can clear this option and configure your system to use encryption after the upgrade process is complete.
6. Click **Configure**. A message is displayed when encryption setup is completed.

The following table indicates the encryption mode that will align successfully across server and client.

		Encryption mode on server		
		Unencrypted	Mixed Mode	Encrypted
Encryption mode on client	Unencrypted	Communication OK	Communication OK	Communication unsuccessful
	Mixed Mode	Communication OK	Communication OK	Communication OK
	Encrypted	Communication unsuccessful	Communication OK	Communication OK

**Note:** If you do not select **Enable Encryption** before you click **Configure**, the following message is displayed:  
"Encryption is currently disabled. It is recommended that you enable encryption."

## See Also

[Runtime System Security](#)

[Connect a Computer to a System Management Server](#)

## Configure a Runtime Computer for Encryption

To prepare a computer for encryption, you need to confirm the following runtime environment settings.

These settings can be adjusted using the **Configurator's Computer Setup** page under Plant SCADA. You can launch Configurator at the completion of the installation procedure, or go to the Windows® **Start** menu and locate **AVEVA | Configurator**.

### Runtime Manager Configuration

This setting instructs Runtime Manager to run as a service. This is required under the following circumstances:

- You are running with encryption enabled. This includes running in mixed mode.
- You want to connect an OPC UA server to your system.
- You want to connect an Industrial Graphics Server to your system.

---

**Note:** If your Plant SCADA server processes are not running as a service, you will not be able to establish an encrypted connection between the server and your clients.

---

#### To run Runtime Manager as a service:

1. In the panel on the left side of the **Configurator**, select Plant SCADA | **Computer Setup**. The **Computer Setup** page appears. Navigate to the **Runtime Manager Configuration** section of the dialog.

##### Runtime Manager Configuration

*The Runtime Manager can be configured to run as a service or as a standard process. When run as a service, Runtime Manager allows projects to be deployed to the machine without having to change any other settings.*

Run Runtime Manager as a service

2. Select **Run Runtime Manager as a Service**.

---

**Note:** If you are using Deployment to run a project and do not select the **Run Runtime Manager as a Service** option, you will need to manually start the Plant SCADA Runtime Manager before deploying the project.

---

3. To apply your settings, click the **Configure** button.

---

**Note:** If you have configured a Plant SCADA OPC DA server on a computer that is running Plant SCADA as a Windows service, you will need to make the additional configuration changes. See the topic *Running an OPC DA Server as a Service* in the *Runtime* section of the Plant SCADA documentation.

---

### See Also

[Enable Encryption](#)

[Troubleshooting Certificate Error Messages](#)

## Advanced Configuration for a System Management Server

The System Management Server page in Configurator includes an **Advanced** button that provides access to the Advanced Configuration dialog.

If you have already configured a **System Management Server**, you can use the tabs on this dialog to adjust the following settings.

### Certificates

By default, the System Management Server is configured to generate its own certificates for distribution to any connecting computers.

You can verify this setting on the Advanced Configuration dialog; the **Certificate Source** field will be set to "Automatically Generated".

However, you may need to enable encryption using certificates provided by a system administrator, a corporate IT department, or a certified third-party vendor. You can use this section of the Advanced Configuration dialog to achieve this. For more information, see [Use Externally Provided Certificates for Encryption](#).

---

**Note:** Automatically generated certificates are renewed automatically. An externally provided certificate will need to be renewed by the provider.

If you are on a computer that has **Connect to an existing System Management Server** selected, the **Certificates** tab will include a field labeled **System Management Server**. This field will display the name of the computer that hosts the active System Management Server. The **Port** field to the right is automatically populated with the port number that the System Management Server is configured to use.

If required, you can connect the local computer to a different System Management Server. From the drop-down list in the **System Management Server** field, select the computer you would like to use to generate and host your certificates.

---

**Note:** You should only specify a computer name or a fully-qualified domain name for a System Management Server. Specifying the name in a different format, for example an IP address, may not be successful.

### Ports

The **Ports** tab on the Advanced Configuration dialog shows the current setting for the following ports:

- **HTTP Port** (not used by Plant SCADA)
- **HTTPS Port**.

These represent the local ports that are used by web services and clients connecting to the computer you are currently using. In the case of Plant SCADA, only the **HTTPS Port** value is considered.

The default value used for an HTTPS Port it is 443. If you need to change the port that is currently being used (for example, you are seeing a "Port number conflict" error message), you can enter a new port number.

#### To enter a new port value:

1. In the **HTTPS Port** field, enter any port number that has not been blocked or is currently in use. HTTPS ports can range from 0 to 65535.
2. Click **OK** to save your settings. The Configurator's main screen will display.

3. Click **Configure** to apply your changes.

The Configuration Messages area displays the steps in the configuration process and the progress. If the configuration is unsuccessful, refer to the error messages in the System Management Console.

See [Troubleshooting Certificate Error Messages](#).

---

**Note:** The functionality described on the **Authentication** tab of the Advanced Configuration dialog is not supported by Plant SCADA.

---

## See Also

[Use SMS Certificates with Web Applications](#)

[Enable Encryption](#)

## Use SMS Certificates with Web Applications

Some of the features supported by Plant SCADA use web-based applications that host content in a web browser. These applications need to trust certificates that have been generated by the System Management Server (SMS).

---

**Note:** The information provided in this topic is intended for a scenario where SMS certificates are used on computers contained within your SCADA system network. If you want to deliver information across multiple domains or externally via the internet, we recommend you seek professional advice on setting up an external web server.

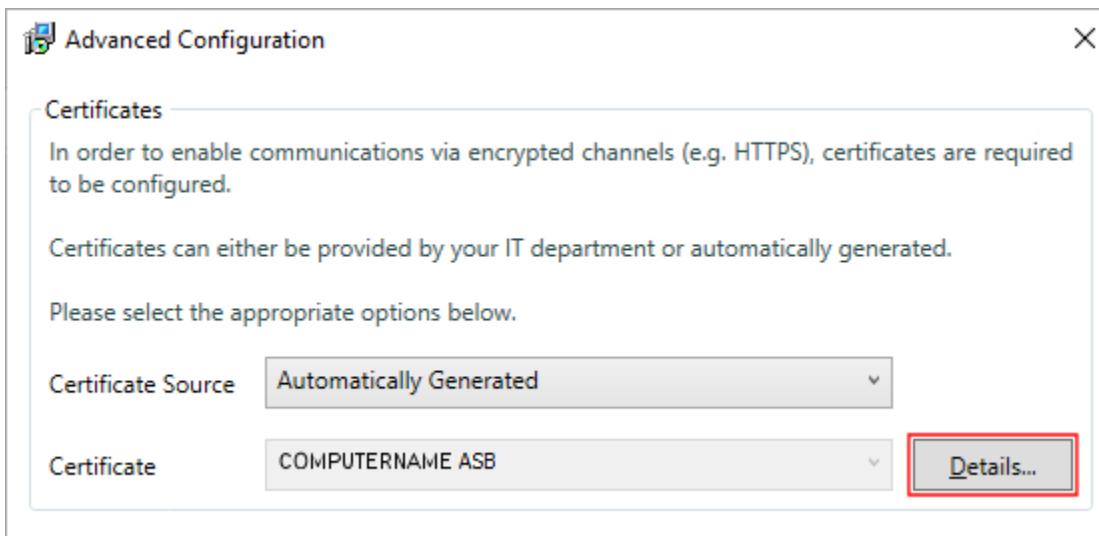
---

The following procedures are required to use SMS certificates with a web-based application.

### Export Certificates from the SMS

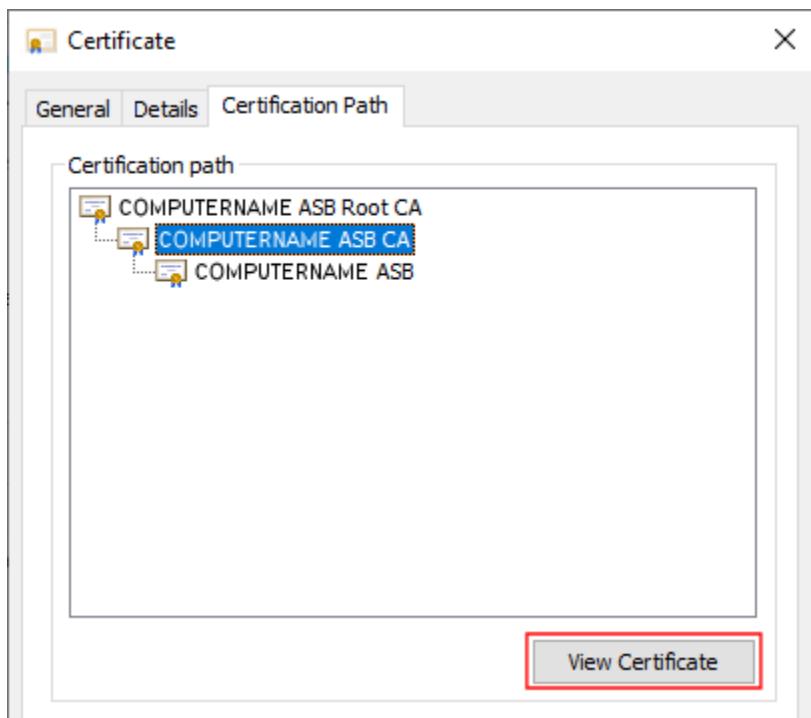
You initially need to export a copy of two certificates created by the SMS.

1. Open Configurator and display the **System Management Server** page.
2. Confirm that the SMS is set up and operational (if required, see [Configure a System Management Server](#)).
3. Click on the **Advanced** button. The Advanced Configuration dialog will appear.
4. Confirm that the required certificate is selected, then click on the **Details** button.



The Certificate dialog will appear.

5. Go to the **Certification Path** tab and select the entry "<computer name> ASB CA", then click the **View Certificate** button.



Another dialog will appear specifically for the "<computer name> ASB CA" certificate. This is the dialog you use to export the certificate to a file.

6. On the **Details** tab, click **Copy to File**. This will open the Certificate Export Wizard.
7. Use the following settings to export the certificate as a CER file.
  - On the **Export Private Key** page, select **No, do not export the private key**.
  - On the **Export File Format** page, select **DER encoded binary X 509 (.CER)**.
  - On the **File To Export** page, enter a path and file name (for example, "c:\temp\<machine name> ASB CA.cer").
8. When you reach the Finish page, review the settings and click **Finish** to export a copy of the certificate. You then need to repeat this process to export a copy of the "<computer name> ASB Root CA" certificate.
9. Go back to step 4 and select "<computer name> ASB Root CA" on the **Certification Path** tab.
10. Repeat steps 5—7.

### Import Certificates on a Client Computer

To complete this process, you need the two CER files exported from the ASB CA and the ASB Root CA certificates.

1. Copy the two CER files to an appropriate location on the client computer.
2. Right click on the CER file created from the ASB Root CA certificate and select **Install Certificate**. This will open the Certificate Import Wizard.
3. Use the following settings.
  - Under **Store Location**, select **Local Machine**.

- On the **Certificate Store** page, select **Trusted Root Certification Authorities Store**.
4. When you reach the last page, review the settings and click **Finish**.  
You then need to repeat this process for the CER file created from the ASB CA certificate.
  5. Right click on the ASB CA file and select **Install Certificate**.
  6. Use the following settings in the Certificate Import Wizard.
    - Under **Store Location**, select **Local Machine**.
    - On the **Certificate Store** page, select **Intermediate Certification Authorities Store**.
  7. When you reach the last page, review the settings and click **Finish**.

## See Also

[Enable Encryption](#)

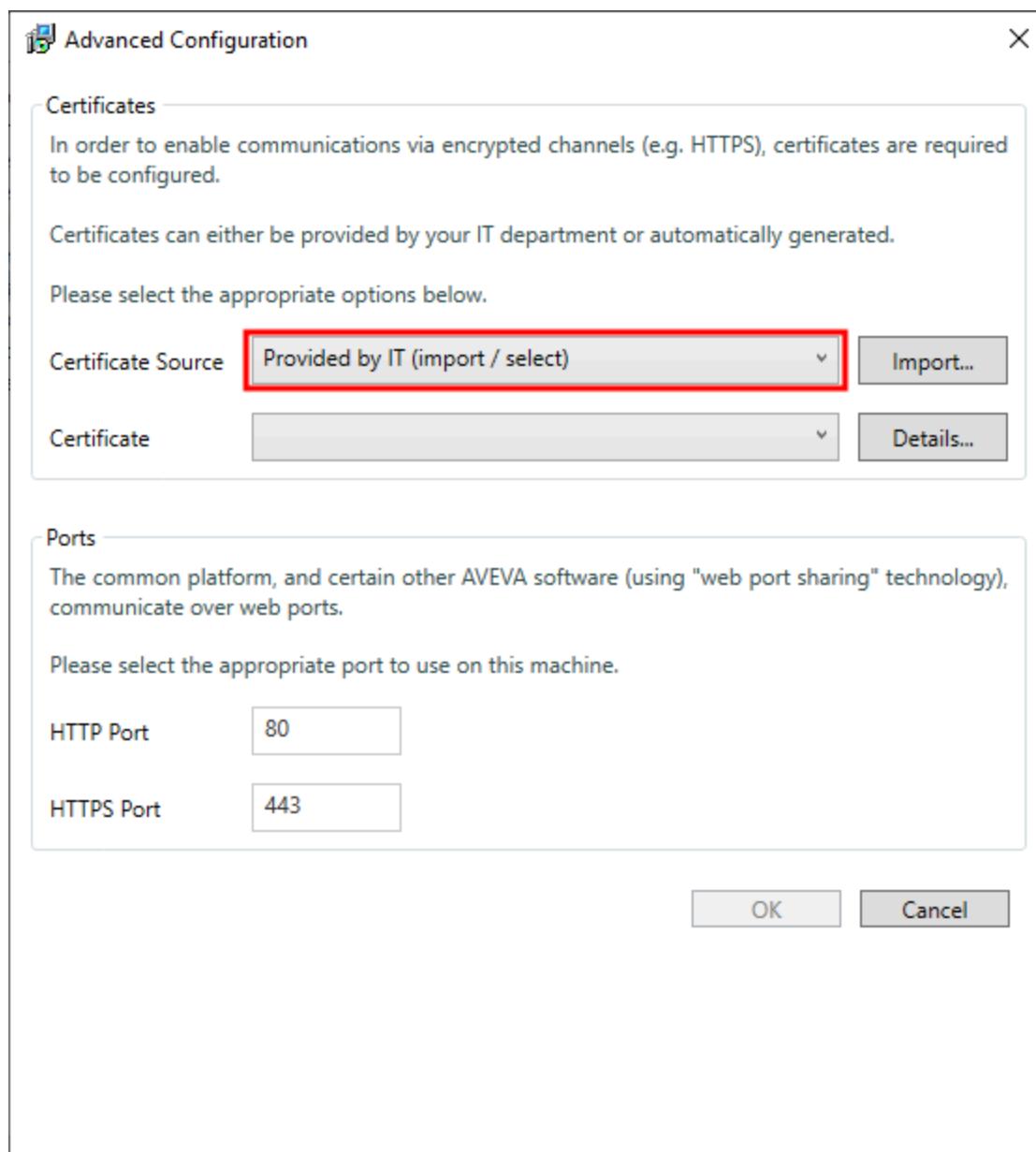
## Use Externally Provided Certificates for Encryption

By default, the System Management Server is configured to generate its own certificates for distribution to any connecting computers. However, you may need to enable encryption using certificates provided by a system administrator, a corporate IT department, or a certified third-party vendor.

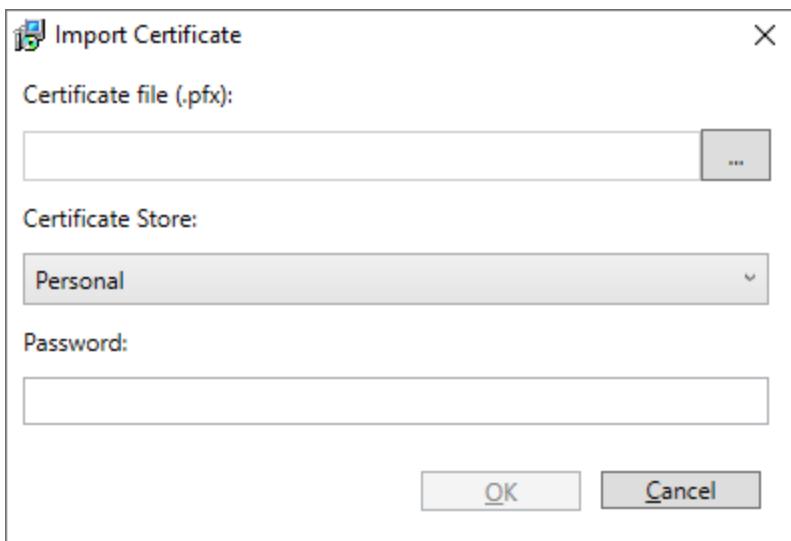
**Note:** If you are provided with a certificate file, it needs to be a .pfx file.

### To use externally provided certificates for encryption:

1. In Configurator, open the System Management Server page.
2. Click on the **Advanced** button. The Advanced Configuration dialog will appear.
3. In the **Certificate Source** field, select "Provided by IT (import / select)".



4. In the **Certificate** field, select the certificate you want to use from the drop-down list.  
To use a certificate that is not included in the list, click **Import**. The **Import Certificate** dialog is displayed.



To import a certificate:

- a. In the **Certificate file** field, browse to the location of the .pfx certificate file you want to import and select it.
- b. In the **Certificate Store** field, select the type of certificate to create; Root, Intermediate or Personal. The certificate will be stored in the Certificate Store associated with the selected type.
  - a. Root Certificate - Local Computer Trusted Root Certification Authorities store
  - b. Intermediate Certificate - Local Computer Intermediate Certification Authorities store
  - c. Personal Certificate - Local Computer Personal store.

If you have only been provided with one certificate, select "Personal". However, check with your certificate provider about where the certificate needs to reside.

Also see [Personal Certificates Requirements](#).

If you have been provided with three certificates (a Root, Intermediate and Personal certificate), repeat this process three times to import each one and place them in the appropriate certificate store.

- In the **Password** field, type the password for the selected **Certificate Store**.
- Click **OK** to save your settings and close the **Import Certificate** dialog.

---

**Note:** The certificate provider will need to renew the certificates they generate as required.

5. To view the information about the selected certificate, click **Details**.
6. Click **OK** to save your settings. The Configurator's main screen will display.
7. Click **Configure** to apply your changes.

The Configuration Messages area displays the steps in the configuration process and the progress. On successful configuration, the Certificate is generated on the selected **System Management Server** and its name is displayed in the Certificate field on the Advanced Configuration dialog.

If the configuration is unsuccessful, refer to the error messages in the System Management Console.

8. Click **Close** to exit the **Configurator**.

### Setting permissions on the binding certificate

To complete the setup of an IT-managed certificate, the "ArchestrAWebHosting" user group needs read access to the certificate's private key.

1. Open the certificates manager (certmgr) in Microsoft Management Console.
2. In the tree view, select **Personal**, then **Certificates**.
3. Locate your IT-managed certificate.
4. Right-click and select **All Tasks**, then **Manage Private Keys**.
5. On the **Security** tab, use the **Add** button to allow read access to the "ArchestraWebHosting" group.

## See Also

[Use Externally Provided Certificates with Web Applications](#)

[Troubleshooting Certificate Error Messages](#)

## Use Externally Provided Certificates with Web Applications

Some of the features supported by Plant SCADA use web-based applications to host content in a web browser. The following procedure is required to enable encryption for a web-based application when the certificates have been provided by a system administrator, an IT department, or a certified third-party vendor.

This procedure also assumes that Plant SCADA's Configurator is not available. If it is, you should follow the procedure described in [Use SMS Certificates with Web Applications](#).

---

**Note:** The information provided in this topic is intended for a scenario where externally-managed certificates are used on computers contained within your SCADA system network. If you want to deliver information across multiple domains or externally via the internet, we recommend you seek professional advice on setting up an external web server.

### To import externally provided certificates on a client computer

To complete this process, you need to be provided an Intermediate Certificate and a Root Certificate.

1. Copy the two CER files to an appropriate location on the client computer.
2. Right click on the Root Certificate, and select **Install Certificate**. This will open the Certificate Import Wizard.
3. Use the following settings.
  - Under **Store Location**, select **Local Machine**.
  - On the **Certificate Store** page, select **Trusted Root Certification Authorities Store**.
4. When you reach the last page, review the settings and click **Finish**.
5. You then need to repeat this process for the Intermediate Certificate.
6. Right click on the file and select **Install Certificate**.
7. Use the following settings in the Certificate Import Wizard.
  - Under **Store Location**, select **Local Machine**.
  - On the **Certificate Store** page, select **Intermediate Certification Authorities Store**.
8. When you reach the last page, review the settings and click **Finish**.

## See Also

[Use Externally Provided Certificates for Encryption](#)

## Personal Certificates Requirements

If you are using certificates from a system administrator, an IT department, or a certified third-party vendor to enable encryption, the personal certificate provided to you needs to meet the following requirements.

Field	Value	Comment
Key Usage	Digital Signature, Key Encipherment, Data Encipherment (b0)	Required.
Enhanced Key Usage	Server Authentication (1.3.6.1.5.5.7.3.1)	Required.
Subject Alternative Name	DNS Name=localhost	Optional. Only required when your client accesses your web applications via localhost.
	DNS Name=[machine name]	Required. Used by your client to access web applications via the machine name. For example: <code>DNS Name=mymachineName</code>
	DNS Name=[fully qualified domain name]	Required if you are using AVEVA Industrial Graphics. The framework uses the FQDN as the endpoint identity to authenticate the web service. This item needs to be the last entry in the SAN list due to a known Microsoft .Net Framework issue. For example: <code>DNS Name=mymachine.mydomain.com</code>
Friendly name	A friendly display name	Required. Configurator displays the friendly name in the certificate drop-down list.

## See Also

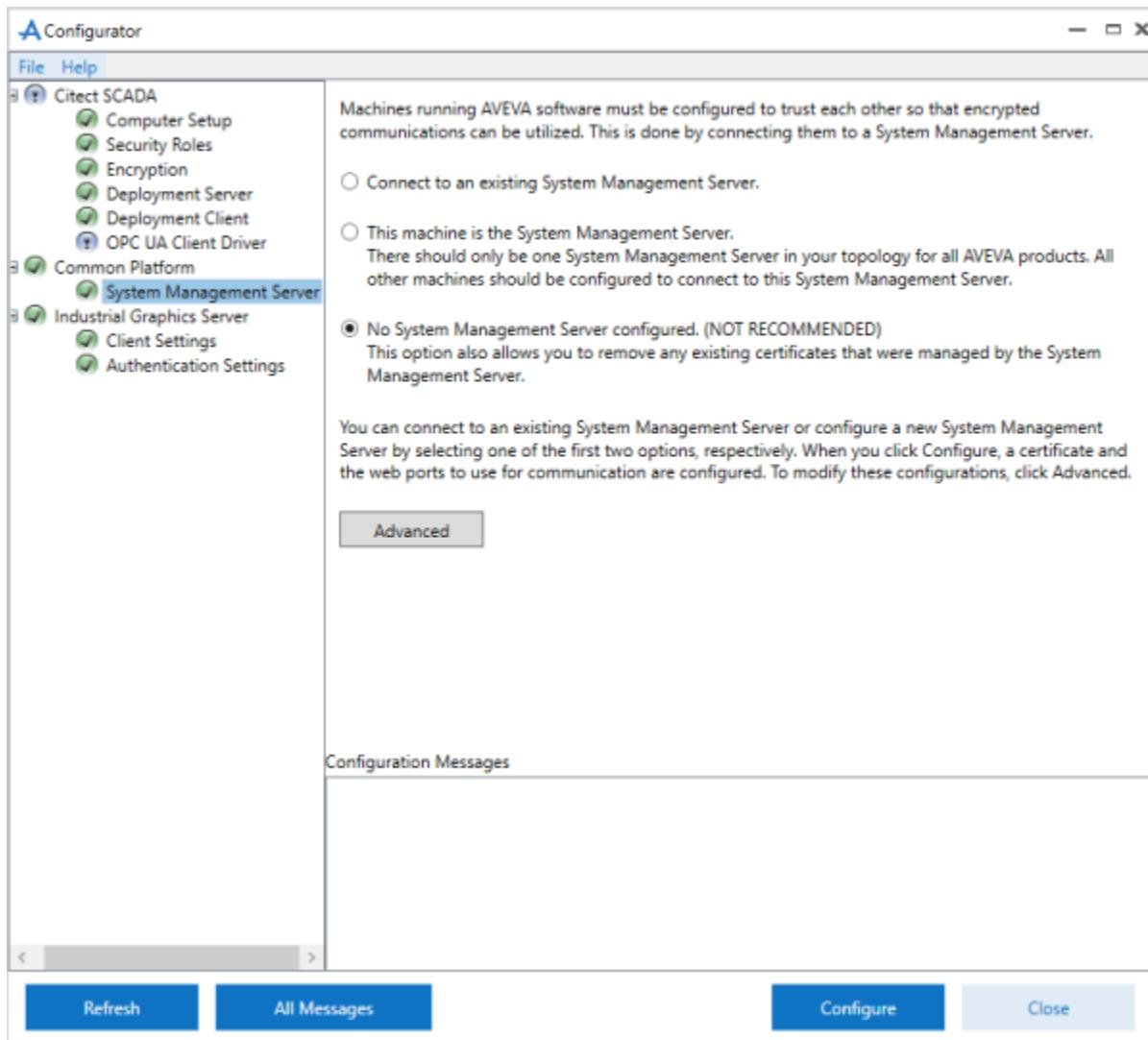
[Use Externally Provided Certificates for Encryption](#)

## Running Products without a System Management Server

**Note:** Running AVEVA products without a **System Management Server** is not recommended.

### To run your products without a System Management Server:

1. Start the **Configurator**.
2. In the left pane, click **Common Platform | System Management Server**.
3. Select **No System Management Server configured**. Selecting this option will result in the application managing trust relationships.



## Troubleshooting Certificate Error Messages

This topic lists some common error messages you might encounter when using encrypted communications.

## Use of IP address instead of machine name

Encrypted communications cannot be configured if you specify an IP address on the CtAPI client for the Plant SCADA machine you want to connect to. Connection will not be established in this case. Specify the machine name, which needs to match the machine name on the certificate. Other reasons for connection errors could be failure to resolve the machine name. You will see the message "Failed to secure connection and unsecured connection is not permitted" in the trace log.

## Unable to establish connection between server and client

The following combination of modes of encryption can be used for a secure connection.

Server	Display Client/CtAPI Client	Communications Occur?	Encrypted Communications?
No Encryption	No Encryption	Yes	No
No Encryption	Mixed	Yes	No
No Encryption	Encryption Enabled	No	No
Mixed	No Encryption	Yes	No
Mixed	Mixed	Yes	Yes
Mixed	Encryption Enabled	Yes	Yes
Encryption Enabled	No Encryption	No	No
Encryption Enabled	Mixed	Yes	Yes
Encryption Enabled	Encryption Enabled	Yes	Yes

**Note:** It is recommended that you enable encryption on both the server and the client.

---

If either machine is not in the mode combination for encrypted communications, you will see the message "**No connection could be made because the target machine actively refused it**" in the trace log.

---

**Note:** For CtAPI clients, you need to restart the CtAPI server if you change the encryption mode from encrypted to mixed in the Configurator.

## Runtime Manager not running as a service

If the **Run Runtime Manager as a service** option is not selected on the Configurator's **Computer Setup** page, you may encounter certificate errors. The syslog will display messages that indicate that the certificate is invalid, while the tracelog displays the message "**Certificate Private key is not accessible**". This will also be displayed on the hardware alarm page with the description "**Cert private key is unreadable**".

You need to configure Runtime Manager to run as a service if you are running any server processes, or if you are running a display client that has been configured to accept CtAPI connections (that is, the [CtAPI]Remote parameter is set to 1).

---

**Note:** The syslog and tracelog also display messages relating to expired certificates.

## Thumbprint errors

If the thumbprint is incorrect, you will see a hardware alarm with the description "**Certificate not found**". Details of this can be viewed in the syslog as well as the tracelog. The syslog will display "**Server thumbprint invalid**", while the tracelog will display "**Certificate not found. Failed to find certificate for thumbprint**".

## Remote client does not connect and generates "TLS Exchange Failed" hardware alarm

A remote client will use a DNS name to validate the certificate it receives from a server in a TLS exchange. The client will attempt to validate the name specified as the "Subject Alternative Name" in the certificate that it receives.

You can find the Subject Alternative Name by going to Configurator on the server and displaying the System Management Server page. To open the certificate properties, click on the **Advanced** button and select **Details**. On the Details tab, scroll down and find the "Subject Alternative Name" property. This will have three entries:

DNS Name=localhost

DNS Name=<computer name>

DNS Name=<FQDN>

As part of the TLS exchange, the computer name in the certificate is validated against the project. If the DNS name field is configured for the computer the server process is running on, the client will resolve the Subject Alternative Name against that field. If this is blank, the address field of the network address is used instead.

You should only use "localhost" when running a standalone server with no remote clients.

If the address from the network address on the server is a computer name, then the easiest way to configure this is to leave the DNS name blank. You will only need to set this to the actual computer name of the server if the network address is an IP Address.

If required, you can use the Plant SCADA Kernel to diagnose runtime connections using Page Table commands. See Page Table Platform.Sessions and Page Table Tran.

## Encrypt Plant SCADA Folders using SMB3

You can use the SMB3 network protocol to protect your Plant SCADA data from eavesdropping occurrences on untrusted networks. This can be achieved by enabling SMB encryption on the Microsoft™ Windows Server that hosts your Plant SCADA data. Encryption can be enabled on Windows Server on a per-share basis, or for the entire file server.

If you are using any features that require you to specify a directory path, you should consider using SMB3. This could include the use of the following parameters:

- [Path Parameters](#)
- [\[CtEdit\]Copy](#)

To use SMB3 encryption with Plant SCADA you need to follow procedures in both Windows Server (2016 or 2019), and Plant SCADA.

## Enable File Server on Microsoft Windows Server

File Server is used to manage shared folders on a Windows Server. You can access it via the **Add Roles and**

### Features Wizard.

Confirm that **File Server** is selected on the **Select server roles** page of the Wizard, then work your way through the wizard to install the required components.

Refer to the Windows Server documentation provided by Microsoft for more information.

## Enable SMB Encryption

You can enable SMB encryption for the entire files server, or only for specific file shares.

### To enable encryption for the entire file server:

You can create a procedure in Windows PowerShell. For example, you could use:

```
Set-SmbServerConfiguration -EncryptData $true
```

You could also confirm the status of the server with the following:

```
Get-SmbServerConfiguration | Select EncryptData
```

---

**Note:** You should confirm the correct usage for PowerShell procedures in the documentation provided by Microsoft.

### To enable encryption for a specific file share:

You can configure file shares in Windows Server using **Server Manager**. Locate **File and Storage Services** in Server Manager to access the **Shares** page.

The Shares page provides access to a **New Share Wizard**. This can be used to configure the following:

- A **Share Profile** for the file share (for example, "SMB Share - Quick").
- The **Share Location** (for example, "C:\ProgramData\AVEVA\Plant SCADA <VersionNumber>\User").
- A **Share Name** (for example, "Plant SCADA Users").
- **Share Settings** including a required option to **Encrypt Data Access**.
- **Permissions** that allow access to the shared files.

When the share is successfully created, it will appear on the Shares page in Server Manager. From here, you can access the properties for the share and make any required adjustments.

You should confirm the correct procedures for configuring shares in the documentation provided by Microsoft.

---

**Note:** You can also use PowerShell to add a new file share and enable SMB encryption. For example, you could use

```
"New-SmbShare -Name User -Path C:\ProgramData\AVEVA -EncryptData $true".
```

## Connect Plant SCADA to a File Share

When you have finished configuring a file share in Windows Server you can access it across an encrypted connection from Plant SCADA.

---

**Note:** The file share you use needs to be high-performance and resilient as any delays (such as network latency) will affect the performance of Plant SCADA's configuration and runtime tools.

For example, you can add a new project link in Plant SCADA Studio that connects to the file share (see [Link to an Existing Project](#)). You can then run the Plant SCADA project directly from the shared folder.

---

**Note:** You need to confirm that you are using the correct version of SMB (SMB3) on the server and client

---

machines. A PowerShell cmdlet is available that can tell you which version of SMB a client has negotiated with the File Server. You can access a remote file server (or create a new mapping to it) and use "Get-SmbConnection" to determine the SMB version that is currently being used. Refer to the PowerShell documentation provided by Microsoft for more information.

---

## Security Roles

To restrict access to Plant SCADA's sensitive components, only users with relevant permissions can perform certain security-related operations. To help manage these permissions, Plant SCADA uses a set of security roles. Each role provides a different level of access to features, applications and project resources.

The following security roles are used on all Plant SCADA computers.

Security Role	Description
Configuration Users	<p>Members of this role can run configuration tools (such as Plant SCADA Studio or Computer Setup Wizard) and start the runtime display client and server processes.</p> <p><b>Note:</b> It is recommended that members of this role only start runtime for development purposes and not in a production system environment.</p>
Runtime Users	<p>Members of this role can run the runtime display client and make local CtAPI connections.</p> <p><b>Note:</b> Any Windows® user account that has to start runtime needs to be assigned to this role. See <a href="#">Add the Required Users to the Runtime Users Role</a>.</p>
Server Users	<p>Members of this role can run Plant SCADA as a server process.</p> <p>If you are not running Plant SCADA as a service, add a member to this role who needs to run a Plant SCADA server (including a display client with [CtAPI]Remote enabled).</p>

**Note:** Plant SCADA computers use a server password to authenticate each other. This creates a trusted network between servers and, optionally, clients. This password is specified for a computer via [Configurator](#) or the Computer Setup Wizard. To set the server password for a computer, you need to be a member of the **Configuration Users** security role. To read the server password you need to be a member of either the **Configuration Users** or the **Server Users** security role. You can be a member of these roles either directly or via an associated domain group.

---

If you have installed deployment server components on a computer, additional three Security Roles will exist.

Security Role	Description
Deployment Administrators	Members of this role can add or remove client computers to/from deployment server. They can also perform upload and deploy operations.
Deployment Users	Members of this role can deploy a new project to a connected deployment client computer.
Deployment Uploaders	Members of this role can upload a new project version to the deployment server.

If you have installed Industrial Graphics Server on a computer, two additional Security Roles will exist.

Security Role	Description
Industrial Graphics Users	<p>Members of this role can connect and authenticate with the Industrial Graphics Server.</p> <p>For more information, see the topic <i>Configure User Access for an Industrial Graphics Web Client</i> in the Plant SCADA documentation.</p> <p><b>Note:</b> You should avoid adding individual users to this security role. To allow authorization in a distributed system, only add domain groups to an Industrial Graphics security role.</p>
Industrial Graphics R/W Users	<p>Members of this role can connect and authenticate with the Industrial Graphics Server. They will also be able to write to variable tags in a Plant SCADA system, provided the tags have been configured to support writes via the <b>Write Roles</b> property.</p> <p>For more information, see the topic <i>Enable Tag Writes for Industrial Graphics Applications</i> in the Plant SCADA documentation.</p> <p><b>Note:</b> You should avoid adding individual users to this security role. To allow authorization in a distributed system, only add domain groups to an Industrial Graphics security role.</p>

During installation, by default the local Windows user groups are associated with these security roles as members (see the information on *Windows User Groups and Security Roles* in the topic [Installation Information](#)).

You can view the current associated members with the security roles via the **Security Roles** page under Plant SCADA in **Configurator**. When you select a role, the existing members associated with the role will be listed in the **Members of...** section.

The following security roles define which Windows users can access specific features, software applications and project resources.

Security Role	Members
Configuration Users	SCADAPC01\SCADA.ConfigUsers
Runtime Users	SCADAPC01\SCADA.RuntimeUsers
Server Users	SCADAPC01\SCADA.ServerUsers, NT SERVICE\...
Deployment Administrators	SCADAPC01\SCADA.DeploymentAdmins
Deployment Uploaders	SCADAPC01\SCADA.DeploymentUploaders
Deployment Users	SCADAPC01\SCADA.DeploymentUsers

**Members of 'Configuration Users'**

Users who can start the configuration tools and set the server password

SCADAPC01\SCADA.ConfigUsers

Add Remove

Configuration Messages

Refresh All Messages Configure Close

It is recommended that you replace the local Windows groups associated with these security roles with your own domain groups. If required, you can also assign additional user accounts or groups to a role (see [Modifying the Members of a Security Role](#)).

## See Also

[Modifying the Members of a Security Role](#)

## Modifying the Members of a Security Role

The **Security Roles** page under Plant SCADA in **Configurator** lists the available Plant SCADA Roles, depending on the components you have selected during Plant SCADA installation.

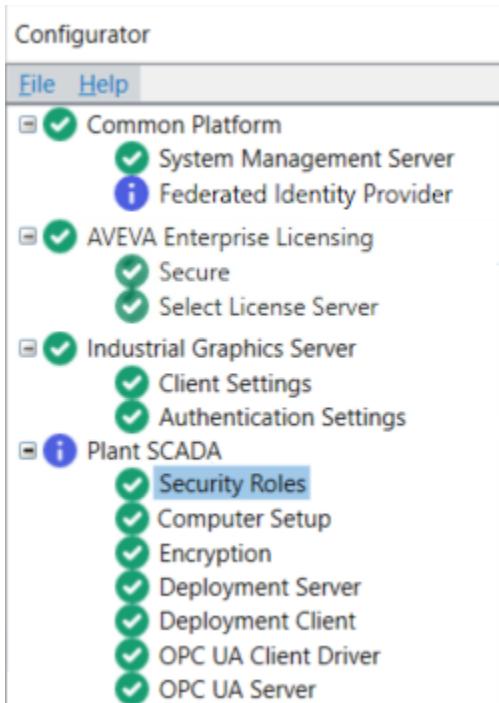
By default, the Windows® User Groups created by the Plant SCADA installer are associated with these roles (see

the information on *Windows User Groups and Security Roles* in the topic [Installation Information](#)).

If required, you can change the members that are associated with each security role. You can also delete the default groups and replace them with your own.

#### Add local or domain groups to a security role:

1. Open the **Configurator**.
2. From left side panel, select Plant SCADA | **Security Roles**.



3. Select a security role displayed in the table. The existing members associated with the role will be listed in the **Members of...** section.

The following security roles define which Windows users can access specific features, software applications and project resources.

Security Role	Members
Configuration Users	SCADAPC01\SCADA.ConfigUsers
Runtime Users	SCADAPC01\SCADA.RuntimeUsers
Server Users	SCADAPC01\SCADA.ServerUsers, NT SERVICE\...
Deployment Administrators	SCADAPC01\SCADA.DeploymentAdmins
Deployment Uploaders	SCADAPC01\SCADA.DeploymentUploaders
Deployment Users	SCADAPC01\SCADA.DeploymentUsers

[Reset All](#)

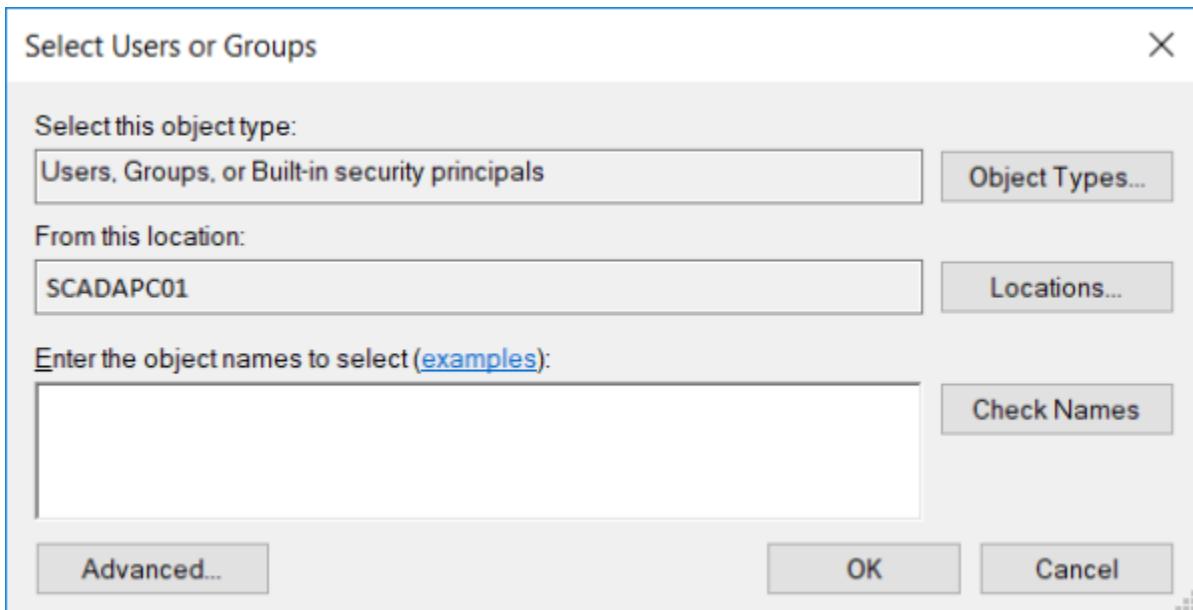
#### Members of 'Configuration Users'

Users who can start the configuration tools and set the server password

SCADAPC01\SCADA.ConfigUsers
-----------------------------

[Add](#)   [Remove](#)

- To add more members to the security role, click **Add**. The **Select Users or Groups** dialog box opens.
- Under **Enter the object names to select** type the name of local or domain group, that you want to assign to the role, and then click **OK**.



You can use the **Check Names** button to verify if the entered names are valid.

**Note:** You can add a maximum of 10 members to a security role. To manage the memberships through normal

---

Windows group management practices, it is recommended that you add a minimal number of members per security role. It is ideal to have one domain group per role.

---

### Remove local or domain groups from a security role:

1. From left side panel of the **Configurator**, select Plant SCADA | **Security Roles**.
2. Select a security role from the table. The existing members associated with that role will be listed in the **Members of...** section.
3. Select the member that you want to remove from the role.
4. Click **Remove**.

The **Plant SCADA Runtime Manager** service account is not removable from the **Server Users** role.

---

**Note:** When removing members from the security roles, make sure that each role has at least one member before you apply the changes.

---

### Applying the Changes in Security Roles:

After modifying the members of a security role, stop all the other Plant SCADA processes and services, then click **Configure** to apply the changes. This will update the security on Plant SCADA's folders and registry keys to match the new members of each role.

---

**Note:** If the Configurator displays the error message "security roles could not be updated as one or more processes are running", you can double-click on it to see all the applications that need to be manually closed and started, as well as any services that need to be restarted.

---

### Resetting the Members of Security Roles:

To reset all security roles to their default members, click **Reset All** from the **Security Roles** page. Stop all the other Plant SCADA processes and services, then click **Configure** to apply the changes. Any default user groups that no longer exist will be recreated.

### Discarding the Changes in Security Roles:

If you have not applied the changes after any modification and want to undo them, click **Close** and **Exit** Configurator.

## See Also

[Security Roles](#)

[Add the Required Users to the Runtime Users Role](#)

## Add the Required Users to the Runtime Users Role

The Runtime Users role is used to control access to Plant SCADA's runtime. Any Windows® user account that has to start runtime needs to be directly added to this role, or one of the User Groups associated with the role (see [Security Roles](#)).

If you have configured Plant SCADA to use any of the following features, you will need to ensure that the relevant user is added to the Runtime Users role so that runtime can be launched.

- If Runtime Manager is running as a service and you are not using the default service account "NT SERVICE\ Citect Runtime Manager".

- If you are using a local CTAPI application that is not being run by the currently logged in user (for example, the CTAPI application is running as a Windows service under a different user account).

To determine which users you need add into this role, check the following:

- If Runtime Manager is running as a service, check the properties of the "Plant SCADA Runtime Manager" service. If this is not running under the "NT SERVICE\Citect Runtime Manager" account, then the user this service is running under will need to be added to the Runtime Users role.
- If you are running a CTAPI application as a different user to the currently logged in user, then the user account this applications runs under needs to be added to the Runtime Users role.

## See Also

[Modifying the Members of a Security Role](#)

## Directory Security

The user and data directories that are installed by Plant SCADA (by default, under "%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>") are protected by Windows® access control lists (ACLs) to avoid unauthorized access.

During installation, access permissions for these directories will be granted (with certain operational restrictions) to the members of the following Plant SCADA Security Roles.

Directory	Security Role	Permission
Config	Configuration Users, Runtime Users and Server Users	Read/Write Access
Data	Configuration Users, Runtime Users and Server Users	Read/Write Access
Logs	Configuration Users, Runtime Users and Server Users	Read/Write Access
User	Configuration Users, Runtime Users and Server Users	Read/Write Access
Starter	Configuration Users	Read Access
Deployment\Client	Configuration Users, Runtime Users and Server Users	Read/Write Access

In addition to the above table, the required permissions will be granted to the service accounts for Plant SCADA's services. Full Control permission will also be granted to the local Administrators group and the System account.

**Note:** You can specify a different location for these directories following installation using Citect.ini parameters. If you do this, you need to manually configure the access control lists for the new folder locations so that they match the permissions that were applied to the original directories during installation. See [Configure Directory Security for Modified Folder Locations](#) in the installed Plant SCADA documentation.

## See Also

[Security Roles](#)

### Configure Directory Security for Modified Folder Locations

During a default installation, Plant SCADA's user and data directories are installed here:

"%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>"

The folders have Windows® access control lists (ACLs) applied to them to avoid unauthorized access and to enable the operation of required services.

However, path substitutions and the following INI parameters can be used to specify a different location for these directories following installation.

- [CtEdit]Config - specifies where configuration files are located.
- [CtEdit]Data - specifies where data files are located.
- [CtEdit]Logs - specifies where the log files are located.
- [CtEdit]Starter - specifies where the starter projects are located.
- [CtEdit]User - specifies where databases are located.

If you have used a path substitution or adjusted the default value for one of these parameters in a Citect.ini file or a profile, you need to manually configure the access control list for the new folder location. This is required to match the permissions that are applied to these folders during installation.

#### To configure the access control list for a new system folder location:

1. In Windows Explorer, browse to the location where the system folder was originally installed.  
For example, the default location for the Config folder is:  
%PROGRAMDATA%\AVEVAPlant SCADA2023\Config
2. Open the properties for the folder and go to the **Security** tab.
3. Keep the properties dialog open and browse to the new location for the system folder (as specified in your INI settings).
4. Open the folder properties for the new location, and go to the **Security** tab.
5. Edit the **Group or user names** list for the new location so that it matches the settings for original folder.  
The groups and users (and the permissions they require) will be different for each folder type. You need to confirm that you copy the appropriate settings to a new folder location.  
Ensure you include any service accounts that have access to the default location. When entering a service account in the **Select Users or Groups** dialog, you must prefix with "NT SERVICE\" (for example, "NT SERVICE\Citect Runtime Manager"). Select the local computer instead of any domain in the **From this location** field.
6. You then need to disable any inherited permissions for the folder via the **Advanced** button on the Security tab.

## See Also

[Directory Security](#)

[Security Roles](#)

## Runtime System Security

You can protect a Plant SCADA runtime system with user-based security. This can be integrated with existing Windows™ user groups, or you can create your own user accounts in Plant SCADA.

In both cases, users are assigned to "roles" that determine the permissions they are granted. Roles use "privileges" to restrict access to specific commands (such as those that operate specialized machinery or acknowledge critical alarms), and "areas" to control access to geographical or logical sections within a plant.

To set up security in Plant SCADA, you firstly need to consider how you will implement **areas** and **privileges** within your system.

- [Areas](#) define sections within a production facility, creating geographical or logical boundaries that can be used to restrict access.
- [Privileges](#) define the level of access that is applied to system elements within your project.

Once you have planned how areas and privileges will be managed (and how they can work in tandem), you can start adding **roles** and **users** to the system.

- [Roles](#) define a set of permissions (based on privileges and areas) that can be assigned to users of the same type.
- [Users](#) represent the individuals (or groups) that need to access to the runtime system.

You can define users in Plant SCADA, or you can integrate Windows™ user groups. Every user needs to be assigned to a role to be granted privileges.

This section of the help also describes how to limit access to a computer while Plant SCADA is running (see [Securing Runtime Computers](#)).

## Areas

Areas allow you to visualize a production facility as a set of discrete sections. You can define areas geographically (especially if parts of the plant are separated by distance or physical barriers), or logically (as discrete processes or individual tasks).

You can define up to 255 separate areas. You can then refer to these areas by number (1 to 255) or use a label to assign a meaningful name to the area (see [Use a Label to Name an Area](#)).

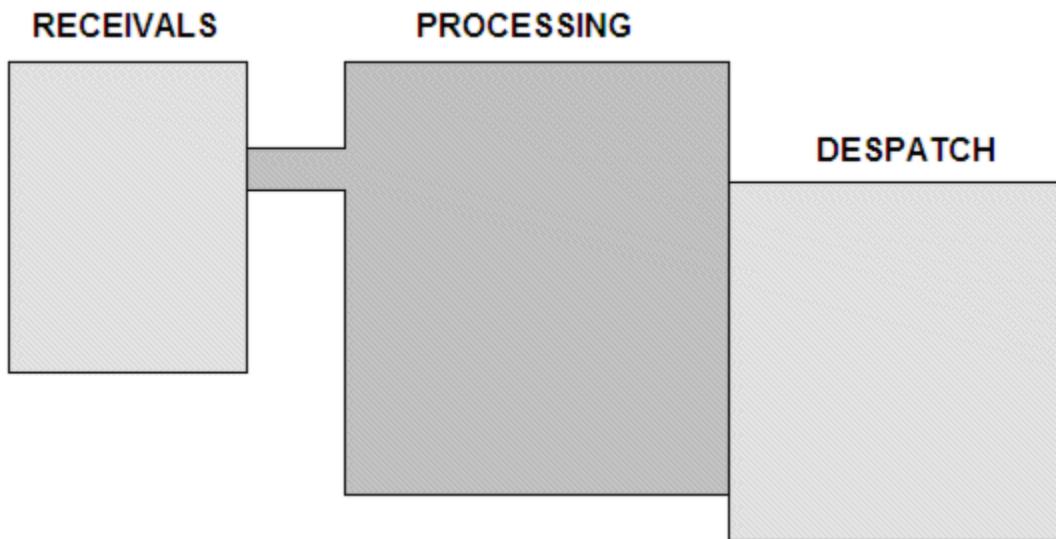
After you have defined your areas, you can then configure the system elements your operators will use in those areas, such as commands, objects, alarms and reports. Refer to [Roles](#) for more information on how areas and roles work together.

---

**Note:** Any system element that is not assigned to an area between 1 and 255 is automatically placed in a default area known as Area 0. Every user can view the system elements in Area 0, but without the matching privilege will be unable to control them.

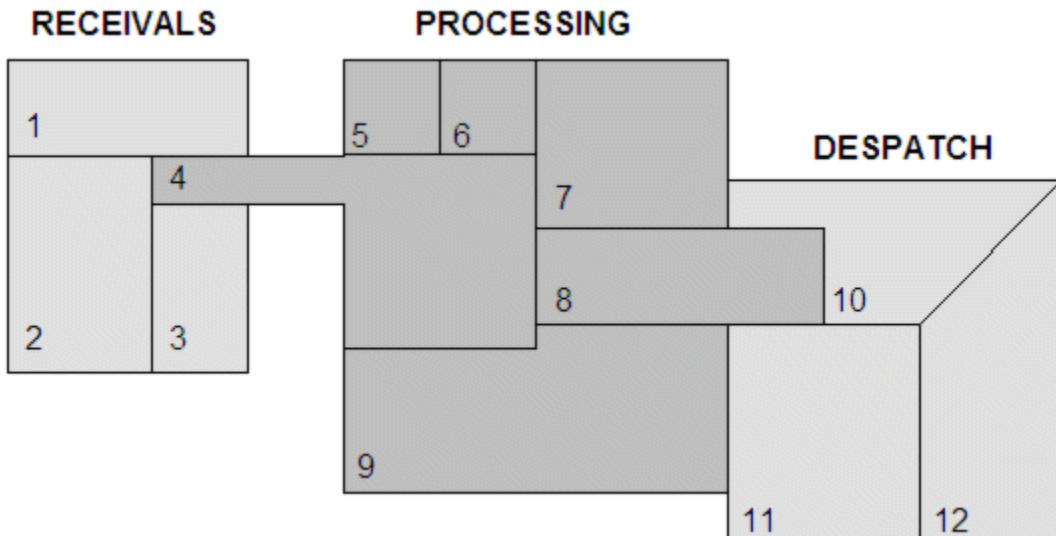
## Example

A simple manufacturing plant could be divided into just three areas - receipts, processing, and dispatch.

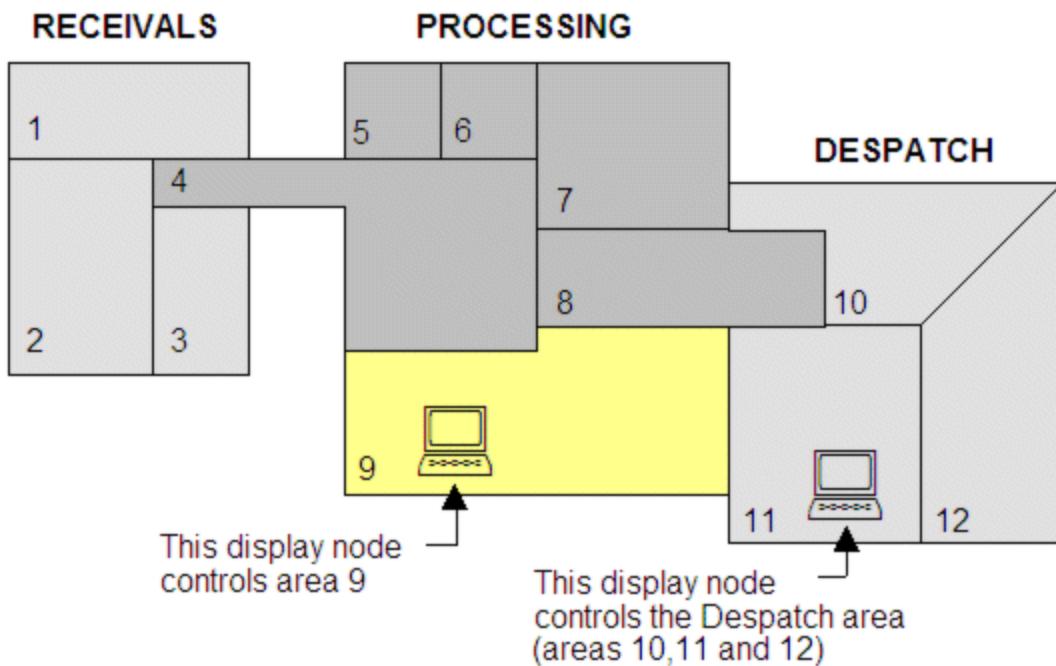


This is an effective scenario if there is a single Plant SCADA control client used to monitor each area.

With a larger or more complex facility, you might need to define several areas, like this:



This demonstrates how you can define smaller areas that are collectively controlled by an operator or control client. This method can increase flexibility, but can introduce a higher level of complexity to your system.



You can simplify the configuration of a complex system by creating a group from multiple areas. For example, areas 10, 11 and 12 in the example above could be grouped and labeled as "Despatch" (see [Create an Area Group](#)).

You can also provide view-only access to an area (see [Configure View-only Access to an Area](#)).

## See Also

[Privilege and Area Combinations](#)

[Runtime System Security](#)

## Use a Label to Name an Area

It can be easier to remember an area by a meaningful label (name) rather than a number. For example:

Label Name	DespatchAccum
Expression	10
Comment	Label Area 10 as "DespatchAccum"

This would allow you to use "DespatchAccum" whenever you need to refer to area 10. For example:

Command	CONVEYOR = 1;
Area	DespatchAccum
Comment	This command belongs to Area 10 (DespatchAccum)

**Note:** If you leave the **Area** field blank on a form, the command does not belong to any particular area - it is assigned to every area of the plant.

**To label an area:**

1. In the **Standards** activity, select **Labels**.
2. In a blank row, click in the **Project** cell and select the required project.
3. Enter a **Label Name** for the label.
4. Enter an expression to be substituted for the label, for example, the number of the area you would like the label to represent.
5. Enter a relevant **Comment**, if required.
6. Click **Save**.

**Note:** To modify an existing label, click in the individual cells and select or enter the required value.

**See Also**

[Create an Area Group](#)

**Create an Area Group**

You can group several areas together and define a name for the group.

In the example below, areas 10, 11, and 12 are associated with the name "Despatch".

Group Name	Despatch
Association 1	DespatchAccum
Association 2	11
Association 3	12
Comment	Areas 10, 11, 12 = "Despatch"

This means any command assigned to "Despatch" will belong to areas 10, 11, and 12.

You can also define a group that includes other groups. In the example below, the name "Plantwide" refers to the areas defined in the "Receivals", "Process", and "Despatch" groups.

Group Name	Plantwide
Association 1	Receivals
Association 2	Process
Association 3	Despatch
Comment	Associate every area with "Plantwide"

**To define a group of areas:**

1. In the **Standards** activity, select **Groups**.
2. Add a row to the Grid Editor.

3. Enter a **Group Name**.
4. Complete the required **Association** fields.
5. Click **Save**.

## See Also

[Add a Group](#)

### Configure View-only Access to an Area

You might need to provide a user access to information from areas within a plant, even though you do not want that user to control any of the processes within those areas. For example, the user may need to monitor the processes in one area as they might directly affect another area.

In the following example, John Smith has been assigned the role of Despatch Handler and as such has been granted control of:

1. System elements located in **Despatch**, with a privilege level of **1**
2. System elements located in **DespatchAccum**, with a privilege level of **4**
3. View-only access Plantwide to track when product is due to arrive in Despatch for packing and distribution.

User Name	J Smith
Global Privilege	
Viewable Areas	Plantwide
Areas for Priv 1	Despatch
Areas for Priv 2	
Areas for Priv 3	
Areas for Priv 4	DespatchAccum
Areas for Priv 5	
Areas for Priv 6	
Areas for Priv 7	
Areas for Priv 8	
Comment	Login for John

Alternatively, you could restrict users access to a group of areas (for example, "Receivals") or to a single area (for example, 12).

## See Also

[Privilege and Area Combinations](#)

## Privileges

Plant SCADA provides eight privileges, numbered 1 to 8, that you can use to restrict access to some of the elements in a runtime system. These elements include:

- Graphics pages
- Graphics page objects
- Alarms
- Trends
- SPC tags
- Accumulators
- Keyboard commands
- Reports.

To implement privileges in a Plant SCADA system, you need to perform the following:

- Assign privileges to the required elements (using the **Privilege** property for each element)
- Assign matching privileges to the [Roles](#) that require access to those elements.

When configuring privileges for a role, you can assign global privileges that apply to all [Areas](#), or you can assign privileges for specific areas.

The properties that define privileges for a role include the following:

Role Property	Description
Privileges	Sets global privileges for the role (that apply to all areas).  If you assign a role a global privilege, that role is also granted view access to every area automatically. Any user assigned that role will be able to view every area of the plant.
Priv 1 Areas ... Priv 8 Areas	Sets privileges for specific areas. For example, if "Priv 1 Areas" is set to "1,2,3", users assigned to this role will have privilege 1 access to Areas 1,2 and 3.
View Areas	Allows users assigned to the role to view the specified areas.

For more information about the configuration of areas and privileges, including some examples, see [Privilege and Area Combinations](#).

Not every system element needs a privilege classification. For example, every user will require access to a login

command. A blank privilege setting (or privilege 0) means that an element has no classification and will be accessible to all users.

**Note:** You can use the Citect.ini parameter [Privilege]Exclusive to implement hierarchical privilege. This means users with privilege 3 have access to commands with a privilege classification of 3, 2, and 1. To allocate every privilege, you only need to specify privilege 8.

## See Also

[Users](#)

[Runtime System Security](#)

## Privilege and Area Combinations

Outlined below are four general rules regarding the use of privileges and areas within Plant SCADA.

1. Global privileges apply to every area.
2. Assigning a privilege to an area within a role means any user assigned that role will gain viewable access to that area automatically. However the user can only operate system elements in that area that have a matching privilege. As a result of the first rule, if users are assigned a global privilege they will also be able to view every area.
3. Area 0 includes every privilege a role may have been assigned in other areas. In other words, if granted a privilege 3 for use in another area that role can also control those system elements in Area 0 that have a privilege set as 3.
4. All users can view Area 0.

The table below outlines numerous scenarios, and the resulting security that is applied to an alarm when accessed by a user assigned to an "operator" role.

Alarm Properties	Operator Role Properties	Result
Area = 0 <All areas> Privilege = 0 <None>	View Areas = 0 (blank) Global privileges = 0 (blank) Privilege 1 Areas = 0 (blank) Privilege 2 Areas = 0 (blank) ...	No privileges are assigned to the alarm, or the operator role. An operator will be able to view and acknowledge the alarm.
Area = 0 <All areas> <b>Privilege = 1</b>	View Areas = 0 (blank) Global privileges = 0 (blank) Privilege 1 Areas = 0 (blank) Privilege 2 Areas = 0 (blank) ...	The alarm is assigned level 1 privileges. An operator will be able to view the alarm, but cannot acknowledge it as the operator role does not have the necessary level 1 privileges.
Area = 0 <All areas> <b>Privilege = 1</b>	View Areas = 0 (blank) <b>Global privileges = 1</b> Privilege 1 Areas = 0 (blank)	An operator can view the alarm and acknowledge it, as the operator role has been granted

Alarm Properties	Operator Role Properties	Result
	Privilege 2 Areas = 0 (blank) ... ...	matching global privileges (level 1 access). The operator will also be able to view and control other system elements that have level 1 privileges across all areas of the plant.
<b>Area = 1</b> Privilege = 0 <None>	View Areas = 0 (blank) Global privileges = 0 (blank) Privilege 1 Areas = 0 (blank) Privilege 2 Areas = 0 (blank) ...	An operator cannot view the alarm, as it is now assigned to Area 1 and the operator role has no permissions for Area 1.
<b>Area = 1</b> Privilege = 0 <None>	<b>View Areas = 1</b> Global privileges = 0 (blank) Privilege 1 Areas = 0 (blank) Privilege 2 Areas = 0 (blank) ...	The View Areas property has been adjusted so that users assigned to the operator role can view Area 1. They can acknowledge the alarm as it has no privilege restrictions.
<b>Area = 1</b> <b>Privilege = 1</b>	View Areas = 1 Global privileges = 0 (blank) Privilege 1 Areas = 0 (blank) Privilege 2 Areas = 0 (blank) ...	An operator can view the alarm in Area 1, but cannot acknowledge it as the operator role does not have the required level 1 privileges.
<b>Area = 1</b> Privilege = 1	View Areas = 1 <b>Global privileges = 1</b> Privilege 1 Areas = 0 (blank) Privilege 2 Areas = 0 (blank) ...	An operator can view the alarm and acknowledge it as the operator role now has global privileges for level 1.
<b>Area = 1</b> Privilege = 1	View Areas = 0 (blank) Global privileges = 0 (blank) <b>Privilege 1 Areas = 1</b> Privilege 2 Areas = 0 (blank) ...	An operator can view the alarm and acknowledge it as the operator role now has level 1 privileges for the matching area (Area 1).
<b>Area = 2</b> Privilege = 1	View Areas = 0 (blank) <b>Global privileges = 1</b> Privilege 1 Areas = 0 (blank) Privilege 2 Areas = 0 (blank)	The alarm is now in Area 2, however, an operator can still view the alarm and acknowledge it as the operator role has global privileges for level 1.

Alarm Properties	Operator Role Properties	Result
	...	

## See Also

[Roles](#)

[Areas](#)

## Roles

Roles define a set of permissions that can be assigned to users of the same type. Before you create a role, determine the permissions required by the users that will be assigned to the role (based on the available [Privileges](#) and [Areas](#)).

To integrate Windows user groups into your Plant SCADA security, use the **Windows Group** property when defining a role. See [Integrate Windows™ User Groups](#).

---

**Note:** Area 0 is assigned by default to every role. This means users can view any system element in Area 0 (no privileges defined).

### To add a Role record:

1. In the **Security** activity, select **Roles**.
2. Add a row to the Grid Editor.
3. Type the required information in each column, or in the fields in the Property Grid.  
For a description of the properties, see below.
4. Click **Save**.

## Role Properties

### General Properties

Property	Description
<b>Role Name</b>	The name of the role. Each name must be unique.
<b>Windows Group</b>	<p>The name of the Windows™ user group associated with this role. You can enter a group name on its own (for example, "PlantOperators"), or you can restrict the group's accessibility by including a local computer name or domain name (for example, "ComputerName\PlantOperators" or "DomainName\PlantOperators").</p> <p>You can only associate Windows user groups with up to 1024 Plant SCADA roles. Duplicated Windows user groups are not supported.</p>

Property	Description
	For more information, see <a href="#">Integrate Windows™ User Groups</a> .
<b>Privileges</b>	<p>The privilege assigned globally to the role. Enter a value of 16 characters or less.</p> <p>In the privilege field you can separate numbers with commas or you can enter a range separated by two periods, for example, 1..8.</p> <p>As you configure your system, you can assign privileges to the various elements, such as graphics objects, alarms, accumulators, commands, and so on. For example, a role with a Global Privilege of 3 will be able to send any command that is assigned a privilege of 3, or action any alarm with a privilege of 3, or click any button that is assigned a privilege of 3, etc. Unless you are using areas, if you do not specify a global privilege, the role cannot access any command with a privilege assigned.</p> <p><b>Note:</b> (For users using windows authentication) When you have completed the fields in this dialog and if you have not already done so, add the users to the group in Windows security that you want to have the privileges of this role.</p>
<b>View Areas</b>	<p>The areas the user assigned the associated role is permitted to view. Enter a value of 16 characters or less.</p> <p><b>Note:</b> Do not set Viewable Areas in conjunction with Global privileges, as global privileges give roles view access to areas automatically.</p> <p>Remember, you need to still assign privileges to the elements in these viewable areas, such as graphics objects, alarms, accumulators, commands, etc. If you do not, the user will have full access to them. For example, if you do not assign a privilege to a command in one of these areas, the user will be able to send it regardless of whether you want them to or not.</p> <p>To make an element (such as a button on a expression) view only for a particular user, assign it an expression and a privilege. Add the area to the user's list of Viewable Areas, but don't give the user the necessary privileges in that area (or the necessary global privilege).</p> <p>Multiple areas can be defined using groups.</p>

Property	Description
	<p>If you do not specify "Viewable Areas", the user will have viewable access to area 0. See <a href="#">Privilege and Area Combinations</a> for more information.</p>
<b>Allow RPC</b>	<p>Determines if a user or group will be restricted from performing remote MsgRPC and ServerRPC calls.</p> <p>From the drop-down, select <b>True</b> or <b>False</b>:</p> <ul style="list-style-type: none"> <li>• True - the user or group will be allowed to run MsgRPC and ServerRPC</li> <li>• False - the user or group will not be allowed to run MsgRPC and ServerRPC.</li> </ul> <p>If the field is left blank, it will default to FALSE. The following compiler warning message will be generated:</p> <p>"'Allow RPC' permission is not defined (defaulting to FALSE)."</p> <p><b>Note:</b> If you want to use MsgRPC to call a procedure on a remote client computer, you will need to set the parameter [Client]AllowRPC to 1 on the client computer.</p>
<b>Allow Exec</b>	<p>Determines whether a user or group will be allowed to run the Exec Cicode function.</p> <p>From the drop-down select True or False.</p> <ul style="list-style-type: none"> <li>• True - user or group allowed to run Exec.</li> <li>• False - user or group not allowed to run Exec.</li> </ul> <p>In the Example project, Allow Exec is set to TRUE for the Engineer role. Note that this is used in conjunction with the Citect INI parameter <b>[Security]BlockExec</b>. Therefore, the parameter also needs to be set as <b>[Security]BlockExec=0</b> so that users with this role can run the Exec Cicode function. For more information about the parameter, refer to the Parameters help.</p> <p>If the field is left blank, Allow Exec will default to FALSE.</p>
<b>Manage Users</b>	<p>Determines if the user is authorized to manage user accounts. From the drop-down select TRUE or FALSE.</p> <p>If TRUE the user is able to:</p> <ul style="list-style-type: none"> <li>• Add, modify, and delete users at runtime.</li> </ul>

Property	Description
	<ul style="list-style-type: none"> <li>• Modify other users passwords without having to know the user's old password.</li> </ul> <p>If FALSE the user will only be able to change their own password. To do this they will need to know their old password.</p> <p>In the Example project, Manage Users is set to TRUE for the Engineer role.</p>
<b>Kernel Access</b>	<p>Determines if a user can launch the Kernel window at runtime. Choose from the following:</p> <ul style="list-style-type: none"> <li>• <b>No Access</b> - the user cannot launch the Kernel.</li> <li>• <b>Read Only</b> - the user can display the Kernel, but cannot perform some privileged commands like running Cicode.</li> <li>• <b>Full Access</b> - the user can display the Kernel and has full access to its functionality and commands.</li> </ul> <p><b>Note:</b> You need to restrict access to the Kernel. Anyone using the Kernel has total control of Plant SCADA (and subsequently your plant and equipment). For more information, see the section <i>Access to Cicode and Cache Commands</i> in the topic <a href="#">The Kernel</a>.</p> <p><b>Note:</b> If you change the Kernel Access setting for a Role and run the recompiled project, you will need to restart any server processes that you want to run the Kernel on.</p>
<b>Comment</b>	Any useful comment.
<b>Entry Command</b>	A Cicode command that is executed when the user assigned this role logs in. You can use any Cicode command or function. Enter a value of 254 characters or less.
<b>Exit Command</b>	A Cicode command that is executed when the user assigned this role logs out. You can use any Cicode command or function. Enter a value of 254 characters or less.
<b>Priv1 Areas... Priv8 Areas</b>	The privileges (by area) assigned to the user. Enter a value of 16 characters or less. Using this combination of areas and privileges, you can assign a user different privileges for different areas. For example, users assigned a role with privilege class 6 in areas 29 and

Property	Description
	<p>30 will only have access to commands in those areas that require privilege class 6.</p> <p>In the privilege field you can separate numbers with commas or you can enter a range separated by two periods, for example, 1..8.</p> <p><b>Note:</b> In assigning a privilege to an area, you are making that area viewable to users assigned that role.</p> <p>If you do not specify areas with associated privileges, access is defined by Viewable Areas or Global Privileges only.</p>

### Project Properties

Property	Description
Project	The project in which the role is included.

### See Also

[Runtime System Security](#)

[Users](#)

### Users

Plant SCADA runtime security supports two types of users:

- Native Plant SCADA users
- Windows™ users.

Every user needs to be assigned to a [Roles](#) to be granted privileges and access to areas.

If you want to use native Plant SCADA users, you define them in the Security activity. See [Add a Plant SCADA User](#).

If you want to use Windows users, they need to be part of a Windows User Group that is associated with a role. See [Integrate Windows™ User Groups](#) for more information.

---

**Note:** Each operator needs to enter a User Name and Password to use the Plant SCADA runtime system. To allow a user to login, you can use the [Login\(\)](#) or [LoginForm\(\)](#) Cicode functions. You can also use the [\[General\]PasswordExpiry](#) parameter to specify when a password will expire.

### See Also

[Runtime System Security](#)

[Roles](#)

## Add a Plant SCADA User

Before you add a Plant SCADA user to your project, check the [Reserved User Names](#) list. This list details built-in users already included in Plant SCADA.

**Note:** You can also use Windows™ user groups to manage runtime security. See [Integrate Windows™ User Groups](#) for more information.

### To add a user:

1. In the **Security** activity, select **Users**.
2. Add a row to the Grid Editor.
3. Type the required information in each column, or in the fields in the Property Grid.  
For a description of the properties, see below.
4. Click **Save**.

You can also group several Plant SCADA users together and define a name for the group. See [Add a Group](#).

**Note:** If you restore a project from a backup, or install a new project from a compiled offline master, the runtime user details might be thrown out of synchronization. For more information, see [User Records and Project Restoration](#)).

## User Properties

### General Properties

Property	Description
<b>User Name</b>	The user's name. Enter a value of 16 characters or less. User Names are restricted to using the same syntax as Tag names. See Tag name syntax.
<b>Full Name</b>	The full name of the user or class of user. Enter a value of 32 characters or less. This name is used as a comment and for display in alarm logs and command logs.
<b>Password</b>	The user's password. When you double click in this cell or when you select <b>Change</b> next to the Password field in the Property Grid, the Set Password dialog is displayed. In the <b>New Password</b> field, enter a value of 16 characters or less. Enter the same password in the <b>Confirm Password</b> field and click <b>OK</b> . When you enter the passwords six dots (.) will be displayed for the password. When you save the user record, the password will be encrypted before it is saved to the Users.dbf. If the contents of the <b>Password</b> and <b>Confirm Password</b> fields are different when the record is saved, a message will be displayed that indicates a mismatch and invites you to try again.

Property	Description
<b>Roles</b>	<p>Each user is assigned roles. The Roles field will accept zero or more comma-separated role names. If zero roles are specified for the user, this is the same as configuring the user with no privileges.</p> <p><b>Note:</b> When a Windows or Plant SCADA user who is linked to multiple roles logs onto runtime the privileges and areas that the user will be assigned are the combined privileges of the linked roles.</p>
<b>Type</b>	<p>The generic type of user. Enter a value of 16 characters or less. For example: Operator, Supervisor or Manager.</p> <p>The Type field is used in configuration only to specify a class of user that can then be used as the basis for creating new users in runtime via the <a href="#">UserCreate()</a> Cicode function. When this function is run it displays a form where you can select the user Type. When you do this, your new user will inherit the properties of the chosen user class record that you have already created. In doing this it uses the first user record with a matching Type value.</p> <p>In configuration, decide what user classes you need. Each class (or Type) would contain any specific Global Privilege, Viewable Areas and Areas for Privilege and Entry and Exit Commands values . Maintain just one user record for each class and then base other individual users on this. If you create new users in configuration in the usual way by using Add, Replace technique, then you will get multiple user records with the same Type field value. Whilst this will not directly cause a problem for individual records it could confuse development or provide scope for inconsistencies and unexpected behavior if other field values are changed. When adding records in configuration, it is therefore recommended that you remove duplicate Type values from additional user records.</p>
<b>Comment</b>	Any useful comment. Enter a value of 48 characters or less.
<b>Password version</b>	Indicates if a password is a legacy password. A new password (created in Version 2015 or later) will display "2". A legacy password will display "1". It is recommended that update any legacy passwords.
<b>Seed</b>	Indicates if each password is uniquely encrypted. If

Property	Description
	<p>you duplicate a user account (by copying and pasting a row), the password for each will have the same seed. This will cause generate a compile error message. To solve this problem, you need to change the password for each account.</p> <p>If you need to keep the same password for the duplicated accounts, you can select all of the duplicated users in the Grid Editor, and <b>Change</b> the password in the Property Grid. This will generate new seeds for each of the duplicated accounts.</p>

### Project Properties

Property	Description
Project	The project in which the user is included.

### Reserved User Names

The following user names have been reserved and should not be used when you add users to your Plant SCADA system.

- driver
- logic
- method call
- schedule
- system
- guest.

If you attempt to log in with one of these reserved user names, a hardware alarm will be raised and the login will be unsuccessful.

**Note:** You should also avoid using these reserved user names for [Integrate Windows™ User Groups](#) that will log in to Plant SCADA.

### See Also

[Add a Plant SCADA User](#)

### User Records and Project Restoration

If you restore a project from a backup, or install a new project from a compiled offline master, any user records created in Plant SCADA are reset to match those originally configured in the project. If the runtime user creation, password change ability, or password expiry functions are used, the runtime details might be thrown out of synchronization with master offline projects.

Here, you need to have procedures in place to use the current Users.dbf file (which is running live in the plant) when offline project compilations are performed. This minimizes the likelihood of either deleting users created at runtime, or of having expired user records locked when a new system is deployed and run.

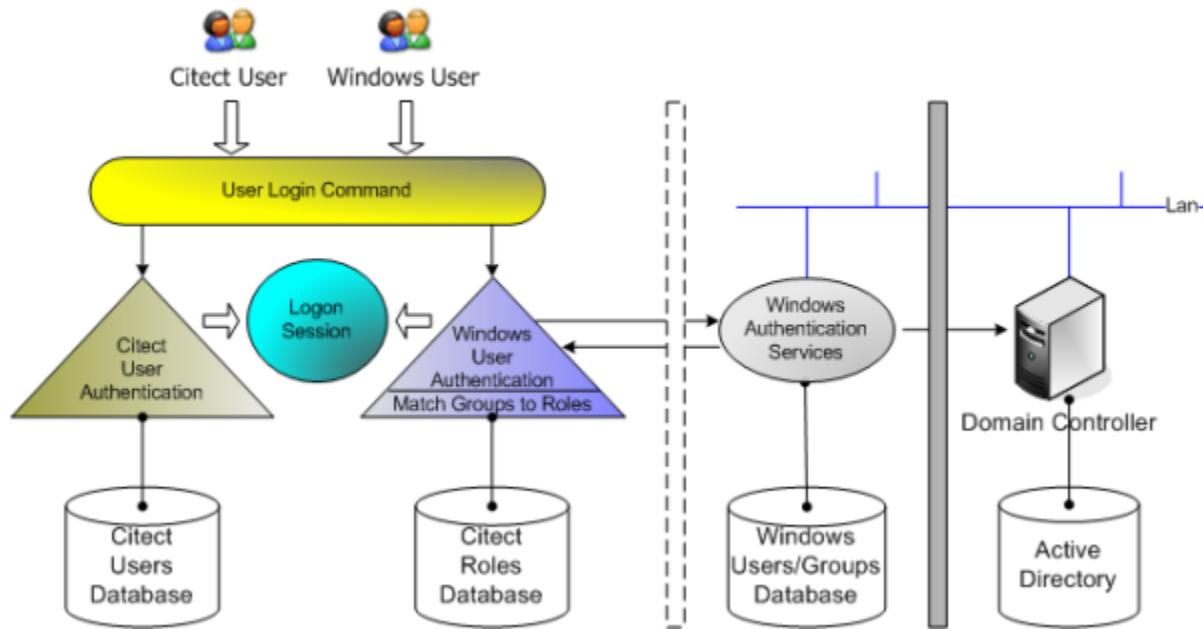
**Note:** Online changes arising from user creations and modifications are reflected only in the local \_Users.rdb and Users.dbf files. Perform user administration activities on a central node so that user records remain synchronized across a distributed network. Other nodes will use the Copy= functionality in Plant SCADA or custom engineered database replication.

## See Also

[Add a Plant SCADA User](#)

## Integrate Windows™ User Groups

Plant SCADA provides the ability to integrate Windows™ user accounts into the security model for a runtime system. You can achieve this by linking a Windows user group to a [Roles](#) defined within Plant SCADA. This allows each user within the Windows user group to log in to Plant SCADA with the permissions defined for the associated role.



To associate a Windows user group with a role, you use the **Windows Group Name** property. You can enter a group name on its own (for example, "PlantOperators"), or you can restrict the group's accessibility by including a local computer name or domain name (for example, "ComputerName\PlantOperators" or "DomainName\PlantOperators").

If you include a domain path with a Windows group name, the Plant SCADA computer where login occurs will need to belong to that domain for the login to be successful. Similarly, if you specify a Windows user group that is local to a particular computer, a Plant SCADA login will only be successful on that computer.

If the **Windows Group Name** property does not specify a local computer name or domain name, any domain computer can be used to authenticate users.

If the same user name exists in both the Plant SCADA users database and an associated Windows user group, authentication will firstly be attempted against the account in the Plant SCADA user database.

For more information on Windows user accounts across a number of different login scenarios, see [Windows™ Security Usage Scenarios](#).

For information on how to add groups and users to Windows, refer to the Windows documentation appropriate to your operating system.

---

**Note:**

- The AutoLogin capability has also been extended to include Windows users that are associated with a role. In order to invoke this functionality for a Windows user you need to set the [\[Client\]AutoLoginMode](#) parameter in the Citect.ini file.
  - Integrated Windows security can be used with the [MultiSignatureForm](#) Cicode function if multiple-user authorization is required.
- 

## See Also

[Roles](#)

### Windows™ Security Usage Scenarios

The following describes how a user will be allowed to negotiate access to a Plant SCADA runtime system when using Windows groups and roles.

---

**Note:** Some scenarios will not be supported if you use a local Windows user to access a Plant SCADA computer. It is recommended that you always use domain Windows user accounts.

---

When a Windows user logs on to a display client, authentication is performed locally on the Plant SCADA process that is running.

If you are running a standalone system with all processes running locally, you can use a local Windows user to log on.

If the display client connects remotely to one or more Plant SCADA servers that are distributed across multiple domains, a domain user will be required. You also need to make sure that all the Plant SCADA computers are on trusted domains. If the domain controllers are unavailable, you will not be able to log on to the display client.

Auto login (for example, a login that occurs following a restart) will work if the current user meets the criteria described above.

## See Also

[Integrate Windows™ User Groups](#)

## Securing Runtime Computers

The Plant SCADA runtime system is a Windows™-based application. Typically, Windows allows you to run several applications at the same time, however, this may impact performance or obscure runtime messages and information if you require a computer to be dedicated to Runtime.

For example, an operator display panel may be used to present alarm notifications. You will not want the runtime screen minimized or hidden behind another window.

There are several different ways can limit access to software other than Plant SCADA.

## See Also

- [Client Startup Restrictions](#)
- [Running a Display Client as a Shell](#)
- [Disabling Windows Keyboard Commands](#)
- [Disabling Control Menu Commands](#)
- [Removing the Cancel Button](#)

### Client Startup Restrictions

On start up Plant SCADA will establish initial communication with the server using a view-only login. By default this communication between the client and the server is in view-only mode. In this mode the user cannot write to any tag, acknowledge any alarm or use Cicode functions.

**Note:** View-only mode is applied to the whole control client process, including any Cicode task that is running.

Write only access is available after a user has successfully logged in. Once the user logs out it returns to view-only mode.

Users can configure login by modifying the [\[Client\]AutoLoginPage](#) parameter.

## See Also

- [Securing Runtime Computers](#)

### Running a Display Client as a Shell

To limit certain operators from switching a client over to a different application during runtime, you can configure Windows™ to launch with Plant SCADA running as the shell. This will deploy Plant SCADA Runtime as the only available interface for a computer, limiting access to within the context of the current project.

For information on how to set up a client as a shell, contact your local support office.

## See Also

- [Disabling Windows Keyboard Commands](#)

### Disabling Windows Keyboard Commands

The Windows™ environment provides commands to switch between applications running on the computer at the same time. When using Plant SCADA, these commands might not be desirable - they allow an operator access to other Windows facilities without your direct control. You might be able to disable some of these commands with the Setup Wizard (see [Keyboard Security Configuration](#)).

## See Also

- [Disabling Control Menu Commands](#)

## Disabling Control Menu Commands

The runtime system's Control Menu can be tailored to give access to several commands specific to Plant SCADA, such as Shutdown (to shut down the runtime system), or Kernel (to display the Kernel).

You can enable and disable these commands with the Setup Wizard (see [Control Menu Security Configuration](#))..

## See Also

[Removing the Cancel Button](#)

### Removing the Cancel Button

When the Plant SCADA runtime system starts, a message box displays the status of the system startup. This message box normally contains a **Cancel** button that allows you to cancel the startup. This button is useful when you are debugging or testing the system. When you have completed testing, you can remove the **Cancel** button from the message box with the Setup Wizard (see [Miscellaneous Security Configuration](#)), so that there will not be an unintentional cancellation of system startup.

## Standards

The Standards activity allows you to create and configure assets that are intended for use across multiple Plant SCADA projects. For example, you may need to configure alarm fonts that adhere to a corporate safety strategy.

The following features are configured in the Standards activity:

- [Labels](#)

Labels allow you to use a series of commands (or expressions) in your system without having to repeat them each time they are used. When you compile the project, the commands (or expressions) defined in the label are substituted in every occurrence of the label.

Labels are similar to macros used in programming languages such as Basic or 'C'.

- [Styles](#)

Styles can be used to manage the appearance of symbols, status animations and numeric values in AVEVA™ Industrial Graphics applications.

- [Groups](#)

Groups can be used for many purposes, for example, to define a set of users, multiple areas or multiple devices.

- [Fonts](#)

Alarm categories and Cicode functions allow you to use predefined fonts, known as **system fonts**, to display text.

## Labels

Labels allow you to use a series of commands (or expressions) in your system without having to repeat them each time they are used. When you compile the project, the commands (or expressions) defined in the label are

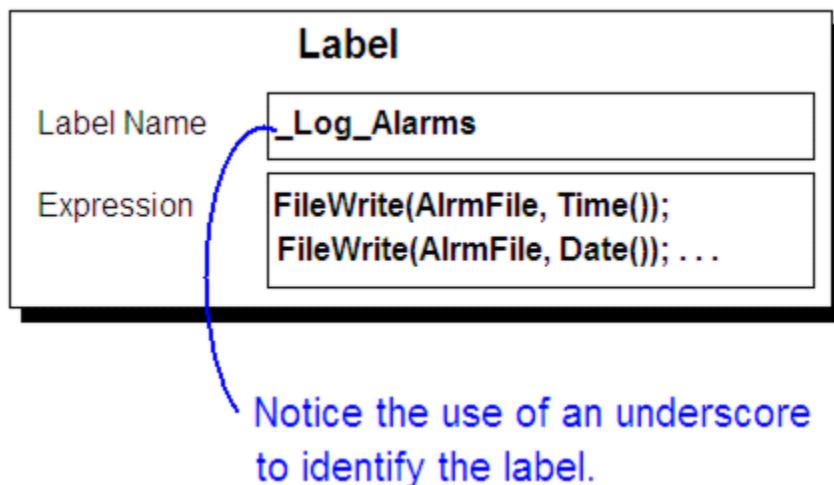
substituted in every occurrence of the label.

Labels are similar to macros used in programming languages such as Basic or 'C'.

You often use the same combination of statements in different commands. For example, when an operator acknowledges an alarm, you might want to log the details to a file. Such a command would require several statements:

Command	FileWrite(AlrmFile, Time()); FileWrite(AlrmFile, Date()); ...
---------	--

Instead of entering the same statements when necessary in a command, you can define a label, and then use the label instead of the statements. When you compile your project, each occurrence of the label is resolved; that is, the expression in the label is substituted for the label name. For example:



Once defined, a label can be used as a statement in a command, for example:

System Keyboard	
Key Sequence	F5 Enter
Command	_Log_Alarms;
Privilege	1

When an operator issues this command, the expression defined in the label is substituted in the command.

**Label**

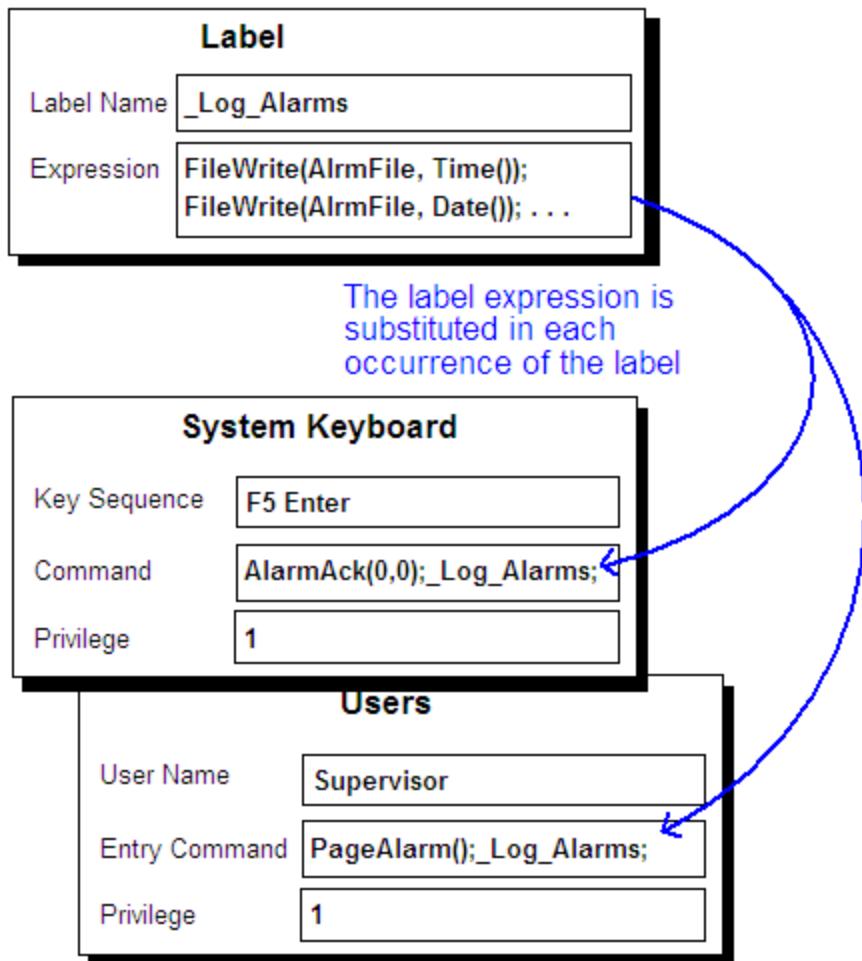
**Label Name** \_Log\_Alarms

**Expression** FileWrite(AlrmFile, Time());  
FileWrite(AlrmFile, Date()); ...

When an operator issues the command, the statements in the expression execute

System Keyboard	
Key Sequence	<u>F5 Enter</u>
Command	<u>_Log_Alarms;</u> ←
Privilege	1

You can also use the label in combination with other statements, for example:



The main advantage of a label is that it is a global definition, recognized throughout the Plant SCADA system. If you want to change something (in the above example you might change the file name or the way the data is logged), you only need to change it in one place - in the label definition. Other occurrences of the label name will reflect the changes.

## See Also

- [Define a Label](#)
- [Use Arguments in Labels](#)
- [Convert Values into Strings](#)

## Define a Label

**Labels** allow you to reference a set of commands (or expressions) that are used repeatedly across a Plant SCADA project.

### To define a label:

1. In the **Standards** activity, select **Labels**.
2. Add a row to the Grid Editor.

3. Type the required information in each column, or in the fields in the Property Grid.
- For a description of the properties, see below.
4. Click **Save**.

## Label Properties

### General Properties

Property	Description
<b>Label Name</b>	A name of 128 characters or less. Whenever this name is used (for example, in Cicode or a field), Plant SCADA will automatically substitutes the expression below.
<b>Expression</b>	The expression to be substituted for the label (maximum length is 254 characters). You can use a label to substitute a name for an entity or a Cicode expression; for instance, when you need to use the entity or Cicode expression in several database records.
<b>Comment</b>	Any useful comment.

### Project Properties

Property	Description
<b>Project</b>	The project in which the label is included.

## See Also

[Use Arguments in Labels](#)  
[Convert Values into Strings](#)

## Use Arguments in Labels

You can [Define a Label](#) that accept arguments enclosed in parentheses (). The following example shows a label that increments a variable by a specific value:

Label Name	Inc(X, STEP)
Expression	X = X + STEP

Here, "X" is the variable to be incremented and "STEP" determines the amount of the increment. You can then use this label in a command, as in the following example:

Key Sequence	FastInc
Command	Inc(SP12, 10);

An operator can use this command to increment the value of SP12 by 10.

You can also use arguments in a label in the following ways:

### Specify a Default Value for an Argument

You can specify a default value for an argument when you define a label, for example:

Label Name	Inc(X, STEP = 10)
Expression	X = X + STEP

When you subsequently use this label without any arguments in a command, the default value is used, for example:

Key Sequence	FastInc
Command	Inc(SP12);

### Use a Substitution String in an Argument

You can pass a string substitution as an argument in a label, for when several variables have part of the variable name in common; for example:

Label Name	SPDev(TAG)
Expression	Prompt("Deviation=" + "IntToStr(CP##TAG## - SP##TAG##));

In the above example, TAG is the common portion of the variable name, and is substituted at each occurrence in the expression. To display the difference between two variables CP123 and SP123, you would specify SPDev(123) in a command, for example:

Command	SPDev(123);
---------	-------------

You cannot use a substitution within a string. In the following example, the DESC Parameter (a text description) will not be substituted as it is between quotation marks:

```
Prompt("Deviation for ##DESC##=" + "IntToStr(CP##TAG## - SP##TAG##))
```

## See Also

[Convert Values into Strings](#)

## Convert Values into Strings

Sometimes, you need to convert a value into a string before it can be used. In the following example, the value of a tag is converted before it is used in the DspStr() function.

Label Name	ShowVariable(TAG)
Expression	DspStr(25, "BigFont", #TAG + "=" + TAG:#.#);

In the above example, only one argument (TAG) is passed to a function that actually requires three arguments

(AN, font and message). When you use this label in a command, the function uses AN 25 and the message displays in "BigFont". Only the third argument (the actual message) varies.

The third argument passed to the function is:

```
...#TAG+"="+TAG:##.#
```

#TAG indicates that the name of the tag (and not its value) is displayed.

TAG:##.# indicates that the value of TAG is converted to a string and displayed. It is formatted with two numbers before the decimal point and one number following the decimal point.

You can use the above label in a command such as:

Command	ShowVariable(SP12);
---------	---------------------

When you use this command in your runtime system, the command displays "SP12=<value>", where **value** is the actual value of SP12 at the time (for example **SP12=42.0**).

## See Also

[Use Arguments in Labels](#)

## Styles

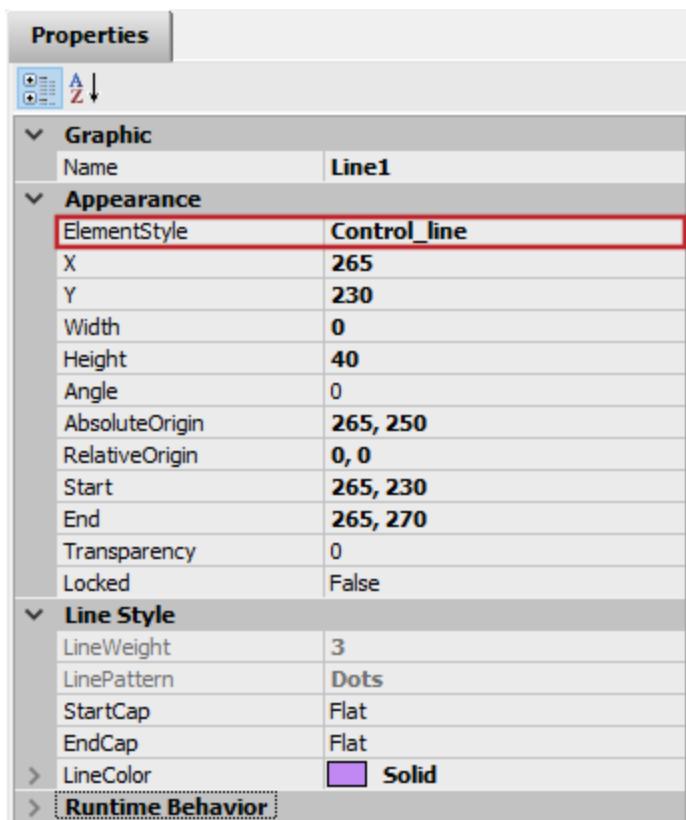
Styles can be used to manage the appearance of elements, status animations and numeric values in AVEVA™ Industrial Graphics applications.

A style defines a set of visual properties that determine the appearance of elements such text, lines, graphic outlines, and interior fill.



When a style is applied to an Industrial Graphics symbol, the specified visual properties take precedence over a symbol's native visual properties. This allows you to establish consistent visual standards.

Style can be applied to selected objects in the Industrial Graphics Editor (see the topic [Applying Element Styles to Elements](#)). For example, you can specify a style in the **Element Style** field in the Properties Editor. This field includes a drop-down list of styles that are currently configured in the active project.



When a style is selected for an object, you are no longer able to modify the value for any properties that have an override configured. In the example above, the **LineWeight** and **LinePattern** fields appear gray to indicate that an override has been configured for these properties in the **Control\_line** style.

Styles are defined in Plant SCADA Studio's **Standards** activity. The **Styles** view hosts the **Configure Styles** dialog, which allows you to configure an overriding style for a selected visual element.

The available elements are distributed across the following three tabs:

- **Quality and Status** — defines the appearance of primitive elements and the status element so they will reflect the quality and/or status of an attached I/O (see [Configure Style Overrides for Quality and Status Elements](#)).
- **Element Styles** — defines visual properties for symbols, including text, lines, outlines, and fill (see [Configure Style Overrides for Element Styles](#)).
- **Format Styles** — defines the format used for numeric values in display animations (see [Configure Format Styles for Value Display Animations](#)).

In each case, pre-configured styles are included for a default set of elements. The predefined values of these element styles can be changed, however, they styles cannot be renamed or deleted.

User-defined styles are also available for Element Styles and Format Styles. You can rename a user-defined style to help identify its intended purpose.

The order of precedence for property styles (from high to low) is:

1. Quality and Status
2. Element style animation
3. Style animations

4. Group-level element style
5. Element-level element style
6. Local element-level style.

---

**Note:** The Industrial Graphics Editor and runtime will only use the styles configured in the active project. If you are editing content in an included project, be aware that the pages and symbols will be displayed using the styles specified in the active project.

You can copy the styles configured for Industrial Graphics from one Plant SCADA project to another. See [Copy Styles to Another Project](#).

## See Also

[AVEVA™ Industrial Graphics](#)

## Configure Style Overrides for Quality and Status Elements

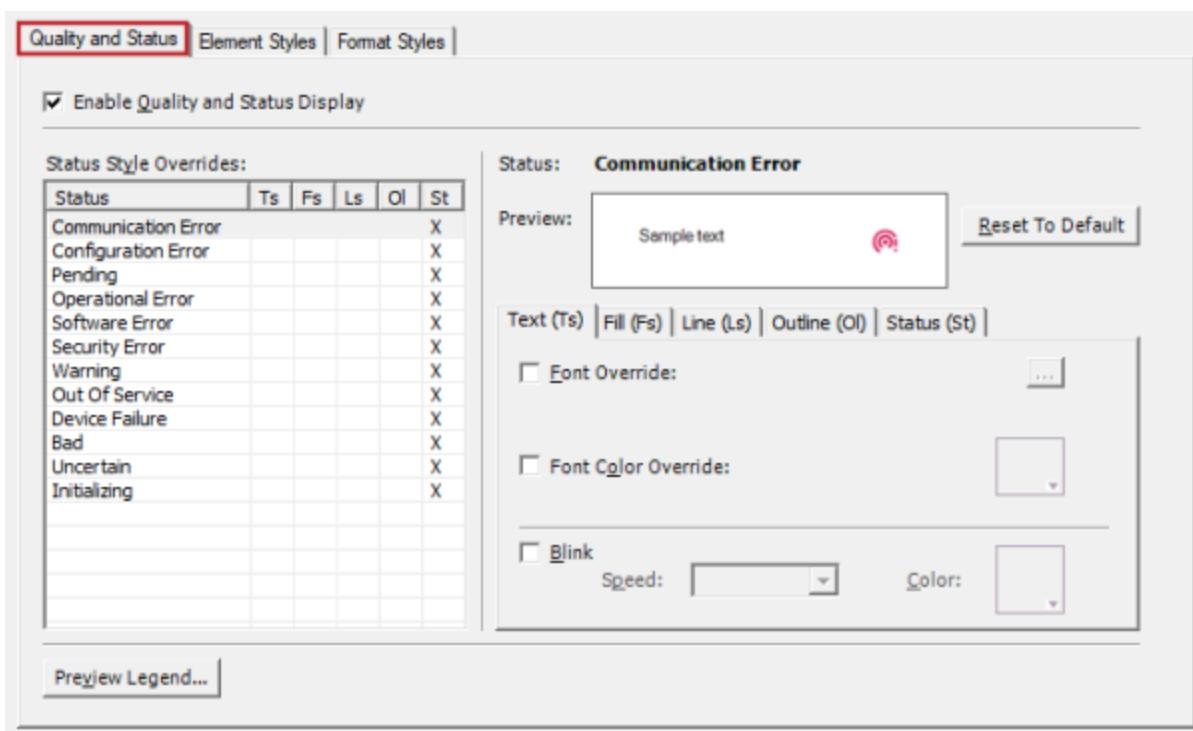
When defining [Styles](#) for AVEVA™ Industrial Graphics objects, you can configure the appearance of elements so that they will reflect the quality and/or status of an attached I/O. This allows you to standardize the appearance of quality and status animations.

There are two ways to indicate status and quality:

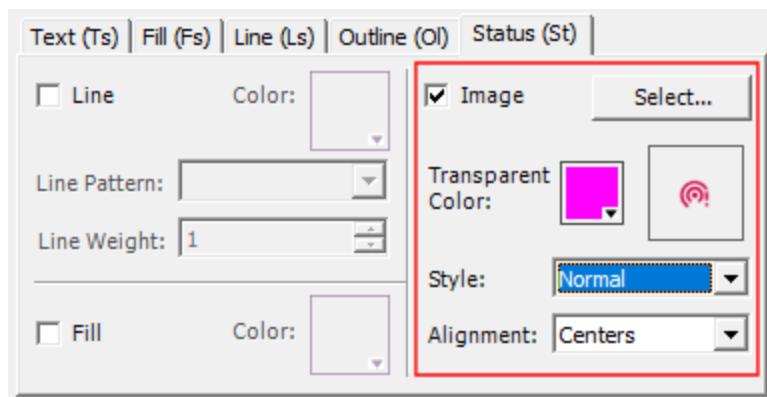
- Use the Status Element, available from the Industrial Graphics Editor Toolbox
- Apply styles to primitive objects so they reflect a particular quality or status.

### To configure style overrides for quality and status elements:

1. In the **Standards** activity, select **Styles**.
2. On the Styles Configuration dialog, select **Edit**. The Configure Styles dialog will appear.
3. Select the **Quality and Status** tab.



4. Select **Enable Quality and Status Display**. This will apply styles to normal primitives that have I/O attached to them.
5. In the **Status Style Overrides** panel, select the status element you would like to modify. The check marks in the Status Style Override panel indicate if an element already has an override configured for it. The columns represent the following:
  - Ts — text
  - Fs — fill
  - Ls — line
  - OI — outline
  - St — status.
6. To view or edit the style override settings for the selected element, click on the corresponding tabs in the section to the right of the dialog.
  - **Text (Ts)** tab — allows you to define a **Font Override**, a **Font Color Override** or **Blink** settings for the text that is used.
  - **Fill (Fs)** tab — allows you to define a **Fill Override** or **Blink** settings for the fill that is used.
  - **Line (Ls)** tab — allows you to define a **Line Pattern Override**, **Line Weight Override**, **Line Color Override** or **Blink** settings for the line that is used.
  - **Outline (OI)** tab — allows you to select to **Show Outline**, and then define a **Line Pattern**, **Line Weight**, or **Blink** settings for the outline.
  - **Status (St)** — allows you to set the **Image** properties for a Status Element (which uses an image to indicate quality and status). You can **Select** the image used, and specify **Transparent Color**, **Style** and **Alignment** properties.



**Note:** The **Line** and **Fill** properties on the Status (St) tab are not supported in Plant SCADA. They will not affect the style applied to a Status Element.

The **Preview** panel will display the outcome of the current configuration when the selected element represents an active state.

7. If required, select the **Reset to Default** button to return the selected element to its default state.
8. Click **OK** to close the Configure Styles dialog.

## See Also

[Configure Style Overrides for Element Styles](#)

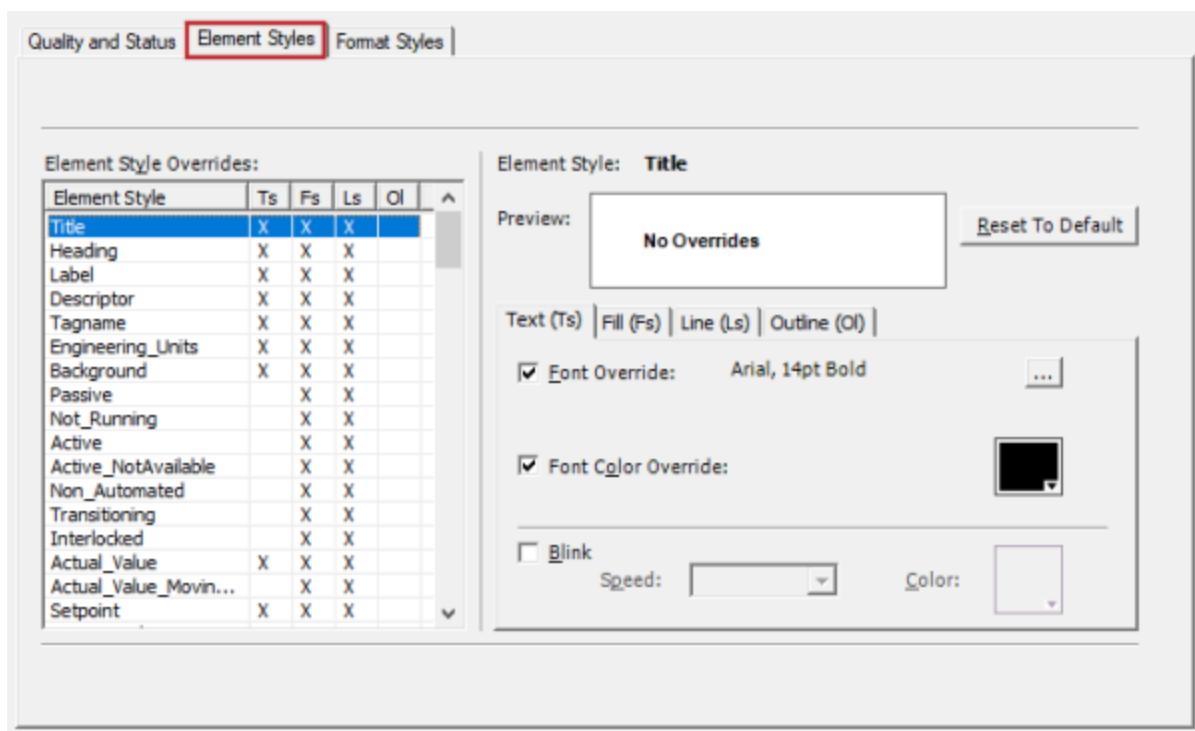
[Configure Format Styles for Value Display Animations](#)

## Configure Style Overrides for Element Styles

When defining **Styles** for AVEVA™ Industrial Graphics objects, you can configure style overrides for the visual elements of symbols. This allows you to standardize the appearance of the symbols included in a project.

**To configure style overrides for elements styles:**

1. In the **Standards** activity, select **Styles**.
2. On the Styles Configuration dialog, select **Edit**. The Configure Styles dialog will appear.
3. Select the **Element Styles** tab.



- In the **Element Style Overrides** panel, select the element style you would like to modify. If you select a user-defined style, you can double click on it to rename it, or right-click and select **Rename**. The check marks in the Element Style Override panel indicate if an element already has an override configured for it. The columns represent the following:
  - Ts — text
  - Fs — fill
  - Ls — line
  - Ol — outline.
- To view or edit the style override settings for the selected element, click on the corresponding tabs in the section to the right of the dialog.
  - Text (Ts)** tab — allows you to define a **Font Override**, a **Font Color Override** or **Blink** settings for the text that is used.
  - Fill (Fs)** tab — allows you to define a **Fill Override** or **Blink** settings for the fill that is used.
  - Line (Ls)** tab — allows you to define a **Line Pattern Override**, **Line Weight Override**, **Line Color Override** or **Blink** settings for the line that is used.
  - Outline (Ol)** tab — allows you to select to **Show Outline**, and then define a **Line Pattern**, **Line Weight**, or **Blink** settings for the outline.
 The **Preview** panel will display the outcome of the current configuration for the selected element.
- If required, select the **Reset to Default** button to return the selected element to its default state.
- Click **OK** to close the Configure Styles dialog.

**Note:** If the Industrial Graphics Editor is currently open, style configuration changes will not appear until after the editor has been restarted.

## See Also

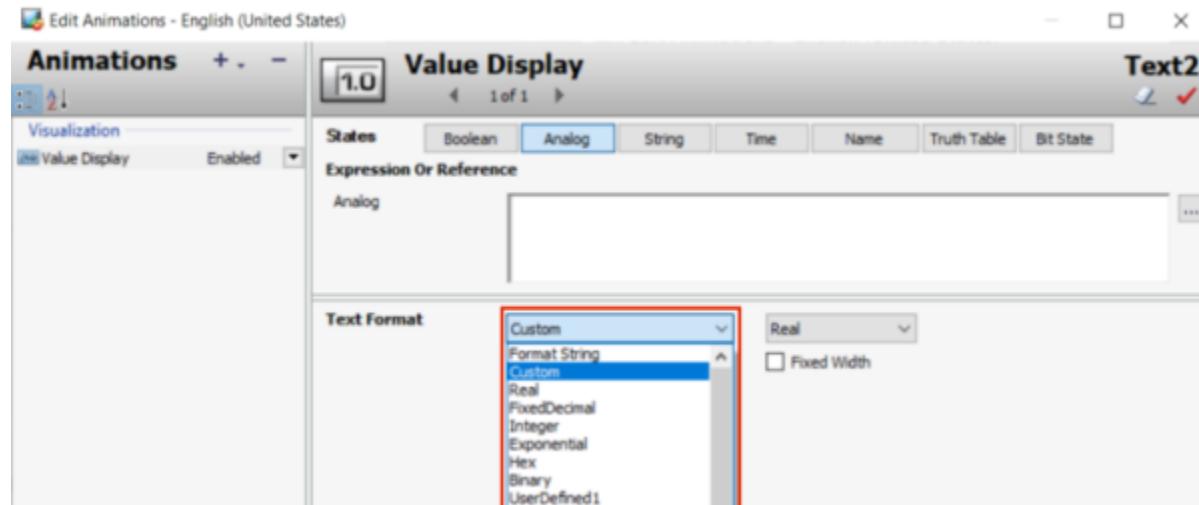
[Configure Style Overrides for Quality and Status Elements](#)

[Configure Format Styles for Value Display Animations](#)

## Configure Format Styles for Value Display Animations

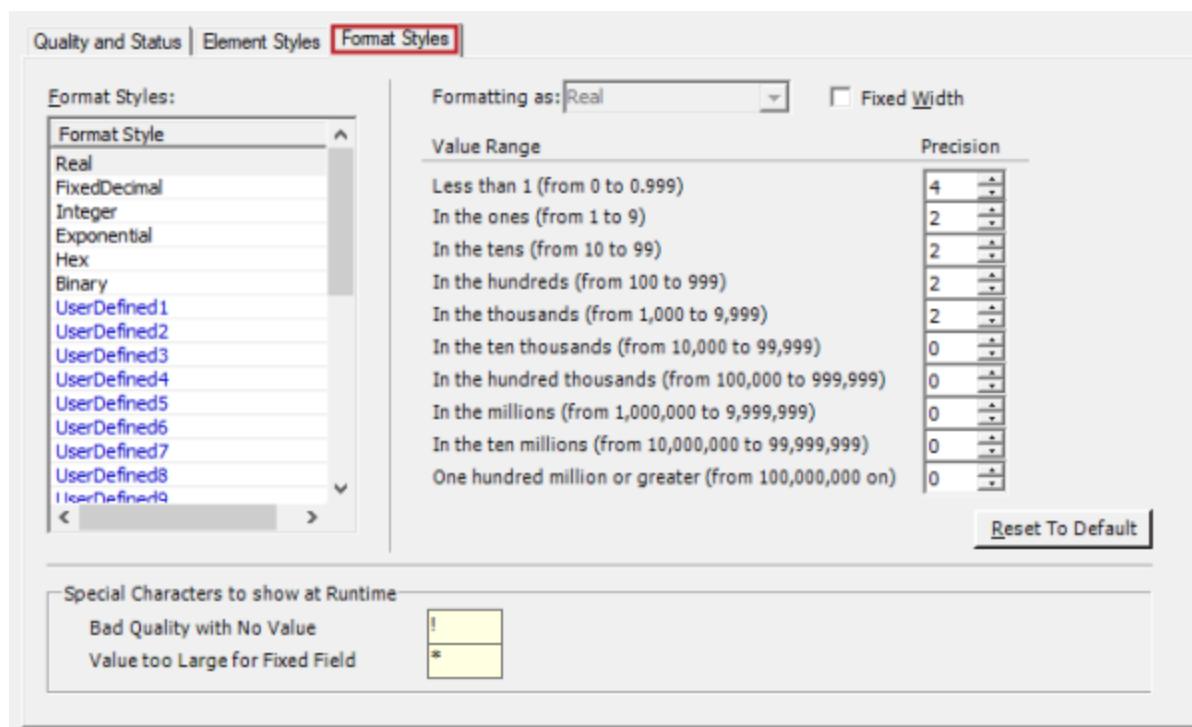
When defining [Styles](#) for AVEVA™ Industrial Graphics objects, you can configure the format used for numeric values in display animations. You can define the value range, precision, and whether or not a number has a fixed width.

Default format styles are available for six numeric value types. You can also configure and rename up to 25 user-defined format styles. You can use the Edit Animations dialog in the Industrial Graphics Editor to apply a format style to an analog Value Display animation.



### To configure format styles for value display animations:

1. In the **Standards** activity, select **Styles**.
2. On the Styles Configuration dialog, select **Edit**. The Configure Styles dialog will appear.
3. Select the **Format Styles** tab.



4. In the **Format Style** panel, select the style you would like to modify.  
If you select a user-defined style, you can double click on it to rename it, or right-click and select **Rename**.
5. Specify the type of value in the **Formatting as** field.
6. Define the level of **Precision** or the number of bits used for the selected value type using the fields that are displayed. For example, if you have selected Real in the **Formatting As** field, you need to specify the Precision used for each Value Range.
7. If required, select **Fixed Width**.
8. Under **Special Characters to show at Runtime**, specify the characters that you would like to use to indicate **Bad Quality with No Value** and **Value Too Large for Fixed Field**. You can only specify one character for each.
9. If required, select the **Reset to Default** button to return the selected format style to its default state.
10. Click **OK** to close the Configure Styles dialog.

**Note:** If the Industrial Graphics Editor is currently open, style configuration changes will not appear until after the editor has been restarted.

## See Also

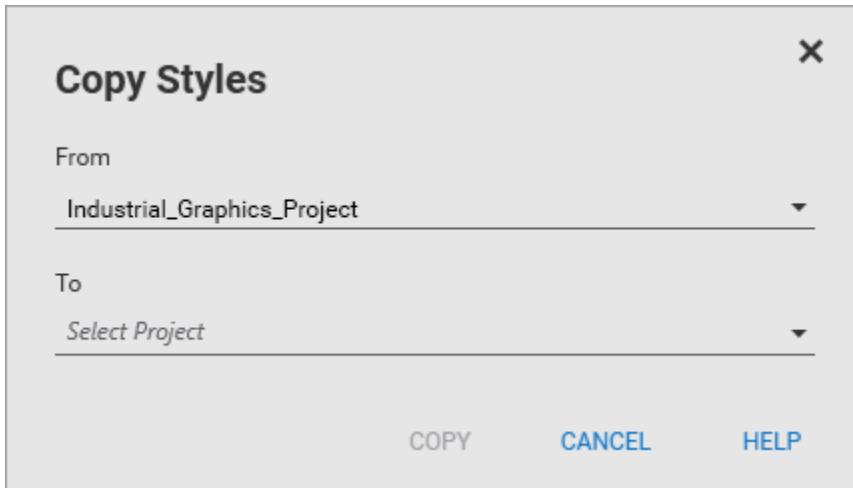
- [Configure Style Overrides for Element Styles](#)  
[Configure Style Overrides for Quality and Status Elements](#)

## Copy Styles to Another Project

You can copy the styles configured for Industrial Graphics from one Plant SCADA project to another.

### To copy Industrial Graphics styles to another project:

1. In the **Standards** activity, select **Styles**.
2. On the command bar, select **Copy Styles**. The Copy Styles dialog will appear.



3. In the **From** field, select the project that includes the styles you want to copy.
4. In the **To** field, select the project to which you want to add the copied styles.
5. Click **COPY**.
6. Click **Yes** on the dialog that appears to confirm that you want to overwrite the styles configured in the destination project.

**Note:** You are not able to copy styles to a system project.

## See Also

- [Configure Style Overrides for Quality and Status Elements](#)
- [Configure Style Overrides for Element Styles](#)
- [Configure Format Styles for Value Display Animations](#)

## Groups

Groups can be used for many purposes, for example, to define a set of users, multiple areas or multiple devices. You can also specify complex groups by defining a group of groups. A group can be used anywhere that an individual entity can be used.

## See Also

- [Add a Group](#)

## Add a Group

To define a **Groups**, you associate a list of entities with a group name.

**To add a group:**

1. In the **Security** activity, select **Groups**.
2. Add a row to the Grid Editor.
3. Type the required information in each column, or in the fields in the Property Grid.  
For a description of the properties, see below.
4. Click **Save**.

## Group Properties

### General Properties

Property	Description
<b>Group Name</b>	The name of the group. Enter a value of 15 characters or less.
<b>Association 1 . . . Association 10</b>	A list of the entities associated with the Group Name. Enter a value of 16 characters or less. An association can be a number, a name, or another group. You can also specify a range of numbers in the format <n1..n2> for example: 4..10 specifies numbers 4,5,6,7,8,9,10.
<b>Comment</b>	Any useful comment. Enter a value of 48 characters or less.

### Project Properties

Property	Description
<b>Project</b>	The project in which the user is included.

## Fonts

Alarm categories and Cicode functions allow you to use predefined fonts, known as **system fonts**, to display text. You also need to define a system font for animated text.

**Note:** If an animation is associated with an I/O device that does not initialize properly at startup or goes offline while the system is running, the associated animation is grayed on the relevant graphics pages (because the values are invalid). You can disable this feature with the [Page]ComBreak parameter. See [Handling Communication Errors in Reports](#).

You can also configure colors for your fonts, see [Configuring Custom Color Fonts](#).

## See Also

[Add a System Font](#)

## Add a System Font

System [fonts](#) can be defined for use with alarm categories and Cicode functions.

### To add a system font:

1. In the **Standards** activity, select **Fonts**.
2. Add a row to the Grid Editor.
3. Type the required information in each column, or in the fields in the Property Grid.  
For a description of the properties, see below.
4. Click **Save**.

## Fonts Properties

### General Properties

Property	Description
<b>Font Name</b>	The name of the font. Unlike background text, strings, and numbers (which can use any standard Windows font), you need to define a font for animated text.
<b>Font Type</b>	<p>Any text font supported by Windows (31 characters maximum). Choose a font type from the menu. You can also specify bold, italic, and underlined text. To specify any of these options, append the appropriate specifier to the font type, for example:</p> <ul style="list-style-type: none"><li>• "Courier,B" specifies bold characters.</li><li>• "Helv,I" specifies italic characters.</li><li>• "TmsRmn,U" specifies underlined text.</li></ul> <p>You can also specify multiple options, for example, "Courier,B,I,U" specifies bold, italic, and underlined characters.</p> <p><b>Note:</b> To use a font that is not displayed in the menu, you need to install the font on your computer before you can use it in your system. To view, install, or remove Windows fonts, use the Windows Control Panel. Refer to your Windows documentation for details. (If your system uses a network, every computer needs to have the font installed.)</p>
<b>Pixel Size</b>	The size of the displayed text. You can specify text

Property	Description
	<p>fonts in pixels or points.</p> <p>To specify a point size, enter a negative number in the <b>Pixel Size</b> field; for example, "—10" specifies a ten-point font size. Be aware that you can only specify a point size as a whole number (integer).</p> <p>If you have not installed the Font Type (or Pixel Size) on your system, a default font and/or size is used that most closely resembles the font and/or size you have specified.</p> <p>If you use a point size, the size remains constant across screen resolutions. On low-resolution screens, the font appears larger than on high-resolution screens, which might cause misalignment of animation. Only use a point size to display text on computer screens of the same resolution.</p>
<b>Foreground Color</b>	<p>The foreground color of the displayed text (i.e., the color of the text characters). You can use a predefined color label, a user-defined custom label, or the RGB encoded number generated by the function <a href="#">MakeColour</a>.</p> <p><b>Note:</b> Avoid confusing predefined labels with the color Name feature associated with the Color Picker. You cannot use color names with this dialog; doing so generates a compile error message.</p>
<b>Background Color</b>	<p>The background color of the displayed text. You can use a predefined color label (accessible via the menu), a user-defined custom label, or the RGB encoded number generated by the function <a href="#">MakeColour</a>.</p> <p>This property is optional. If a background color is not specified, it defaults to transparent.</p>
<b>Comment</b>	Any useful comment (48 characters maximum).
<b>Foreground Flash</b>	<p>The secondary color applied to the font if using flashing color for your text characters. You can use a predefined color label (accessible via the menu), a user-defined custom label, or the RGB encoded number generated by the function <a href="#">MakeColour</a>.</p>
<b>Background Flash</b>	<p>The secondary color applied to the background if using flashing color. You can use a predefined color label (accessible via the menu), a user-defined custom label, or the RGB encoded number generated by the function <a href="#">MakeColour</a>.</p>

Property	Description
	If a color is not specified, the text remains solid.

**Project Properties**

Property	Description
<b>Project</b>	The project in which the font is included.

**See Also**

[Configuring Custom Color Fonts](#)

## Configuring Custom Color Fonts

Colors are referenced by using hex codes for red, green and blue (RGB) values. You can view RGB values (in decimal) of a selected color by choosing **Tools | Edit Favorite Colors** in **Graphics Builder**.

If, for example, the color you want to use has the values R = 128, G = 170, B = 213, you can convert these to their hex equivalents (R = 0x80, G = 0xAA, B = 0xD5).

If you define a label for your color, you can use decimal or hex values. For example:

- **Label Name:** Plate\_Blue
- **Expression:** 0x80AAD5 (or 8432341)

Once you have defined your label, you can create fonts to use in your project for alarm categories, Cicode functions, and button objects. For example, use the Fonts dialog box to define a font with the following properties:

- **Font Name:** Plate\_Font
- **Font Type:** Arial
- **Pixel Size:** 12
- **Foreground Color:** Plate\_Blue

**See Also**

[Fonts](#)

## Setup

The Setup activity is intended to manage assets that are unlikely to require ongoing configuration once they have been set up.

The following features are configured in the Setup activity:

- [Alarming](#)

- Project Database Parameters
- Devices
- Triggering Events
- Languages
- Keyboard Keys
- Custom Files.

## Alarming

The **Alarming** view in the **Setup** activity is where you configure project-wide settings that relate to the management of alarms. These settings include:

- Alarm categories (see [Add an Alarm Category](#))
- Priority display properties (see [Configure Display Properties for an Alarm Priority](#))
- Alarm mode display properties (see [Configure Display Properties for an Alarm Mode](#))
- Cause and response information (see [Add Cause and Response Information to Alarms](#)).

## See Also

[Manage Alarms](#)

## Add an Alarm Category

You can use alarm categories to group the alarms in your Plant SCADA system (see [Categorize Alarms](#)). Up to 16376 alarm categories can be created.

**To add an alarm category to a project:**

1. In the **Setup** activity, select **Alarming**.
2. On the menu below the Command Bar, select **Alarm Categories**.
3. Type the required information in each column, or in the fields in the Property Grid (see below for a description of the fields).
4. Click **Save**.

You will now be able to assign alarms to the category you have created in the System Model activity (see [Assign an Alarm to an Alarm Category](#)).

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

When configuring an alarm category, do not use a blocking Cicode function in the following fields:

- ON Action

- OFF Action
- ACK Action.

A blocking function or a lengthy operation will affect the polling of alarms, and may result in slow or delayed alarm processing. Therefore, do not configure long running functions in these fields but keep the functions simple or they may not be executed correctly. To execute complicated functions, it is recommended to use TaskNew() function to initiate a new task which performs the required function in a separate task.

If the configuration of these fields is updated and reloaded by the alarm server, the new actions will be executed. However, the updated configuration can not call any new or updated user functions which did not exist before reloading, even if a new task is used to run it. If used, nothing will happen until the alarm server is restarted. Hence, confirm that the updated actions do not refer to brand new or updated functions or tasks prior to reloading the alarm server.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Alarm Category Properties

### General Properties

Field	Description
Category	<p>The alarm category as a numeric value (0-16375). The following category numbers are reserved:</p> <ul style="list-style-type: none"><li>• Category 0 is reserved for selecting all the alarms with all categories.</li><li>• Category 254 is reserved for user-created alarm summary entries.</li><li>• Category 255 is reserved for hardware alarms.</li></ul> <p>This field supports up to 16 characters.</p>
Priority	<p>The priority applied to alarms assigned to this alarm category (0- 255). Alarm priority determines the order in which alarms are displayed, acknowledged, enabled and so on.</p> <p>Priority 1 is the highest priority, and priority 255 is the lowest. For example, if alarms with priorities 1 to 8 were displayed, priority 1 alarms would be displayed first in their time/date order, then priority 2 alarms, then priority 3, and so on up to priority 8.</p> <p>Priority 0 is the default priority. It is reserved for selecting alarms of all priorities when referencing priorities. For example, if you use the Cicode function <b>AlarmSetInfo</b> to change an alarms list so that alarms of all priorities are displayed, you would use the following settings:</p>

Field	Description
	<p><i>Type = 7, Value = 0.</i></p> <p><b>Note:</b> When priority 0 is used to display alarms of priorities, priority 0 only alarms will display first, followed by priority 1 alarms, then priority 2, and so on. You can also customize the order in which alarms are displayed on an alarm summary page using the SummarySort and SummarySortMode parameters. (This order will override the alarm category priority order.)</p>
<b>Show on Active</b>	<p>Determines if alarms assigned to this category display on active alarms pages.</p> <p>You can set this field to TRUE or FALSE. The default value is TRUE.</p> <p><b>Note:</b> For Situational Awareness projects, <b>Show On Active</b> must be blank (TRUE) for associated alarm counts to appear on the Page Navigation Zone and Tree Views.</p>
<b>Show on Summary</b>	<p>Determines if alarms assigned to this category display on alarm summary pages.</p> <p>You can set this field to TRUE or FALSE. The default value is TRUE.</p>
<b>Comment</b>	Any useful comment.

#### Font Properties

Field	Description
<b>UnAck On Font</b>	<p>Defines the font used to display alarms that are unacknowledged and active.</p> <p>This property is optional. If no font is specified, the font defaults to 10pt YELLOW.</p> <p>This field supports up to 16 characters.</p>
<b>UnAck Off Font</b>	<p>Defines the font used to display alarms that are unacknowledged and no longer active.</p> <p>This property is optional. If no font is specified, the font defaults to 10pt BROWN.</p> <p>This field supports up to 16 characters.</p>
<b>Ack On Font</b>	<p>Defines the font used to display alarms that have been acknowledged and are still active.</p> <p>This property is optional. If no font is specified, the font defaults to 10pt CYAN.</p>

Field	Description
	This field supports up to 16 characters.
<b>Ack Off Font</b>	Defines the font used to display alarms that have been acknowledged and are no longer active. This property is optional. If no font is specified, the font defaults to 10pt WHITE. This field supports up to 16 characters.
<b>Disabled Font</b>	Defines the font used to display disabled alarms. This property is optional. If no font is specified, the font defaults to 10pt WHITE. This field supports up to 16 characters.

*Format Properties*

Field	Description
<b>Alarm Format</b>	Defines the screen display format used on active alarm pages for alarms in this category. <b>Note:</b> Alarm pages based on the tab-style templates will not consider individual formats for each category. The screen display format used for an alarm page is set by the first available of the following definitions: <ul style="list-style-type: none"> <li>• [Format]Alarm parameter</li> <li>• The Alarm Format specified for category 0</li> <li>• The default alarm display format.</li> </ul> When applied to pages based on a standard template, Alarm Format specifies how the data for alarms in this category are displayed on alarms pages (on the screen only). Each alarm displays on the alarms page in a single line, for example: 12:32:21RFP3 Raw Feed pump 3 Overload When alarms are displayed using variable width fonts (such as Arial or Helvetica), the alarm fields may not align properly across different rows. This can be avoided by using a field separator in the alarm format configuration instead of just a space. Tab characters, denoted by either " <code>^t</code> " (horizontal tab) or " <code>^v</code> " (vertical tab), between the alarm fields will act as alignment points in your alarm display. If you leave the Alarm Format field blank, the format defaults to: <code>{Time,12} {Tag,10} {Name,20} {Desc,32}</code> You can change this default setting with the parameter

Field	Description
	<p>[Alarm]DefDspFmt.</p> <p>See <a href="#">Alarm Format Fields</a> for details about each field type.</p> <p><b>Note:</b> If an alarm value is longer than the field it is to be displayed in, it will be truncated or replaced with the #OVR ("overflow of format width") alert message. When the alarm is logged to a device (i.e. printed or written to a file or database), the format specified for the logging device overrides the display format.</p>
<b>Summary Format</b>	<p>Defines the screen display format used on alarm summary pages for alarms in this category.</p> <p><b>Note:</b> Alarm summary pages based on the tab-style templates will not consider individual formats for each category. The screen display format used for an alarm summary page is set by the first available of the following definitions:</p> <ul style="list-style-type: none"> <li>• [Format]Alarm parameter</li> <li>• The Alarm format specified for category 0</li> <li>• Default Alarm display format.</li> </ul> <p>When applied to pages based on a standard template, Summary Format is defined the same way as <b>Alarm Format</b> (see above). However, you can also use additional data fields.</p> <p>See <a href="#">Alarm Format Fields</a> for details about each field type.</p> <p>If you leave the Summary Format field blank, the format defaults to:</p> <pre>{Name,20} {OnTime,8} {OffTime,8} {DeltaTime,8} {Comment,22}</pre> <p>You can change this default setting with the parameter <b>[Alarm]DefSumFmt</b>.</p> <p><b>Note:</b> When an alarm is logged to a summary device (i.e. printed or written to a file or database), the format specified for the logging device overrides the display format.</p>
<b>SOE Format</b>	<p>Defines the screen display format used on sequence of event (SOE) pages for alarms in this category.</p> <p><b>Note:</b> SOE pages based on the tab-style templates will not consider individual formats for each category. The screen display format used for an SOE page is set by the first available of the following definitions:</p> <ul style="list-style-type: none"> <li>• [Format]Alarm parameter</li> </ul>

Field	Description
	<ul style="list-style-type: none"> <li>The Alarm format specified for category 0</li> <li>Default Alarm display format.</li> </ul> <p>When applied to pages based on a standard template, SOE Format is defined the same way as <b>Alarm Format</b> (see above). However, you can also use additional data fields.</p> <p>See <a href="#">Alarm Format Fields</a> for details about each field type.</p> <p>If you leave the SOE Format field blank, the format defaults to:</p> <pre>{DATE,16} {TIME,16} {TAG,10} {NAME,15} {MESSAGE,64} {STATE,16} {CLASSIFICATION,13} {USERNAME,16} {USERLOCATION,16}</pre> <p>You can change this default setting with the parameter <b>[Alarm]DefSOEFmt</b>.</p>

**Actions Properties**

Field	Description
<b>ON Action</b>	<p>Specifies a Cicode command that is executed when an alarm of this Category becomes active (ON).</p> <p><b>Example:</b> Where <b>ON Action</b> is <b>STOP_PROCESS = 1;</b> In this example the digital variable <b>STOP_PROCESS</b> is set to ON when an alarm in this category is triggered.</p> <p><b>Note:</b> Do not put a blocking Cicode function in this field. The alarm system executes ON, OFF, or ACK actions within the polling loop. A blocking function will affect the polling of alarms, and may result in slow or delayed alarm processing.</p> <p>A special case of this command occurs when the ON Action is self-referring, with a form such as TAG1 = TAG1 + 1. This command will not work properly since tags are not reread before processing the ON action (for performance reasons). This particular command will therefore initially set the value of TAG1 to 1 rather than incrementing it.</p> <p>To correctly run a command of this type in the ON Action, use TaskNew() to run your own Cicode function to perform the tag command. For example: ON Action is TaskNew("MyFunc","Data",5);</p>
<b>OFF Action</b>	Specifies a Cicode command that is executed when an

Field	Description
	<p>alarm of this Category is reset (OFF).</p> <p><b>Example:</b></p> <p>Where <b>OFF Action</b> is <b>ENABLE_PROCESS = 1;</b></p> <p>In this example the digital variable ENABLE_PROCESS is set to ON when an alarm in this category is reset.</p> <p><b>Note:</b>Do not put a blocking Cicode function in this field. The alarm system executes ON, OFF, or ACK actions within the polling loop. A blocking function will affect the polling of alarms, and may result in slow or delayed alarm processing.</p>
<b>ACK Action</b>	<p>A Cicode command that is executed when an alarm of this Category is acknowledged.</p> <p><b>Note:</b> Do not put a blocking Cicode function in this field. The alarm system executes ON, OFF, or ACK actions within the polling loop. A blocking function will affect the polling of alarms, and may result in slow or delayed alarm processing.</p>

### Storage Properties

Field	Description
<b>Alarm Acquisition Error</b>	<p>Acquisition error is an error related to acquiring data from I/O sub-system for the underlying tag and/or expression, for example, Device Offline, Tag Unknown, etc. (Integer length 6).</p> <p>All acquisition errors are stored for each alarm record separately within the alarm server as a new AcqError field. They are not logged and no hardware alarm is raised from them.</p> <p>The AcqError field stores the error as the Plant SCADA error code (for example, NO ERROR). This field is made accessible through the alarm browse functionality. If there are multiple acquisition errors for a single alarm record, then the first of these is stored within the system.</p>
<b>Summary Device</b>	<p>Specified the device to which the alarm summary is sent.</p> <p>The format specified in the device is used instead the display format. If not specified, alarm summaries are not logged.</p>
<b>Log Device</b>	<p>Specifies the device to which alarm state changes are logged. An alarm entry is made in the log device each</p>

Field	Description
	<p>time an alarm assigned to the category changes state (on, off, acknowledged, enabled, or disabled).</p> <p>When the alarm is printed, or written to a file or device, the format specified in the device overrides the display format.</p> <p>If not specified, alarm state changes are not logged.</p>
<b>Log Transitions: ON</b>	<p>Determines if the alarm details are logged when an alarm assigned to this category becomes active.</p> <p>You can set this field to TRUE or FALSE. The default value is TRUE.</p>
<b>Log Transitions: OFF</b>	<p>Determines if the alarm details are logged when an alarm assigned to this category becomes inactive.</p> <p>You can set this field to TRUE or FALSE. The default value is TRUE.</p>
<b>Log Transitions: ACK</b>	<p>Determines if the alarm details are logged when an alarm assigned to this category is acknowledged.</p> <p>You can set this field to TRUE or FALSE. The default value is TRUE.</p>

### Project Properties

Property	Description
<b>Project</b>	The project in which the specified alarm category is included.

### See Also

[Prioritize Alarms](#)

## Configure Display Properties for an Alarm Priority

You can configure optional display properties for each alarm priority value (1-255). This allows you to:

- Specify a name for a priority value to provide a meaningful representation of its purpose.
  - Define background and foreground colors that provide a visual representation of priority on a graphics page.
- 
- Note:** Transparent colors are not supported.
- Associate Genies with a priority that you can use as an alarm flag or icon.

**To configure display properties for an alarm priority:**

1. In the **Setup** activity, select **Alarming**.
2. On the menu below the Command Bar, select **Alarm Priorities**.
3. Add a row to the Grid Editor.
4. Type the required information in each column, or in the fields in the Property Grid (see below for a description of the fields).
5. Click **Save**.

---

**Note:** Though the functionality is supported, it is advised that you do not configure the top three alarm priorities with **Show On Indicator** set to **FALSE**.

---

## Alarm Priority Properties

### General Properties

Field	Description
<b>Priority</b>	The priority value (1- 255).
<b>Display Name</b>	Can be used to provide a meaningful representation of a priority value.
<b>Short Name</b>	Can be used to provide a short-form name that represents the priority value. For example, you can use this short name as a label on an alarm flag.
<b>Comment</b>	Any useful comment.
<b>Show on Indicator</b>	<p>Specifies if the priority will be ignored or not on Alarm Indicator. Default is TRUE (priority is not ignored). If this value is FALSE, Alarm Indicator will ignore this priority and take the next prioritized alarms. For example, if there are two alarms raised with priority 3 and 4 in the same graphic object, and priority 3 has Show on Indicator set to FALSE, the alarm border will ignore alarms set to priority 3 and display alarm border using priority 4 color codes.</p> <p>When this value is set to FALSE, it will not be included in the alarm counts on any alarm equipment tree views or navigation areas.</p> <p><b>Note:</b> You still need to configure correct color codes on color properties even if "Show on Indicator" value is set to FALSE as these colors are used when there is server – client property mismatch caused by alarm server reloading. For example, when that field on server side configuration has been changed from TRUE to FALSE on server side only by reloading, these colors</p>

Field	Description
	may appear on display client's alarm borders even if client configuration has that field set to FALSE. As this situation indicates configuration mismatch, operators should get updated projects from engineers and restart the client.
<b>UnAck On Foreground Color</b>	Specifies the foreground color to display when an alarm of this priority is in an ON state and unacknowledged.
<b>UnAck On Background Color</b>	Specifies the background color to display when an alarm of this priority is in an ON state and unacknowledged.
<b>Ack On Foreground Color</b>	Specifies the foreground color to display when an alarm of this priority is in an ON state and acknowledged.
<b>Ack On Background Color</b>	Specifies the background color to display when an alarm of this priority is in an ON state and acknowledged.
<b>UnAck Off Foreground Color</b>	Specifies the foreground color to display when an alarm of this priority is in an OFF state and still unacknowledged.
<b>UnAck Off Background Color</b>	Specifies the background color to display when an alarm of this priority is in an OFF state and still unacknowledged.
<b>Library Name</b>	The name of library to which the associated Genies belong (see below).
<b>Genie Name</b>	The name of a Genie that you want to use as a visual representation of this priority. This allows the Genie to be used as an alarm flag (see <a href="#">Use Alarm Indicators</a> ).
<b>Small Genie Name</b>	<p>The name of a small Genie that you want to use as a visual representation of this priority. This allows the Genie to be used as a small icon that indicates alarm priority. For example, it can be used for:</p> <ul style="list-style-type: none"> <li>• Priority And State symbols in an alarms list (see <a href="#">Create Priority and State Symbols for an Alarms List</a>).</li> <li>• The alarm counts on a tree-view (see <a href="#">Add a Tree View to a Page</a>).</li> </ul>

Field	Description
	<ul style="list-style-type: none"><li>The alarm counts in the Navigation Zone (see <a href="#">Navigation Zone</a>).</li></ul>

### Project Properties

Property	Description
Project	The project in which the alarm priority is included.

## See Also

[Prioritize Alarms](#)

## Configure Display Properties for an Alarm Mode

An alarm mode provides a way to group together any alarms that are currently in the same mode of operation. For example, you can use a mode to represent any alarms that are currently shelved or disabled. This allows you to configure global display properties that can be applied to an alarm as it moves into the specified mode of operation. For example, you can:

- Define background and foreground colors that provide a visual representation of a mode on a graphics page.
- Associate Genies with a mode that you can use as an alarm flag or icon.

**Note:** You can only use the predefined modes that are included with Plant SCADA, that is, Shelved/Disabled.

### To configure display properties for an alarm mode:

- In the **Setup** activity, select **Alarming**.
- On the menu below the Command Bar, select **Alarm Modes**.
- Add a row to the Grid Editor.
- Type the required information in each column, or in the fields in the Property Grid (see below for a description of the fields).
- Click **Save**.

## Alarm Mode Properties

### General Properties

Field	Description
<b>Display Name</b>	Can be used to provide a meaningful representation of a mode. You need to select a name from those available in the drop-down list, otherwise a compile error will be generated.
<b>Short Name</b>	Can be used to provide a short-form name that represents the mode. For example, you can use this short name as a label on an alarm flag.
<b>Comment</b>	Any useful comment.
<b>Foreground Color</b>	Defines the foreground color to display when an alarm is in the specified mode of operation. Transparent colors are not supported.
<b>Background Color</b>	Defines the background color to display when an alarm is in the specified mode of operation. Transparent colors are not supported.
<b>Library Name</b>	The name of library to which the associated Genies belong (see below).
<b>Genie Name</b>	The name of a Genie that you want to use as a visual representation of this mode. This allows the Genie to be used as an alarm flag (see <a href="#">Use Alarm Indicators</a> ).
<b>Thumbnail Name</b>	The name of a small Genie that you want to use as a visual representation of this mode. This allows the Genie to be used as a small icon that indicates alarm priority.

**Project Properties**

Property	Description
<b>Project</b>	The project in which the alarm mode is included.

**See Also**[Use Alarm Indicators](#)[Shelve Alarms](#)[Disable Alarms](#)**Add Cause and Response Information to Alarms**

You can add cause and response information to an alarm that describes the circumstances that would cause the alarm and the most appropriate course of action to take. This information can be presented to an operator at

runtime. See [Provide Cause and Response Information to Operators](#).

You can configure up to eight alarm responses for each alarm tag.

### To add cause and response information for alarms:

1. In the **Setup** activity, select **Alarming**.
2. On the menu below the Command Bar, select **Cause and Response**.
3. Add a row to the Grid Editor.
4. Type the required information in each column, or in the fields in the Property Grid (see below for a description of the fields).
5. Click **Save**.

## Cause and Response Properties

### General Properties

Field	Description
<b>Cluster</b>	The name of the cluster that runs the alarm. This field needs to be defined if your project has more than one cluster.
<b>Alarm Tag</b>	The alarm tag associated with the cause and response information.

### Cause and Response Properties

Field	Description
<b>Cause</b>	A description of the cause of the alarm (maximum of 254 characters).
<b>Response</b>	A description of the appropriate response to the alarm (maximum of 254 characters).
<b>Response Time</b>	A description of the period of time in which the specified response should be acted upon (maximum of 254 characters).
<b>Consequence</b>	A description of the likely outcome if the suggested response is not acted upon within the specified response time (maximum of 254 characters).

### Project Properties

Property	Description
<b>Project</b>	The project in which the cause and response information is included.

## See Also

[Manage Alarms](#)

## Project Database Parameters

Project database parameters can be used to set nominal defaults for parameters that are local to a specific Plant SCADA project, rather than a particular computer. For example, you could use the project database parameters to specify login requirements for all computers in a Plant SCADA system.

**Note:** Parameter settings in the Citect.ini file take precedence over project database parameters. For more information, see [About Parameters](#).

### To set parameters in the project database:

1. In the **Setup** activity, select **Parameters**.
2. Add a row to the Grid Editor.
3. Type the required information in each column, or in the Property Grid (see below for a description of the properties).
4. Click **Save**.

If you set (or change) parameters in the project database, you need to recompile the project before the new parameter settings are applied.

## Parameters Properties

### General Properties

Field	Description
<b>Section Name</b>	The parameter section. Enter a value of 48 characters or less.
<b>Name</b>	The name of the parameter for which you want to define a value. Enter a value of 64 characters or less. Numeric and digital variables have a default value of 0 and string variables default to "" (empty string). If you do not specify a data type, the local variable will be treated as 16-bit integer.
<b>Value</b>	The value of the parameter. Enter a value of 254 characters or less.
<b>Comment</b>	Any useful comment (254 characters).

### Project Properties

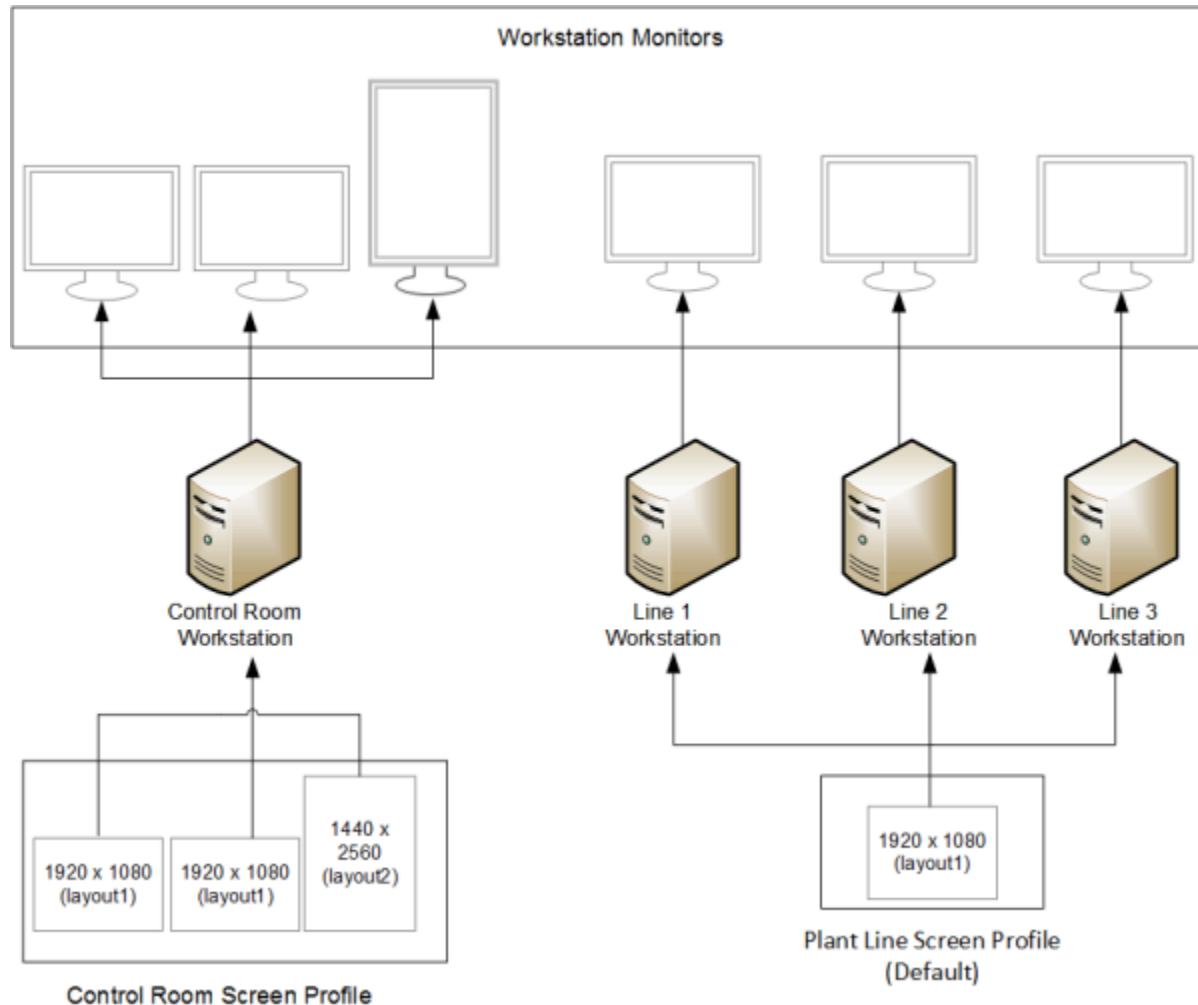
Property	Description
<b>Project</b>	The project in which the project database parameter is configured.

## Screen Profiles

A screen profile specifies the physical characteristics of one or more workstation screens and how these screens are arranged with respect to each other. Each screen icon that appears in a screen profile represents a physical screen.

You can create, manage and configure screen profiles with the Screen Profile Editor, which is displayed below the Screen Profile table in the **Setup** activity. You can create as many screen profiles as you want.

A screen profile is applied at runtime using the Setup Wizard (see [Screen Setup](#)).



### Add a screen profile to a project:

1. In the **Setup** activity, select **Screen Profiles**.
2. Type the required information in each column, or in the fields in the Property Grid (see below for a

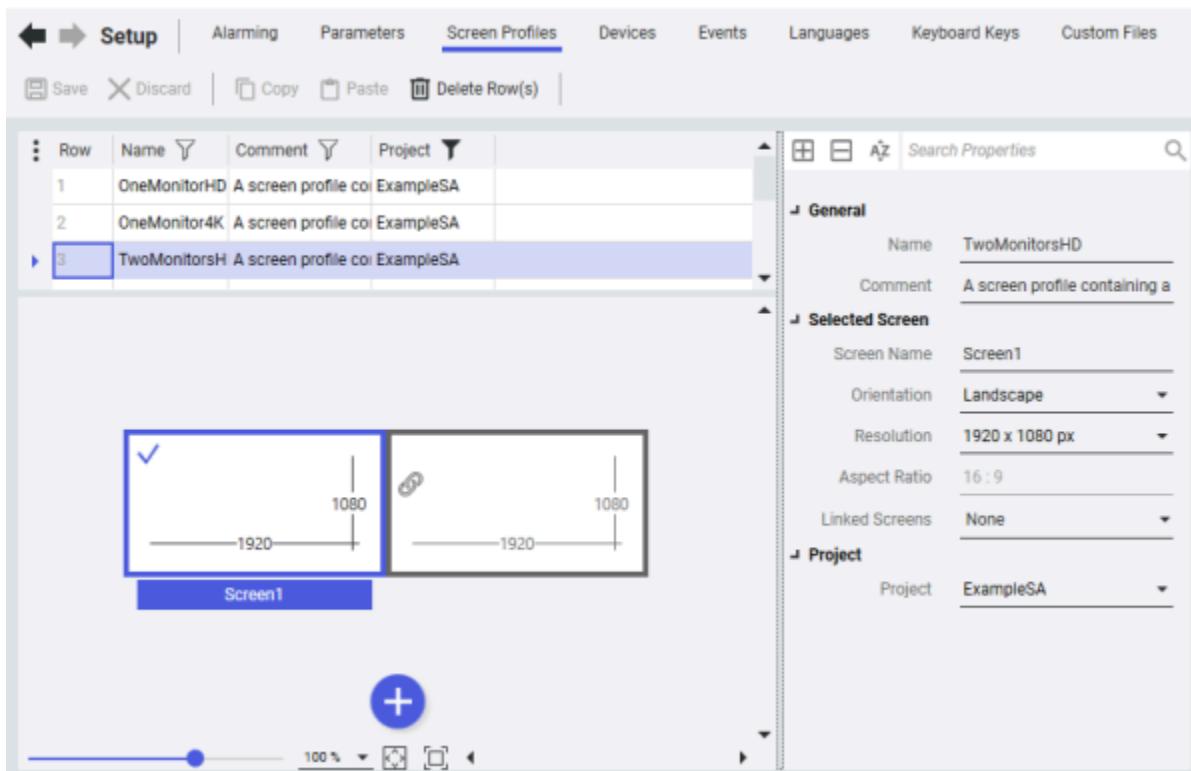
description of the fields).

3. Click **Save**.

Selecting a screen profile in the grid displays the arrangement of monitors for that profile.

#### Add a screen profile using the editor:

The editor that allows you to manage screens is located below the Screen Profiles table in the **Setup** activity. A maximum of 20 screens can be added to a profile.



Click to add a new screen (monitor). A new screen is always added to the right of the selected screen and its resolution is the same as the selected screen. For each screen defined in a profile, you need to specify a name, the screen's orientation and resolution. Also, you need to arrange the screen icons in a screen profile to represent the configuration of actual client workstation screens. You can do this by dragging and dropping the screen icon to match its physical location. The dragged rectangle is red when in an invalid position and green when valid.

By default, the first screen that you add is named "Primary" and the following screens are named "Screen1", "Screen2" and so on. However, these names may be changed in the Screen Profile Property Grid or table. The primary screen displays a solid green tick.

To set a screen as the primary screen, click the clear green tick mark displayed in the top left hand corner of the screen icon. The tick mark changes to a solid green mark, and the screen becomes the primary screen.

---

**Note:** The primary screen should match the primary monitor configuration on Windows on the client workstation.

To delete a screen, click on the top right corner of the screen icon. Alternatively, select the screen by clicking on it and press the **Delete** keyboard button.

To zoom in/out to view the arrangement of screens, use the slider to increase or decrease the zoom. You can also

select the zoom from the list of values next to the slider, or type the required value.

Clicking  adjusts the arrangement of screens to fit the screen. Clicking  sets the zoom to 100%.

## Screen Profile Properties

### General Properties

Field	Description
<b>Name</b>	Name to identify the screen profile. Names should be unique within a project, and cannot contain symbols or characters such as /, \, ?, *, ", <, > and  . Names can contain a maximum of 64 characters.
<b>Comment</b>	Any useful comment about or description of the screen profile.

### Project Properties

Property	Description
<b>Project</b>	The project for which the specified screen profile is created.

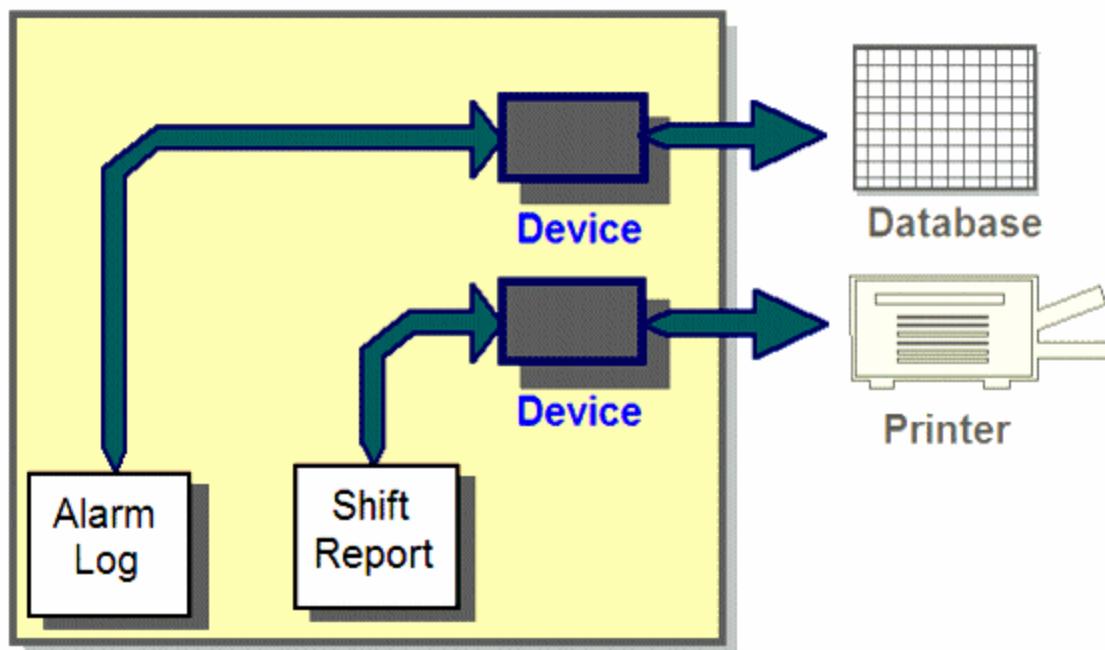
### Selected Screen Properties

Property	Description
<b>Screen Name</b>	<p>The name given to the screen. By default, the primary screen that you add is named "Screen1" and the following screens are named "Screen2", "Screen3" and so on. However, these names may be changed.</p> <p><b>Note:</b> Screen1 cannot be deleted. However, it can be renamed and re-positioned.</p> <p><b>Note:</b> The first screen (Screen1) should match the Windows primary monitor configuration on the client workstation.</p> <p>The screen name should follow these naming rules:</p> <ul style="list-style-type: none"> <li>• Should be less than 32 characters</li> <li>• Can contain only letters, numbers and _ (underscore)</li> <li>• Cannot start with a number</li> <li>• Should contain at least one letter</li> <li>• Should be unique within a screen profile</li> </ul>
<b>Orientation</b>	Whether the display is in Portrait or Landscape mode.

Property	Description
<b>Resolution</b>	<p>Display resolution of the selected screen. You can set the resolution to values between 768 x 768 and 16384 x 16384, or select from the following:</p> <ul style="list-style-type: none"> <li>• XGA (1024 x 768)</li> <li>• SXGA (1280 x 1024)</li> <li>• HD768 (1366 x 768)</li> <li>• WSXGA (1440 x 900)</li> <li>• HD900 (1600 x 900)</li> <li>• UXGA (1600 x 1200)</li> <li>• HD1080 (1920 x 1080) - This is the default.</li> <li>• WUXGA (1920 x 1200)</li> <li>• QHD (2560 x 1440)</li> <li>• WQXGA (2560 x 1600)</li> <li>• 4K UHD (3840 x 2160)</li> </ul>
<b>Aspect Ratio</b>	<p>This is automatically set depending upon the resolution of the screen. This cannot be modified.</p>
<b>Linked Screens</b>	<p>Select screen(s) from the list to link to the current screen. The current screen and linked screens share the same context. Linked screens function as a group and can be linked (as a group) to an individual screen.</p> <p><b>Note:</b> An unlinked screen can be linked only to another unlinked screen individually.</p> <p>Linked screens display a link icon when selected.</p>

## Devices

In a Plant SCADA system, a device is used to transfer high-level data such as reports, command logs or alarm logs. For example, you could use a device to send the output from a report to a printer, or you could use a device to save your alarm logs to a database.



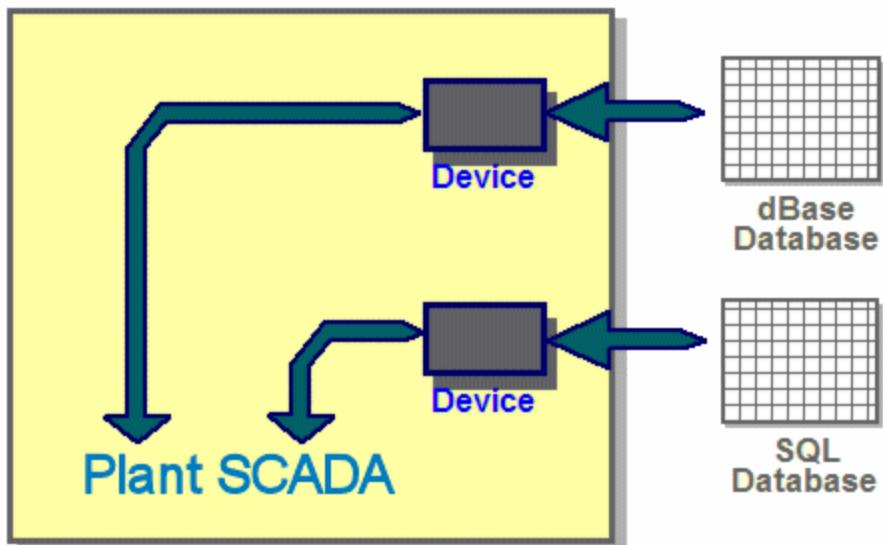
**Note:** A device is very different to an I/O device. An I/O device allows Plant SCADA to exchange plant floor data with a physical piece of equipment via a communications cable. If you are looking for information on I/O devices, see [I/O Devices](#).

A device can write data to the following:

- RTF files
- ASCII files
- dBASE databases
- SQL databases (through ODBC-compliant drivers)
- Printers (connected to your Plant SCADA computer or network).

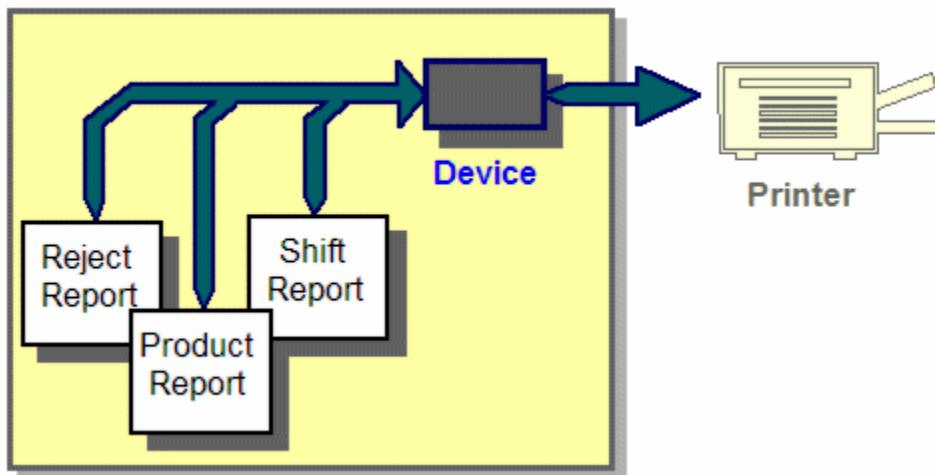
You can also read data from the following sources using a device and Cicode functions:

- ASCII files
- dBASE databases
- SQL databases

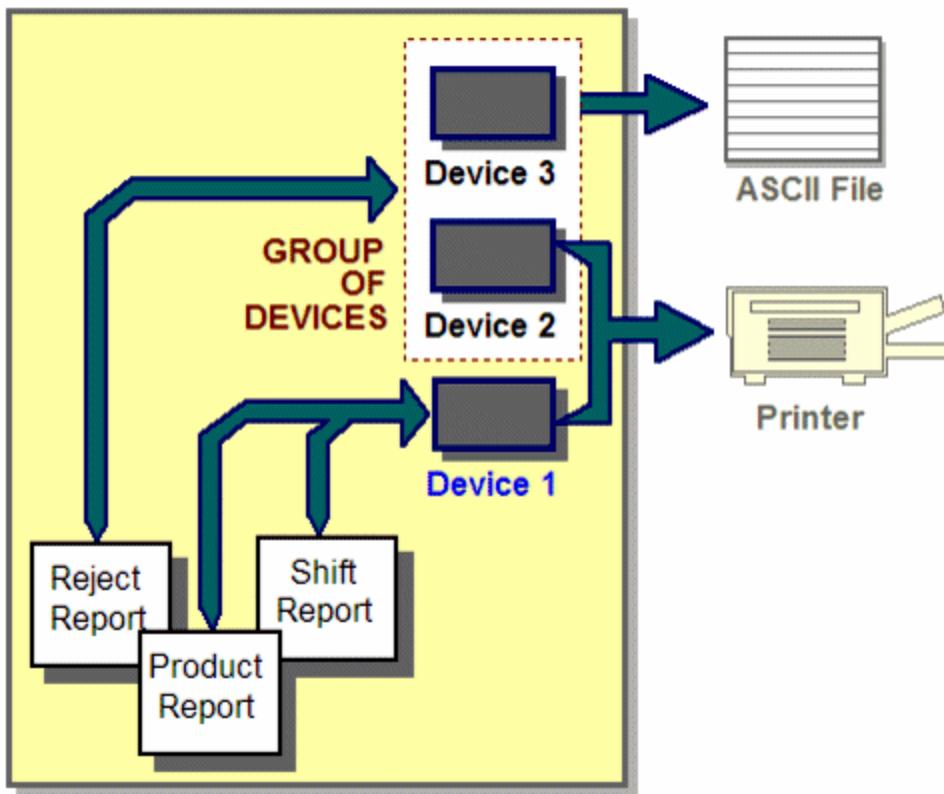


**Note:** You can also use ActiveX data objects (ADO.NET) to create an interface between Plant SCADA and an SQL database. For more information, see [Using Structured Query Language](#).

You can configure any number of devices in a Plant SCADA system. However, a device can be used as a common resource. This means you could configure a single device that sends the output of all your reports to a printer (when they are requested).



You can also configure device groups. A group of devices allows you to export the same data to two (or more) locations.



You can read from a devices group, however, data is only read from the first device in the group.

**Note:** The Plant SCADA Include project contains a set of predefined devices. For a description of the devices that are available, see [Predefined Devices](#).

## See Also

- [Add a Device](#)
- [Format Data in the Device](#)
- [Use Device History Files](#)
- [Print Management](#)

## Add a Device

[Devices](#) are used to transfer high-level data such as reports, command logs or alarm logs.

### To add a device:

1. In the **Setup** activity, select **Devices**.
2. Add a row to the Grid Editor.
3. Type the required information in each column, or in the fields in the Property Grid.  
For a description of the properties, see below.
4. Click **Save**.

**Note:** SQL devices are configured differently to other types of devices. For information, see [Configure SQL](#)

---

[Devices.](#)

## Device Properties

### General Properties

Property	Description
Name	The name of the device. The field is not case-sensitive.
Format	<p>Specifies how the data is formatted in the device. The format is determined by the type of device, and the data that is sent to the device (see <a href="#">Format Data in the Device</a>). Enter a value of 120 characters or less.</p> <p>If you are logging alarms or command messages, you need to specify a format, or no data is written to the device.</p> <p>See <a href="#">Command Fields</a> for a list of the fields (or field combinations) you can use to format a command logging device.</p> <p>For a list of the fields you can use to format an alarm logging device, see the following:</p> <ul style="list-style-type: none"><li>• <a href="#">Alarm Display Fields</a></li><li>• <a href="#">Alarm SOE Fields</a></li><li>• <a href="#">Alarm Summary Fields</a>.</li></ul> <p><b>Note:</b> The log device for a command is specified wherever the command is defined. The log device for an alarm is specified at the Alarm Categories form.</p> <p>When producing reports, the format is ignored. (The format defined for the report is used to write the report to the device.)</p>
Header	Additional information for the device. Enter a value of 254 characters or less.
File Name	The file name of the device. Enter a value of 253 characters or less.
Type	<p>The type of device. Enter a value of 16 characters or less.</p> <ul style="list-style-type: none"><li>• ASCII_DEV - ASCII file *</li><li>• PRINTER_DEV - Printer</li><li>• DBASE_DEV - dBASE file</li><li>• SQL_DEV - SQL database</li></ul>

Property	Description
	<p>* When defining RTF report properties, an ASCII device would be selected if the report was to be saved as a file.</p> <p>This property is optional. If you do not specify a type, the device <b>Type</b> is ASCII_DEV unless:</p> <ul style="list-style-type: none"> <li>• The file name is a printer device (LPT1: to LPT4: or COM1: to COM4: or a UNC name), where <b>Type</b> is PRINTER_DEV, or</li> <li>• The file name extension is .DBF, where <b>Type</b> is dBASE_DEV.</li> </ul> <p>See <a href="#">Print Management</a>.</p>
<b>No. Files</b>	<p>The number of history files (see <a href="#">Use Device History Files</a>). Enter a value of 4 characters or less.</p> <p>By default, Plant SCADA creates a single data file for each device. (This data file is called &lt;filename.TXT&gt; or &lt;filename.DBF&gt;, depending whether the device is an ASCII device or database device.) The number of history files you specify here are in addition to the data file.</p> <p><b>Note:</b> If you do not want history files created, you need to enter 0 (zero) here, and set the <a href="#">[Device]CreateHistoryFiles</a> parameter to 0; otherwise, 10 history files will be created as a default. You need to also verify that the data file is of a fixed size. (If the data accumulates, the file eventually fills the hard disk.)</p> <p>If you specify -1 the data is appended to the end of one file.</p> <p>If you are logging alarm, keyboard commands, or reports to the device, specify the number of files to be created, and the time of each file.</p>
<b>Time</b>	<p>The time of day to synchronize the beginning of the history file, in hh:mm:ss (hours:minutes:seconds). Enter a value of 32 characters or less. If you accepted the default number of history files above, and you specify a time and period, 10 history files will be created. If you do not specify a time, the file is synchronized at 0:00:00 (i.e. midnight).</p> <p>If you omit both the time and the period, additional history files will still be created (with the default time and period). If you don't want history files to be</p>

Property	Description
	created, you need to set the <a href="#">[Device]CreateHistoryFiles</a> parameter to 0 (zero).
<b>Period</b>	<p>The period of the history file, in hh:mm:ss (hours:minutes:seconds). Enter a value of 32 characters or less. Alternatively you can:</p> <ul style="list-style-type: none"> <li>Specify a weekly period by entering the day of the week on which to start the history file, for example Monday, Tuesday, Wednesday, etc.</li> <li>Specify a monthly period by entering the day of the month on which to start the history file, for example 1st, 2nd, 3rd, 4th, 5th, etc.</li> <li>Specify a yearly period by entering the day and month on which to start the history file, for example 1st January, 25th February, etc. The day and month needs to be separated by a space.</li> </ul> <p>If you accepted the default number of history files above, and you specify a time and period, 10 history files will be created.</p> <p>If you do not specify a period, the time field will be ignored and the device will assume users will invoke device logging manually.</p> <p>If you omit the time but have specified a period, additional history files will still be created (with the default time mid-night at 00:00:00). If you don't want history files to be created, you need to set the <a href="#">[Device]CreateHistoryFiles</a> parameter to 0 (zero).</p>
<b>Comment</b>	Any useful comment.
<b>Cluster Name</b>	Select a cluster from the list of clusters defined previously with the in the <b>Topology</b> activity of Plant SCADA Studio. If this is a single cluster system this field can be left blank.
<b>Process</b>	Select the type of server (or client) on which the process runs that sends data to the device. This field is used to prevent a history file being created while the device is active. If there is no history processing then this field can be left blank.

## Project Properties

Property	Description
Project	The project that includes the device.

## See Also

[Configure a Device Group](#)

## Configure SQL Devices

SQL devices are configured and used in a slightly different way than other types of devices.

### To configure an SQLdevice:

1. In the **Setup** activity, select **Devices**.
2. Add a row to the Grid Editor.
3. Complete the following fields:
  - **Name** - Specify a name for the device, for example mySqlDevice.
  - **Format** - Specify the format of the device. The format is the definition of the column names in the target database table. For example:  
`{date,10}{day,15}{month,20}{year,4}`
  - **Header** - Specify the connection string for the database. For example, Driver={SQL Server};Server=MyServer.
  - **File name** - Specify the complete database table name. For example, TestDatabase.dbo.MySqlCalendar.
  - **Type** - The type should be set to SQL\_DEV.
4. Configure the rest of the fields as per other devices, see [Add a Device](#).
5. Click **Save**.

## Example

When configured, the device can be associated with SCADA elements, or used in Cicode. SQL devices are used in Cicode in a slightly different way to other types of devices. For example, the order of DevSetField and DevAppend is different with SQL devices.

The following example makes a connection to a SQL device, cleans the records, adds new records to the device, finds records matching the condition, then closes the connection:

```
FUNCTION
DataSQLDeviceLogging()
  STRING strValue = "";
INT hConnection = DevOpen("mySQLDevice") // open the device
  IF hConnection <> -1 THEN
    DevZap(hConnection); //clean the records
    DevSetField(hConnection, "date_","30-03-2011");
    DevSetField(hConnection, "day_","Wednesday");
    DevSetField(hConnection, "month_","March");
    DevSetField(hConnection, "year_","2011");
    DevAppend(hConnection);
```

```
DevSetField(hConnection, "date_","31-03-2011");
DevSetField(hConnection, "day_","Thursday");
    DevSetField(hConnection, "month_","March");
DevSetField(hConnection, "year_","2011");
    DevAppend(hConnection);
    DevSetField(hConnection, "date_","01-04-2011");
DevSetField(hConnection, "day_","Friday");
    DevSetField(hConnection, "month_","April");
DevSetField(hConnection, "year_","2011");
    DevAppend(hConnection);
    DevSetField(hConnection, "date_","02-04-2011");
DevSetField(hConnection, "day_","Saturday");
DevSetField(hConnection, "month_","April");
    DevSetField(hConnection, "year_","2011");
    DevAppend(hConnection);
    IF DevFind(hConnection, "April", "month_") = 0 THEN
        strValue = StrTrim(DevGetField(hConnection, "day_"));
    END;
    DevClose(hConnection); //close the connection
END;
END
```

## Configure a Device Group

A device group allows you to send data to a number of devices using a single group name. The following example shows the property settings for a group called "AlarmInfo".

Group Property	Value
Group Name	AlarmInfo
Association 1	AlarmPrint
Association 2	AlarmLog
Association 3	AlarmDBF

In this case, when the group name "AlarmInfo" is used, the information will be sent to three devices - "AlarmPrint", "AlarmLog", and "AlarmDBF".

### To configure a device group:

1. In the **Security** activity, select **Groups**.
2. Add a row to the Grid Editor.
3. Configure the fields as required. For a description of the fields, see Add a Group.
4. Click **Save**.

**Note:** You can read from a devices group, however, data is only read from the first device in the group.

## See Also

[Add a Device](#)

## Format Data in the Device

The device format specifies how to format the data in the device. The format is determined by the type of device, and the data that is sent to the device.

### Printer and ASCII Devices Format

The format specifies how each line of data is printed on the printer or written to the ASCII file, for example:

```
RFP3 Raw Feed pump 3 Overload 12:32:21  
RFP9 Secondary Feed Overtemp 13:02:45
```

To include Plant SCADA data you need to specify the field name and (optionally) a width for each field to be printed or written to the file. The format has the following syntax:

```
{<field name>, [width[, justification]]}
```

You need to enclose each field in braces {}, for example:

Format	{Tag,8}{Name,32}
--------	------------------

In this case, two fields are printed or written to the file - Tag, with 8 characters, and Name, with 32 characters. The width specifier is optional - if you do not specify a width, the width of the field is determined by the number of characters between the braces, for example:

Format	{Name }
--------	---------

In this case, Name is followed by four spaces - the field is printed or written to the file with 8 characters.

#### List and Tables

To set the justification of the text in each field, use a justification specifier. You can use three justification characters, L (Left), R (Right), and N (None) - for example:

Format	{Tag,8,L} {Name,32,R}
--------	-----------------------

The justification specifier is optional - if it is omitted, the field is left justified. If you use a justification specifier, you need to also use the width specifier.

To display field text in columns, use the tab character (^t) - for example:

Format	{Tag,8}^t{Name,32}^t{Desc,32} {Time,8,R}
--------	--

#### Fixed text

You can include fixed text by specifying the text exactly as it is to be printed or written to the file - for example:

Format	Name of Alarm:
--------	----------------

Any spaces that you use in a text string are also included in the string.

**Note:** When producing reports, the device format is ignored. The format for the report is defined in the report format file.

## **dBASE and SQL Database Devices Format**

The format specifies the structure (field names and field widths) of the database. The format has the following syntax:

{<field name>, <width>}

You need to use braces (`{ }`) to enclose each field, for example:

Format	{Tag,8}{Name,32}
--------	------------------

In this case, the database is created with two database fields - Tag (with 8 characters) and Name (with 32 characters). The size of each database field is determined by the width you specify in the format. (Justification character are ignored.) Every database field is character (string) field types.

You can define your own fields (as well as the standard Plant SCADA fields) for the database device, for example:

Format	{Name,16}{Water,8}{Sugar,8}{Flour,8}{Salt,8}{Yeast,8}{Milk,8}
--------	---

This database device has the following structure:

Do not leave any spaces between each field definition or at the end of the format string, or Plant SCADA creates extra fields for each space (in the device). Do not specify a field name longer than 10 characters, or Plant SCADA truncates the name to 10 characters. (The dBASE file format limits field names to a maximum of 10 characters.)

In this example, the database is created with seven database fields. To access the above dBASE device, use a Cicode function similar to the following:

```
hDev = DevOpen("Recipe");
DevFind(hDev, "Name", "Bread");
PLC_Water = DevGetField(hDev, "Water");
PLC_Sugar = DevGetField(hDev, "Sugar");
. . .
. . .
DevClose(hDev);
```

## DBase Devices

If the database does not exist, it is created with the specified format. If you do not specify a format, and if the file name specifies an existing dBASE file, Plant SCADA uses the existing fields in the database.

## SQL Devices

You need to create the SQL database by an external application before it can be used.

If you edit a dBASE or SQL device record (in an existing project), the associated physical device is not edited. For example, if the device is a dBASE type device and you add an extra field in the device, the extra field is not added to existing database files (when you run Plant SCADA).

If you want to make changes to the file structure of an existing database and retain existing data, you need to manually edit the data using dBASE, Excel or some other database tool.

If you want to make changes to the file structure of an existing database and don't want to keep the existing data:

1. Close the device.
2. Manually delete the device file.
3. Make your modifications.
4. Open the device.

When the device is opened, no device file is available, and a new data file is created with the necessary changes.

## See Also

[Command Fields](#)

### Command Fields

You use the following fields (or combination) to format a command logging device:

Field Name	Description
{UserName,n}	The name of the user (User Name) who was logged on when the command was issued.
{FullName,n}	The full name of the user (Full Name) who was logged on when the command was issued.
{Time,n}	The time (in short format) when the command was issued (hh:mm).
{TimeLong,n}	The time (in long format) when the command was issued (hh:mm:ss).
{Date,n}	The date (in short format) when the command was issued (dd:mm:yy).
{DateLong,n}	The date (in long format) when the command was issued (day month year).
{DateExt,n}	The date (in extended format) when the command was issued (dd:mm:yyyy).
{Page,n}	The page that was displayed when the command was issued.
{MsgLog,n}	The message sent as the <i>Message Log</i> property (of the command record).

You can use the following fields (in the command field) for **Keyboard commands only**:

{Arg1,n}	The first keyboard command argument (if any).
{Arg2,n}	The second keyboard command argument (if any).
...	
{Arg8,n}	The eighth keyboard command argument (if any).
{Native_MsgLog,n}	The native language version of the message sent as the <i>Message Log</i> property (of the command record).

Where n specifies the display field size.

For example, you could have a device configured as follows:

Name	KeyLog
Format	{Date,9} {MsgLog,27} {Arg1,3} by {FullName,11}

Then a keyboard command (object, page, or system) could be created with the following configuration:

Log Device	KeyLog
Key Sequence	### ENTER
Log Message	Density setpoint changed to

Resulting in an output of the following kind: "01/01/99 Density setpoint changed to 123 by John Smith".

## Use a Database Device

You can use a database device to do the following:

### Write dBase Records

To write data to a database device, first append a record to the end of the device, then add data to the fields of the record. For a dBASE database, the DevAppend() function appends the record, and the DevSetField() function writes data to a field. The following function writes a recipe record to a device:

```

FUNCTION
WriteRecipeData(INT hDevice, STRING sName, INT Water, INT Sugar,
    T Flour, INT Salt, INT Yeast, INT Milk)
    DevAppend(hDevice);
    DevSetField(hDevice, "NAME", sName);
    DevSetField(hDevice, "WATER", IntToStr(Water));
    DevSetField(hDevice, "SUGAR", IntToStr(Sugar));
    DevSetField(hDevice, "FLOUR", IntToStr(Flour));
    DevSetField(hDevice, "SALT", IntToStr(Salt));
    DevSetField(hDevice, "YEAST", IntToStr(Yeast));
    DevSetField(hDevice, "MILK", IntToStr(Milk));
END

```

## Write SQL Records

To use an SQL device in Plant SCADA, you cannot use every the Cicode Device function; you can only use the following functions:

DevOpen()	DevClose()	DevGetField()	DevFind()	DevWrite()
DevNext()	DevSeek()	DevAppend()	DevWrite()	DevZap()
DevControl()	DevSetField()			

To write Plant SCADA data to an SQL database, use the DevWrite() function, but you need to add a new record and write to fields in the new record. No data is written to the database if you do not write to all fields.

For example:

```
FUNCTION
WriteRecipeData(INT hDevice, STRING sName, INT Water, INT Sugar,
                INT Flour, INT Salt, INT Yeast, INT Milk)
    DevWrite(hDevice, sName);
    DevWrite(hDevice, Water);
    DevWrite(hDevice, Sugar);
    DevWrite(hDevice, Flour);
    DevWrite(hDevice, Salt);
    DevWrite(hDevice, Yeast);
    DevWrite(hDevice, Milk);
END
```

The following functions return error 267 (File mode is invalid) when the device type is SQL:

DevFlush()
DevPrev()
DevDelete()

The follow function returns error 263 (Cannot read file) when the device type is SQL:

DevRead()
-----------

The following functions return error -1 when the device type is SQL:

DevSize()
DevRecNo()

## Locate and Read Database Records

To read data from a dBASE or SQL database device, use the DevFind() function to locate the record, and then the DevGetField() function to read each field:

```
FUNCTION
GetRecipe(STRING sName)
    INT hDev;
    hDev = DevOpen("Recipe");
    IF hDev >= 0 THEN
```

```
IF DevFind(hDev, sName, "NAME") = 0 THEN
    PLC_Water = DevGetField(hDev, "WATER");
    PLC_Sugar = DevGetField(hDev, "SUGAR");
    PLC_Flour = DevGetField(hDev, "FLOUR");
    PLC_Salt = DevGetField(hDev, "SALT");
    PLC_Yeast = DevGetField(hDev, "YEAST");
    PLC_Milk = DevGetField(hDev, "MILK");
ELSE
    DspError("Cannot Find Recipe " + sName);
END
DevClose(hDev);
ELSE
    DspError("Cannot open recipe database");
END
END
```

## Delete Records

You can delete dBASE records with the DevDelete() function. The following Cicode function deletes records from a dBASE device:

```
FUNCTION DeleteRecords(INT hDev)
    WHILE NOT DevEOF(hDev) DO
        DevDelete(hDev);
        DevNext(hDev);
    END
END
```

To delete every record from a dBASE database, use the DevZap() function:

```
FUNCTION DeleteRecords(INT hDev)
    DevZap(hDev);
END
```

To delete records from an SQL database, use the Cicode SQL functions.

To open a database device or devices:

Use the DevOpen() function. this function returns an integer handle to identify each device, as in the following example:

```
INT hRecipe;
hRecipe = DevOpen("Recipe");
```

When you are finished with a device, close it to free Cicode system resources by using the DevClose() function. For example:

```
DevClose(hRecipe);
```

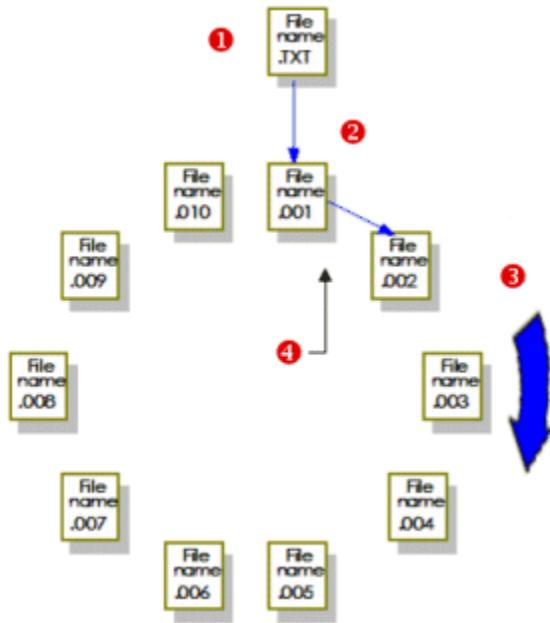
## See Also

[Use Device History Files](#)

## Use Device History Files

To make the long-term storage of logged data easier to organize and more accessible, Plant SCADA uses a system of rotational history files to store historical data. To use this system, you need to specify how many device history

files you want to keep. For example, if you want to keep 10 history files, they would be saved rotationally as illustrated below:



1. When Plant SCADA begins logging, data is written to a file called <filename>.txt or <filename>.dbf, depending on the type of device.
2. At midnight the following Sunday, the file <filename>.txt is renamed to <filename>.001 and a new <filename>.txt is created.
3. At midnight the following Sunday, the file <filename>.001 is renamed to <filename>.002, <filename>.txt is renamed to <filename>.001, and then a new <filename>.txt is created.
4. After week 10, the first file is overwritten (week 11 in the first cycle).

The history files will not be created if the device is not being used.

---

**Note:** To archive your data for long-term storage, manually back up your history files before they are overwritten. The 10 history files are in addition to the default data file that is saved for each device.

---

By default, Plant SCADA uses 10 files (if history files are specified). You can change the default by specifying the number of files to use, for example:

No. Files	20
Comment	Plant SCADA uses twenty files for the data

The maximum number of files you can specify is 999.

You can also specify the period between files, that is, when a new history file is used, for example:

Period	Comment
1:00:00	Use a new file each hour
6:00:00	Use a new file every six hours

Period	Comment
72:00:00	Use a new file every three days
Monday	Use a new file each week beginning on Monday
15th	Use a new file every month beginning on the 15th of each month
25th June	Use a new file every year beginning on the 25th of June

**Note:** Marked improvement in system performance is observed when a period of one week or more is specified.

You can also specify the time of day to synchronize the beginning of the history file, for example:

Time	Comment
6:00:00	Synchronize the file at 6:00 am
12:00:00	Synchronize the file at 12:00 midday
18:30:00	Synchronize the file at 6:30 pm

The first file does not actually begin at this time: the first file begins when you start your runtime system. The time and period together determine when new history files are created, for example:

Time	Period
6:00:00	Monday

In the above example, Plant SCADA creates a new file each Monday at 6:00am. If you start your runtime system at 7:30am on Sunday, your first file only contains 22.5 hours of data. If you leave your system running, subsequent files start each Monday at 6:00am, and contain one full week of data.

## Predefined Devices

This section describes devices that are predefined in the [Include Project](#).

### Devices database

The table below shows the devices supported by Plant SCADA.

Device Name	Type	Description
ASCII_DEV	0	Ascii Device number
PRINTER_DEV	1	Printer Device number
dBASE_DEV	2	dBASE device number
SQL_DEV	4	SQL device number
AlarmDisk	0 (ASCII File)	Default alarm log file

Device Name	Type	Description
AlarmPrint	0 (ASCII File)	Default alarm print device
KeyDisk	0 (ASCII File)	Default keyboard log file
KeyPrint	1 (Printer)	Default keyboard printer log
Printer1	1 (Printer) LPT1:	Printer 1 device
Printer2	1 (Printer) LPT2:	Printer 2 device
SummaryPrint	0 (ASCII File)	Default alarm summary printer device
SummaryDisk	0 (ASCII File)	Default alarm summary log file
_Trend	3 (dBASE)	Trend RDB device
Scratch	0 (ASCII File)	Device for DevModify function

**Note:** Do not modify the Include Project. Changes to the Include project are lost when you reinstall or upgrade Plant SCADA.

## Print Management

The Windows printer management has been designed for page-based printers: laser printers and shared network printers. The printer driver does not print anything on the printer until the page is full; it then prints the page.

This is the preferred printing method (when printers are shared on a network), because it minimizes the likelihood of conflict of data when more than one operator uses the print facility at the same time.

However, this method is inappropriate when logging alarms or keyboard commands. If you send alarm logging to this type of printer, Plant SCADA flushes the data to the printer when the current activity completes, or when the [DEVICE]FlushTime parameter has been exceeded (it defaults to 10 seconds). If, for example, you have one alarm occurring each minute, each alarm is printed on a new page (because the default flush time is less than the alarm frequency).

You can bypass the Windows print management by writing the output to a file. Set the device type to ASCII\_DEV and specify the file name as lpt1.dos, lpt2.dos or lpt3.dos (depending on the port to which your printer is connected). The printer needs to be either on a local port, or a captured network printer. When you log to this device, the data is printed immediately on the printer with no extra form feeds.

For correct logging operation, reserve one printer to be your logging printer. This printer has to be a local printer, not on the network server. You can then send any other non-logging printouts, (for example, reports) to a shared network or local printer.

## See Also

[Devices](#)

## Triggering Events

You can use a triggering event to initiate an action, such as a command or set of commands. For example, an operator can be notified when a process is complete, or a series of instructions can be executed when a process reaches a certain stage.

A triggering event can be run automatically:

- At a specified time and period.
- When a trigger condition becomes TRUE.
- When a trigger condition is TRUE at a specified time and period.

Triggering events needs to be enabled to operate. To do this, use the Setup Wizard (custom setup) to enable **Events** (see [Events Configuration](#)). If using a network, you can process triggering events on any computer (or every computer).

**Note:** The events system is not redundant. That is, it is not possible to assign primary and standby events servers. If you require a redundant event reporting system, use reports instead. See [Reports](#).

## See Also

[Add a Triggering Event](#)

[Add a Triggering Event](#)

[Use Event Triggers](#)

## Add a Triggering Event

You can use a triggering event to initiate an action, such as a command or set of commands.

**To configure a triggering event:**

1. In the **Setup** activity, select the **Events**.
2. Add a row to the Grid Editor.
3. Type the required information in each column, or in the fields in the Property Grid.  
For a description of the properties, see below.
4. Click **Save**.

**Note:** The events server needs to be enabled for triggering events to work. See [Triggering Events](#).

## Event Properties

### General Properties

Property	Description
Name	For a single computer system, specify <b>GLOBAL</b> for the event name.

Property	Description
	<p>If you are using a network and want to run an event on every computer, specify <b>GLOBAL</b> for the event name. If you want to run an event only on specific computers, specify an event name and use the Setup Wizard (Custom setup) to specify which computer(s) will run the event. The event name does not have to be unique: you can specify many events with the same name.</p> <p>Enter a value of 16 characters max.</p> <p><b>Note:</b> Make event names conform to correct <a href="#">Tag Name Syntax</a>. The use of any other characters such as spaces will result in a compiler error. Instead of a space, use an underscore character (_).</p>
<b>Time</b>	<p>The time of day to synchronize the <b>Period</b> in hh:mm:ss (hours:minutes:seconds). If you do not specify a time, <b>Period</b> is synchronized at 00:00:00 (that is, midnight). Enter a value of 32 characters maximum.</p>
<b>Period</b>	<p>The period to check for the event, in hh:mm:ss (hours:minutes:seconds). Enter a value of 32 characters maximum. Alternatively you can specify:</p> <ul style="list-style-type: none"> <li>• A weekly period by entering the day of the week to check for the event, for example, Monday, Tuesday, Wednesday.</li> <li>• A monthly period by entering the day of the month to check for the event, for example, 1st, 2nd, 3rd, 4th, 5th.</li> <li>• A yearly period by entering the day and the month to check for the event, for example, 1st January, 25th February, and so on. The day and month needs to be separated by a space.</li> </ul> <p>If you do not specify a period or time, the period defaults to one second. If you do not specify a period, but do specify the time, the period defaults to one day.</p>
<b>Trigger</b>	<p>The Cicode expression (or Variable tag) which is used to determine whether the event Action is executed. This expression is checked every one second. Enter a value of 254 characters maximum.</p>
<b>Action</b>	<p>The commands to execute. Enter a value of 64 characters maximum. These commands will execute in</p>

Property	Description
	<p>the following circumstances:</p> <ul style="list-style-type: none"> <li>When the specified <b>Time</b> and <b>Period</b> occurs, and the <b>Trigger</b> condition is TRUE or blank.</li> <li>When the <b>Trigger</b> becomes TRUE, and the <b>Time</b> and <b>Period</b> field are blank. The Trigger needs to become FALSE and TRUE again for the action to re-execute.</li> </ul>
<b>Comment</b>	Any useful comment. Enter a value of 48 characters maximum.
<b>Cluster Name</b>	The default cluster to be used when the event action and event trigger Cicode is run. Any tags in these Cicode expressions will resolve to this default cluster if they don't have their cluster specified inside the expression. This field can be left blank in single cluster systems or if every tag inside the expressions are declared in the <ClusterName>.<TagName> format.

### Project Properties

Property	Description
<b>Project</b>	The project in which the triggering event is included.

### See Also

[Specify Event Times and Periods](#)

[Use Event Triggers](#)

## Specify Event Times and Periods

The **Period** determines when a [Triggering Events](#) is run. You can specify the period in hh:mm:ss (hours:minutes:seconds), for example:

Period	Comment
1:00:00	Run the event every hour
6:00:00	Run the event every six hours
72:00:00	Run the event every three days
Monday	Run the event each Monday

Period	Comment
15th	Run the event on the 15th of each month
25th June	Run the event on the 25th of June

You can also specify the time of day to synchronize the event, for example:

Period	Comment
6:00:00	Synchronize the event at 6:00 am
12:00:00	Synchronize the event at 12:00 midday

The Time synchronizes the time of day to run the event and, with the Period, determines when the event is run; for example:

Time	6:00:00
Period	1:00:00

In this example, the event is run every hour, on the hour. If you start your runtime system at 7:25am, your event is run at 8:00am, and then every hour after that.

## See Also

[Use Event Triggers](#)

## Use Event Triggers

You can use any Cicode expression (or variable tag) as a trigger for an event. If the result of the expression (in the **Trigger** field) becomes TRUE, and if the **Time** and **Period** fields are blank, the event is run. For example:

Time	
Period	
Trigger	RCC1_SPEED<10 AND RCC1_MC

This event is only run when the expression (Trigger) becomes TRUE, that is, when the digital tag RCC1\_MC is ON and the analog tag RCC1\_SPEED is less than 10. The expression needs to become FALSE and then TRUE again before the event is run again.

If you use the Time and/or Period fields, the Trigger is checked at the Time and/or Period specified, for example:

Time	6:00:00
Period	1:00:00
Trigger	RCC1_SPEED<10 AND RCC1_MC

This event is run each hour, but only if the expression (Trigger) is TRUE (that is, if the digital tag RCC1\_MC is ON and the analog tag RCC1\_SPEED is less than 10).

## See Also

[Add a Triggering Event](#)

## Languages

Plant SCADA supports a language switching capability that allows text within the runtime environment to be translated and displayed into a variety of languages.

Language switching is configured in two very different ways, depending on the type of content you are creating.

- **Standard Projects**

With a standard Plant SCADA project you can flag items for translation and then display them in a specified language at runtime. A project can support multiple languages, and you can dynamically switch between them.

To achieve this, you need to provide the required translations by building a localized database for each supported language.

For more information, see [Configure Languages for a Standard Project](#).

- **Industrial Graphics Applications**

If you are producing content for an AVEVA™ Industrial Graphics application, you can apply different set of strings to a graphic object for each supported language. The content that gets displayed is then determined by the language that is selected in the Industrial Graphics Web Client.

For more information, see [Configure Languages for Industrial Graphics Applications](#).

## Configure Languages for a Standard Project

In a standard Plant SCADA project, items such as alarm descriptions, button text, keyboard/alarm logs, graphics text and Cicode strings can be marked for translation and then displayed in a specified language at runtime. A project can support multiple languages, and you can dynamically switch between them.

---

**Note:** If you want to configure language switching for an Industrial Graphics application, see [Configure Languages for Industrial Graphics Applications](#).

Text translation does not happen automatically. A language will only be supported if the required translations are manually inserted into a localized language database.

To enable your runtime system to switch between different languages, your firstly need to mark the text you would like translated and locate the required translations. See [Prepare a Project for Multi-language Support](#).

You will then be able to:

- [Set the Local Language Used at Runtime](#)
- [Change the Local Language at Runtime](#).

---

**Note:**

- Plant SCADA can support both neutral languages (for example, "French") and region-specific variations (for example, "French (Belgium)"). However, each of these will generate and use its own language database ("French.dbf" and "French(Belgium).dbf"). You cannot refer to a regional variation of a language using just the neutral language name.

- 
- It is recommended that you use languages that specify a regional variation, particularly if you plan to migrate your system to the AVEVA Industrial Graphics platform.
- 

## See Also

[Officially Supported Languages](#)

### Prepare a Project for Multi-language Support

To prepare a project for multi-language support, you need to consider the following procedures:

- **Defining the languages supported by a project**

The **Languages** view in Plant SCADA Studio's **Setup** activity allows you to specify which local languages you would like a project to support. During compilation, a local language database is created for each language you define in this view.

See [Define the Languages Supported by a Project](#).

- **Translating the language databases**

Local language databases are used to define the translations from native language to local language. When a project is compiled, any text that is marked for translation is drawn into the supported language databases. You can then manually edit a databases to define the translations required for the specified language.

Recompiling the project will then capture the translations you have input.

See [Translate a Local Language Database](#).

- **Marking text and alarm text for translation**

Language change indicators are used to identify any text you would like translated at runtime.

Plant SCADA distinguishes between a project's *native* language (that is, the language used by the developer during configuration), and the *local* language (the language displayed to the end user). Based on this principle, any text marked with a language change indicator can be converted from native language to a local language at runtime.

See [Mark Text for Translation](#) and for alarm specific strings [Mark Alarm Text for TransltionP](#).

## See Also

[Set the Local Language Used at Runtime](#)

### Define the Languages Supported by a Project

To set up multi-language support in your project, complete the following:

1. In the **Setup** activity, define the required **Languages** for the main project. For example, "French (France)" and/or "Japanese (Japan)".
- 

**Note:** You cannot refer to a regional variation of a language (for example, "French (France)") using just the neutral language name (for example, "French"). These two variations will generate and use separate databases ("French(France).dbf" and "French.dbf"). It is recommended that you use languages that specify a regional variation.

---

2. In the **Setup** activity, select **Parameters** and define the default language for your project. For example, French (France).
3. In the include project (if any and not a standard Include project) define languages in the **Setup** activity. For example, French (France) and Japanese (Japan).
4. Compile and generate the language DBFs.
5. Open each language DBF and insert translation (for main and include projects).
6. Recompile to include the new translations.

**Note:** The language defined in the Citect.ini file will take precedence over the default language defined in your project.

During compilation, a local language database (for example, French.dbf) is generated for each of the defined languages. If the project does not have an existing database for one of the specified languages, one will be created with its native column populated with native strings and local column empty.

Once this process is complete, an .rdb file is added to the master project for each defined language to facilitate the operation of multi-language switching during runtime.

The following table provides an example of how the defined languages would impact the project "MasterProject1" when compiled.

Project name	Defined languages	Existing and newly defined language databases	.dbf files created during compilation	.rdb files created during compilation
		Local Language Databases		Runtime Database
MasterProject1	French(France) English Japanese(Japan)	English.dbf	French(France).dbf Japanese(Japan).dbf	lanen.rdb lanfr-fr.rdb lanja-JP.rdb Plus RDBs for languages defined in default include projects.

If your master project includes an Include project (not one of the include projects included with Plant SCADA) you will need to define the supported languages within your include project as well as in the master project.

The following table provides an example of how the defined languages would impact the project "MasterProject1" when linked to "IncludeProject1" which has its own existing language databases.

Project name	Defined languages	Existing and newly defined language databases	.dbf files created during compilation	.rdb files created during compilation
		Local Language Databases		Runtime Database
MasterProject1	French(France) English Japanese(Japan)	English.dbf	French(France).dbf (1) Japanese(Japan).dbf	lanen.rdb lanfr-fr.rdb (1 & 2 are merged) lanja-JP.rdb

				Ianko-KR.rdb Plus RDBs for languages defined in default include projects
IncludeProject1	French (France) English Korean(Korea)		English.dbf French.dbf (2) Korean(Korea).dbf	

If a project and one of its included projects have variations of the same language database, the translations will be merged into the '.rdb file', in the table above the French files (1 & 2) were merged.

You can also see that during compilation the Korean.dbf was created and added to the "MasterProject". Once you have defined the required languages, and defined the default language for your project you can open the language DBFs and insert translation (for main and include projects).

---

**Note:** If you do not define any languages in Plant SCADA Studio, Plant SCADA will create a default file "\_lanEN.rdb". The native text will be used for the associated translations.

---

### To define the languages supported by a project:

1. In the **Setup** activity, select **Languages**.
2. Add a row to the Grid Editor.
3. Configure the fields as required. For a description of the fields, see [Languages Properties](#).
4. Click **Save**.

---

**Note:** When setting the properties for you will be required to associate a setting for the parameter **[CtEdit]ANSIToOEM** for each language. This parameter defines whether or not the language requires translation to OEM. See the topic [OEM Character Sets](#) for more information.

---

### See Also

[Translate a Local Language Database](#)

[Officially Supported Languages](#)

[Define an Unsupported Language](#)

## Languages Properties

Languages have the following properties (see [Define the Languages Supported by a Project](#)).

### Language

Select the language from the drop down list provided.

### Source File (Optional)

A file which contains translation for the chosen language. Enter the complete file name (with extension) For example, Spanish.DBF.

The compiler raises an alert for any specified file names with the wrong extension or without an extension.

If a file is specified but cannot be found, the compiler creates a DBF file in that name with empty translations.

If this field is empty and "English(Australia)" has been chosen, the compiler will look for "English(Australia).DBF" first, if not found, then "English.DBF", if still not found, it will create a file named "English(Australia).DBF" with empty translations.

#### ANSIToOEM

This field corresponds to the Citect.ini parameter [\[CtEdit\]ANSIToOEM](#), which determines if ANSI to OEM translation is required to support a language. Once you associate a setting for this parameter with a language, the parameter will be adjusted accordingly whenever the language is implemented. See [OEM Character Sets](#) for more information.

#### Comment

Any useful comment (optional).

## See Also

[Officially Supported Languages](#)

# Officially Supported Languages

Technically any languages that are supported by the OS can be used in the project (by defining it in the **Languages** view in Plant SCADA Studio's **Setup** activity). However, the subset of languages that Plant SCADA "officially" supports is listed below.

**Note:** Plant SCADA can support both neutral languages (for example, "French") and region-specific variations (for example, "French (Belgium)"). However, each of these will generate and use its own language database ("French.dbf" and "French(Belgium).dbf"). You cannot refer to a regional variation of a language using just the neutral language name. It is recommended that you use languages that specify a regional variation.

Language	Language Code	Charset
Chinese	ZH	134
Chinese (Simplified, PRC)	ZH-CN	134
English (Australia)	en-AU	0
English (Belize)	en-BZ	0
English (Canada)	en-CA	0
English (Caribbean)	en-CB	0
English (India)	en-IN	0
English (Ireland)	en-IE	0
English (Jamaica)	en-JM	0
English (Malaysia)	en-MY	0

Language	Language Code	Charset
English (New Zealand)	en-NZ	0
English (Philippines)	en-PH	0
English (Singapore)	en-SG	0
English (South Africa)	en-ZA	0
English (Trinidad & Tobago)	en-TT	0
English (United Kingdom)	en-GB	0
English (United States)	en-US	0
English (Zimbabwe)	en-ZW	0
French	fr	0
French (Belgium)	fr-BE	0
French (Canada)	fr-CA	0
French (France)	fr-FR	0
French (Luxembourg)	fr-LU	0
French (Monaco)	fr-MC	0
French (Switzerland)	fr-CH	0
German	de	0
German (Austria)	de-AT	0
German (Germany)	de-DE	0
German (Liechtenstein)	de-LI	0
German (Luxembourg)	de-LU	0
German (Switzerland)	de-CH	0
Italian	it	0
Italian (Italy)	it-IT	0
Italian (Switzerland)	it-CH	0
Japanese	ja	128
Japanese (Japan)	ja-JP	128

Language	Language Code	Charset
Korean	ko	129
Korean (Korea)	ko-KR	129
Portuguese	pt	0
Portuguese (Brazil)	pt-BA	0
Portuguese (Portugal)	pt-PT	0
Russian	ru	204
Russian (Russia)	ru-RU	204
Spanish	es	0
Spanish (Argentina)	es-AR	0
Spanish (Bolivia)	es-BO	0
Spanish (Chile)	es-CL	0
Spanish (Colombia)	es-CO	0
Spanish (Costa Rica)	es-CR	0
Spanish (Dominican Republic)	es-DO	0
Spanish (Ecuador)	es-EC	0
Spanish (El Salvador)	es-SV	0
Spanish (Guatemala)	es-GT	0
Spanish (Honduras)	es-HN	0
Spanish (Mexico)	es-MX	0
Spanish (Nicaragua)	es-NI	0
Spanish (Panama)	es-PA	0
Spanish (Paraguay)	es-PY	0
Spanish (Peru)	es-PE	0

## See Also

[Logging Data in Different Languages](#)  
[Define an Unsupported Language](#)

# Define an Unsupported Language

If necessary, you can define languages that are not [officially supported](#) by Plant SCADA in the **Setup | Languages** view in Plant SCADA Studio.

The format for specifying a non-supported language needs to follow Windows language and region definitions.

The format for specifying an unsupported language is:

<LanguageName>(<RegionName>)

For example,

"Hebrew(Israel)", where the language name is "Hebrew", and the region name is "Israel".

"Chinese (Traditional)(Taiwan)", where the language name is "Chinese (Traditional)", and the region name is "Taiwan".

When an unsupported language is specified and is used at login (that is, with the [LoginForm\(\)](#), [Login\(\)](#) or [UserLogin\(\)](#) Cicode functions), the runtime will display the prompt line "Unsupported language" and display the content in the specified language.

If the specified language is unsupported, some system texts are displayed in English rather than the specified language, such as the SOE message field. It is possible that the system may not be able to determine the character set of a non-Plant SCADA-supported language. In this case, you need to provide their desired character sets by:

- Setting [\[Language\]CharSet = <value>](#),
- Calling [ParameterPut\("Language", "CharSet", <value>\)](#)

Allowable values for <value> are:

- 0 - ANSI ASCII
- 1 - System Default Character Set
- 128 - Japanese - Shift JIS
- 129 - Korean - Hangul
- 130 - Korean - Johab
- 134 - Chinese - simplified GB2312
- 136 - Chinese - traditional Big5
- 161 - Greek
- 162 - Turkish
- 163 - Vietnamese
- 177 - Hebrew
- 178 - Arabic
- 186 - Baltic
- 204 - Russian
- 222 - Thai
- 238 - East European

## See Also

[Set the Local Language Used at Runtime](#)

### Translate a Local Language Database

The local language databases created during compilation are located in the project directory. Each consists of two fields: NATIVE and LOCAL.

Any text that is marked with a language change indicator is automatically entered in the NATIVE field when a project is compiled. Text that has been marked since the last compile is appended to the database; the rest of the database remains unchanged.

To include the required translations for a specific language, open the associated database in a database editor (such as Microsoft Excel™ with DBF Add-in) and enter the localized translations in the LOCAL field.

For example, French.dbf may have the following translations added to it:

NATIVE	LOCAL
Line Disconnected Alarm at Line Speed	alarme "Ligne deconnectee" sur la vitesse de la ligne
Main Menu page	Page du menu principal
Conveyor Belt Trip	Tapis roulant declenche

When the project is run, the French translations will display in place of the associated native text.

If you do not enter a local equivalent for a native text string, the native text is displayed by default. You can specify to display "#MESS" instead of the native text by setting the [\[Language\]DisplayError](#) parameter to 1 (one); the default is 0 (zero).

**Note:** The [\[Language\]DisplayError](#) parameter does not apply to alarm strings.

A language database can contain entries which are not actually included in a project. This means that a single language database can be developed that is applicable to many projects.

Once you have completed adding translations to a database, you will need to recompile the project to implement the changes.

**Note:** After translation, review the foreign language interface and verify that translated strings fit in their graphics elements correctly.

## See Also

[Set the Local Language Used at Runtime](#)

### Mark Text for Translation

During project development, you need to mark any text you want to change to another language at runtime with a language change indicator, like this:

`@( Native Text [,Width [,Justify]])`

where *Native Text* is the text to be translated. This text will be replaced by the local equivalent at runtime.

Be aware that the brackets are necessary as they specify the extent of the native language text; *Width* and *Justify*

are optional (indicated by the square brackets).

For example, if English is the native language, you could enter the following alarm description:

Equipment Desc:	@(Pump, 20, R)
-----------------	----------------

This indicator serves two purposes; it flags the text as native, and tells Plant SCADA to change the text from native to local at runtime.

By default, the text that you enter here can be in any combination of upper- and lowercase. In other words, *Motor Inoperative* will be considered the same string as *motor inoperative* or *MOTOR Inoperative*, and they will have the same local language translation. Case-sensitivity can be introduced by setting the [\[Language\]CaseSensitive](#) parameter to 1.

---

**Note:**

1. The [\[Language\]CaseSensitive](#) parameter does not apply to alarm strings. Alarm strings are always case sensitive. For example Motor Inoperative is different to MOTOR INOPERATIVE.
2. Alarm fields no longer support Width and Justify options.

Refer to the topic [Mark Alarm Text for TranslationP](#) for more information

---

*Width* can be assigned any value from 0 to 254. If the local text is longer than specified, it is truncated and left-justified. If a width is not specified, the field is the length of the local text and the text left-justified.

*Justify* specifies the text justification and can only be used with *Width*.

*Justify* can be one of the following values:

- **I** or **L** - Left
- **r** or **R** - Right
- **c** or **C** - Center
- **n** or **N** - None

For example, to limit the local text in the previous case to 20 characters with right justification:

Equipment Desc:	@(Pump, 20, R)
-----------------	----------------

Characters that are normally part of the formatting - **@** , **(** - can also be used within the native text. To do this, place a caret (^) character before them. For example, to include a comma without introducing a formatting error:

Equipment Desc:	@(Pump^, overload, 20, R)
-----------------	---------------------------

---

**Note:** The caret (^) character does appear at runtime or in the language database.

## See Also

[Define the Languages Supported by a Project](#)

## Mark Alarm Text for TranslationP

During project development, you need to mark any alarm text you want change to another language at runtime with a language change indicator.

Partial translation of alarm related strings such as "ABC@(DEF)" or "@(ABC)DEF" and multiple markers per sentence such as "@(ABC)DEF@(IJK)" is no longer supported. During compilation, the @ at the beginning of the

sentence is accepted and the brackets removed so that the whole sentence will be translated. Any "@(" encountered in the middle or end of the sentence (anywhere not at the beginning) will trigger a compiler alert message and will be removed.

**Note:** For alarm related strings the Escape character ^ is removed during compilation. When used immediately preceding a valid translation marker such as ^@{(text)}, it will skip translation and will be removed.

The following table outlines translation marker scenarios and what will happen if:

Scenario	Translated	Display as	Language	Compiler Error message
@(Text)	Yes	Text	In selected language	None
@(Text) some text	No	Text some text	In default language e.g. English	Unsupported syntax for marking translation text, it will not be translated.
Some text @(at the end)	No	Some text at the end	In default language e.g. English	Unsupported syntax for marking translation text, it will not be translated.
Some text @{(in the middle) and more	No	Some text in the middle and more	In default language e.g. English	Unsupported syntax for marking translation text, it will not be translated.
@(Text) and @{(more text)	No	Text and more text	In default language e.g. English	Unsupported syntax for marking translation text, it will not be translated.
@@(Text)	No	Text	In default language e.g. English	Unsupported syntax for marking translation text, it will not be translated.
@(Text @ Text)	Yes	Text @ Text	In selected language	<p>None</p> <p><b>Note:</b> @ by itself is regarded as a normal character, translation marker requires @ to be immediately</p>

Scenario	Translated	Display as	Language	Compiler Error message
				followed by ( and a ) to follow later. The string will appear in the local language DBFs as "Text @Text". Double layers of translation markers are invalid.

## See Also

[Define the Languages Supported by a Project](#)

# Set the Local Language Used at Runtime

The default local language that gets displayed at runtime is determined by the Citect.ini parameter [\[Language\]LocalLanguage](#). The language entered in this parameter should be pre-defined in the [Languages](#) view in Plant SCADA Studio's [Setup](#) activity.

Once this parameter is set (and the steps outlined in [Prepare a Project for Multi-language Support](#) are complete), you can then configure either the [Login\(\)](#), [UserLogin\(\)](#) or [LoginForm\(\)](#) functions to set the preferred language for the project at runtime. When the user logs in to the project, any marked native text will be replaced by the preferred language defined in the corresponding .dbf file. Be aware that changing languages at runtime will cause any open pages to reload.

**Note:** To use characters for Baltic, Central European, Cyrillic, Greek, Turkish, and Asian languages, or right-to-left languages (Arabic, Hebrew, Farsi, and Urdu) the operating system needs to have the corresponding language version of Windows, or have installed system support for that language.

## See Also

[Logging Data in Different Languages](#)

[Change the Local Language at Runtime](#)

# Logging Data in Different Languages

Alarm and keyboard logs can be processed in both the native and the local language. This means that both native and local users can read the historical logs. The logs can employ the same device, or separate devices.

Logs in the local language are produced using the standard field names. For example, if {NAME} {DESC} {COMMENT} is entered in the format field of an alarm category, the alarm name, description and comment of alarms in that category will be logged in the local language.

All fields which support the automatic language change facility can also be logged in the native language. To do

so, just precede the field name with **NATIVE**. For example, to log the name, description and comment of a category of alarms, enter {NATIVE\_NAME} {NATIVE\_DESCRIPTION} {NATIVE\_COMMENT} in the format field for that category.

To log both native and local to the same device, just enter the standard fields, and the native fields together in the format field. To log them to different devices, use a Group of two devices, and enter the local fields as the format for one, and the native fields as the format for the other.

## See Also

[ASCII and ANSI Character Sets](#)

### Change the Local Language at Runtime

The language of runtime display items such as alarm descriptions, button text, keyboard/alarm logs, graphic text, Cicode strings and so on can be changed using one of the following functions:

- Login
- UserLogin
- LoginForm.

Normal operations of the project continue unaffected.

Local translations that are missing from the specified language database are replaced by the native equivalent. You can specify to display "#MESS", instead of the native text, by setting the [\[Language\]DisplayError](#) parameter to 1 (one); the default is 0 (zero).

---

**Note:**

1. The [\[Language\]DisplayError](#) parameter does not apply to alarm strings.
  2. For alarm data where there are multiple clients and one user name, a scenario may occur where the runtime display shows a variety of languages. For more information refer to [Alarm Data Localization](#).
- 

## See Also

[Logging Data in Different Languages](#)

### Alarm Data Localization

Alarm data localization is saved in user accounts on alarm servers. For each user account, the last login action across the network determines the current language setting of the user. If the same user account is used simultaneously with different languages on each of the display clients the runtime display may show a mixture of languages.

For example: Client A , Client B, Client C all log in (different clients) with the user name "Operator":

1. Client A logs into "Operator" with French. Once login has been successful, the display contents in Client A are in French, including both UI and alarm data.
2. Client B then logs into "Operator" with English. Once login has been successful, the display contents in Client B are in English, including both UI and alarm data. However, the alarm data in Client A is in English instead of French. Thus Client A would display both French and English even though Client A has logged in with French.

3. Client C then logs into "Operator" with German. Once login has been successful, the display contents in Client C are in German, including both UI and alarm data. However, the alarm data in Client A and Client B are in German instead of French or English. Thus Client A would display both French and German even though Client A has logged in with French, and Client B would display both English and German.

In this scenario allow Client A, Client B and Client C to use separate accounts, such as "OperatorA", "OperatorB", "OperatorC". This will then allow each user to view the runtime display in their preferred language.

## See Also

[Set the Local Language Used at Runtime](#)

## ASCII and ANSI Character Sets

Each screen character is defined by a code (number). Operating systems and applications need to know these codes to attach meaning to individual characters. A character set provides a code for every character. For your operating system/application to interpret a character correctly, you need to use the correct character set.

---

**Note:** Character sets are distinct from fonts. A font defines the visual/appearance properties of a character, not its meaning.

ASCII (American Standard Code for Information Interchange) is a widely adopted 7-bit code specifying the basic alphanumeric character set of the English language. For example, the character capital "A" has the ASCII value of 65, the character lowercase "a" a value of 97.

The ASCII character set contains 96 characters and is commonly used as a standard for protocols and files.

Much of Plant SCADA uses ANSI (American National Standards Institute) character sets. ANSI character sets are language-based, with each different language version of Windows (French, Korean and so on) requiring a specific ANSI character set. Codes 32 to 127 contain standard ASCII characters.

Windows uses Unicode, but still supports ANSI character sets. Several of Plant SCADA's utilities have been created for Unicode, for instance Process Analyst. Unicode accommodates known character sets by having one 16-bit (worldwide) character encoding standard.

## See Also

[OEM Character Sets](#)

## OEM Character Sets

OEM character sets are those which are used by MS-DOS or Console applications (they are operating system dependent). Most OEM character sets do not match the ANSI character sets. For example, line drawing characters commonly used in MS-DOS character sets were replaced with language characters in ANSI character sets.

As explained below, when building multiple language projects give adequate consideration to the role that ANSI and OEM character sets play, in the way the language strings are stored and interpreted.

Language configuration information is stored in dBASE files (a database standard defined primarily for MS-DOS applications) where string information is customarily stored as OEM characters. When using a Windows application (such as Excel) to edit dBASE files, the characters on screen are in the ANSI character set. When you

save this information to the dBASE file, Excel converts it to an OEM equivalent. For this conversion to work correctly the OEM character set needs to be compatible with the ANSI character set used in Excel. For example, if you have prepared strings for a project in Russian (using Excel), the OEM character set needs to support the Russian (Cyrillic) character set. The OEM character set used by Windows is primarily determined by your system setup and cannot easily be changed. This presents a challenge for multi-language projects.

For example, consider a project intended to support Russian, French, and English. Excel is used to prepare the language dBASE files. When saving information from Excel, it is translated from the respective ANSI character sets to the OEM equivalent. To display this information, Plant SCADA will need to convert it from OEM back to ANSI. However, Russian requires a Cyrillic OEM character set and French and English requires a Latin OEM character set. Because Windows can only use one OEM character set at a time (which cannot be changed dynamically), only one of the three project languages can be correctly supported during any given session.

The only way to support multiple languages with differing character sets within one Plant SCADA project, is to verify that the language information you store in dBASE files is stored in the ANSI (not the OEM) format. Further, the [\[CtEdit\]ANSIToOEM](#) parameter needs to be set to 0 (zero) to prevent a conversion from occurring automatically. The challenge for the developer in preparing the project is in saving this information in the first place, because most applications store the language information in OEM format.

---

**Note:** A multi-language project is included in the samples directory on your installation CD. This project allows you to enter information into the language dBASE files in ANSI format.

---

## Configure Languages for Industrial Graphics Applications

AVEVA™ Industrial Graphics applications can be configured to support multiple languages. You can apply a different set of strings to a graphics object for each language that is configured in Plant SCADA Studio. The content that gets displayed is then determined by the language that is selected in the Industrial Graphics Web Client.

---

**Note:** If you are looking for information on language switching for a standard Plant SCADA project, see [Configure Languages for a Standard Project](#).

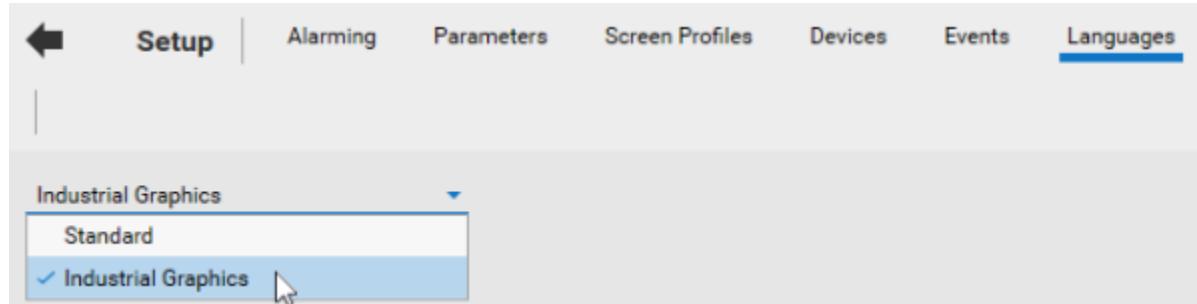
---

Perform the following steps to enable language switching for an Industrial Graphics application.

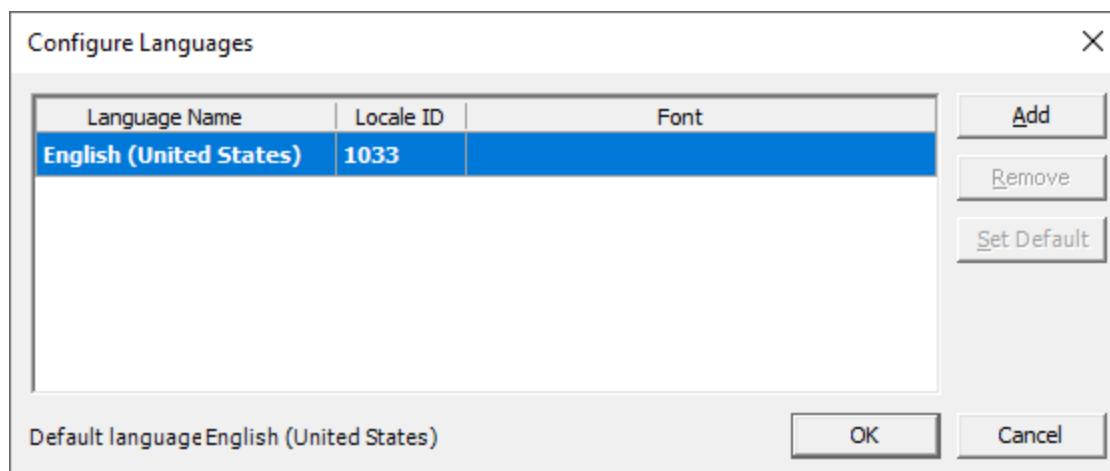
### [Set up the languages you want to support](#)

To configure the list of languages supported by an Industrial Graphics application, you need to use Plant SCADA Studio's **Setup** activity.

1. In the **Languages** view, select **Industrial Graphics** from the menu below the command bar.



2. On the Languages Configuration dialog, select **Edit**. The Configure Languages dialog will appear.



By default, the current Windows® locale setting is used to set the default language and font.

3. To add an additional language, click the **Add** button.
4. Use the Add Language dialog to select a language **By Name** or **By Locale ID**.
5. Specify a **Font** for the language.
6. Click **OK**.

The language you selected will be now appear in the Configure Languages dialog.

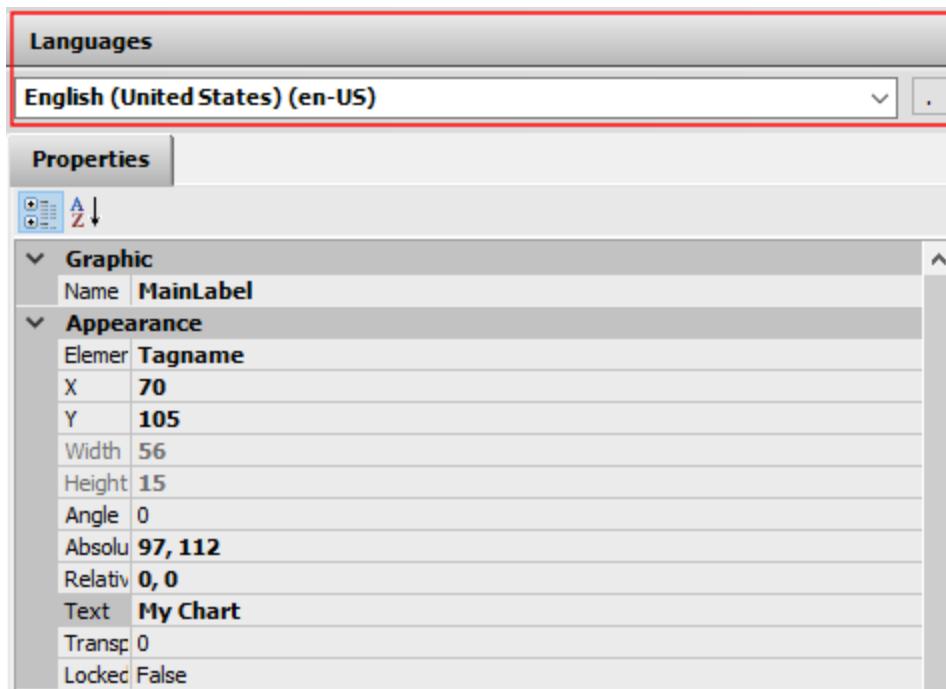
7. If required, use the **Set Default** button to change the language that will be used by default in the Industrial Graphics Editor.
8. Click OK to update the list of supported languages.

If an instance of the Industrial Graphics Editor is currently open, any changes you make to the language settings will not be applied until after the editor has been restarted.

**Note:** If you delete a language, the translations for a graphic will still be available until you specifically remove them in the Industrial Graphics Editor. See the topic *Removing a Language from a Graphic* in the *Creating Industrial Graphics* section of the help.

#### Select the required language in the Industrial Graphics Editor

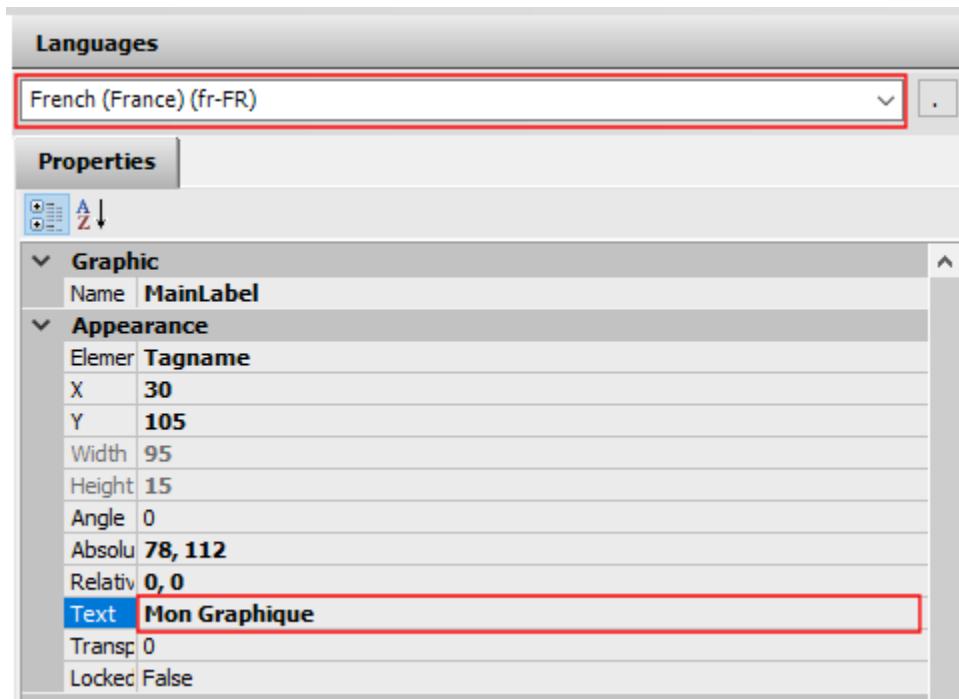
You can specify the language that is currently applied to the Industrial Graphics Editor using the **Languages** panel.



Select the required language from the drop-down menu to the right of the field. Any changes you make to the properties for a selected object will now be associated with this language.

#### Substitute the required strings

You can then select an object and make any required changes to the strings you would like to update for the selected language.



You will need to repeat these steps for each language you would like to support.

## Select a language in the Industrial Graphics Web Client

By default, your project will load using the language set as default in the browser. If the default language of the browser is not supported by your project and/or the Web Client, the default language configured in your project will be used.

**To change the current language:**

1. Go to the title bar and click .
2. Click the **Language** icon to select a language.



When you change the language, the application in the browser will be restarted.

**Note:** If you use an included project that has its own translations for graphic objects, those translation will be retained. However, their use will be limited by the languages that are configured in the active project. For more information, see [Working with Industrial Graphics Languages in Included Projects](#).

## See Also

[Use the Industrial Graphics Web Client](#)

## Working with Industrial Graphics Languages in Included Projects

If you use an included project that has its own translations for graphics objects, those translations will be retained. However, only languages that are configured in the active project will be available for use.

If you open an include project graphic that does not contain the default language of the active project, then the active project's default language is added to the list of languages for that graphic and is set as the new default language. The text strings from the last saved default language for a graphic are transferred over to the new default language for the graphic.

For example, if a graphic is part of an included project that only has the French language configured and it is used by an active project where English is the default language, the French strings are transferred over to the English language when the graphic is opened, and English is made the default language for the graphic. The French language still exists for the graphic, but it will no longer be the default language.

If a graphic is part of an included project that has both English and French configured (with French as the default language), and the graphic is opened via an active project with English as the default language, the default language for the graphic will be set to English.

## See Also

[Configure Languages for Industrial Graphics Applications](#)

## Keyboard Keys

Keyboard keys allow you to define a meaningful name for any key on a keyboard. This creates a label for the key that you can then use in keyboard commands.

For example, you could define the **F2** key as the "Login" key. When the Login key is subsequently referenced in the system, it will automatically refer to the F2 key.

Keyboard Commands are configured in the **Visualization** Plant SCADA Studio (see [Keyboard Commands](#)).

## See Also

[Define Keyboard Keys](#)

## Define Keyboard Keys

[Keyboard Keys](#) allow you to define a meaningful name for any key on a keyboard, making it easier to issue commands.

**To define a keyboard key:**

1. In the **Setup** activity, select **Keyboard Keys**.
2. Add a row to the Grid Editor.
3. Type the required information in each column, or in the fields in the Property Grid.  
For a description of the properties, see below.
4. Click **Save**.

## Keyboard Keys Properties

### General Properties

Property	Description
<b>Key Name</b>	The name assigned to the key (16 characters or less). You can define a meaningful name for any key on your keyboard, and use these keys in any keyboard command. For example, you can define the F2 key as the Login key. When the Login key is subsequently referenced in the system, it refers to the F2 key.  You can assign a key name to any key on the keyboard (including the alphanumeric keys A-Z, a-z, and 0-9). However, after a key is defined as a command key it can only be used as a command key. If you do assign a definition to an alphanumeric key (for example the character A), then that key can not be used as a data key. Each time it is pressed, the definition for the key is recognized and not the keyboard character itself. Keyboard key definitions are usually only used with non-alphanumeric characters (for example the function keys).
<b>Key Code</b>	The code assigned to the key name (16 characters or less). The Key Code is what links your Key Name to the

Property	Description
	actual key. You can specify the key code either as a hexadecimal value or use the standard label already associated with the key.
<b>Echo</b>	<p>Determines if the key is echoed on the screen (when the key is used).</p> <p>If set to TRUE, the <b>Key Name</b> is displayed (echoed) when the key is pressed. The key name is displayed on the graphics page in the keyboard entry line (AN1).</p> <p>If set to FALSE, the <b>Key Name</b> does not display when the key is pressed.</p> <p>This property is optional. If not specified, Echo defaults to TRUE.</p>
<b>Keyboard Type</b>	<p>The type of keyboard. This field is only necessary if you have more than one type of keyboard on the same system.</p> <p>If you do have more than one type of keyboard, use Keyboard Type 1 for your first type of keyboard, use a separate number for each additional type (1, 2, 3, and so on), and define keys for each keyboard. You can use the default (Type 0) for common keys.</p>
<b>Comment</b>	Any useful comment (48 characters or less).

#### Project Properties

Property	Description
<b>Project</b>	The project in which the keyboard key is included.

#### See Also

[Keyboard Commands](#)

## Custom Files

If a Plant SCADA project relies on externally-created files that you need to incorporate into a runtime deployment (such as DBF files, HTML files, or ActiveX objects), you will need to identify these files as **Custom Files** in the **Setup** activity.

When a project is compiled, any files you have identified in this way will be compressed into one of the following zip files:

- **custActiveX.zip** - includes the associated ActiveX files
- **custFiles.zip** - includes other associated files.

These zip files are then incorporated into a project. Each time an operator starts Plant SCADA Runtime, the client process checks the project and its included projects directories. If custFiles.zip or custActiveX.zip is found locally, the compressed archive is unzipped. (If [CtEdit]Copy is specified, then the zip files may be copied from the copy path first.)

Files in CustFiles.zip are unpacked to the project directory and its sub directories, depending on the value specified in the **Destination Location** field.

ActiveX control files in CustActiveX.zip are unpacked to the project directory and its sub directories first, then moved to "%PROGRAMFILES(X86)%\Common Files\AVEVA Plant SCADA" and, if the file is a valid ActiveX control, it is registered.

---

**Note:** Custom files can be transferred to remote computers via a deployment server using Plant SCADA's Deployment feature.

---

## See Also

[Include Custom Files](#)

## Include Custom Files

[Custom Files](#) are external or user-created files that can be included in a project when it compiles. These files could include CSV or DBF files associated with tables presented on project pages, or HTML content. It should also include any required ActiveX objects.

### To include custom files in a runtime deployment:

1. Determine the name and location of any custom files that you want to include in the project.

---

**Note:** The following files cannot be included as custom files as they are used by the compiler when preparing the zip files:

- Changes.dbf
- Custfiles.dbf
- Errlog.dbf
- Custfiles.zip
- Custactivex.zip
- \_library.rdb
- \_project.rdb

- 
2. In the **Setup** activity, select **Custom Files**.

3. Add a row to the Grid Editor.

4. Type the required information in each column, or in the fields in the Property Grid.

For a description of the properties, see below.

5. Click **Save**.

If you are adding an ActiveX object, make sure the **Is ActiveX** field is set to TRUE.

---

**Note:** If an ActiveX object has an associated data source, verify that the data source can be located by the computer hosting the client. See the topic [Managing Associated Data Sources](#) under the section on ActiveX objects in the Plant SCADA Help.

---

## Custom Files Properties

### General Properties

Field	Description
<b>File Name</b>	<p>The name of a file you would like to incorporate into to a runtime deployment. For example, a CSV or DBF file associated with a table presented on a project page, an HTML file, or a required ActiveX object. You can use wild cards ('*' or '?') when specifying a file name.</p> <p>To identify a custom file, you need to precede the file name with:</p> <ul style="list-style-type: none"> <li>• an absolute path</li> <li>• the Windows relative path</li> <li>• a Plant SCADA directory</li> </ul> <p>A Windows relative path should navigate from the current project directory. A Plant SCADA directory path can be one of the following:</p> <ul style="list-style-type: none"> <li>• [BIN] – where the Plant SCADA binary files are located</li> <li>• [Config] – where the Plant SCADA configuration files are located</li> <li>• [DATA] – where the Plant SCADA data files are located</li> <li>• [USER] – where databases are located</li> <li>• [COMMONFILES] – where Plant SCADA common files are located.</li> </ul> <p><b>Note:</b> You will need to manually configure the access control list for a folder location so that it matches the permissions that were applied to the default location during installation. See Configure Directory Security for Modified Folder Locations.</p>
<b>Is ActiveX</b>	<p>This field specifies if a file is an ActiveX object or not, which determines if it should be included in custActivex.zip or custFiles.zip. If the value you select is <b>False</b>, the file will be added to custFiles.zip; if you select <b>True</b> it will be added to custActiveX.zip.</p> <p>If you leave this field blank, any file with the extension ".ocx", will be automatically added to custActiveX.zip by the compiler. This is because ActiveX objects</p>

Field	Description
	typically use this file extension. Similarly, any files that don't have this extension will be automatically added to custFiles.zip  If you select <b>False</b> in this field where an .ocx file has been identified, it will force the .ocx file to be included in custFiles.zip.
<b>Destination Location</b>	This field specifies the folder structure that should be used to host the file when it is extracted from the one of the delivered zip files. It should represent a directory path in relation to the location where extraction occurs. For example: \Files\  The base folder for extraction is %PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\User\Example
<b>Comment</b>	Any useful comment (254 characters).

### Project Properties

Property	Description
<b>Project</b>	The project in which the custom file is configured.

## Compile

The Plant SCADA compiler brings together elements of your project - the configuration databases, graphics and Cicode files - to create a runtime system.

Compilation checks the project for errors and optimizes your system for fast and efficient operation. The time necessary to compile a project depends on its size and on the speed of your computer. Typically, compiling only takes several minutes.

When the Plant SCADA compiler runs, it normally opens files in exclusive mode. In this mode only Plant SCADA has access to the files (while the compiler is running). This improves the performance of the compiler, but can also result in an incomplete compilation if two people try to compile different projects at the same time, and these projects have one or more Include projects in common.

The [\[General\]ShareFiles](#) parameter tells the compiler to open files in shared mode. This option allows shared network users to run the compiler at the same time, but it can increase the time necessary for the compilation.



### UNINTENDED EQUIPMENT OPERATION

Restart the client process if the hardware alarm "Cicode library timestamp differs" is raised after a page is opened.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

---

**Note:** A hardware alarm of "Cicode library timestamp differs" will be raised if the Cicode library used by a page has a different timestamp from the one in memory. The timestamps will be different if the project has been fully recompiled, the project has been incrementally recompiled after the page has been modified, or if the project has been incrementally recompiled after any Cicode has been modified.

---

## See Also

- [Compile a Project](#)
- [Compile Messages](#)
- [Incremental Compilation](#)
- [Debug a Compilation](#)

## Compile a Project

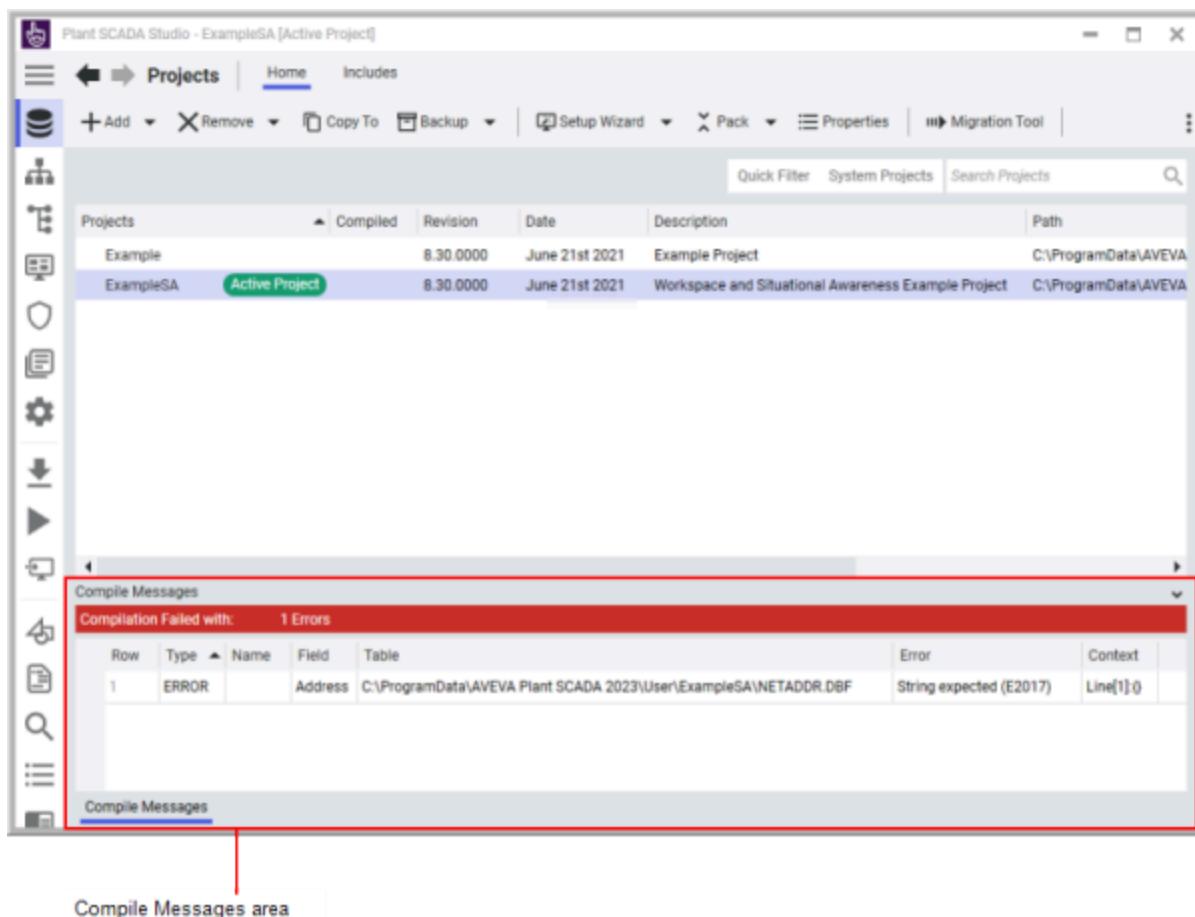
### To compile a project:

1. Open Plant SCADA Plant SCADA Studio.
2. On the Activity Bar, select **Compile**.



The **Plant SCADA Compiler** dialog box will appear and track the progress of the compile as it runs. If required, click the **Cancel** button to stop the compile.

When the compile is complete, the [Compile Messages Area](#) will open.



If any alert conditions are detected during compilation they will be listed in the **Compile Messages** area sorted by **Type**.

A Plant SCADA project can compile successfully if warning messages are generated. However, error messages and fatal messages will prevent a project from compiling successfully. You will need to fix these in order to recompile (see [Debug a Compilation](#)).

If you require further information about a particular compile message, see [Compile Messages](#).

3. To go to a database record that relates to a compile message, double click within the message row. The Grid Editor will display with the associated field selected.

---

**Note:** Plant SCADA automatically compiles the project (if uncompiled) when you try to run it.

---

## See Also

[Debug a Compilation](#)

[Incremental Compilation](#)

[Plant SCADA Studio Options](#)

## Debug a Compilation

The Plant SCADA Compiler dialog will notify you of any error messages as it compiles, and you can opt to cancel

the compilation at any stage. If there are multiple or severe errors, the compiler might automatically cancel. Once the compiler is finished, the compile messages will be listed in the [Compile Messages Area](#) sorted by **Type**. This can include the following types:

- Error messages
- Fatal messages
- Warning messages.

A Plant SCADA project can compile successfully if warning messages are generated. However, error messages and fatal messages will prevent a project from compiling successfully.

To go to a database record that relates to a compile message, double click within the message row. The Grid Editor will display with the associated field selected.

If you require further information about a particular compile message, see [Compile Messages](#).

The compiler does not verify the operation of your project. Just because your project compiles does not mean it will work correctly at runtime. For example, the compiler checks that the tags you use are defined correctly, and that your Cicode has acceptable syntax. But, it does not check your tags for incorrect scaling, or that your Cicode has no potential divide by zero errors.

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

- Do not attempt to run your system until you have resolved errors reported during project compilation.
- Always perform a complete compilation before promoting your project to the test or live environments.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

The compiler supports a number of options that modify the output of the compile error log, making it more practical to assess specific error types and entries. For more information, see [Compilation Options](#).

## **See Also**

[Incremental Compilation](#)

## **Compile Messages**

You might encounter error messages when you compile a Plant SCADA project.

Compile errors can be viewed in the [Compile Messages Area](#) in Plant SCADA Studio.

Each compile message includes the following fields:

Property	Description
Type	The type of error. Three types can occur during compilation. These are:

Property	Description
	<ul style="list-style-type: none"> <li>• <b>ERROR</b> - The compilation process continues, however the project will not compile successfully until you have corrected the error. For a complete list of error messages, see <a href="#">Compile Error Messages</a></li> <li>• <b>FATAL</b> - The severity of this error is such that it halts the compilation process. The project cannot be compiled until you correct the error. For a complete list of fatal messages, see <a href="#">Compile Fatal Messages</a>.</li> <li>• <b>WARNING</b> - The error was not serious enough to stop the project being compiled successfully, however investigate and correct the compile error. For a complete list of warning messages, see <a href="#">Compile Warning Messages</a>.</li> </ul>
<b>Name</b>	The name of the graphics page library, or report format file where the error has been detected.
<b>Field</b>	The database field where the error has been detected.
<b>Table</b>	The database table where the error has been detected.
<b>Error</b>	A brief description of the error.
<b>Context</b>	The location in the database field, report format file, or Cicode library where the error has been detected. The context of the error is enclosed in braces { . . . }.

## See Also

[Incremental Compilation](#)

## Compile Error Messages

The following table describes the [Compile Messages](#) you may encounter when compile a project. For information on how to address an ERROR message, use the code included in the error description (for example, "E2001") to locate the error message in the following table.

If the **Type** specified for a compile error message is WARNING, see the topic [Compile Warning Messages](#).

If the **Type** specified for a compile error message is FATAL, see the topic [Compile Fatal Messages](#).

Error Code	Error Message	Description
E2001	Tag not found	The specified tag does not exist. Check that the tag name and 'equipment.item' reference is correct, or add the tag to your project.  If the tag does exist in the variables database, the index to the database might be incorrect. This can occur if you have edited the variables database using Excel or some other database editor. To re-index the database, go to the Projects activity and select <b>Pack</b> on the Command Bar.
E2002	Database is empty	The database does not contain any records.
E2003	Cannot write to read only variable	This error indicates that the address of the variable tag, which refers to a protocol-specific area on the device, is read only. This means it is not possible to write to the variable tag.
E2004	Too many records in database	Too many records have been specified in the database. This error should only occur if the Citect.frm file has been changed or become corrupt. Contact Technical Support.
E2005	Too many fields in database	Too many fields have been specified in the database. This error will only occur if the Citect.frm file has been changed or become corrupt. Contact Technical Support.
E2006	Bad integer value	An integer value cannot be found where one is expected, or the integer value is out of range.
E2007	Bad floating point value	A floating-point number cannot be found where one is expected, or the floating-point value is out of range.
E2008	Tag expected	A tag name was not found where

Error Code	Error Message	Description
		one was expected, or an expression has been passed to a function that expects a tag. Check that the tag name is correct, or add the tag to your project.
E2009	Cannot read from write only variable	This error indicates that the address of the variable tag, which refers to a protocol-specific area on the device, is write only, so the variable tag does not have a value that can be read.
E2010	Primary IO device not found	The specified I/O device does not exist or cannot be located. Check that the device reference is correct, or add the I/O device to your project.
E2011	Close bracket expected	The Cicode statement has a different number of open and close brackets. Another close bracket ')' or ']' is expected in the statement.  For example: This error can be generated when a Genie is configured to be used by DspSym Cicode function but still has genie substitution syntax such as '%substitution%' on expressions that are unsupported by that configuration.
E2012	Symbol search failed	A database record does not exist. Check that the record name is correct.
E2013	Read remap is not supported for this variable	A mapped variable cannot be read when Remap Read is disabled. Check the remapping configuration.
E2014	Bad IO Device variable	The address of the variable is invalid for the protocol specified for the I/O device.
E2015	Bad raw data type	An invalid raw data type or a mismatch of data types is specified, for example, an attempt was made

Error Code	Error Message	Description
		<p>to convert an integer into a string. This may be due to a variable address implying a data type that is not compatible with the data type specified.</p> <p><b>Example 1</b>  Protocol: S7NT or PSDIRECT  Tag Data Type: REAL  Address: DB101,54.3[32]  In this case, the address implies that the variable is an array of 32 DIGITALs (via bit access) but the tag data type is REAL.</p> <p><b>Example 2</b>  Protocol: ABMLXEIP  Tag Data Type: INT  Address: T4:0/0  In this case, the address implies that the variable is a DIGITAL (via bit access) but the tag data type is INT.</p>
E2016	Array size exceeded	A tag is indexed but that tag is not declared as an array, or no index has been specified when a tag is declared as an array, or the wrong number of dimensions are specified for an array, or more than four dimensions are specified for an array.
E2017	String expected	Only strings can be used in database fields.
E2018	Protocol expected	The protocol field in the I/O Devices database is blank. You need to select a protocol for the I/O Device.
E2019	Open bracket expected	<p>You need to use parentheses () in Cicode functions, even when they have no parameters, for example MyFunction().</p> <p>This error can be generated when a</p>

Error Code	Error Message	Description
		variable tag is placed on a graphics page and the name of the tag is identical to the name of a Cicode function in the project or an included project.
E2020	Syntax error	A malformed Cicode expression has been specified. Check the structure of the expression.
E2021	Record already defined	Tag names, alarms, trends, users and Cicode functions all need to be unique within a project and any included projects. Check for duplicated names.
E2022	Incorrect number of arguments for function	Too many or not enough arguments have been specified for a Cicode function. Check the number of arguments being passed to the function and make the required adjustments.
E2023	Trailing characters in Cicode	There are extra trailing characters in a Cicode statement, following the semi-colon.
E2024	Incompatible types	<p>There is a mismatch of data types in a statement. For example, a string is specified where a number is expected.</p> <p>This error can be generated when a variable tag is placed on a graphics page and the name of the tag is identical to the name of a Cicode function in the project or an included project.</p> <p>This error can also be generated when a Genie is configured to be used by the DspSym Cicode function and has genie substitution syntax such as '%substitution%' on expressions that are unsupported by that configuration.</p>
E2025	Too many arguments	Too many arguments are specified in a Cicode function. The maximum

Error Code	Error Message	Description
		number of arguments allowed is 32.
E2026	Invalid font name	The font does not exist in the project. Check that the font name is correct, or specify the font in the <b>Standards</b> activity.
E2027	Invalid BOOLEAN value	A non-integer value was found where a TRUE or FALSE value is expected. For example, the controlling expression in an IF, WHILE, or FOR statement needs to be an integer.
E2028	Analog address not supported	An INT or other analog tag is specified where a DIGITAL tag is expected. Check that the tag name is correct, or that a DIGITAL data type is specified for the tag.
E2029	Group not found	The device listed in the report configuration does not exist. Check that the device or group name exists.
E2030	Group expected	
E2031	FUNCTION expected	A function needs to be declared with the FUNCTION keyword.
E2032	THEN expected	A THEN statement needs to be used in an IF statement.
E2033	DO expected	A DO statement needs to be used in a WHILE statement.
E2034	END expected	An END statement needs to be used at the end of a conditional statement or function definition.
E2035	Bad string conversion parameter	An invalid format parameter is specified in a string conversion. Check the format specification of the variable.
E2036	Cannot return value from void function	A RETURN statement cannot be used in a function that does not return a value. Remove the

Error Code	Error Message	Description
		RETURN statement or declare a return data type for the function.
E2037	Must return value from function	If a Cicode function is declared to return a value, it needs to have a RETURN statement.
E2038	Label is defined twice	Label names needs to be unique. Check the labels defined in the <b>Standards</b> activity for duplicated names.
E2039	Cannot use RETURN outside of functions	A RETURN statement can only be used within a function.
E2040	Statement expected	Plant SCADA is expecting a statement. Check the Cicode for syntax errors.
E2041	Operand expected	A Cicode operator needs to be followed by an operand.  This error can be generated when a Genie is configured to be used by the DspSym Cicode function and has genie substitution syntax such as '%substitution%' on expressions that are unsupported by that configuration.
E2042	Invalid time format	The time is incorrectly specified in the Time, Period or Sample Period field of a Reports, Events, Trend Tags, SPC Trend Tags, or Device.  Time formats needs to be in the format HH:MM:SS and needs to be in the range of 0:00:00 - 23:59:59. Only the hour is necessary, for example a value 16 means 16:00 (4:00 PM). Also 24:00:00 is accepted for historical purposes, and maps directly to 0:00:00.  Period formats needs to be either a valid date or a time in the format HH:MM:SS with the minutes and seconds in the range of 0 - 59. Only the seconds are necessary, for example a value of 22 means 22

Error Code	Error Message	Description
		seconds.  Sample Period formats need to either be a milliseconds value (for example 0.200 for 200 milliseconds) or a time in the format HH:MM:SS with the minutes and seconds in the range of 0 - 59. Only the seconds are necessary, for example a value of 22 means 22 seconds.
E2043	Bad analog format	The format is incorrectly specified for an analog variable. Check the Format field in the Variable Tags form.
E2044	Maximum report size exceeded	The report file size needs to be less than 63 kb. Reduce the size of the report or configure two reports.
E2045	Bad factor specification	A Cicode expression that contains an invalid expression has been used. Check the syntax of the expression.
E2046	Semicolon expected	Cicode statements need to be separated with semi-colons (;).
E2047	Page name cannot start with underscore	A Page Name needs to start with an alphanumeric character (A - Z, a - z, or 0 - 9).
E2048	Invalid group definition	A group does not exist in the project. Check that the group name is correct, or specify the group with the Groups form.
E2049	Cicode data limit reached	An array in a Cicode module cannot exceed 60 KB. Reduce the size of the array.
E2050	Expression too big	An expression is too large for the compiler. Reduce the length of the expression by splitting the expression into two or more smaller expressions.
E2051	Error reading file	An include file specified in a

Error Code	Error Message	Description
		database field cannot be found or cannot be opened. Check that the file name is correct, and that the file has been specified correctly (for example, <@FILENAME>).
E2052	Cannot use an array inside function	You cannot declare an array within a function. Arrays can only be declared as library variables, at the beginning of the library file.
E2053	Trailing characters in Name	The database record name contains invalid characters. Remove any invalid characters from the record name.
E2054	Close quotation mark expected	The Cicode statement has a different number of open and close quotation marks. Another close quotation mark ("") is expected in the statement.
E2055	String too big	The string size has been exceeded. The maximum size of the string not to exceed 255 characters.
E2056	Close comment delimiter expected	A comment opened with /* needs to be closed with the */ delimiter. Add the */ delimiter or use a single line comment that starts with an exclamation mark (!) and has no end delimiter.
E2057	Label argument error	The syntax of the argument is incorrect, or the incorrect number of arguments has been specified, or the number of characters in an argument is incorrect.
E2058	Invalid number	
E2059	Label too big	The label is too big. The size of a label cannot exceed 8kb.
E2060	Address on bad boundary	When reading a long or real from the memory of an I/O device, addresses need to be on odd or even boundaries. Addresses cannot

Error Code	Error Message	Description
		be mixed and all I/O devices need to be configured on the same boundary. You can disable checking with the [General]CheckAddressBoundary parameter.
E2061	Super Genie must be on a Page	Super Genie syntax (?) can only be used on pages. You cannot use a Super Genie in a report or Cicode function library. Use the TagRead() and TagWrite() functions instead.
E2062	Write remap is not supported for this variable	A mapped variable cannot be written to when Remap Write is disabled. Check the remapping configuration.
E2063	Digital Array must start on 16 bit boundary	
E2064	Remap is defined more than once	Duplicate mappings are configured for a variable. Check the remapping configuration.
E2065	CASE expected	A SELECT statement in your Cicode function needs to be followed by a CASE statement.
E2066	Unexpected trend sample period	
E2067	Too many function variables defined	Too many variables have been defined for a Cicode function. Check the configuration of the function and make the required adjustments.
E2068	Too many Cicode files defined	The project includes more than the allowable number of Cicode files.
E2069	MODULE function is not allowed, use PRIVATE	You have declared a Module function within your Cicode. Module is not a valid function type. Instead, the type Private needs to be used.
E2070	GLOBAL function is not allowed, use PUBLIC	You have declared a Global function within your Cicode. Global is not a

Error Code	Error Message	Description
		valid function type. Instead, the type Public needs to be used.
E2071	PUBLIC variable is not allowed, use GLOBAL	You have declared a Public variable within your Cicode. Public is not a valid variable type. Instead, the type Global needs to be used.
E2072	PRIVATE variable is not allowed, use MODULE	You have declared a Private variable within your Cicode. Private is not a valid variable type. Instead, the type Module needs to be used.
E2073	VB compiler error	
E2074	CiVBA code not allowed here	The compiler has detected CiVBA code where it is not supported.
E2076	Unexpected FUNCTION declaration found, are you missing an END?	The number of END statements in your functions is incorrect.
E2077	Tag already defined, tags must be unique	Tag names need to be unique across a project, even if they are different types. Change the tag name property for one of the duplicated tags.
E2078	Cluster not found	A cluster cannot be found. Check that the cluster reference is correct, or create the specified cluster. Clusters can be defined by selecting <b>Computers</b> in the <b>Topology</b> activity.
E2079	No Clusters defined	Every project needs to be configured to use at least one cluster. Clusters can be defined by selecting <b>Computers</b> in the <b>Topology</b> activity.
E2080	Invalid mode specified	The mode specified for the alarm, report or trend server is invalid.
E2081	More than one server of this type exists in this cluster	Each cluster can only include one pair of redundant alarm servers, one pair of redundant report servers, and one pair of redundant trend servers. Check the servers that are configured for the cluster.

Error Code	Error Message	Description
E2082	A server with this name already exists	Server names need to be unique. Rename one of the servers with the duplicated name.
E2083	Cannot use format on Expression with no result	
E2084	Network Address not defined in Topology/Network Addresses	The compiler has detected a network address that has not been defined in your project. Edit the address if it incorrect, or add it to your project.
E2085	Network Address not defined in Topology/Network Addresses	The compiler has detected a network address that has not been defined in your project. Edit the address if it incorrect, or add it to your project.
E2086	Duplicate network address for same type of server in a cluster	Two servers of the same type within a shared cluster have been configured with the same network address. Change the address for one of the servers.
E2087	Duplicate network address for IO server	Two I/O servers have been configured with the same network address. Change the network address for one of the servers.
E2088	Invalid port number	The port is not a valid number.
E2089	Invalid port number (2080-2088,2073,3073,5482,20222 are reserved)	The specified numbers are reserved. Please change these port numbers to another value.
E2090	Cluster name expected	Every project needs to be configured to use at least one cluster. Clusters can be defined by selecting <b>Computers</b> on the <b>Topology</b> activity.
E2091	Trailing characters in Expression	There are extra trailing characters following an expression.
E2092	A server with this network address and port already exists	A report, trend, alarm or I/O server is configured in your project with the same network address and port

Error Code	Error Message	Description
		number. If you intend for these servers to be configured on the same machine, change the port number, otherwise update the network address.
E2093	Cluster not found for the IO Device	Every project needs to be configured to use at least one cluster. Clusters can be defined by selecting <b>Computers</b> on the <b>Topology</b> activity.
E2094	Deadband value must be expressed as a percentage (0.0 - 100.0)	The compiler cannot decipher the value specified for a deadband. Adjust the value so that it represents a percentage (0.0 - 100.0).
E2095	Tag already defined as a local variable tag	Tag and local variables cannot have the same name. Please rename one or the other.
E2096	Cluster must be specified when two servers of the same type exist at the same net addr.	More than one server of the same type has been defined on the same machine. Please define a cluster for each server.
E2097	Invalid combination of specified & unspecified network addresses for the defined servers	The network addresses for the servers in a project need to all be blank, or all configured.
E2098	No Server of proper type is defined for the specified cluster	No server has been defined for this type in this cluster. Please define the appropriate server by selecting <b>Computers</b> on the <b>Topology</b> activity.
E2099	Primary IO devices having same network number cannot belong to the same cluster	Two primary I/O devices within the same cluster have been configured to use the same network number. If the devices have the same number to support redundancy, change the priority and startup mode. Otherwise, change the network number or move the devices to a different cluster.
E2100	Primary IO device not found for the	This error occurs when the I/O

Error Code	Error Message	Description
	cluster	device listed in the tag configuration does not exist.
E2101	Function is obsolete and cannot be used anymore	The Cicode function that you are trying to use can no longer be used. Refer to the upgrade topic for details.
E2102	Standby IO device should have a primary on the same cluster	A cluster has a standby I/O device configured, but no primary I/O device. Check the configuration of your redundant devices to confirm that a primary I/O device is assigned to the cluster.
E2103	Primary IO device must have a priority of 1	Set the Priority field for the primary I/O device to 1.
E2104	IO device has duplicate priority	The Priority field allows you to set precedence for the primary and standby devices within a cluster. Check the priority setting for the I/O devices with the same number in the specified cluster and remove any duplications.
E2105	Compound statement too large; refactor code. Please see KB article Q5070	
E2106	A duplicate menu entry already exists	Check the configuration of your menus in the <b>Visualization</b> activity and remove any duplications.
E2107	Role not found	A user or variable refers to a role that does not exist. Check the <b>Roles</b> setting for the specified user account, or the <b>Write Roles</b> setting for the specified variable.
E2108	Comma expected	
E2109	Function name is longer than 250 characters	A Cicode function name must be shorter than 250 characters. Rename the Cicode function.
E2110	Malformed device format	The format of the device is in an incorrect format. This could mean

Error Code	Error Message	Description
		<p>one of two issues is occurring:</p> <ol style="list-style-type: none"> <li>1. There is incorrect number of open and close braces ('{' and '}') in the device format. For example:  {Date,10}^v{TimeLong,12}  ^v{MsgLog,50}^v{Arg1,7}^v{FullName,20}</li> <li>2. A comma is missing between either the field name, field width or justification. For example:  {Date,10}^v{TimeLong,12}^v{MsgLog,50}^v{Arg1,7}^v{FullName,20}"</li> </ol>
E2111	Minimum update rate value must be blank or same value as defined by the primary device	The <b>Min Update Rate</b> field is not configured correctly for a standby device. Adjust the current setting so that is blank, or has the same value as the primary I/O device.
E2112	Staleness period value must be blank or same value as defined by the primary device	The <b>Staleness Period</b> field is not configured correctly for a standby device. Adjust the current setting so that is blank, or has the same value as the primary I/O device.
E2113	The setting for Background Poll must be the same for all I/O devices on the same server	The <b>Background Poll</b> field is not configured correctly on an I/O device. Check that the setting is the same on all devices that share a common I/O server.
E2114	The setting for Background Rate must be the same for all I/O devices on the same server	The <b>Background Rate</b> field is not configured correctly on an I/O device. Check that the setting is the same on all devices that share a common I/O server.
E2115	Tag element and item names cannot be used in this context	
E2116	Tag element and item names cannot be used in the context of a push data event trend	

Error Code	Error Message	Description
E2117	No User has been defined for this project. At least one user must be defined	Use the <b>Security</b> activity to add at least one user to your project.
E2118	Standby server has been defined without primary server in the cluster	A cluster has a standby server configured, but no primary server. Check the configuration of your redundant servers to confirm that a primary server is assigned to the cluster.
E2120	Two IO devices with Persist set to true have the same File Name	This occurs when two I/O devices have the <b>Persist</b> property set to "True", and the <b>File Name</b> property set to the same value. Change the file name for one of the I/O devices.
E2121	Invalid path	
E2122	Same primary IO device name used with two different network numbers	Two primary I/O devices have the same name, but different network numbers. If the devices have the same name to support redundancy, confirm that they have the same network number and have the priority and startup mode set correctly, otherwise assign them to a different cluster.
E2123	Undefined equipment name	The compiler has detected a reference to undefined equipment. Check that the reference is correct. If required, add the equipment definition to your project in the <b>System Model</b> activity.
E2124	Undefined equipment state name	The compiler has detected a reference to an undefined equipment state. Check that the reference is correct. If required, add the equipment state to your project in the <b>System Model</b> activity.
E2125	Invalid default state	
E2126	Invalid DR Mode value	

Error Code	Error Message	Description
E2127	Equipment State contains cyclic reference	
E2128	State defined without default DR Mode value	
E2129	Invalid destination location	This occurs when the destination location specified for a custom file is invalid. Check that the destination setting is correct.
E2130	Duplicate network address name for OPC Server	The compiler has detected two OPC servers with the same name. Each OPC server must have a unique name and run on a separate computer. Check the configuration of your OPC servers to make sure they use different names and network addresses.
E2131	Duplicate network IP address for OPC server	The compiler has detected two OPC servers with the same network address. Each OPC server must run on a separate computer. Check the configuration of your OPC servers to make sure they use different network addresses.
E2132	Cluster name present in Equipment name	A name used for root level equipment may not be the same as any cluster names. Check your equipment definitions.
E2133	An equipment record with this name is already present	Equipment names need to be unique within the same cluster. Check your equipment definitions.
E2134	Wrong hierarchy for OPC Server	The browsing hierarchy setting for the OPC server is invalid.
E2135	Equipment hierarchy too deep	
E2136	Equipment object name too long	The length of the complete name for the equipment needs to be less than 254 characters. Check your equipment definitions.
E2137	Equipment Name and Item too	Indicates the combination of

Error Code	Error Message	Description
	long. Specify or change the Item field	"Equipment.Item" is too long. Change the name of the Equipment and/or Item.
E2138	Equipment Name and Item not unique. Specify or change the Item field	Indicates the combination of "Equipment.Item" is not unique within the cluster. Change the name of the Equipment and/or Item.  This message will also occur if on the same level within the equipment hierarchy, the equipment object and the item object name are identical.
E2139	Equipment root has same name as an Alarm Tag	A name used for root level equipment may not be the same as any alarm tag names. Check your equipment definitions.
E2140	The Storage Method is not defined	The <b>Storage Method</b> property has not been defined for the trend.
E2141	Equipment not specified for tag item. Please specify equipment or remove tag item	Indicates that the Item field on the Tag Form (Variable Tags, Trend Tags, and all Alarm Tags) was specified without an associated Equipment Name.
E2142	Item name is also a node within the equipment hierarchy	An Item name cannot have the same name as a node on the equipment hierarchy.
E2143	No report server defined for this cluster	Equipment requires a report server to be configured. Add a report server to the cluster.
E2144	Invalid user defined	The user name conflicts with a set of <a href="#">Reserved User Names</a> . Use a different name.
E2145	Invalid role defined	
E2146	Tag name exceed length limit	
E2147	Conflict of parameters in included projects	A parameter setting in an included project is conflicting with the same parameter in the main project.

Error Code	Error Message	Description
E2150	Invalid Genie format	
E2151	Password is invalid due to duplication of user record	
E2152	Invalid Setpoint	
E2153	Set the parameter [General]TagStartDigit=1 if your tags start with a digit	If your project includes tags, tag items, or equipment names that have a number as the first character, you need to set the parameter [General]TagStartDigit to "1" to enable a compile.
E2154	Duplicate ScheduleID for IODevice	
E2155	Profile name is invalid	Profile name is invalid as it may contain a space, or any of these characters: < > : \ " / \\   ? *. Rename the profile.
E2156	Cicode IODevices do not support memory mode, memory must be set to FALSE	A Cicode I/O device will not compile if memory mode is set to TRUE. Set memory mode to FALSE for the Cicode I/O device.
E2157	Equipment Expected	An Equipment Reference refers to Equipment or Referenced Equipment that does not exist. Update the Equipment Reference, or add the missing equipment to your project.
E2158	Duplicate Equipment Reference	An Equipment Reference is configured twice. Check the configuration of the duplicated entries.
E2159	Too many alarm responses are defined for an alarm	Confirm that only eight sets of Cause and Response properties are defined for each alarm tag.
E2160	Content Type not found	
E2161	Invalid Alarm Priority	The Value property for an Alarm Priority must be between 1 and 255. If a label is used, confirm that it validates to a value between 1 and 255.

Error Code	Error Message	Description
E2162	Duplicate Alarm Priority	An Alarm Priority is configured twice. Check the configuration of the duplicated entries (including any that may occur in an included project).
E2163	Alarm Priority or Alarm Mode color must be specified	Display colors need to be configured for an Alarm Priority or an Alarm Mode. Check that the 'Foreground Color' and/or 'Background Color' properties have not been left blank.
E2164	Port Name can be no longer than 31 characters	A Port Name is too long. Use a name that is no longer than 31 characters.
E2165	Duplicate Alarm Mode display name	The Display Name specified for an Alarm Mode is configured twice. Remove one of the duplicated entries.
E2166	Invalid Alarm Mode display name	The Display Name specified for an Alarm Mode is invalid. You can select a valid name for this field from the drop-down list in the Grid Editor.
E2167	Screen Profile Name is Invalid	The specified screen profile name is invalid. Names should be less than 65 characters long and must not contain any invalid characters (< > : " / \   ? *).
E2168	Duplicate Equipment Runtime Parameter	A runtime equipment parameter is configured twice for the same equipment on the same cluster. Locate and change the duplicate entries.
E2169	Invalid equipment item name	The specified equipment item name is invalid. Use a valid item name that is not one of tag extensions keyword. If the item name is automatically generated by an equipment template, use Equipment Editor to rename the item name on the template and run

Error Code	Error Message	Description
		the <b>Update Equipment</b> command.
E2170	Network address can be no longer than 15 characters	The length of the network address is too long. Enter a network address that is less than or 15 characters in length.
E2171	Network addresses cannot be a fully qualified domain name	Network addresses cannot be a fully qualified domain name. Enter a valid computer name or an IP address instead.
E2172	Duplicate trend or SPC file name	When configuring the <b>File Name</b> property for a trend tag or SPC data file, ensure that each file uses a unique file name.
E2176	Duplicate Industrial Graphics page or symbol name	The name used for an Industrial Graphics page or symbol is not unique within the current project and its included projects. Enter a valid name for one of the duplicate items.
E2177	Embedded symbol was found but outside its project's include hierarchy	This error message will appear if an embedded symbol has been deleted or its host include project has been removed. It indicates that a new symbol with the same name as the deleted symbol has been created in a higher-level project.
E2178	Embedded symbol not found but a page with the same name was found	This error message will appear if an embedded symbol has been deleted, or its host include project has been removed. It indicates that a new page with the same name as the deleted symbol has been created.
E2179	<This error message is replaced with arbitrary text>	The compiler has detected a problem with the syntax for an object property (such as an incomplete expression). The error message will identify the object, and provide a brief explanation of what was detected.

Error Code	Error Message	Description
E2180	Page not found but a symbol with the same name was found	This error message will appear if a page has been deleted, or its host include project has been removed. It indicates that a new symbol with the same name as the deleted page has been created.
E2181	Reserved keyword is being used	Change the keyword to something that is not a reserved word. See <a href="#">Reserved Words</a> .
E2182	Invalid Unique ID format	Unique IDs for variables need to use the following format: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX  This error indicates that a unique ID is not valid.  This may occur if you modify the variables database outside of Plant SCADA Studio.  For more information, see <a href="#">Unique IDs for Variable Tags and Equipment</a> .
E2183	Duplicate Unique ID for variable	The compiler has detected duplications among the unique IDs used for the variables in your project.  This error may occur if you modify the variables database outside of Plant SCADA Studio.  For more information, see <a href="#">Unique IDs for Variable Tags and Equipment</a> .

## Compile Fatal Messages

If the **Type** specified for a [Compile Messages](#) is FATAL, it means something was detected that stopped the project from compiling successfully. For information on how to address a message of this type, use the code included in the error description (for example, "F4001") to locate the fatal message in the following table.

If the **Type** specified for a compile error message is ERROR, see the topic [Compile Error Messages](#).

If the **Type** specified for a compile error message is WARNING, see the topic [Compile Warning Messages](#).

Code	Fatal Message	Description
F4001	Out of file handles. Increase value of the parameter [CtEdit]DbFiles	<p>Plant SCADA uses a file handle to open each file. When you try to open too many files or databases simultaneously, Plant SCADA can need more file handles than are available.</p> <p>You are likely to run out of file handles if you have many included projects. When Plant SCADA compiles your project, it will open several files in each include project at the same time, so each extra project you include will increase the usage of file handles. If you get this alert message when you have added another include project, you have run out of file handles. To verify this, remove one of the included projects to see if Plant SCADA can then compile your project.</p> <p>With Windows running on a network, the setup of the number of file handles is located in various places. To increase the number of file handles in DOS, the setup is in the CONFIG.SYS file. If you are using Novell Netware you need to also increase the file handles in the NET.CFG or SHELL.CFG file. You need to also increase the number used by Plant SCADA with the [CtEdit]DbFiles parameter. Adjust the following settings the associated files:</p> <p>CONFIG.SYS FILES=120 NET.CFG or SHELL.CFG file handles=120</p>
F4002	Cannot open file	The compiler cannot open a file that is referenced by the project you are trying to compile. The file does not exist, or it has become corrupt, or your system is out of file handles.

<b>Code</b>	<b>Fatal Message</b>	<b>Description</b>
F4003	Cannot read from file	The compiler cannot read a file that is referenced by the project you are trying to compile. The end of file was found, or it has become corrupt. You should confirm that the file is functioning as expected.
F4004	Cannot write to file	The compiler cannot write to a file that is referenced by the project you are trying to compile. The file has become corrupt or the disk is full. You should confirm that the file is functioning as expected and that the required disk space is available.
F4005	Unknown field	A field is being referenced that does not exist. The database has been modified or has become corrupt. Pack the databases. If the error persists, contact Technical Support of this product.
F4006	Unexpected end of file	The end of the database has been reached or the database has become corrupt. Pack the database. If the error persists, contact Technical Support for this product.
F4007	Unexpected beginning of file	The beginning of the database has been reached or the database has become corrupt. Pack the database. If the error persists, contact Technical Support for this product.
F4008	Out of memory	Plant SCADA has run out of memory. Increase the amount of memory in the computer or use smaller databases.
F4009	Software error	An internal Plant SCADA software error has been detected. Contact Technical Support for this product.
F4010	Not database format	The database has become corrupt or the file format is unknown. Pack the database.

<b>Code</b>	<b>Fatal Message</b>	<b>Description</b>
F4011	File is read only	A file the compiler is attempting to access is set to read only. Check that the file name is correct or change the attributes of the file.
F4012	Unknown dba error	An internal Plant SCADA software error was detected. Contact Technical Support of this product.
F4013	Unknown bin error	An output file could not be opened during compilation. Pack the database. If the error persists, contact Technical Support for this product.
F4014	Disk full	The compiler is unable to function correctly due a lack of available disk space. Remove unwanted files from the disk, or replace the existing disk with a larger disk.
F4015	File locked	A file that is referenced by the project you are trying to compile appears to be locked. Check if the file is in use by another network user.
F4016	The include project was not found	The include project specified in the Context field cannot be located in Plant SCADA Studio. You should add the project, or update the reference.
F4017	Project includes itself	The project specified in the Context field contains a circular reference. Check the structure of your include projects.
F4018	Index key has changed	The database has a corrupted index. Pack the database.
F4019	File not indexed	The database needs to be indexed, but the index file associated with the database cannot be found. Pack the database.
F4021	Database table full	The database is full. If the error persists, contact Technical Support

<b>Code</b>	<b>Fatal Message</b>	<b>Description</b>
		for this product.
F4022	Reached the end of table	The end of the database has been reached or the database has become corrupt. Pack the database. If the error persists, contact Technical Support for this product.
F4023	File does not exist	The compiler cannot locate a file that is referenced by the project you are trying to compile. You should confirm that the file is in the specified location, or remove the reference to the file.
F4024	Too many files open	The maximum number of DBF files that can be open simultaneously has been exceeded. Increase the limit by changing the [CtEdit]DbFiles parameter.
F4025	File already opened in SINGLE mode	The file has been opened by another user. Set the [General]ShareFiles parameter to 1 in the Citect.ini file to open files in shared mode.
F4026	Too many include projects	The project you are trying to compile has more than 240 included projects defined. Reduce the number of included projects.
F4027	Unknown protocol	The project you are trying to compile refers to a protocol that does not exist.
F4028	Cannot compile all functions	The function library did not compile correctly.
F4029	Too many Cicode functions	More than 4500 user functions have been defined. To increase the number of functions allowed (up to 10000), use the parameter [CtEdit]MaxCicodeFunctions This error is often due to Cicode functions being defined in a number of included projects.

Code	Fatal Message	Description
		Extending this parameter might affect system performance. Only set it when advised by Citect Customer Service.
F4030	Point limit reached	The maximum number of points that can be referenced has been reached. The maximum point limit is determined by your Plant SCADA license. Contact Technical Support for this product.
F4031	Bad point limit	The incorrect point limit is specified in the Citect.ini file. The point limit needs to correspond to your Plant SCADA license.
F4032	Specification file invalid	A system file has become corrupt, or been deleted. Re-install Plant SCADA on your system. If the error persists, contact Technical Support for this product.
F4033	File size error	A Cicode functions file, report format file, or an include file is too big. The maximum file size is 1 MB.
F4034	Cicode Function greater than 64K	The code generated by a Cicode file is too large. Place the functions within the file into separate Cicode files.
F4035	OID Overflow. Set [OID]Reset=1 to regenerate OIDs	An internal error has occurred within a DBF file. Set the parameter [OID]Reset=1 to regenerate the OIDs in the database.
F4036	OID out of sequence. Set [OID]Reset=1	An internal error has occurred within a DBF file. Set the parameter [OID]Reset=1 to regenerate the OIDs in the database.
F4037	Project ID not found - each project must have an ID	The project you are trying to compile (or one of its included projects) does not appear to have a project ID. Specify an ID for the project in the Project General Properties (see <a href="#">Edit a Project's</a>

Code	Fatal Message	Description
		<a href="#">Properties</a> ).
F4038	Invalid Project ID - must be between 1 and 500	The project you are trying to compile (or one of its included projects) does not appear to have a valid project ID. Specify an ID for the project (between 1 and 500) in the Project General Properties (see <a href="#">Edit a Project's Properties</a> ).
F4039	Duplicate Project ID	The project you are trying to compile has the same project ID as one of its included projects. Specify a new ID for one of the projects in the Project General Properties (see <a href="#">Edit a Project's Properties</a> ).
F4040	The specified protocol is no longer supported	The project you are trying to compile references a protocol that is no longer supported. You should determine where the point of reference occurs and implement a supported protocol.
F4041	Citect.ini parameter specified is no longer supported	The Citect.ini file includes a parameter that is no longer supported. You should determine how the parameter is used and implement an alternative solution.
F4042	Alarm and Memory PLC devices are obsolete. Run the Migration Tool from the Tools menu	Alarm devices no longer need to be defined to use Alarm Properties as tags. Please refer to the help for the <a href="#">Project Migration Tool</a> to make the required changes to your project.
F4043	Total code size too large; refactor code. Please see KB article Q5600	The total size of your code is too large. Refer to Tech Note TN7730 via the AVEVA Knowledge and Support Center.

## Compile Warning Messages

If the **Type** specified for a compile message is WARNING, it means something was detected that did not stop the project from compiling successfully, however, further investigation is recommended. For information on how to address a WARNING message, use the code included in the error description (for example, "W1001") to locate

the warning message in the following table.

If the **Type** specified for a compile error message is ERROR, see the topic [Compile Error Messages](#).

If the **Type** specified for a compile error message is FATAL, see the topic [Compile Fatal Messages](#).

**Note:** You can disable a warning using the parameter [\[CtEdit\]SuppressCompilerWarning](#).

Code	Warning Message	Description
W1001	No IO Devices defined	No I/O devices have been defined in this project or any of its included projects (see <a href="#">Add an I/O Device</a> ).
W1002	Include project not found	An included project (specified in the Included Projects database) does not exist. Check the name of the included project. For more information, see the <i>Included Projects</i> section in <a href="#">Project Types</a> .
W1003	Argument with default found before argument with no default	<p>This error will occur when a Cicode function is defined with a default argument before an argument without a default. For example:</p> <pre>INT FUNCTION LogFieldValue(string sFile, string sField = "      ", string sValue) ... END</pre> <p>In this case, you will need to review how this function is used and either add a default value to argument sValue or remove the default from the argument sField. For example:</p> <pre>INT FUNCTION LogFieldValue(string sFile, string sField = "      ", string sValue = "      ") ... END or  INT FUNCTION LogFieldValue(string sFile, string sField, string sValue) ...</pre>

Code	Warning Message	Description
		END
W1004	Incorrect number of arguments for function	Too many or too few arguments have been passed to a Cicode function.
W1005	Tag may be unused	A tag that is included in your project is not referenced and appears to serve no purpose. Consider removing the tag, or add the required reference to the tag.
W1006	Function has the same name as built-in function	A Cicode function name within the project has the same name as an in-built Cicode function. It is recommended that you rename your function. You can then use the find and replace dialog to update any references to the original function name.
W1007	Tag not defined	The compiler has detected a reference to a tag that does not exist. You should create the required tag, or update the reference.
W1008	Function deprecated, legacy function will be removed in a future release	A Cicode function that is used in your project has been deprecated, which means it will be made obsolete in a future release. You should consider replacing the Cicode function.
W1009	Function deprecated, can only be called on the Server	A Cicode function that is used in your project has been deprecated, which means it will be made obsolete in a future release. Currently, you can only call the function on a server. You should consider replacing the Cicode function.
W1010	Function deprecated, does not support online changes made to the system	A Cicode function that is used in your project has been deprecated, which means it will be made obsolete in a future release. Currently, the function does not

Code	Warning Message	Description
		support online changes. You should consider replacing the Cicode function.
W1011	Alarm Tag not found, but valid property detected. Tag may not be available at runtime	The project refers to an alarm property that is associated with an alarm tag that does not exist. Create the specified alarm tag, or confirm that it is available on the alarm server.
W1012	Standby IODevice did not have a priority specified	The specified standby I/O device requires a priority setting.
W1013	Could not launch post compile command	Plant SCADA was unable to execute a command, script or batch file that was specified to run when compilation is complete. Check the replacement strings used for the parameters [CtEdit]CompileSuccessfulCommand and [CtEdit]CompileUnsuccessfulCommand to determine what occurred.
W1014	The tag and a cluster have the same name. It may lead to unintended behavior	Plant SCADA may behave unexpectedly if you use a tag name that is the same as a cluster name. You should rename the specified tag.
W1015	IODEvices for this unit number mix different protocols	I/O devices with the same unit number must all use the same protocol.
W1016	IODEvices for this unit number mix disk and non-disk types	I/O devices with the same unit number must either be all disk devices, or all non-disk devices.
W1017	IODEvices for this unit number mix memory and non-memory modes	I/O devices with the same unit number must either be all memory or all non-memory.
W1018	File does not exist	The compiler has detected a reference to a file cannot be found. Check that the file is available, and that name used is correct.

Code	Warning Message	Description
W1019	Multiple translations for the same native string	Multiple translations for a native string have been detected in the language database(s). You should review the language databases associated with your project (including those in any included projects), and remove any duplications.
W1020	Duplicate language	A language has been defined multiple times. You should check the languages specified in your project (including those in any included projects) and remove the duplication.
W1021	Possible missing operand between tags	The compiler has detected some incorrect Cicode syntax. For example, an operand may be missing between a tag and a constant.
W1022	Possible address on bad boundary	When the compiler finds a double-register variable, it remembers which boundary it is on and checks that all other double register variables are on the same boundary. This message will appear if the address of a long or real variable is not aligned correctly. For more information, see [General]CheckAddressBoundary.
W1023	Void functions are not supposed to return values	<p>The Cicode function uses the keyword RETURN but the function header does not contain a return type.</p> <p>For example:</p> <pre>FUNCTION LogFieldValue(string sFile, string, sField, string sValue) ... Return = FileOpen(sFile, "a+") IF iReturn = -1 THEN RETURN -1; END ... END</pre>

Code	Warning Message	Description
		<p>Change this function to the following:</p> <pre>INT FUNCTION LogFieldValue(string sFile, string sField, string sValue) ... iReturn = FileOpen(sFile, "a+") IF iReturn = -1 THEN RETURN -1; END ... END</pre>
W1024	Duplicate settings for LocalLanguage	<p>The compiler has detected that the local language setting is duplicated across the main project and its included projects. This means the language databases associated with each project would have been merged during compilation. For more information see <a href="#">Define the Languages Supported by a Project</a>.</p>
W1025	The specified language is not defined in the language form	<p>Your project refers to a language that is not specified in the <b>Languages</b> view in the <b>Setup</b> activity. You need to define the language in your project (see <a href="#">Define the Languages Supported by a Project</a>).</p>
W1026	Unsupported file extension type	<p>A file that is referred to in your project includes an extension type that is not recognized by Plant SCADA. Firstly check that the file extension is correct in the point of reference. If it is, it means the file type is not supported; you should remove the reference.</p>
W1027	The specified language is not supported	<p>Your project refers to a language that is not supported by Plant SCADA by default. You should use a supported language instead, or add support for the language to your project (see <a href="#">Define an Unsupported Language</a>).</p>

Code	Warning Message	Description
W1028	Unsupported syntax for marking translation text, it will not be translated	The compiler has detected that incorrect syntax has been used to mark some text for translation. Review the syntax used and make any required changes. For more information on correct syntax for marking translation text, see <a href="#">Mark Text for Translation</a> and/or <a href="#">Mark Alarm Text for TranslationP</a> .
W1029	Not a valid language	A language that has been defined in your project is not recognized by the compiler. Check the Languages Properties settings (see <a href="#">Define the Languages Supported by a Project</a> ).
W1030	This parameter overrides another with different value in one of the subprojects	A system parameter setting in the project database has also been detected in an included project. It is recommended you remove any duplicated system parameter settings from the included project(s).
W1032	'Allow RPC' permission is not defined (defaulting to FALSE)	<p>The <b>Allow RPC</b> property for a Role determines if a user or group of users can perform remote MsgRPC or ServerRPC calls. When upgrading projects, this field is left blank which will raise the compiler warning message.</p> <p>To allow existing users to continue using MsgRPC and ServerRPC, you need to manually change the value of <b>Allow RPC</b> to TRUE in Plant SCADA Studio's <b>Security   Roles</b> view.</p> <p>If these functions are used in your project, the roles that execute the functions will also need to have the permissions enabled.</p>
W1033	Length of the vertex path is zero	The From Offset and To Offset values specified for the polyline vertex are the same; therefore, the length of the vertex path is zero.

Code	Warning Message	Description
W1034	Transparent colors are not supported	A transparency setting is applied to a color that has been specified for an Alarm Priority display property. Transparency is not supported for Alarm Priority color settings.
W1035	'Allow Exec' permission is not defined (defaulting to FALSE)	"Allow Exec" on the Role form determines if a user or group of users can perform Exec calls in runtime. When upgrading projects if this field is left blank, the compiler warning message will be displayed.
W1036	'Manage User' permission is not defined (defaulting to FALSE)	"Manager User" on the Role form determines if a user or group of users can update user security settings at runtime. When upgrading projects if this field is left blank, the compiler warning message is displayed.
W1037	IO Server is not defined	The Server Name referenced by the I/O device is not defined in the IO servers table.
W1039	Tag usage is ambiguous; Tag exists for two or more clusters. Add cluster name to resolve.	An ambiguous tag is detected because multiple tags with the same name exist in different clusters, and the cluster name has not been specified as part of the tag name.
W1040	Numeric user names are not supported	When the user name contains only digits, for example, "12345", this message is displayed at runtime because a numeric user name is not valid.
W1041	TLS exchange, if enabled, cannot succeed using an IP address	The TLS exchange does not succeed if encryption is enabled for your system and the <b>Address</b> field in the <b>Network Addresses</b> table is set to an IP address.  To resolve this, configure the computer name of the server in the <b>DNS Name</b> field in the <b>Computers</b>

Code	Warning Message	Description
		<p>table.</p> <p><b>Note:</b> This step is not required if the <b>Address</b> field in the network address table is configured with a computer name.</p>
W1048	Reserved Keyword is being used	<p>Tag, equipment or cluster name contains <a href="#">Reserved Words</a>.</p>
W1049	Equipment parameters should be migrated to Configuration Parameters	<p>The compiler has detected legacy custom parameters in your project. It is recommended that you migrate these to configuration parameters (introduced in Plant SCADA 2023). For more information, see <a href="#">Migrate Custom Parameters to the Configuration Parameters Database</a>.</p> <p><b>Note:</b> This warning message will only be reported the first ten times an occurrence is detected.</p>
W1050	Unique ID expected. Please run Migration Tool.	<p>The compiler has detected equipment and/or variables that do not have a unique ID assigned to them. Run the <a href="#">Project Migration Tool</a> with the <b>Create unique IDs for Variables and Equipment</b> option selected.</p> <p>For more information, see <a href="#">Unique IDs for Variable Tags and Equipment</a>.</p> <p><b>Note:</b> This warning message will only be reported the first ten times an occurrence is detected.</p>
W1051	User 'Kernel' does not have Full Access to the Kernel window.	<p>The legacy 'Kernel' user that enabled Kernel access in previous versions of Plant SCADA no longer has default access to the Kernel window in version 2023 R2 (or later).</p> <p>It is recommended that you delete this user and access the Kernel through a Plant SCADA or Windows user account. See <a href="#">Display the</a></p>

Code	Warning Message	Description
		<a href="#">Kernel Window.</a>
W1052	Parameter name is empty	The <b>Name</b> field for an Equipment Configuration Parameter is empty. The equipment runtime database will not include this parameter.

## Incremental Compilation

You can compile the project incrementally. With incremental compilation, Plant SCADA only compiles the database records that were added (or changed) since the last compilation. The remainder of the project is not re-compiled.

**Note:** Some database records are dependent on other database records. If you change a dependent record, Plant SCADA compiles the entire database.

Before you run a system on a live plant, perform a complete compilation (switch off incremental compile) as the compilation will be more rigorous. Similarly, when you restore a project, you need to perform a complete compilation the first time (switch off incremental compile).

### To switch to incremental compile:

1. Open Plant SCADA Plant SCADA Studio.
2. On the Activity Bar, select **Options**.



The **Options** dialog box will appear.

3. Select the **Incremental compile** check box.
4. Click **OK**.

The compiler also supports a number of other options that modify the output of the compile error log. For more information, see [Compilation Options](#).

## See Also

[Debug a Compilation](#)

## Compilation Options

The compiler has a number of options available to help simplify the process of resolving compilation issues. Typically, these options modify the output of the compile error log, making it more practical to assess specific error types and entries.

The options listed here can be adjusted via the [Plant SCADA Studio Options](#), accessible via the [Activity Bar](#).

- **Log "tag not defined" warnings during compile**

If you select this option, the compiler will generate a "tag not defined" warning in the error log for any tags detected that are not defined in the variable database.

As Plant SCADA allows you to include undefined tags on your graphic pages, this warning may be redundant and impractical. By clearing this option, the warnings are still included in the displayed warning count, but they are not added to the error log.

- **Log deprecated warnings during compile**

If you select this option, the compiler will generate a warning to identify any deprecated elements it detects in a project, i.e. any functions, parameters, or Kernel commands that are no longer supported.

By clearing this option, the warnings are still included in the displayed warning count, but they are not added to the error log.

- **Warn about unused tags during full compile**

This option enables the generation of warning entries for unused tags that are not used directly in a Plant SCADA project. The warning entries are included in the compile errors form when a full compile is run. By default this option is not selected.

## See Also

[Incremental Compilation](#)

## Post Compile Commands

After a project has compiled successfully you can execute an optional command, script or batch file. This offers useful functionality if you have tasks that could be automated after a successful compile. This provides an expansion point for you to add your own script or command to perform additional tasks. An example of this could be an application or script to merge the language files that your project is using into a single file in the root project to aid in localization.

You can also launch an optional command, script or batch file to execute after an unsuccessful compile. Generally this would be used to create a log file to show warnings or errors generated during a compile.

To use either of these functions, add the command line containing the command or script to either the [\[CtEdit\]CompileSuccessfulCommand](#) or [\[CtEdit\]CompileUnsuccessfulCommand](#) parameter in the Citect.ini file.

## See Also

[Compile a Project](#)

[Debug a Compilation](#)

## Distribute the Project

After compiling a project, you can distribute the runtime files to your live system in the following ways:

- Using the Deployment functionality (recommended) - see [Deployment](#).
- Using the archive feature - see [Back Up a Project](#) and [Restore a Project](#) projects.
- Using a manual file transfer - pushing the runtime files directly onto your live system at the operating system

level.

- Using automatic file transfer - pulling runtime files from the configuration machine to the live server using [CtEdit]Run and [CtEdit]Copy.
- Using a shared folder on a network drive.

---

**Note:** Shared folders are not recommended for servers or control computers.

---

### Automatic file transfer

This is where runtime files are pulled down to the runtime system using the [CtEdit]Run and [CtEdit]Copy parameters.

- [CtEdit]Run points to the location of the runtime files. To avoid running a system over the network and potentially slowing down the system, it is recommended that this is a local folder under the [user] directory.
- [CtEdit]Copy points to a copy of the runtime files. This can be a remote shared location (for example the [user] folder shared on another computer) or another local folder. When Plant SCADA detects that the files in the RUN directory have a different timestamp to those in the COPY directory, the files are copied across to the RUN directory.

By updating the runtime files in the COPY directory, they will be automatically copied to the RUN directory whenever they are changed.

---

**Note:** Do not use the [CtEdit]Run and [CtEdit]Copy parameters if you intend to use Plant SCADA's Deployment functionality.

---

### Shared Folders

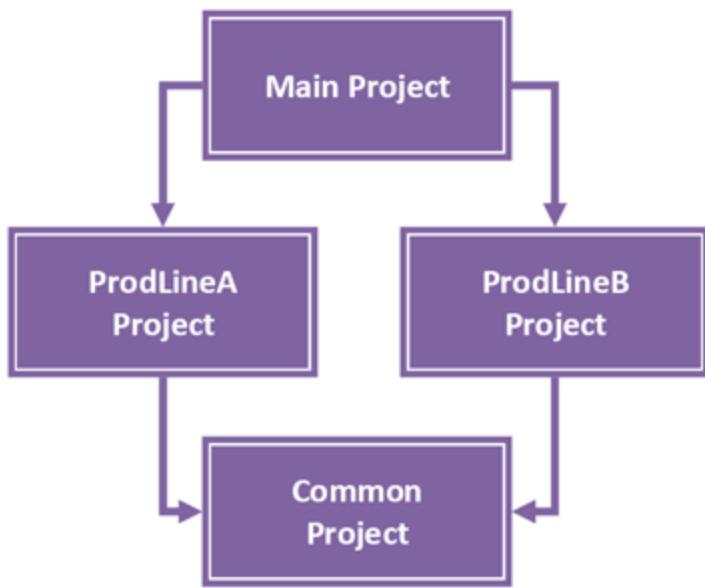
This is where the runtime files are stored in a network shared location, with a backup server specified in case the main files are not available.

- [CtEdit]Run points to the main remote shared location of the runtime files.
- [CtEdit]Backup points to the alternate remote shared location.

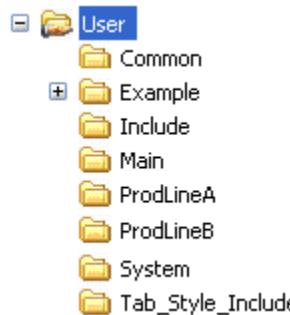
## Example — Using Run/Copy to distribute project runtime files

### File Server (PC1)

Consider the example project structure below. The overall system comprises four parts: a main project which features two separate production line projects, each of which draws on some common elements from a common project.



Run/Copy can be used to update project files from the File Server (PC1) to the File Client (PC2). Run/Copy does not support nested project folders, therefore project folders need to be on the same level as shown below.



The [User] folder (or a parent of the [User] folder) needs to be shared on the network. The shared folder needs to be accessible for machines to download the project runtime files. In this example, [User] is shared.

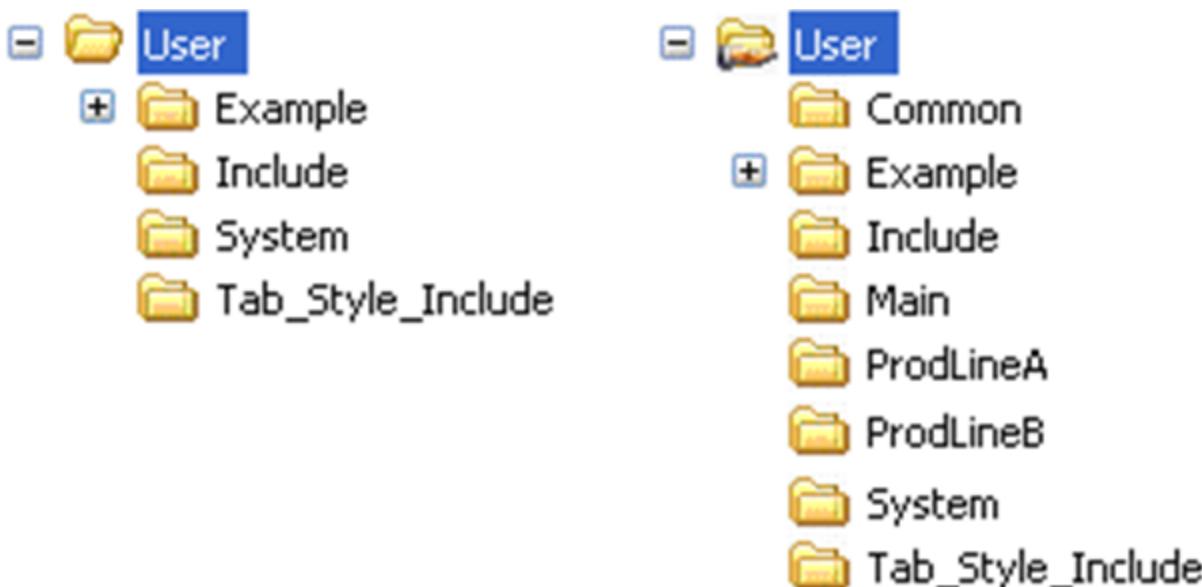
### File Client (PC2)

File Client (PC2) will use Run/Copy to get the latest updated projects from the File Server.

The settings for Run/Copy on the client are:

```
[ctEdit]Copy = \\PC1\User\Main
[ctEdit]Run = %PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\User\Main
```

The [Run] folder needs to be a child of the [User] folder on the target machine. The following pictures shows the project folders before and after the first project start-up with Run/Copy.



The project folders are automatically created by the Run/Copy process if they don't exist. In this example, the basic runtime files are copied to the [Run] folder. Any other files will be copied over on demand, such as page \*.ctg and \*.rdb files. After a successful project start-up, it is also possible for the project to start up when the [Copy] path is not available (if, for example, the File Server is uncontactable). In this case, only pages that have been previously viewed (that have been copied over previously on demand) are available.

If the target machine has the project development environment installed, do not launch Plant SCADA Studio after Run/Copy are set up. Doing so will override the [Run] path setting and you may end up running the wrong project.

## See Also

[File Server Redundancy](#)

## Deployment

Deployment allows you to distribute projects (and any subsequent updates) to computers in your Plant SCADA system.

This is enabled by a deployment server, a computer that stores multiple "versions" of a project's runtime files. From here, you can deploy a specific version to any computer that has been configured as a deployment client.

Deployment offers the following benefits:

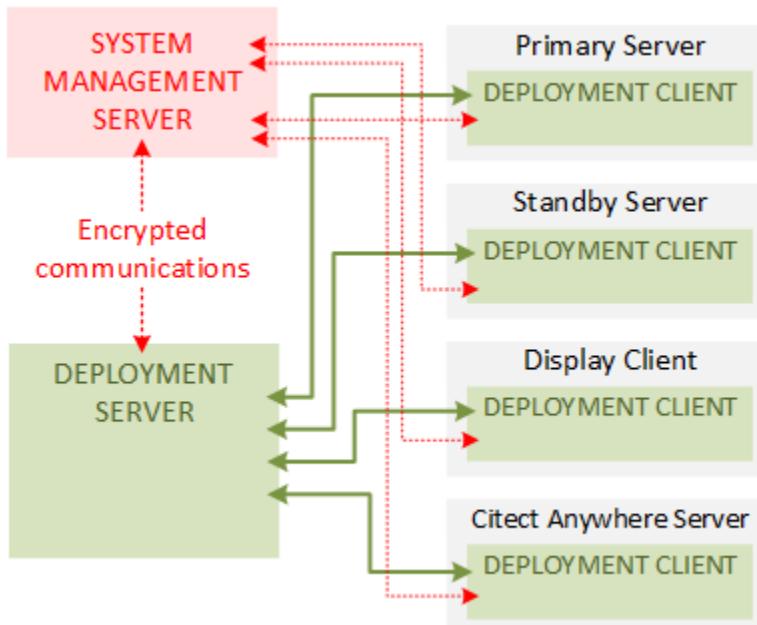
- Store multiple versions of a project's runtime files in a central network location.
- Deploy a specific version of a project to a runtime computer, or a predefined group of computers.
- Roll back a computer to a previous version of a project.
- Manage the restart options on the destination computer when a new version of a project is received.
- Tune the level of network utilization by deployment activities.
- Send only delta changes for version updates.

There are two key processes required to use deployment for your Plant SCADA system:

## 1. Set up the deployment server and deployment clients

To enable deployment, you firstly need to set up a deployment server. Deployment uses encrypted communications, which means you will require access to a System Management Server to do this (see [Set Up a System Management Server](#)).

You then need to establish a connection with each deployment client to which project versions will be delivered. You will also need to connect each deployment client to the System Management Server.



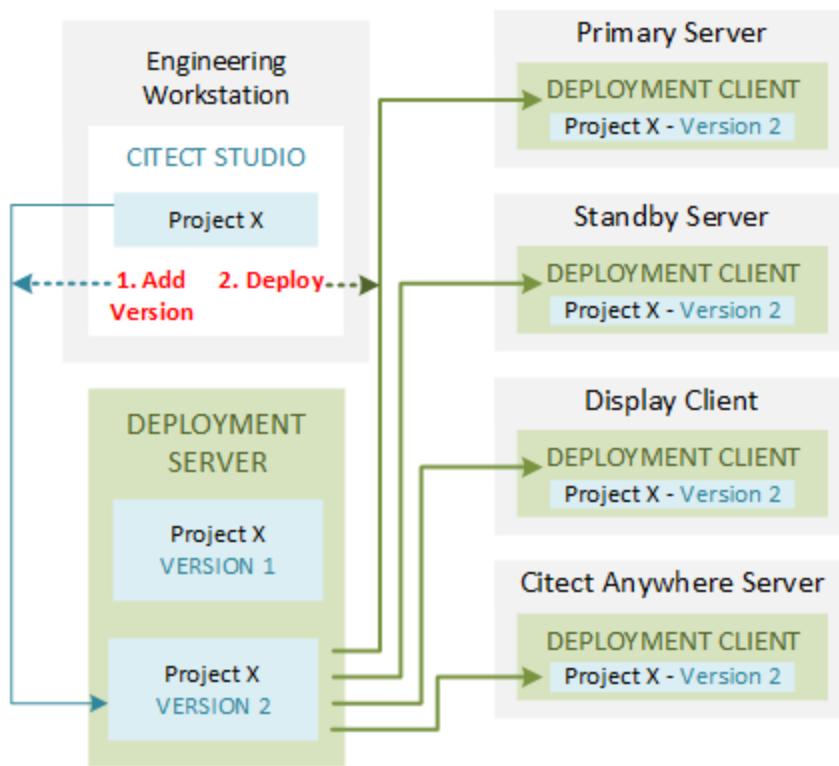
Each computer requires the installation of the relevant deployment components available from the **Customize the default components** page of the Plant SCADA installer. You can then configure these components with a tool called the Configurator. See [Prepare Your System for Deployment](#).

## 2. Manage Computers and Versions in Plant SCADA Studio

You can use the **Deployment** activity in Plant SCADA Studio to add project versions to the deployment server and manage their distribution.

The Deployment activity has two primary views:

- Use the **Versions** view to add versions of projects to the deployment server. Version numbers are incremented automatically.
- Use the **Computers** view to deploy a particular version of a project to selected computers (or a group of computers).



You can specify the **Update Method** that will be used to restart the Runtime Manager on the destination computer. Restart can occur automatically, or the operator will be prompted to accept the update and restart Runtime Manager when appropriate.

A **Settings** view is also available in the Deployment activity. You can use this view to configure a deployment server, or to manage multiple deployment servers if required.

---

**Note:** A deployment server uses Windows user groups to control access. These groups define a number of roles that support specific functionality: admin, deploy, upload and read-only. If a particular user needs to perform deployment tasks in Plant SCADA Studio, you will need to confirm that they are added to the appropriate Windows user group on the deployment sever. For more information, see [Provide Deployment Access to Additional Users](#).

## See Also

[Prepare Your System for Deployment](#)

[Deployment in Plant SCADA Studio](#)

## Prepare Your System for Deployment

Deployment uses encrypted communications, which means you will require access to a System Management Server (see [Set Up a System Management Server](#)).

To prepare your system for deployment, you need to:

- Configure or connect to an existing System Management Server.
- Setup a deployment server.

- Setup each deployment client.

The deployment clients include any computers in your Plant SCADA system to which you will deploy project versions.

**Note:**

- Ideally the Deployment Server should be installed on the same computer as the System Management Server. See [Install a Deployment Server](#).

- If you install the Deployment Server on a different computer to the System Management Server, you cannot use the default local user groups assigned to the Deployment Security Roles as they will not work in a distributed environment. You need to create domain user groups that align with the existing local groups, and add these to the Deployment Security Roles. The domain groups can then be used instead of the following default local user groups.

Deployment Administrators role: 'SCADA.DeploymentAdmins' or 'Asb.Deployment.AdminRole'.

Deployment Uploaders role: 'SCADA.DeploymentUploaders' or 'Asb.Deployment.UploadRole'.

Deployment Users role: 'SCADA.DeploymentUsers' or 'Asb.Deployment.DeployRole'.

See [Modifying the Members of a Security Role](#).

- When installing a Deployment Server as part of a workgroup, it must be installed on the same computer as the System Management Server. If you need the Deployment Server on a separate computer to the System Management Server, you will need to move your IT infrastructure to a domain.

You need to install the required components on each computer. These are available via the Plant SCADA installer. You can then use the **Configurator** to perform the following steps:

- [Configure a Deployment Server](#)
- [Configure a Deployment Client](#)
- [Configure a Runtime Computer for Deployment](#).

**Note:** The deployment server uses Windows user groups to control access to some of its functionality. If a particular user needs to perform deployment tasks in Plant SCADA Studio, you will need to make sure they are added to the appropriate Windows group on the deployment sever. For more information, see [Provide Deployment Access to Additional Users](#).

When this setup is complete, each deployment client will be ready to receive projects (and ongoing updates) from the deployment server.

## See Also

[Connect Plant SCADA Studio to a Deployment Server](#)

## Configure a Deployment Server

Deployment uses encrypted communications, which means you will require access to a System Management Server (see [Set Up a System Management Server](#)).

To configure a deployment server, you also need to have the deployment server components installed. This is an option on the **Customize the default components** page of the Plant SCADA installer (see [Install a Deployment Server](#)).

You can then use Configurator to configure a deployment server and set the deployment database password. Launch Configurator at the completion of the installation procedure, or from the Windows™ Start menu (**Programs | AVEVA | Configurator**).

On the **Security Roles** page under Plant SCADA, make sure that your user account is assigned to the Deployment Administrator role, either directly or via an associated domain group (see [Security Roles](#)).

---

**Note:** The deployment architecture installed with Plant SCADA 2023 is not compatible with deployment servers running an earlier version. See the note at the end of the topic [Install a Deployment Server](#) for more information.

### To configure the deployment server (on a local computer):

1. In the panel on the left side of the **Configurator**, select Plant SCADA | **Deployment Server**. The [**START**] page appears.
2. Select one of the following options:
  - **Configure the Deployment Server** - this option is available when no deployment server has been configured.
  - **Update the configuration of the current Deployment Server** - this option is available once a deployment server has been configured. Select it to update the database password and transfer speed for a deployment server; however you cannot update the port number.
  - **Import configuration file from previous version of the Deployment Server** - If you have an earlier version of the deployment server configured, you can import the settings. Browse for the following Configurator file in the provided field:  
"SE.Asb.Deployment.Server.WindowsService.exe.config".

---

**Note:** The default location is "%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Config"

After importing the Configurator file, you can continue using the deployment database and file repository created previously in the current or earlier versions of Plant SCADA.

3. Click **Next**. The **AUTHORIZE** page appears.

Enter the **User Name** and **Password** for the Windows account that will be used to connect the deployment server to the System Management Server.

The user account needs to be assigned to the 'Administrators' or 'aaAdministrators' group on the System Management Server.

4. Click **Next**. The **SETTINGS** page appears.

This page allows you to set the password used by the deployment database. It also allows you to set the transfer rate to limit the network bandwidth used when deploying a project.

5. Enter the **Password** for the database. Confirm the password.

---

**Note:** To change an existing database password you need to reconfigure the deployment server.

6. In the **Transfer Speed (KB/s)** field, enter a value between 0 and 2147483647 (0 being unrestricted). The default is 10000 (KB/s). By limiting the transfer speed, you allow other processes to use the remaining network bandwidth. This value may affect the overall duration of a deployment operation. For example, if your project is 20MB with a limit set to 1000 KB/s, the project will take approximately 20 seconds to transfer.

---

**Note:** Settings may vary according to your network infrastructure.

7. Click the **Next**. The **FINISH** page appears.

If required, you can use the **Previous** button to make any changes to your settings before you complete the configuration process.

8. Click **Configure**. The Configuration Messages panel will indicate if the deployment server configuration is

successful.

---

**Note:** If you change the computer name for the deployment server, it will cause a loss of communications between the deployment server and all of the deployment clients.

If any deployment server activities are not functioning correctly, check that the **NT Service\** **CitectDeploymentServer** virtual service account has read and write permissions for the deployment server folder (%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Deployment\Server).

Right click the **Server** folder, go to **Properties | Security**, and check that the permissions are assigned correctly.

## See Also

[Configure a Deployment Client](#)

[Configure a Runtime Computer for Deployment](#)

[Connect a Computer to a System Management Server](#)

## Configure a Deployment Client

Deployment uses encrypted communications, which means you will firstly need to connect each deployment client to the System Management Server used by the deployment server (see [Connect a Computer to a System Management Server](#)).

To configure a deployment client you use Configurator. This allows you to enter the login credentials required to register the client.

---

**Note:** To configure a deployment client you need to have the deployment client components installed. This is an option on the **Customize the default components** page of the Plant SCADA installer.

Launch the Configurator at the completion of the installation procedure, or from the Windows™ **Start** menu (**Programs | AVEVA | Configurator**).

On the **Security Roles** page under Plant SCADA, make sure that your user account is assigned to the Deployment Administrator role, either directly or via an associated domain group (see [Security Roles](#)).

---

**Note:** The current user on a deployment client needs to be assigned to the Runtime Users role to run a deployment client. If this is not always possible, you can configure Runtime Manager to run as a service (see [Configure a Runtime Computer for Deployment](#)).

### To configure a deployment client (on a local computer):

1. In the panel on the left side of the **Configurator**, select Plant SCADA | **Deployment Server**. The **[START]** page appears.
2. Select one of the following options:
  - **Configure the Deployment Client** - This option is available when no deployment client has been configured.
  - **Update the configuration of the current Deployment Client** - This option is available if a deployment client has been configured. The Deployment Client configuration needs to be updated whenever there is a change in the System Management Server configuration.
  - **Import configuration file from previous version of the Deployment Client** - If you have an earlier version of the deployment client configured (from Plant SCADA 2016 or 2018), you can import the client settings. Browse for the following Configurator file in the provided field:

"SE.Asb.Deployment.Node.WindowsService.exe.config".

---

**Note:** The default location is "%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Config"

---

3. Click **Next**. The **CONNECT** page appears.

In the **Deployment Server** field, select the deployment server to which you would like to connect. If the name does not appear in the list, type the Windows computer name of the machine on which the System Management Server is installed.

To change the deployment server to which a deployment client is connected, repeat the steps needed to configure a deployment client.

If the deployment client is also configured as a deployment server, the deployment client should only connect to the local deployment server.

4. Click **Next**. The **AUTHORIZE** page appears.

Enter the **User Name** and **Password** for the Windows user account that will be used to register the client computer with the deployment server.

The user account you enter needs to be assigned to the Deployment Administrator role on the deployment server (either directly or via an associated domain group). For more information, see [Security Roles](#).

If the System Management Server is on a different computer to the Deployment Server, the user account also needs to be assigned to the 'Administrators' or 'aaAdministrators' group on the System Management Server.

5. Click **Next**. The **SETTINGS** page appears.

On the **SETTINGS** page, you can set the Unpack Rate, and specify the location for the projects to be deployed.

- In the **Unpack rate (KB/s)** field, enter a value between 0 and 2147483647 (0 being unrestricted). If you limit the unpack rate, your system will still be able to run other processes. This value may affect the overall duration of a deployment operation.
- In the **Deployed project location** field, specify the folder path where the deployed projects from the Deployment Server will be stored. The default location is: %PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Deployment\Client\Projects. You can use the browse button to change the folder path.

6. Click **Next**. The **FINISH** page appears.

This page informs you that the **Configurator** is ready to send a request to the deployment server for registration.

If required, you can use the **Previous** button to make any changes to your settings before you initiate the registration process.

7. Click **Configure**.

If registration is successful, the configured client information will be stored on the computer and a connection will be established. If registration is not successful, you will be notified via the Configuration Messages panel.

If there is any issues with deployment client activities, check that the **NT Service\CitectDeploymentClient** and **NT Service\Citect Runtime Manager** virtual service accounts have read and write permissions for the deployment client folder (%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Deployment\Client).

Right click the **Client** folder, go to **Properties | Security**, and check that the permissions are assigned correctly.

## See Also

[Configure a Runtime Computer for Deployment](#)

[Connect a Computer to a System Management Server](#)

## Configure a Runtime Computer for Deployment

To prepare a computer for deployment, you need to confirm the following runtime environment settings.

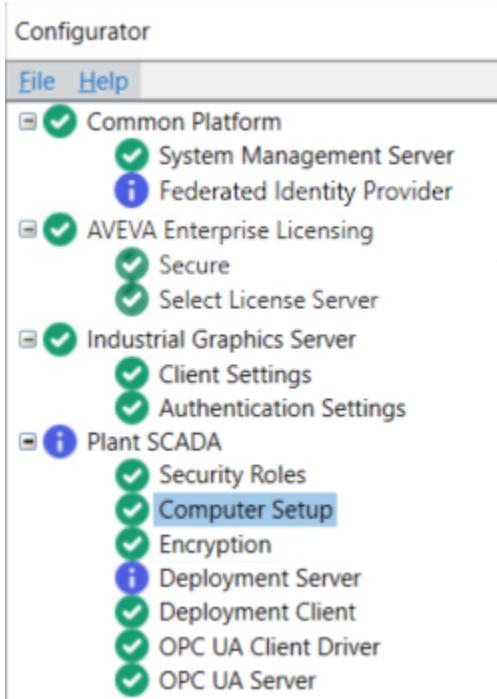
These settings can be adjusted using the **Configurator's Computer Setup** page under Plant SCADA. You can launch Configurator at the completion of the installation procedure, or go to the Windows® Start menu and locate **AVEVA | Configurator**.

## Server Authentication

If the computer will be used to host a Plant SCADA server process, you will need to specify a password for server authentication. Server processes with mismatching server passwords will not communicate with each other.

[To configure server authentication:](#)

1. In the panel on the left side of the **Configurator**, select **Plant SCADA | Computer Setup**.



2. The **Computer Setup** page appears. Navigate to the Server Authentication section. **Server Authentication** section.

### Server Authentication

*Running a server process requires the configuration of a server password. Setting this password allows servers to authenticate and create trust with each other.*

<input checked="" type="checkbox"/> Configure Server Password
Password
Confirm Password

3. Select the **Configure Server Password** check box.
4. Enter the required password and confirm it in the fields provided.

If the **Password** and **Confirm Password** fields already contain an entry, it means a server password has already been configured on the local computer. If required, you can enter a new password.

The password you use needs to match the password configured for the other server processes included in your Plant SCADA system. This password can be set on each computer using Configurator or the Setup Wizard.

5. To apply your settings, click the **Configure** button.

---

**Note:** If you change the server password while the runtime system is active, it will not disrupt any established connections as the password is only retrieved when authentication needs to occur. However, it is recommended that you apply the new password as soon as possible. To do this, you need to restart the runtime system.

---

## Project Run Path

This setting instructs Runtime Manager to run a deployed project from a specified directory.

### To set the Project Run Path:

1. In the panel on the left side of the **Configurator**, select **Plant SCADA | Computer Setup**.  
The **Computer Setup** page appears.
2. Navigate to the **Project Run Path** section.

#### Project Run Path

*Runtime can run the project selected in Plant SCADA Studio, or the project that has been deployed by the Deployment Server*

- Run the project selected in Plant SCADA Studio.
- Run the project deployed from the Deployment Server.

3. Select one of the following options to determine which project will be launched by Runtime Manager:
  - **Run the project currently selected in Plant SCADA Studio** — This is selected by default and will run the project from Plant SCADA Studio.
  - **Run the project deployed from the Deployment Server** — Select this option to set up a computer for deployment. This option allows you to run the deployed projects from the deployment server.
4. To apply your settings, click the **Configure** button.

---

**Note:** If Plant SCADA Runtime is running as a service and the **Project Run Path** option in the Configurator is changed to **Run the project deployed from the Deployment Server**, you need to restart the Runtime Manager service for deployment to function correctly.

---

## Runtime Manager Configuration

This setting instructs Runtime Manager to run as a service. This allows a project to be deployed and run on a computer without having to change other settings.

You need to select **Run Runtime Manager as a service** under the following circumstances:

- When a deployment client is configured on a computer and you want to push project updates using "Force" mode.
- If you are using encrypted communications, and the current user on a deployment client may not have required privileges.

---

**Note:** The current user on a deployment client needs to be a member of the **Configuration Users** security role to run the deployment client (see [Security Roles](#)).

---

### To run Runtime Manager as a service:

1. In the panel on the left side of the **Configurator**, select Plant SCADA | **Computer Setup**. The **Computer Setup** page appears. Navigate to the **Runtime Manager Configuration** section of the dialog.

#### Runtime Manager Configuration

*The Runtime Manager can be configured to run as a service or as a standard process. When run as a service, Runtime Manager allows projects to be deployed to the machine without having to change any other settings.*

**Run Runtime Manager as a service**

2. Select **Run Runtime Manager as a Service**.

---

**Note:** If you are using Deployment to run a project and do not select the **Run Runtime Manager as a Service** option, you will need to manually start the Plant SCADA Runtime Manager before deploying the project.

---

3. To apply your settings, click the **Configure** button.

---

**Note:** If you have configured a Plant SCADA OPC DA server on a computer that is running Plant SCADA as a Windows service, you will need to make the additional configuration changes. See the topic *Running an OPCDA Server as a Service* in the *Runtime* section of the Plant SCADA documentation.

---

## See Also

[Configure a Deployment Server](#)

[Configure a Deployment Client](#)

## Provide Deployment Access to Additional Users

When you install deployment server, additional security roles will be created. By default the local Windows user groups are associated with the security roles as members. These members control access to the deployment server's functionality. You can add additional users to the roles using Configurator (see [Modifying the Members of a Security Role](#)).

**Note:** If your Deployment Server is on a separate computer to the System Management Server, you need to run your IT infrastructure as a domain. To allow authorization in this type of distributed system, only add domain groups to a deployment security role. Avoid adding individual users to these roles.

The security roles and local Windows user groups will be created and mapped as following:

Security Roles	Windows User Group	Description
Deployment Administrators	SCADA.DeploymentAdmins	Members of this role can add or remove client computers to/from deployment server. They can also perform upload and deploy operations.
Deployment Uploaders	SCADA.DeploymentUploaders	Members of this role can upload a new project version to the deployment server.
Deployment Users	SCADA.DeploymentUsers	Members of this role can deploy a new project to a connected deployment client computer.

When the deployment server is installed, the user that is logged in to the computer is automatically added to each of these groups.

For example, you may have a number of engineers that need to add project versions to the deployment server. To enable this, each engineer's Windows user account needs to be added directly or via a domain group to the "Deployment Uploaders" security role locally on the deployment server.

Refer to the documentation provided by Microsoft® to determine how to add user accounts and domain groups to a Windows™ user group.

## See Also

[Configure a Deployment Server](#)

[Connect Plant SCADA Studio to a Deployment Server](#)

## Connect Plant SCADA Studio to a Deployment Server

To deploy project versions to selected deployment clients, you need to connect the computer running Plant SCADA Studio to a deployment server.

There are two different scenarios to consider.

- **Connect Plant SCADA Studio to a local deployment server**

If you have successfully configured a local deployment server on the computer where Plant SCADA Studio is running, the local computer should be added to the **Deployment Servers** on Deployment activity's **Settings** view. You just need to confirm that your local computer is set as the active deployment server (see [Make a Deployment Server Active](#)).

If the local computer was removed from the list of deployment server at some point, you will have to add it again. See [Add a Deployment Server](#).

If you do not have a deployment server configured at all, the Deployment activity you will prompt you to set one up. You can select **Configure Local Server** on the Command Bar to launch the Configurator. See [Configure a Deployment Server](#) for further instructions.

---

**Note:** If a different Windows™ user was logged on to the local computer at the time the deployment server was configured, you may not have access to the deployment server. If this is the case, your Windows user profile needs to be added to the user groups that restrict access to the deployment server. See [Provide Deployment Access to Additional Users](#) for more information.

- **Connect Plant SCADA Studio to a remote deployment server**

The connection between the computer hosting Plant SCADA Studio and a remote deployment server needs to be encrypted. You firstly need to connect the development workstation running Plant SCADA to the System Management Server used by the deployment server (see [Connect a Computer to a System Management Server](#)).

You can then connect Plant SCADA Studio to a remote deployment server by following the procedure described in [Add a Deployment Server](#).

You will also need to confirm that your Windows user has appropriate access to the deployment server (see [Provide Deployment Access to Additional Users](#)).

When you have connected Plant SCADA Studio to a deployment server (and established appropriate access permissions), you will able to perform the tasks described in [Using the Deployment Activity](#).

---

**Note:** If you would like to synchronize your source Plant SCADA projects across a number of engineering workstations, you need to use the [Back Up a Project](#) and [Restore a Project](#) functionality. Deployment only delivers the runtime components of a project.

## See Also

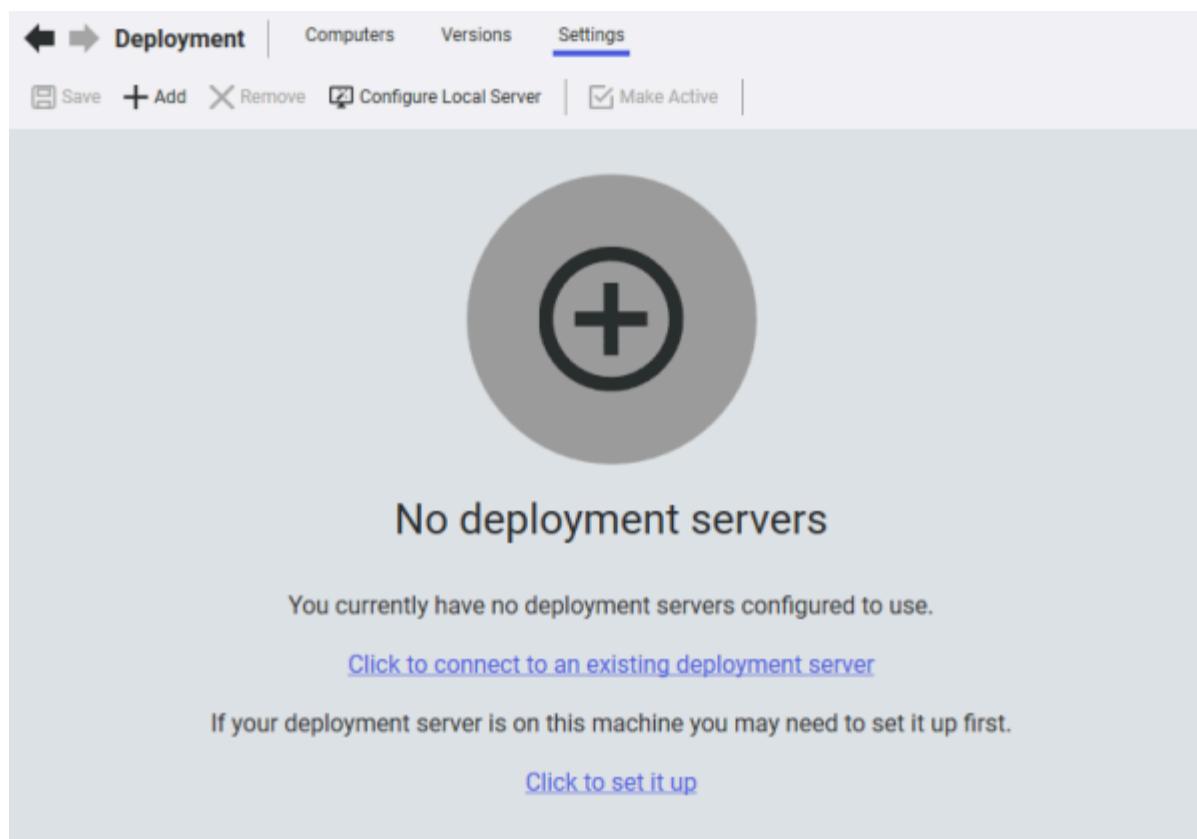
[Deployment in Plant SCADA Studio](#)

## Deployment in Plant SCADA Studio

The Deployment activity in Plant SCADA Studio includes the following views:

- [Computers](#) — use this view to deploy a version of a project to selected computers. You can also assign computers to groups in this view.
- [Versions](#) — use this view to add versions of projects to the deployment server.
- [Settings](#) — use this view to configure a deployment server, or to manage multiple deployment servers if required.

When you first open the Deployment activity, the following may appear



This indicates that Plant SCADA Studio has not been connected to a deployment server. For further instructions on how to proceed, see [Connect Plant SCADA Studio to a Deployment Server](#).

## See Also

[Using the Deployment Activity](#)

## Computers

The **Computers** view lists the deployment clients connected to the active deployment server.

The screenshot shows the 'Deployment' view in AVEVA Plant SCADA Studio. On the left is a sidebar with icons for Deployment, Computers, Versions, and Settings. The main area has tabs for Deploy, Add Group, and Remove. The 'Computers' tab is selected. At the top right, it says 'Deployment Server: deployserver.domain.com' and 'User: userid'. The main table lists four computers under the 'Computer' column: S1\_A1\_PRIMARY, S1\_A1\_STANDBY, S1\_A2\_PRIMARY, and S1\_A2\_STANDBY. The 'Description' column shows 'Area 1 Primary', 'Area 1 Standby', 'Area 2 Primary', and 'Area 2 Standby' respectively. The 'Projects' column shows SITE1\_A1 and SITE1\_A2. The 'Target' column shows the number 2. The 'Active' column shows the project name followed by '(2)'. The 'Status' column shows 'Running' for all. The 'Profile' column shows SITE1\_A1, SITE1\_A1, SITE1\_A2, and SITE1\_A2. The 'Update Method' column shows 'Prompt' for all.

Computer	Description	Projects	Target	Active	Status	Profile	Update Method
S1_A1_PRIMARY	Area 1 Primary	SITE1_A1	2	SITE1_A1(2)	Running	SITE1_A1	Prompt
S1_A1_STANDBY	Area 1 Standby	SITE1_A1	2	SITE1_A1(2)	Running	SITE1_A1	Prompt
S1_A2_PRIMARY	Area 2 Primary	SITE1_A2	2	SITE1_A2(2)	Running	SITE1_A2	Prompt
S1_A2_STANDBY	Area 2 Standby	SITE1_A2	2	SITE1_A2(2)	Running	SITE1_A2	Prompt

**Note:** If Plant SCADA Studio is not connected to a deployment server, you will not be able to access the Computers view. See [Connect Plant SCADA Studio to a Deployment Server](#). You will also need to log in to the deployment server to access and operate this view. If you have not been assigned appropriate access, see [Provide Deployment Access to Additional Users](#) or contact your system administrator.

For each computer you can select one of the projects that have versions available for deployment, and the target version you would like to deploy (see [Deploy a Version](#)). The project that is currently active on the computer is also displayed, as well as its operational status.

You can also:

- Assign computers to groups (see [Create a Computer Group](#))
- Set the method used to restart the Runtime Manager on a deployment client (see [Specify the Update Method](#))
- Deploy a profile to a computer (see [Profiles](#))
- Log out of the deployment server.

Computers have the following properties:

Column	Description
<b>Computer</b>	The name that is defined for the computer in Windows™. You are not able to change this name.
<b>Description</b>	A description of the computer. By default, this will be the description defined for the computer in Windows. To change the description, double-click in the column.
<b>Projects</b>	The name of the project that was most recently deployed to the computer, or the project you have recently selected to be deployed. The drop-down list includes all the projects that are available for deployment to the computer.
<b>Target</b>	The version that was most recently deployed to the

Column	Description
	computer, or the target version you have recently selected to be deployed. The drop-down list includes all the versions that are available for deployment to the computer. See <a href="#">Versions</a> .
<b>Active</b>	<p>The name of project (and version number) that is currently deployed to the computer. The number in brackets indicates the version number.</p> <p><b>Note:</b> The Industrial Graphics and OPC UA servers use only the "Force" method for deployment. If you have installed any of these two servers on the same computer with a Plant SCADA Runtime server or client, and using the "Prompt" or "Notify" update method for deployment, then the <b>Active</b> column will display "?". See <a href="#">Deploy a Version</a>.</p>
<b>Status</b>	<p>Indicates the current deployment status on the computer. One of the following will be displayed:</p> <ul style="list-style-type: none"> <li>• <b>Deploying</b>— a deployment is currently in progress. A progress bar is displayed.</li> <li>• <b>Not Running</b> — the active project is not running. It indicates that, Runtime Manager and Connectivity Server are not running on the computer.</li> <li>• <b>Offline</b> — Runtime Manager and/or Connectivity Server are offline (or cannot be contacted).</li> <li>• <b>Activating</b> — the project is in the process of being activated. This is dependent on the type of update method that is selected (see below). It may indicate that an operator has not restarted Runtime Manager and/or Connectivity Server (as instructed to by a prompt or notification).</li> <li>• <b>Disabled</b> — deployment is disabled on the client computer due to the INI parameter [Deployment]Enable=0.</li> </ul> <p><b>Note:</b> If deployment is unsuccessful, an error icon will display. Click on the down arrow next to the icon to reveal the deployment error message. Refer to <i>Status Column Messages</i> in <a href="#">Deployment Troubleshooting</a> for more information.</p>
<b>Profile</b>	The name of the profile that was most recently deployed to the computer, or the profile you have recently selected to be deployed (see <a href="#">Profiles</a> ). The

Column	Description
	drop-down list includes all the profiles that are available for deployment to the computer. If "(No Profile)" is displayed, the parameter settings within the local Citect.ini will be applied to the deployment client.
<b>Update Method</b>	When a version is delivered to a deployment client, Runtime needs to be restarted to apply the included changes. The Update Method allows you to specify how a restart will be initiated. Choose from the following options: <ul style="list-style-type: none"><li>• <b>Prompt</b></li><li>• <b>Notify</b></li><li>• <b>Force</b></li></ul> See <a href="#">Specify the Update Method</a> .

## See Also

[Versions](#)

## Versions

To deploy a Plant SCADA project, you need to add a "version" of the project to a deployment server.

A version comprises the files that are required to run a project on a deployment client. You can add multiple versions of the same project to a deployment server, and deploy a different version to each client. You can also roll back to a previously deployed version.

The **Versions** view in the Deployment activity displays a list of the projects that have versions available for deployment.

The screenshot shows the 'Deployment' view in AVEVA Plant SCADA Studio. On the left, there's a sidebar with icons for Deployment, Computers, Versions, and Settings. Under 'Deployments', there are two entries: 'SITE1\_A1' with 2 versions and 'SITE1\_A2' with 3 versions. The 'Versions' tab is selected at the top. For 'SITE1\_A1', the table shows two rows: Version 2 (Tuesday, 25 January 2022) and Version 1 (Tuesday, 25 January 2022). For 'SITE1\_A2', it shows three rows: Version 3 (Tuesday, 25 January 2022), Version 2 (Tuesday, 25 January 2022), and Version 1 (Tuesday, 25 January 2022). The table includes columns for Version, Date, Description, Projects, Revision, Created in, and Date.

Deployments		Versions	Details					
		Version	Date	Description	Projects	Revision	Created in	Date
	SITE1_A1	2	Tuesday, 25 January 2022	S1_A1 V2	SITE1_A1	1.0	Plant SCADA 8.40	25/01/2022
		1	Tuesday, 25 January 2022	S1_A1 V1	SITE1_A1	1.0	Plant SCADA 8.40	25/01/2022
	SITE1_A2	3	Tuesday, 25 January 2022	S1_A2 V3	SITE1_A2	1.0	Plant SCADA 8.40	25/01/2022
		2	Tuesday, 25 January 2022	S1_A2 V2	SITE1_A2	1.0	Plant SCADA 8.40	25/01/2022
		1	Tuesday, 25 January 2022	S1_A2 V1	SITE1_A2	1.0	Plant SCADA 8.40	25/01/2022

**Note:** If Plant SCADA Studio is not connected to a deployment server, you will not be able to access the Versions view. See [Connect Plant SCADA Studio to a Deployment Server](#). You will also need to log in to the deployment server to access and operate this view. If you have not been assigned appropriate access, see [Provide Deployment Access to Additional Users](#) or contact your system administrator.

You can use the Versions view to add versions of the active Plant SCADA project to the deployment server (see [Add a Version](#)).

You can also:

- Remove a version (see [Remove a Version](#))
- View the included projects for a version (see the **Projects** column description below)
- Log out of the deployment server.

The **Deployments** column to the left includes each project that has versions added to the deployment server. Each is represented by a green card that also displays the most recent version number and the date it was created. The versions associated with a project are listed to the right.

Versions have the following properties:

Column	Description
<b>Version</b>	The unique number that is automatically incremented with each new version of the project added to the deployment server.
<b>Date</b>	The date the version was added to the deployment server.
<b>Description</b>	A description of the version.
<b>Projects</b>	A list of the included projects in the version. Click on the arrow within the field to reveal the list of included projects.
<b>Revision</b>	The value specified for the project in the Revision

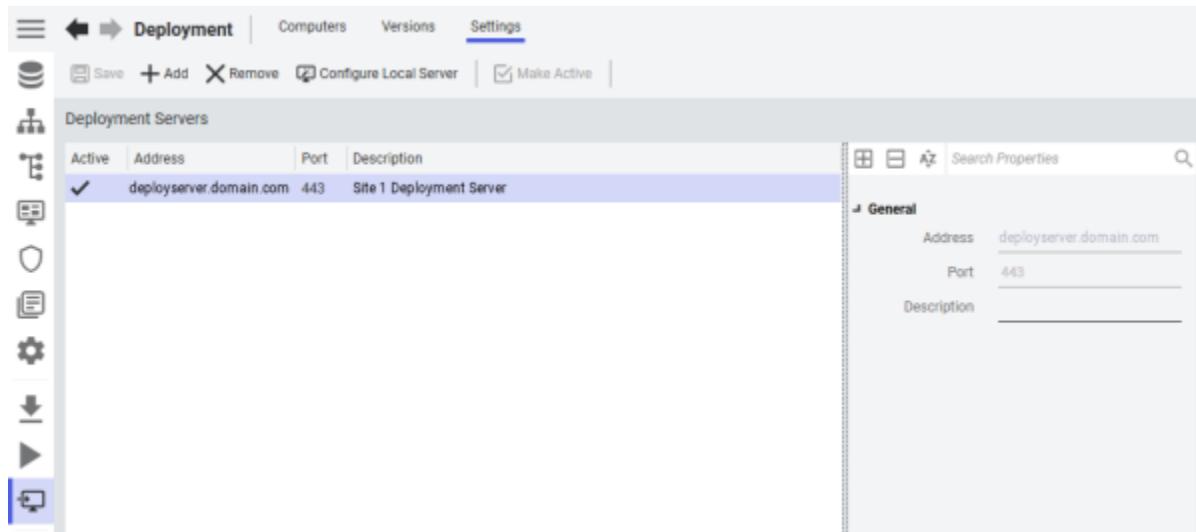
Column	Description
	column of the Projects activity.
<b>Created In</b>	The version of Plant SCADA that was used to create the project.
<b>Date</b>	The date specified for the project in the Date column of the Projects activity.

## See Also

[Deploy a Version](#)

## Settings

The **Settings** view in the Deployment activity lists the deployment servers that you have connected to Plant SCADA Studio. A selected check box indicates which of the servers is currently "active".



**Note:** If the Settings view displays a message prompting you to connect to a deployment server, see [Connect Plant SCADA Studio to a Deployment Server](#).

If you have configured a local deployment server on the computer where Plant SCADA Studio is running, the local computer needs to be added. To include additional servers, you need to add them (see [Add a Deployment Server](#)) to the Settings view.

You can also use the Settings view to:

- Change the deployment server that is currently active (see [Make a Deployment Server Active](#))
- Remove a deployment server (see [Remove a Deployment Server](#))
- Change the **Description** for a deployment server.

Each deployment server has the following properties:

Property	Description
<b>Address</b>	The name of machine hosting the deployment server. This is name defined for the computer in Windows™. You are not able to change this name.
<b>Port</b>	The port used by the deployment server for https communication with the deployment clients. This value is set in the Configurator, you are not able to edit this value.
<b>Description</b>	A description of the deployment server. To change the description, double-click in the column or edit the field in the property grid to the right.

**Note:** You need to log on to an active deployment server to access its **Computers** and **Versions**. If you have not been assigned appropriate access on the deployment server, see [Provide Deployment Access to Additional Users](#) or contact your system administrator.

## See Also

[Deploy a Version](#)

## Using the Deployment Activity

You can perform the following tasks Plant SCADA Studio's **Deployment** activity:

- [Add a Deployment Server](#)
- [Make a Deployment Server Active](#)
- [Remove a Deployment Server](#)
- [Create a Computer Group](#)
- [Rename a Computer Group](#)
- [Remove a Computer Group](#)
- [Remove a Deployment Client](#)
- [Add a Version](#)
- [Deploy a Version](#)
- [Remove a Version](#)
- [Specify the Update Method.](#)

**Note:** You need to log in to the deployment server to access and operate the **Computers** and **Versions** views. If you have not been assigned appropriate access, see [Provide Deployment Access to Additional Users](#) or contact your system administrator.

## See Also

[Deployment in Plant SCADA Studio](#)

## Add a Deployment Server

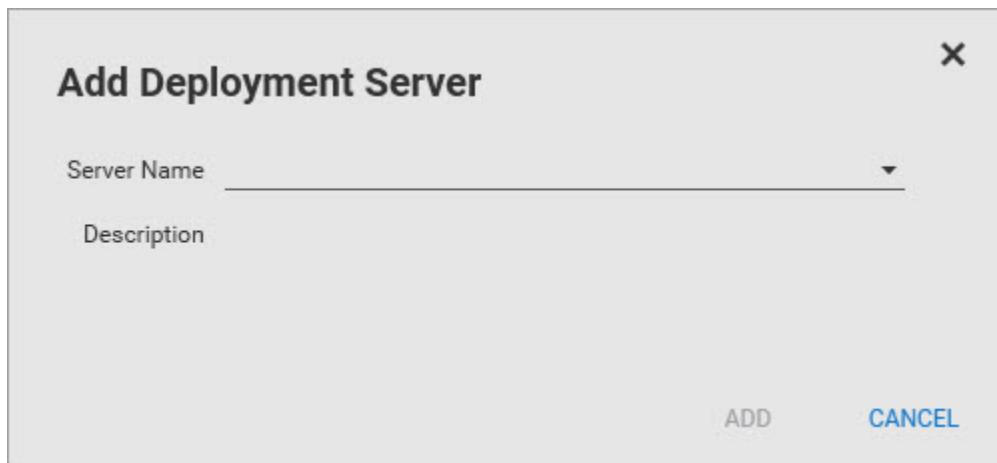
To use a deployment server, you firstly need to add it to the Deployment activity. To do this, use the [Settings](#) view.

**Note:** Plant SCADA Studio requires an encrypted connection to the deployment server to perform deployment operations. Before you add a deployment server, you need to connect the development workstation running Plant SCADA to the System Management Server used by the deployment server (see [Connect a Computer to a System Management Server](#)).

### To add a deployment server:

1. From the **Deployment** activity, select **Settings**.
2. If no deployment servers have been added, select **Click to connect to an existing deployment server**. Otherwise, click **Add** on the Command Bar.

The **Add Deployment Server** dialog will appear.



3. From the **Server Name** list, select the deployment server to which you want to connect. For more information, see [Configure a Deployment Server](#). If the configured deployment server is not listed, enter the Windows computer name. If the deployment server is not configured, click **Configure Local Server** on the Command Bar to configure deployment through the Configurator.
4. Enter an optional **Description** for the server.
5. Click **Add**.  
The deployment server will appear in the **Deployment Servers** list.

6. To use the deployment server, you need to make it the active server (see [Make a Deployment Server Active](#)). Once active, you will be able to log on to the deployment server through the [Versions](#) or [Computers](#) view.

**Note:** You may need to confirm that the current Windows user has appropriate access to the deployment server (see [Provide Deployment Access to Additional Users](#)).

## See Also

[Configure a Deployment Server](#)

[Remove a Deployment Server](#)

## Make a Deployment Server Active

To use a deployment server, it needs to be the selected as the "active" server. To do this, use the [Settings](#) view in the **Deployment** activity.

### To make a deployment server active:

1. From the **Deployment** activity, select **Settings**.
2. In the **Deployment Servers** list, select the required server.
3. On the Command Bar, click **Make Active**.  
The check box next to the server will be selected.
4. Click **Save**.

Once active, you will be able to log on to the deployment server through the [Versions](#) or [Computers](#) view.

## See Also

[Add a Deployment Server](#)

[Remove a Deployment Server](#)

## Remove a Deployment Server

You can remove a deployment server from the [Settings](#) view in the Deployment activity.

### To remove a deployment server:

1. From the Deployment activity, select **Settings**.
2. From the list of **Deployment Servers**, select the server you would like to remove.
3. On the Command Bar, click **Remove**.  
The server will be removed from the Deployment Servers list.
4. The server will be removed from the grid.
5. Click **Save**.

## See Also

[Add a Deployment Server](#)

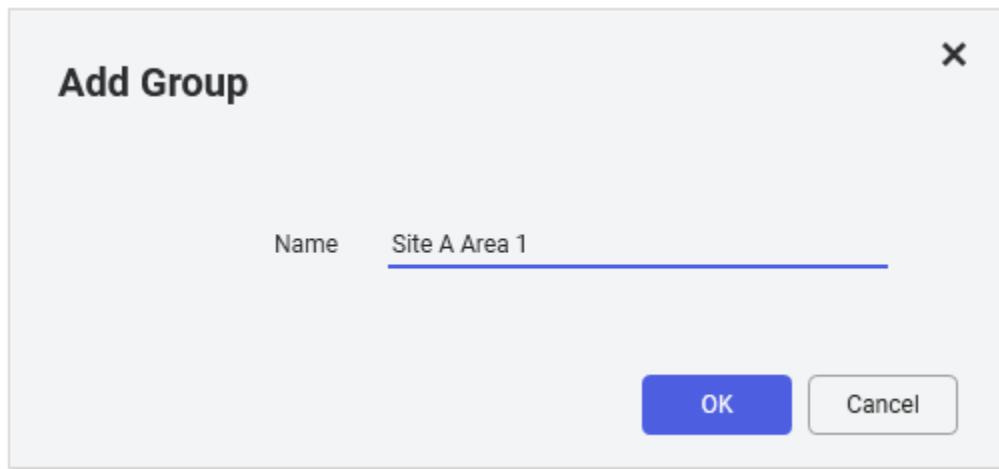
## Create a Computer Group

You can organize the deployment clients in the [Computers](#) view into groups.

### To create a computer group:

1. From the **Deployment** activity, select **Computers**.
2. On the Command Bar, select **Add Group**.

3. In the **Add Group** dialog, enter a name for the group and click **OK**.



The group will display in the **Computer** column.

4. To add computers to the group, select the required computers and drag them in to the group you have created.  
To select multiple computers, select the first computer and hold the **Shift** key while you use the up and down arrow key to select adjacent rows.

You can use the arrow next to a group name or hide or display the computers in a group.

## See Also

[Remove a Computer Group](#)  
[Rename a Computer Group](#)

## Rename a Computer Group

### To rename a computer group:

1. From the **Deployment** activity, select **Computers**.
2. In the **Computer** column, double-click the name of a group to make it editable.
3. Enter a new name for the group.  
A prompt confirming the name change will display.
4. Click **OK**.

## See Also

[Remove a Computer Group](#)  
[Create a Computer Group](#)

## Remove a Computer Group

### To remove a computer group:

1. From the **Deployment** activity, select **Computers**.
2. In the **Computer** column, select the group you want to remove.
3. On the Command Bar, select **Remove**.  
The Remove Group dialog is displayed.
4. Click **OK** to remove the group.

**Note:** When you remove a group, the computers included in the group will remain in the **Computer** column.

## See Also

[Create a Computer Group](#)

[Rename a Computer Group](#)

## Remove a Deployment Client

If a deployment client is no longer used by a deployment server, you can remove it from the [Computers](#) view.

### To remove a deployment client:

1. In the **Deployment** activity, select **Computers**.
2. In the **Computer** list, select the deployment client you would like to remove.
3. On the Command Bar, select **Remove**.
4. On the Remove Computer dialog, click **OK** to confirm the removal.

**Note:** This process only removes the client from the Computers view. Any project versions that have been deployed to the client will still be available.

If you want to replace a removed deployment client, run the Configurator (on the deployment client) to configure the client again.

## See Also

[Configure a Deployment Client](#)

## Add a Version

To deploy a Plant SCADA project, you need to add a version of the project to the deployment server (see [Versions](#)).

### To add a version:

1. From the **Deployment** activity, select **Versions**.

2. In the **Deployments** list, select the project for which you want to add a version.

If no versions exist, you will be prompted to create one for the active project.

- Click the hyperlink **Let's create the first deployment of <Name of Project>**.
- If the active project has not been compiled, you will be prompted to compile it. Click **OK**.
- When compilation is complete, proceed to step **4**.

3. From the Command Bar, select **Add Version**.

If the selected project has not been compiled, you will be prompted to compile it. Click **OK**.

When compilation is complete, the Add Version dialog will appear.

4. Add a **Description** for the new version and click **OK**. (Be aware that you cannot change a version description after it has been added.)

The Version Deployment Preparation dialog will appear. If required, you click **Abort** to stop the preparation progress.

When version preparation is complete, a dialog will inform you if the process was successful.

5. **Close** the notification dialog.

If successful, the new version will appear in the **Versions** view.

Deployment	Versions	Details				
Version	Date	Description	Projects	Revision	Created in	Date
SITE1_A1	3	Tuesday, 25 January 2022	> SITE1_A1	1.0	Plant SCADA 8.40	25/01/2022
		Tuesday, 25 January 2022 S1_A1 V2	> SITE1_A1	1.0	Plant SCADA 8.40	25/01/2022
		Tuesday, 25 January 2022 S1_A1 V1	> SITE1_A1	1.0	Plant SCADA 8.40	25/01/2022

Versions are stored on the deployment server with the file extension ".aaspkg".

## See Also

[Deploy a Version](#)

[Remove a Version](#)

## Deploy a Version

**Note:** To successfully deploy a version of a project, the Connectivity Service or the Runtime Manager needs to be running on the deployment client computer either as a service or as a standard application. To determine which project Runtime Manager is running, refer to the [Deployment Troubleshooting](#) question "**Runtime Manager is not running the deployed project**".

### To deploy a version:

1. From the **Deployment** activity, select **Computers**.
2. Select the check box next to the computers or groups to which you want to deploy a version. To make multiple selections, you can:

- Select adjacent rows by holding the **Shift** key while you use the up or down arrow keys.
- Select distributed rows by holding the **Ctrl** while selecting the required rows. (You can also cancel selection by clicking on a row again.)
- Select all computers by using the check box next to the **Computers** column header.

---

**Note:** If you have selected multiple computers, any setting changes you make to a particular computer will apply to every selected computer.

3. In the **Projects** column, select the name of the project you would like to deploy.

The **Target** column will display the latest available version of that project.

4. In the **Target** column, select the version of the project you would like to deploy from the drop-down menu. You can select a version that was previously deployed if required.

If the target version has been previously deployed, you can rollback to an earlier version of the project directly from the client computer using the Computer Setup Wizard. See [Project Configuration](#) for more information.

5. In the **Profile** column, select the profile you would like to deploy from the drop-down menu (see [Profiles](#)). The settings in the associated <ProfileName>.ini file will be implemented on the deployment client after the Runtime Manager has been restarted. If no profile is selected, the local Citect.ini file will be used.

If you have configured two profiles for the one project, both profiles are deployed with the version of the project; however, only the selected profile will be used.

To determine which profile is active on the deployment client, check the setting for the Citect.ini parameter [RuntimeManager]Profile=<ProfileName>.

---

**Note:** The Industrial Graphics and OPC UA servers do not support profiles. If you have installed the Industrial Graphics server and/or OPC UA server on the same computer with a Plant SCADA Runtime server or client, then only the Runtime server and client will use the selected profile. The Industrial Graphics and OPC UA servers will ignore the selected profile.

6. In the **Update Method** column select either Prompt, Notify or Force (see [Specify the Update Method](#)).

If no project versions have previously been deployed to the client computer, Plant SCADA will force an update to the Runtime Manager and/or the Industrial Graphics server.

---

**Note:** The Industrial Graphics and OPC UA servers only support the "Force" update method. If you have installed the Industrial Graphics server and/or OPC UA server on the same computer with a Plant SCADA Runtime server or client, then only the Runtime server and client will use the selected update method. The Industrial Graphics and OPC UA servers will apply the "Force" method.

If the operator is prompted to update and clicks **Cancel** instead of accepting the new version, a message will be displayed in Runtime Manager. This message alerts the operator that a new deployment of a project is available. When Runtime Manager is restarted, the project will be activated.

7. Ensure that the Connectivity Service and/or the Runtime Manager are running on the deployment client computer.

8. Click **Deploy**.

9. The status column will update with the progress of the deployment.

10. When the version has been deployed, the **Active** column will display the project name with the active version number in brackets.

If you have installed the Industrial Graphics server and/or OPC UA server on the same computer as a Plant SCADA Runtime server or client, and using the "Prompt" or "Notify" update method for deployment, then the **Active** column in the deployment computer will display:

- "?(?)" for a new deployed project.
- "<Project name>(?)" for a new deployed version.

Restarting the Runtime Manager will run the new deployed project or version, and the **Active** column will display the deployed project name with the active version number.

**Note:** When the Industrial Graphics server and/or OPC UA server are installed on the same computer as a Plant SCADA Runtime server or client, they will update to a new deployed version at different speeds. So, the **Active** column in the deployment computer will be updated, only when the deployment is complete.

- 
11. After successful deployment, Plant SCADA will start the relevant processes. The Runtime Manager and/or the Industrial Graphics server on the deployment client will be updated with the deployed project version.

**Note:** If deployment is unsuccessful, an exception icon will display. Click on the down arrow to reveal the error message and click **OK** to close. For information on each error message, refer to the [Deployment Troubleshooting](#) section.

---

## See Also

[Versions](#)

[Create a Computer Group](#)

## Remove a Version

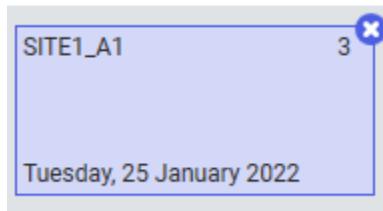
If a version is no longer required on a deployment server, you can remove it.

### To remove a version:

1. From the **Deployment** activity, select **Versions**.
2. Select the version of the project you would like to remove.
3. On the Command Bar, click **Remove Version**.

Or:

If you want to remove all versions of a project, click the "x" icon in the top right corner of the project card.



4. A Remove Version notification will appear. Click **Yes** to confirm.

**Note:** You cannot remove a version that is currently active on a deployment client. To remove an active version, you firstly need to replace it by deploying a different version.

---

## See Also

[Versions](#)

[Add a Version](#)

## Specify the Update Method

When a version is delivered to a deployment client, the Runtime Manager needs to be restarted to apply the included changes. The **Update Method** allows you to specify how a restart will be initiated.



### WARNING

#### LOSS OF CONTROL

- If you set the Update Method to **Force**, the runtime system will automatically shutdown on the deployment client when a deployment occurs.
- If you set the Update Method to **Prompt**, the runtime system may be shutdown at an inappropriate time if an operator does not fully understand the implications of the dialog that appears.

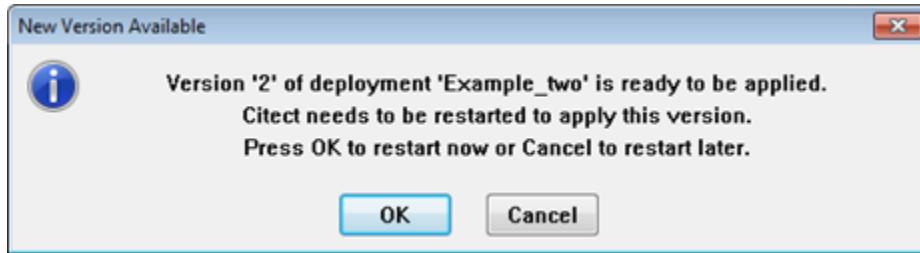
You need to carefully consider how an update method will impact the runtime system before you deploy a project version.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

There are three options you can use for the Update Method.

- **Prompt**

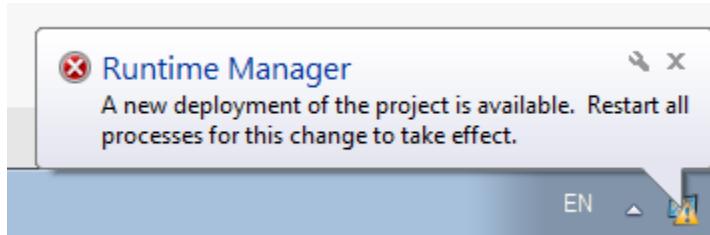
When the Update Method is set to **Prompt**, the client machine accepts the project and a message box will display prompting the operator to restart Runtime Manager.



**Note:** To use your own prompt instead of the default prompt, you can create custom Cicode and implement it with the parameters [Deployment]AskRestartFunc and [Deployment]AskRestartArgs. Refer to [Create a Custom Deployment Prompt](#) for more information.

- **Notify**

When the Update Method is set to **Notify**, the client machine accepts the project and a notification is added to the system tray indicating a new project has been deployed and that Runtime Manager needs to be restarted.



- **Force**

When the Update Method is set to **Force**, the client machine accepts the project and restarts Runtime Manager automatically.

**Note:** If you do not restart the Runtime Manager after using "Prompt" or "Notify" updated method, it will continue with the older version and project. If you have installed Industrial Graphics server and/or OPC UA Server in the same machine, then the Industrial Graphics Server and/or OPC UA Server automatically activates the newly deployed project and version. So, the Runtime manager and Industrial Graphics Server and/or OPC UA Server may not sync with each other.

---

If Runtime Manager is not running on a deployment client, a deployed project will be transferred but will not run. You will need to start Runtime Manager to launch the project.

## See Also

[Deploy a Version](#)

## Create a Custom Deployment Prompt

If the **Update Method** for deployment is set to **Prompt**, the deployment client normally displays a default "New Version Available" message box when deployment occurs (see [Specify the Update Method](#)). However, you can customize the way an operator is notified that Runtime Manager requires a restart.

For example, you could:

- Build a delay function that allows an operator to postpone a restart for a period of time.
- Add a simple animated notification to a graphics page.

To do this:

1. Create custom Cicode that produces the result you require.
2. Implement the Cicode using the parameters [\[Deployment\]AskRestartFunc](#) and [\[Deployment\]AskRestartArgs](#).

## See also

[Deploy a Version](#)

## Using Citect Anywhere to View Deployed Projects

Citect Anywhere™ is a browser-based client that you can use to view a Plant SCADA project using a mobile device. You can use Citect Anywhere clients to view deployed projects at runtime, and to acknowledge deployed project notifications.

Once you acknowledge a notification regarding a new deployment, Runtime Manager will restart and your Citect Anywhere session will end. On relaunching your Citect Anywhere client, you will be able to view the newly deployed project.

---

**Note:** To view deployed projects, Runtime Manager needs to be running on the deployment client either as a service or as a standard application.

## See Also

[Deployment in Plant SCADA Studio](#)

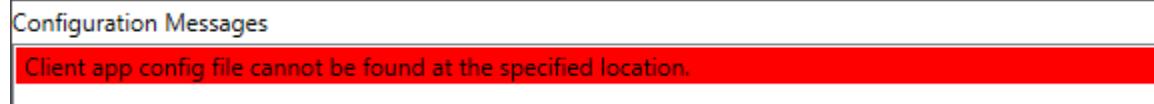
# Deployment Troubleshooting

The suggested solutions described here are organized into the following sections:

1. The Configurator
2. Deployment Activity
3. Status Column Messages
4. System Management Server

## The Configurator

When you try to configure a deployment server or deployment client with the Configurator, the **Configuration Messages** panel may display a message with a red background.



This indicates that the task you were attempting has not been successful. For a suggested solution, select the relevant message from those listed below.

---

**Note:** If you double-click on a message in the Configuration Messages panel, a dialog box will appear that provides more details about the message you have selected.

### An existing database has been detected and it is password protected.

The Configurator stores the deployment database password to the ArchestraA™ Data Store (ADS) service. This message appears when the Configurator is unable to access the deployment database using the stored password provided by the ADS service (or a stored password does not exist).

This could be the result of manually restoring another deployment database, or the removal of the password data from the ADS service. To repair the deployment server:

1. Enter the last known password that can access the password-protected database.  
Or
2. Delete the deployment database and reconfigure the deployment system.

### Configuration of the Deployment Server was unsuccessful.

The Configurator stores the deployment database password in the ArchestraA™ Data Store (ADS) service. This message appears when the Configurator cannot establish communication with the ADS service.

To repair the deployment server:

1. Ensure the ADS service is running via Windows™ Services console.  
Or
2. Check the Archestra System Management Console for any error messages.

**Failed to access the System Management Server due to invalid user name or password.**

Confirm that the user specified on the Authorize page is a member of either the "aaAdministrators" or "Administrators" Windows group on the System Management Server.

**Deployment Client registration was unsuccessful.**

There are a few solutions to consider when this message appears:

1. **Deployment server administrator credentials** — If the action suggested in the detailed configuration message tells you to check the "Deployment Server Administrator credentials", it indicates that the User Name and Password entered on the AUTHORIZE page of the Configurator were not valid. Confirm that valid user credentials have been entered. For more information, see [Provide Deployment Access to Additional Users](#).
2. **SSL/TLS exception** — If the detailed configuration message indicates that an SSL/TLS exception occurred, it means the deployment certificate is not available in the expected location on the local workstation (Trusted Root Certification Authorities). The certificate is required to establish https communication between the computer running the Configurator and the deployment server.
3. **Server refuses connection** — If the detailed configuration message indicates that the target server "actively refused" a connection or "failed to respond", try the following solutions:
  - The Plant SCADA Deployment Server Service may not be running, or is not running on the designated network port. Check the deployment server service is running and is bound to the correct network port.
  - A firewall on the deployment server service may be blocking inbound connections. Check the firewall settings and allow the network inbound connection for the deployment server service.
  - A firewall on the workstation running the Configurator may be restricting an outbound connection. Allow the outbound connection.
  - A proxy setting may present on the workstation running the Configurator. Check the proxy setting on the workstation and confirm that the deployment server can be contacted.
  - The DNS (Domain Name System) did not recognize the deployment server by machine name, or the IP does not match the machine name with the DNS record. This may be due to an IP address change, as the deployment server was not in contact with DNS server for a period of time. The deployment server/client address uses a full computer name address convention; Host name + DNS suffix (for example, "https://computer\_name.example.com"). If the deployment server is not recognized by its computer name, try to restart the deployment server and wait a period of time (this may take up to 15 minute or depending on your network infrastructure). If you are still unable to resolve the machine name, contact your network administrator.

---

**Note:** If you want your system to use secured features such as encryption or deployment, you should not install Plant SCADA on the computer that acts as the domain controller within your Windows® domain.

**The "DeploymentClient" could not be started**

You will see this error if the authentication details to access the Deployment Server have changed since the previous login.

Delete the Deployment Client configuration file located in "%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Config". Re-configure the Deployment client. For more information, see [Configure a Deployment Client](#).

**The system cannot find the file specified.**

Any Windows™ 10 computer that has upgraded to Version 1511 (and Build 10586.420) will have this issue.

This is caused by missing registry entries ("RegisteredOwner" and "RegisteredOrganization") in:

HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\Microsoft\Windows NT\CurrentVersion

Add the registry keys, restart the Configurator, and try to configure the deployment server again.

## Deployment Activity

Review the descriptions below.

### Unable to deploy projects following upgrade to Plant SCADA 2023

The deployment architecture installed with Plant SCADA 2023 is not compatible with deployment servers running an earlier version. If you are trying to deploy a 2023 project to a computer that has not been upgraded to 2023, the following limitations will apply.

- You cannot use Plant SCADA Studio to add projects to a deployment server that is running a different version of Plant SCADA.
- You cannot deploy a project from a deployment server to a deployment client if they are running different versions of Plant SCADA.

If you have a distributed system that includes some computers on an earlier version of Plant SCADA, you should reinstate your existing deployment server to distribute projects to these computer until the roll out of 2023 is complete. You will also need to configure a new deployment server to deploy 2023 projects to your upgraded computers.

If you upgraded from version 2020 R2, you can uninstall 2023 and reinstate a 2020 R2 deployment server. You can use your existing configuration files to connect with any deployment clients that are still running 2020 R2.

If you upgraded from version 2018 R2, you can uninstall 2023 and reinstate the 2018 R2 deployment server. You will need to generate new certificates to reconfigure the deployment client connections.

### Unable to log into deployment server: "Service Unavailable"

When you attempt to log in to a deployment server, a popup with the following message appears:

"A problem occurred while performing 'LOG IN'. Service Unavailable (503)"

This means the deployment server cannot be reached by Plant SCADA Studio. This can be caused by:

- A firewall is present on the deployment server service blocking inbound connections. Check the firewall settings and allow inbound connections.
- A firewall on the computer running Plant SCADA Studio may be restricting an outbound connection. Check the firewall settings and allow outbound connections.
- The deployment server service may not be running, or is not running on the designated network port. Check that the deployment server service is running and is bound to the correct network port.
- A proxy setting may present on the computer running Plant SCADA Studio. Check the proxy setting on the computer and confirm that the deployment server can be pinged.
- The DNS (Domain Name System) did not recognize the deployment server by machine name, or the IP does not match the machine name with the DNS record on the workstation running Plant SCADA Studio. This may be due to an IP change as the deployment server was not in contact with DNS server for a period of time.

The deployment server address uses a full computer name address convention; Host name + DNS suffix (for example, "[https://computer\\_name.example.com](https://computer_name.example.com)"). If the deployment server is not recognized by the computer name, try to restart the deployment server and wait a period of time (this may take up to 15

minute or depending on your network infrastructure). If you are still unable to resolve the machine name, contact your network administrator.

---

**Note:** If you want your system to use secured features such as encryption or deployment, you should not install Plant SCADA on the computer that acts as the domain controller within your Windows® domain.

---

### **Unable to log into deployment server: "The underlying connection was closed."**

When you attempt to log in to a deployment server, a popup with the following message appears:

"An error occurred while sending the request. The underlying connection was closed. Could not establish trust relationship for the SSL/TLS secure channel."

Plant SCADA Studio connects with the deployment server via an encrypted connection to perform deployment operations. The System Management Server is used to establish a trust relationship between the computer running Plant SCADA Studio and the deployment server.

The above error may be shown when the connection between the computer hosting Plant SCADA Studio and a remote deployment server is not encrypted.

To resolve this, you need to connect the development workstation running Plant SCADA to the System Management Server used by the deployment server (see [Connect a Computer to a System Management Server](#)).

You will also need to confirm that your Windows user has appropriate access to the deployment server (see [Provide Deployment Access to Additional Users](#)).

### **The deployment client service is running, but the status of the computer shows "Offline"**

This means the deployment server cannot be reached by the deployment server. This can be caused by:

- A firewall is present on the deployment client service blocking inbound connections. Check the firewall settings and allow inbound connections.
- A firewall on the deployment server may be restricting an outbound connection. Check the firewall settings and allow outbound connections.
- A proxy setting may be present on the deployment server. Check the proxy setting on the computer and confirm that the deployment client can be contacted.
- The DNS (Domain Name System) did not recognize the deployment client by machine name, or the IP does not match the machine name with the DNS record on the deployment server. This may be due to an IP change as the deployment client was not in contact with DNS server for a period of time.

The deployment client address uses a full computer name address convention; Host name + DNS suffix (for example, "https://computer\_name.example.com"). If the deployment client is not recognized by the computer name, try to restart the deployment client and wait a period of time (this may take up to 15 minute or depending on your network infrastructure). If you are still unable to resolve the machine name, contact your network administrator.

---

**Note:** If you want your system to use secured features such as encryption or deployment, you should not install Plant SCADA on the computer that acts as the domain controller within your Windows® domain.

---

- A trusted connection cannot be established between deployment server and client. To repair the connection, you need the original authentication file that was generated when the deployment server was configured. Use the Configurator to reconfigure deployment client using the original authentication file.

### **Deployment is enabled on a client, but the status of the computer shows "Disabled"**

If you use the Configurator to change a deployment client's Project Run Path from "Run the project selected in Plant SCADA Studio" to "Run the project deployed from the Deployment Server", it should enable deployment

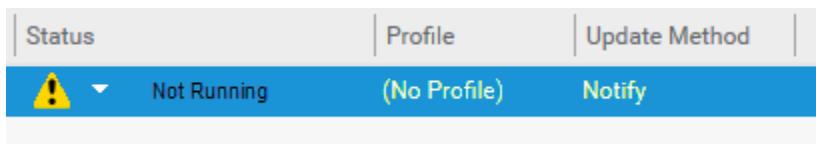
on the computer. However, the status of the computer may still show "Disabled" in Plant SCADA Studio.

Selecting the "Run the project deployed from the Deployment Server" option sets the [Deployment]Enabled INI parameter to 1. However, you need to restart the running Plant SCADA processes for the change to take effect.

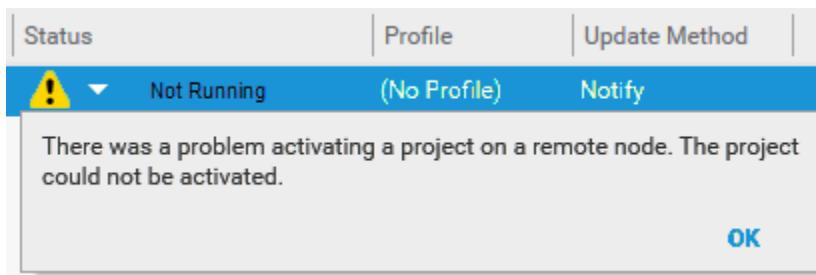
- If any Plant SCADA processes, AVEVA Plant SCADA Runtime Manager or AVEVA Plant SCADA Connectivity Server are not running, restart the Plant SCADA Deployment Client Service or start AVEVA Plant SCADA Runtime Manager and AVEVA Plant SCADA Connectivity Server on the deployment client machine.
- If any Plant SCADA processes are running, restart all the processes via AVEVA Plant SCADA Runtime Manager and AVEVA Plant SCADA Connectivity Server.
- If there are no Plant SCADA processes listed in AVEVA Plant SCADA Runtime Manager, restart AVEVA Plant SCADA Runtime Manager (or the AVEVA Plant SCADA Runtime Manager Service) and AVEVA Plant SCADA Connectivity Server.

## Status Column Messages

The **Status** column on the [Computers](#) view may display a notification icon. It will typically appear when you attempt to deploy a project version.



This indicates that runtime is not operating as expected on the deployment client. To view a message describing the circumstances, click the arrow to the right of the icon.



For a suggested solution, select from the messages listed below.

### **One or more Plant SCADA processes are stopped.**

One or more Plant SCADA processes in AVEVA Plant SCADA Runtime Manager and/or AVEVA Plant SCADA Connectivity Server are not running. Start these processes, or restart AVEVA Plant SCADA Runtime Manager and AVEVA Plant SCADA Connectivity Server on the deployment client to complete deployment.

### **An unexpected error occurred in one or more Plant SCADA processes.**

One or more Plant SCADA processes in AVEVA Plant SCADA Runtime Manager and/or AVEVA Plant SCADA Connectivity Server are not operating as expected. Restart AVEVA Plant SCADA Runtime Manager and AVEVA Plant SCADA Connectivity Server on the deployment client.

### **One or more Plant SCADA processes are not responding.**

One or more Plant SCADA processes in AVEVA Plant SCADA Runtime Manager and/or AVEVA Plant SCADA Connectivity Server are not operating as expected. Restart AVEVA Plant SCADA Runtime Manager and AVEVA Plant SCADA Connectivity Server on the deployment client.

**There was problem transferring a project version onto a remote node. The transfer did not finish successfully.**

This message can indicate:

- The connection between the deployment server and deployment client is not working.  
Or
- There is not enough disk space on the deployment client to download the project version.

To resolve this:

- Check that the connection between deployment server and deployment client is functional and stable.
- Confirm that there is enough disk space available for the project version.

The default location of the cache folder that stores downloaded versions is:

%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Deployment\Client\Cache

This is defined as a value of "DeploymentNode.BaseStoragePath" under the "appSettings" section of SE.ASb.Deployment.Node.Windows.Service.exe.config.

If you manually change this value, you need to restart the Plant SCADA Deployment Client Service.

**There was a problem activating a project on a remote node. The project could not be activated.**

This message can indicate:

- AVEVA Plant SCADA Runtime Manager (or the AVEVA Plant SCADA Runtime Manager Service) is not running.  
Or
- AVEVA Plant SCADA Connectivity Server is not running.  
Or
- Deployment is disabled on the deployment client.

To resolve this, first confirm that AVEVA Plant SCADA Runtime Manager (or the AVEVA Plant SCADA Runtime Manager Service) and AVEVA Plant SCADA Connectivity Server are running.

To enable deployment on the deployment client:

1. Launch the Configurator.
2. Select **Plant SCADA | Computer Setup** in the left panel.
3. Select **Run the project deployed from the Deployment Server** under **Project Run Path**.
4. Click **Configure**.

When resolved, you can try to deploy the version again.

**An internal error occurred while deploying a project version onto a remote node. The deployment did not finish successfully.**

This message can indicate:

- There is not enough disk space on the deployment client to unpack the version.
- The Deployment Client Service is running as a Network Service user, and the following has occurred:
  - The Network Service user does not have full control permission to the Cache folder.

- The Network Service user does not have full control permission to the Deployment folder.

To resolve this, firstly confirm that there is enough space on the disk for the version. The deployment folder path is defined as value of [CtEdit]Deploy in Citect.ini. To change deployment folder path:

1. Launch the Configurator.
2. Select **Plant SCADA | Deployment Client** in the left panel. The **START** page appears.
3. Check that the radio button **Update the configuration of the current Deployment client** is selected. Click **Next**. The **SETTINGS** page appears.
4. Enter a valid path in the field under **Deployed project location**.
5. Click **Configure**.

When resolved, you can try to deploy the version again.

To resolve the Network Service user issue:

- Confirm that the Network Service user has full control permission to the cache folder.  
The default location of the cache folder that stores downloaded versions is:  
`%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Deployment\Client\Cache`  
This is defined as a value of "DeploymentNode.BaseStoragePath" under the "appSettings" section of `SE.ASb.Deployment.Node.Windows.Service.exe.config`.
- Confirm that the Network Service user has full control permission to the deployment folder.  
The deployment folder path is defined as value of [CtEdit]Deploy in Citect.ini.

When resolved, you can try to deploy the version again.

## System Management Server

### Changing the System Management Server port after a Deployment Server and Client have been configured

The Deployment Client, Deployment Server and the System Management Server use port sharing. This allows all these services to use the same default https port number (443).

After configuring these services, we recommend that you do not change the port. If you need to do this, perform the following steps:

1. On the **Advanced Settings** page of Configurator's System Management Server plugin, change the port number on the **Ports** tab.
2. Click on **Configure** button to configure the SMS with the new port.
3. Restart Configurator.
4. Stop the Deployment Server and Deployment Client services.
5. Delete the following files from the default location "`%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Config`", or from the location specified during installation.
  - `SE.ASb.Deployment.Node.WindowsService.exe.config`
  - `SE.ASb.Deployment.Server.WindowsService.exe.config`
6. Launch the Configurator.

7. Reconfigure the Deployment Server and Deployment Client using the “Configure...” option. The Deployment Server and Deployment Client services will start up without any error messages.
8. If you have also changed the HTTP port, then restart the connectivity service.
9. Launch Plant SCADA Studio.
10. In the **Settings** view of the Deployment activity, add a Deployment Server with the new port number and make it active.
11. Log in to the Deployment Server and delete the computer that belongs to the old port. The status of that computer will be "Offline".

All deployment activities will now work with the new System Management Server port.

## Runtime

Runtime activates a compiled project and engages it with the production system. When launched, runtime allows your operators to visually monitor a system, initiate production processes and respond to alarm conditions. Historical and trend data can also be collated while runtime is active.

You can launch runtime for the active project in Plant SCADA Studio via the Activity Bar (see [Run a Project](#)).

You can also use the [Runtime Manager](#) to run individual processes in a multi-process system, allowing a particular server or client to be launched independently. This can be useful for hardware testing and system analysis. In some cases, this also allows you to implement project configuration changes without restarting your runtime system (see [Online Changes](#)).

If required, you can temporarily transfer the communications of a server to another computer to assist with hardware maintenance and system analysis (see [Server Redirection Using Address Forwarding](#)).

You can also configure actions to occur when runtime commences (see [Startup and Runtime Configuration](#)), and run the system using a specific configuration file ( see [Using an Alternative INI File](#)).

There are tools available to [Monitor Runtime](#) and [Debug Runtime](#).

---

**Note:** Before you attempt to run a project, check that the host computer has been appropriately configured using the Setup Wizard. See [The Setup Wizard](#) for more information.

---

## See Also

[Firewall Settings and Plant SCADA](#)

[Virtualization Host Support](#)

[Anti-virus Software Setup](#)

[Running the System as a Windows Service](#)

## Run a Project

### To launch runtime:

1. Check that the project you would like to run is currently selected as the active project (see [View Projects](#)).
2. On the Activity Bar, select **Run**.



If Plant SCADA detects that the active project is not compiled, it will compile the project.

When this is complete, the [Runtime Manager](#) will launch and display the status of each process as they start up. If required, you can use the **Cancel** button on the Runtime Manager to stop runtime from launching.

Runtime Manager (Service Mode)

 Special Build: 8.30.0.405  
ExampleSA  
Encryption: Enabled

CPU	Process ID	Process	Type	Status	Message
All	14456	System Services	System Services	Running (Demo)	
All	20928	Cluster1.IOServer1	IOServer	Running	
All	3428	Cluster1.AlarmServer1	Alarm	Running	
All	14092	Cluster1.TrendServer1	Trend	Running	
All	24360	Cluster1.ReportServer1	Report	Running	

[Restart All](#) [Shutdown All](#) [Hide](#) [Help](#)

**Note:** You can also use Runtime Manager to stop and start individual processes in a multi-process system. This can be useful for hardware testing and system analysis. See [Launch Runtime Manager from Plant SCADA Studio](#).

## See Also

[Runtime Manager Interface](#)

## Runtime Manager

Runtime Manager is used to start, stop, monitor, and shutdown Plant SCADA processes during runtime.

Runtime Manager

 Special Build: 8.30.0.405  
ExampleSA  
Encryption: Enabled

CPU	Process ID	Process	Type	Status	Message
All	10916	Cluster1.IOServer	IOServer	Running	
All	12100	Cluster1.AlarmServer1	Alarm	Running	
All	3828	Cluster1.TrendServer1	Trend	Running	
All	4680	Cluster1.ReportServer1	Report	Running	
All	9336	Client	Client	Running	

[Restart All](#) [Shutdown All](#) [Hide](#) [Help](#)

When you run a Plant SCADA project, the [Runtime Manager Interface](#) will appear. It reviews your project settings, and the Citect.ini file, then starts the required processes on the specified CPUs. The **Message** column

displays verbose startup messages for each process as they start.

The Runtime Manager also displays the status of encryption configured on the workstation. This can be one of the following depending upon the setup defined in the [Encryption](#) node of Configurator:

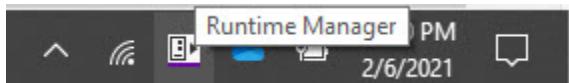
- Enabled
- Mixed
- Disabled.

If the encryption status is **Invalid Configuration**, it indicates that encryption has been enabled but the System Management Server has not been configured. [Configure a System Management Server](#) using Configurator before running your project again.

**Note:**

- If the Runtime Manager encounters a licensing issue or an invalid configuration in the Citect.ini file during startup, an alert message is displayed, an error is logged and the system does not start.
- If networking is disabled, you can only start up Plant SCADA as a single process.
- When working with deployed versions of projects, you can restart a single process or all processes if the deployed project that is running has not been changed; however if a new version has been deployed, you will not be able to restart a single process. You will need to restart all processes.

When all the Plant SCADA processes have started, the Runtime Manager window minimizes to an icon in the notification area on the Windows™ task bar. It will continue to be operational while your project is running.



Clicking on the icon in the notification area restores Runtime Manager to the foreground, allowing you check the status of the Plant SCADA processes.

Runtime Manager can also be launched during the configuration of a project (see [Launch Runtime Manager from Plant SCADA Studio](#)). Launching Runtime Manager in this way presents it in an idle state, allowing you to start each process independently.

When displayed, you can use Runtime Manager to perform the following tasks:

- [Monitor Runtime Processes](#)
- [Start a Process](#)
- [Stop a Process](#)
- [Restart a Process](#)
- [Cancel a Process](#)
- [Use the Kernel with a Selected Process](#)
- [Hide Runtime Manager](#)
- [Shut Down Runtime Manager](#).

You can also reload updated database files for a process during runtime, allowing you to implement some project configuration changes without performing a full restart. For more details see [Reload a Process](#).

For more information on using Runtime Manager, click on its **Help** button.

**Note:** If the title bar of Runtime Manager displays "Runtime Manager (Service Mode)", it means the local Plant SCADA servers are currently being run as a Windows™ service. While in Service Mode, the Runtime Manager will operate with limited functionality. For more information about Service Mode, see [Operate Runtime Manager in Service Mode](#)

---

[Service Mode.](#)

## Runtime Manager Interface

Runtime Manager represents each individual process on the local computer as a row in a table. Each row (process) includes the following columns:

Column	Description
CPU	The CPU number(s) on which the process is running, or 'All' to indicate the process is running on all CPUs.
ProcessID	The Windows ProcessID.
Process	Lists each Plant SCADA component running in the process. Where Plant SCADA is running in single process mode and all components are sharing a process, they appear comma-separated on a single line in this column.
Type	The type of process (Alarm, Report, Trend, I/O Server or Client). Where Plant SCADA is running in single process mode and all components are sharing a process, the type is known as 'MIXED'.
Status	The status of each process. See <a href="#">Monitor Runtime Processes</a> for a list of each status reported by Plant SCADA Runtime Manager.
Message	Displays message information relevant to the status of each process, such as the date and time of project compilation, as well as project name. For example, during startup of each process the verbose startup message appears in this column.

The Plant SCADA Runtime Manager interface also includes the following buttons:

- **Restart All** - Restarts all the processes on the workstation, including any that are not currently running (see [Restart a Process](#)).
- **Start All** - If no processes are currently running, this button starts all the processes on this workstation (see [Start a Process](#)).

As soon as a process is running, this button changes to **Shutdown All** which stops all processes and shuts down Runtime Manager (see [Shut Down Runtime Manager](#)).

While a process is starting, this button will display **Cancel**. This allows you to stop any processes that are currently starting (see [Cancel a Process](#)).

- **Hide** - minimizes Runtime Manager (see [Hide Runtime Manager](#)).
- **Help** - opens the Runtime Manager documentation.

## See Also

- [Stop a Process](#)
- [Reload a Process](#)
- [Use the Kernel with a Selected Process](#)

## Launch Runtime Manager from Plant SCADA Studio

You can start [Runtime Manager](#) during the configuration of a project from the following locations:

- The System Menu (click on the following icon in Plant SCADA Studio's top left corner)  

- Graphic Builder's **Tools** menu.

This will launch the Runtime Manager in an idle state, allowing you to start processes independently in a multi-process system. This can be useful for system testing and analysis, as it allows you to start a particular server or client without having to perform a full compile of a project.

**Note:** Only one instance of Plant SCADA Runtime Manager can run on a machine at any time.

Once a process is started (by right-clicking on it and selecting **Start**), the Runtime Manager displays the startup log for the selected processes and launches runtime.

## See Also

- [Runtime Manager Interface](#)

## Monitor Runtime Processes

Plant SCADA Runtime Manager monitors the status of each Plant SCADA process running on the local workstation. It presents the status of each process in the Status column. This can include the following states.

Status	Definition
Exception	The Plant SCADA process being monitored has stopped functioning and is currently generating an exception report, which can be used to diagnose the situation.
Starting	The Plant SCADA process being monitored is currently starting up.
Stopped	The Plant SCADA process being monitored has: <ul style="list-style-type: none"><li>• Been stopped due to a shutdown.</li></ul>

Status	Definition
	<ul style="list-style-type: none"> <li>• Stopped unexpectedly.</li> <li>• Disappeared due to a shutdown via Windows™ Task Manager.</li> </ul>
Running	The Plant SCADA process being monitored is currently running.
Stopping	The Plant SCADA process being monitored is currently shutting down.

When the status of a process changes to exception, a notification balloon appears from the task bar icon to inform you a process has stopped unexpectedly. A further notification balloon appears when the process has finished generating the exception report and the state has transitioned to stopped.

When the status of a process changes to **stopped** abnormally, a notification balloon appears from the task bar icon to inform you a process has stopped.

If the status of a process changes to **not responding**, a notification balloon appears from the task bar icon to inform you a process is not responding.

While a process is running, the Message column indicates the project being run and the date and time that the project was compiled. A tool tip is also available on each process which indicates how long the process has been running.

## See Also

- [Start a Process](#)
- [Stop a Process](#)
- [Restart a Process](#)
- [Cancel a Process](#)
- [Reload a Process](#)

## Start a Process

If a Plant SCADA process is not running, has stopped or is not responding, you can use Runtime Manager to start the process.

---

**Note:** If a new version of a project has been deployed to the computer where Runtime Manager is operating, you will not be able to restart a single process. You will need to restart all processes.

### To start a process:

- Right-click the process to start and choose **Start** from the shortcut menu.

Runtime Manager will start the selected process. The verbose startup message appears for that process until the startup completes, and the status for the process is set to "Running".

---

**Note:** By default, a process that has stopped abnormally is restarted automatically. This behavior is controlled by the Citect.ini parameter [\[RuntimeManager\]AutoRestartTrigger](#).

If all the processes are in a stopped state, you can click the **Start All** button to start all processes.

## See Also

- [Stop a Process](#)
- [Restart a Process](#)
- [Cancel a Process](#)
- [Reload a Process](#)

## Stop a Process

If a Plant SCADA process is running, you can use Runtime Manager to stop the process.

### To stop a process:

- Right-click the process to stop and choose **Stop** from the menu that appears.  
Runtime Manager stops the Plant SCADA process. Its status is changed to "Stopped".

## See Also

- [Start a Process](#)
- [Restart a Process](#)
- [Cancel a Process](#)
- [Reload a Process](#)

## Restart a Process

You can restart a selected Plant SCADA process without interrupting the current state of other processes.

### To restart a process:

- Right-click the required process and choose **Restart** from the menu that appears.  
Runtime Manager will stop the process and start it again. The verbose startup messages appear, then the status for the process is set to "Running".

## See Also

- [Start a Process](#)
- [Stop a Process](#)
- [Cancel a Process](#)
- [Reload a Process](#)

## Cancel a Process

If a Plant SCADA process is currently starting, you can right-click on the process and choose **Cancel** from the menu that appears.

A **Cancel** button is also available on the Runtime Manager while a process is starting.

---

**Note:** If the Citect.ini parameter [\[Page\]StartUpCancel](#) is set to 0, the Cancel option will be unavailable in Runtime Manager.

---

## See Also

[Start a Process](#)

[Stop a Process](#)

[Restart a Process](#)

[Reload a Process](#)

## Reload a Process

You can reload the configuration for a server processes during runtime. This will reload any updated database files, allowing you to implement some project configuration changes without performing a full restart. For more details see the topic *Server Side Online Changes*.

### To reload a process:

- Right-click on the process and choose **Reload** from the shortcut menu.

The **Status** column will change to indicate that a reload is in progress. Once the reload is complete (successfully or otherwise), the **Message** column describes the result of the operation.

Message	Description
Reload could not start because another reload was being commenced.	Check that no reloads are currently active before you attempt the reload again.
The last reload was completed at {time}, on {date}.	Indicates the reload was successful.
Reload could not start because the requested components in this process are disabled.	All components in the selected process are disabled due to license restrictions or a Citect.ini file setting (for example, <a href="#">[Trend]Disable</a> is set to 1).
Server reload failed. Please check Citect syslog for more information.	View for the syslog.dat file to determine why the reload was unsuccessful.
Unknown error: {0}	{0} represents the error code returned by the system when a reload of the database files was attempted.

---

**Note:** It is recommended that the [ServerGetProperty](#) Cicode function be used with the *LibRDBMemTime* and *LibRDBDiskTime* properties to check if there is a change to the Cicode library before attempting a reload.

---

Following a reload please check the corresponding server's syslog.dat file for any reload messages. The Cicode changes will not be reloaded, therefore a restart may be more appropriate.

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

Restart the server process if a "Cicode library timestamp differs" error is detected. The library mismatch is indicated on the server in either the hardware alarm or the server's syslog.dat file.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** A message in the Syslog.dat file and hardware alarm of "Cicode library timestamp differs" (error code 454) will be raised if the Cicode library used by one or more server runtime databases is different from the one in memory. The timestamps will be different if the project has been fully recompiled (with or without Cicode modification), or if the project has been incrementally recompiled after any Cicode has been modified.

## **See Also**

[Start a Process](#)

[Stop a Process](#)

[Restart a Process](#)

[Cancel a Process](#)

## **Use the Kernel with a Selected Process**

If a Plant SCADA process is running, the Kernel window can be used to diagnose the runtime information for that process.

To enable the Kernel to be launched from Runtime Manager, you need to confirm following settings:

- The Citect.ini parameter [\[Debug\]Menu](#) needs to be set to 1. By default, this parameter is set to 0 which means the Kernel menu option will not be available.  
You can also use the Setup Wizard to configure this setting. Run the Setup Wizard, select **Custom** mode, and select the **Kernel on menu** option on the [Control Menu Security Configuration](#) page.
- Users who access the Kernel need to be assigned to a [Role](#) that has the **Kernel Access** property set to "Full Access" or "Read Only". A Read Only user can access the Kernel, but cannot perform some privileged commands like running Cicode.

**Note:** If you attempt to display the Kernel for a display client process from Runtime Manager, it will not succeed if no one is logged in or the current user is assigned to a role with **Kernel Access** set to "No Access". An error message will be displayed in this case.

### **To view the Kernel for a process:**

1. In Runtime Manager, right click on the process for which you would like to view the Kernel window.
2. Select **Kernel** from the context menu which appears.
3. The Kernel window for the selected process will appear and a login dialog will display. Enter a valid user name and password (see above).

## ⚠ WARNING

### UNINTENDED EQUIPMENT OPERATION

- Do not use the Kernel for normal Plant SCADA operation. The Kernel is only for diagnostics and debugging purposes.
- Configure your security so that only approved personnel can view or use the Kernel.
- Do not view or use the Kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of AVEVA Support.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Dump Kernel Data for a Running Process

You can dump Kernel data for a running process directly from the Runtime Manager.

1. Right-click the running process.
2. Select **Dump Kernel**.

When complete, the message "Kernel data has been dumped successfully" is displayed. Kernel data for only one process may be dumped at a time. Kernel data is dumped in non-verbose mode. This menu option can be disabled by setting the INI parameter [\[RuntimeManager\]AllowDumpKernel](#) to 0.

## See Also

[Display the Kernel Window](#)

## Hide Runtime Manager

The **Hide** button minimizes Runtime Manager to the notification area on the Windows™ task bar.



Runtime Manager does not stop running, the interface is just hidden until you require it again.

Clicking the task bar icon restores Runtime Manager to the foreground.

## See Also

[Runtime Manager Interface](#)

## Shut Down Runtime Manager

All runtime processes need to be stopped before Runtime Manager can shut down.

### To shut down Runtime Manager:

- Click the **Shutdown All** button on the Plant SCADA Runtime Manager interface.

Runtime Manager stops all processes that are currently running, not responding, or stopped. It displays verbose shutdown messages for each process.

---

**Note:** When Log Off | Reboot | PowerOff options are executed from Cicode during the shutdown of Plant SCADA, Runtime Manager will appear and display the verbose shutdown messages for each instance. Then, when complete, it automatically shuts down Runtime Manager.

---

If the Citect.ini parameter [\[RuntimeManager\]ExitAfterShutdown](#) is set to 1 (its default value), Runtime Manager automatically closes when all processes have stopped.

---

**Note:** For security reasons the **Shutdown** button in Runtime Manager can be disabled. To do this, set the Citect.ini parameter [\[Debug\]Shutdown](#) to zero (0).

---

### See Also

[Runtime Manager Interface](#)

[Launch Runtime Manager from Plant SCADA Studio](#)

### Operate Runtime Manager in Service Mode

Plant SCADA can be run as a Windows™ service, allowing for unattended operation of system servers. The associated service, named "Plant SCADA Runtime Manager", is configured using the Microsoft Management Console. For more information, see the topic [Configure a System to Run as a Windows Service](#).

When Plant SCADA is running as a service, the [Runtime Manager](#) will operate in "Service Mode". This is indicated by the application title bar, which displays "Runtime Manager (Service Mode)".

---

**Note:** If you are running as part of a trusted network (for example, [\[Client\]PartOfTrustedNetwork](#) is set to 1), you will not be able to run your project if the current user is not assigned to the Configuration Users or Server Users security role (see [Security Roles](#)). Similarly, if you are running an /X client process as part of a trusted network ([\[Client\]AdditionalClientsArePartOfTrustedNetwork](#) is set to 1), the current user will also need to be assigned to one of these security roles.

---

When operating in Service Mode, the Runtime Manager will demonstrate the following functionality changes:

- The display client process will not appear in the list of processes.
- A process named "System Services" will appear in the list of processes. This process performs system functions for the service. If you stop the System Services process, it will cause the other Plant SCADA processes to close down as the system will no longer be able to detect a license.
- When the display client is manually shut down, it will not close Runtime Manager.
- After the display client is manually shut down, clicking the **Restart All** or **Start All** button (if they are available) will not start the display client. You can start the display client again from Plant SCADA Studio, or by launching Plant SCADA Runtime from Windows **Start** menu.
- Clicking the **Shutdown All** button will shut down all the managed processes that are currently running. When all the processes have stopped, you can manually close Runtime Manager via the close button in the top right-hand corner.

Running Plant SCADA as a service in single process mode is not recommended. If you run Plant SCADA in this condition, the following limitations will apply:

- Hardware alarms originating from the server process will not appear.
- The following Cicode functions will not be executed within the server process:
  - AlarmSumAppend
  - AlarmSumCommit
  - AlarmSumDelete
  - AlarmSumFind
  - AlarmSumFindExact
  - AlarmSumFirst
  - AlarmSumGet
  - AlarmSumLast
  - AlarmSumNext
  - AlarmSumPrev
  - AlarmSumSet
  - AlarmSumSplit
  - AlarmSumType
  - DriverInfo
  - IODeviceControl
  - IODeviceInfo
  - SPCAlarms
  - ServerInfoEx
  - TrnAddHistory
  - TrnDelHistory

---

**Note:** If you attempt to switch Plant SCADA between Service Mode and normal operation, you need to close down the display client and the Runtime Manager before you restart Plant SCADA in a different mode.

---

## Online Changes

Online changes refer to the ability to implement project configuration changes in a runtime system without the need to restart the processes that are currently running.

These types of changes fall into two categories:

- [Client Side Online Changes](#)
- [Server Side Online Changes](#).

## See Also

[Restarting the System Online](#)

## Client Side Online Changes

During runtime, Plant SCADA clients only require graphics, code and communications information to operate successfully. Other configuration information is deployed to an appropriate server.

This means the following configuration changes can be made to a project without the need to restart clients:

- I/O devices
- Tags
- Alarms
- Trends
- Reports
- Accumulators.

Following any configuration changes to these areas of functionality, you can compile and distribute a project as you normally do. However, despite the fact a client restart is not required, you will have to restart the following servers to implement the changes in the runtime system.

- I/O devices - require a restart of the I/O server
- Tags - require a restart of the I/O server
- Alarms - require a restart of the alarm server
- Trends - require a restart of the trends server
- Reports - require a restart of the reports server
- Accumulators (require a restart of the reports server).

---

**Note:** You are able to reload a page at runtime even if the associated Cicode library has changed. However, the hardware alarm "Cicode library timestamp differs" will be raised if the Cicode library that was used to create the page does not match the one that is in memory. Under these circumstances, it is recommended that you restart a client. See [Handling Cicode Changes during Runtime](#).

---

## See Also

[Server Side Online Changes](#)

## Handling Cicode Changes during Runtime

You are able to reload a page at runtime even if the associated Cicode library has changed. However, the system will compare the Cicode library that was used to create the page to the one that is in memory. If they do not match, the hardware alarm "Cicode library timestamp differs" is raised. The page is displayed, but the Cicode that is executed will be from the memory version of the Cicode library and not the latest compiled version.

A restart of the client is needed to update the Cicode library version to the latest compiled version.



**UNINTENDED EQUIPMENT OPERATION**

Restart the client process if the hardware alarm "Cicode library timestamp differs" is raised after a page is opened.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** A hardware alarm of "Cicode library timestamp differs" will be raised if the Cicode library used by a page has a different timestamp from the one in memory. The timestamps will be different if the project has been fully recompiled, the project has been incrementally recompiled after the page has been modified, or if the project has been incrementally recompiled after any Cicode has been modified.

It is recommended that the [ServerGetProperty](#) Cicode function be used with the *LibRDBMemTime* and *LibRDBDiskTime* properties to check if there is a change to the Cicode library before attempting a reload. Following a reload, it is recommended that you check the corresponding server's syslog.dat file for any reload messages.

## See Also

[Client Side Online Changes](#)

[PageDisplay](#)

[PageGoto](#)

## Server Side Online Changes

Records can be modified during runtime using Plant SCADA Studio, recompiled and the resulting database files updated. You can initiate a configuration reload for each server using the Runtime Manager or Cicode functions.

**Note:** It is recommended that the [ServerGetProperty](#) Cicode function be used with the *LibRDBMemTime* and *LibRDBDiskTime* properties to check if there is a change to the Cicode library before attempting a reload.

Following a reload please check the corresponding server's syslog.dat file for any reload messages. The Cicode changes will not be reloaded, therefore a restart may be more appropriate.

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

Restart the server process if a "Cicode library timestamp differs" error is detected. The library mismatch is indicated on the server in either the hardware alarm or the server's syslog.dat file.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** A message in the Syslog.dat file and hardware alarm of "Cicode library timestamp differs" (error code 454) will be raised if the Cicode library used by one or more server runtime databases is different from the one in memory. The timestamps will be different if the project has been fully recompiled (with or without Cicode modification), or if the project has been incrementally recompiled after any Cicode has been modified.

## See Also

[Restart an Alarms Server](#)

[Restart a Trends Server](#)

[Restart a Reports Server](#)

## Restart an Alarms Server

The following fields are reloaded on alarms servers:

- Digital alarms
- Time stamped alarms
- Analog alarms
- Advanced alarms
- Multi-digital alarms
- Time stamped digital alarms
- Time stamped analog alarms

The alarm category is not reloaded on the client side.

The following alarm category fields are used by the server and will be reloaded:

- ON Action
- OFF Action
- ACK Action
- Summary Device
- Log Device, ON, OFF and ACK log devices

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

On updating the Equipment Name or Equipment Item for an alarm, the update will not be reflected on the client side until the alarm servers have been restarted. The reload procedure will indicate a reload error on Runtime Manager. Reloadlog.dat and Syslog.dat of the server process will show the details of the unsupported changes. You can use this information to revert the changes, or restart all servers after deploying your updated project.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

The numbers of **added**, **modified** and **deleted** categories are recorded in the system log. Alarm categories are identified by their category numbers.

- **Added** — Indicates how many categories with new category numbers have been added to the alarm server.
- **Modified** — Indicates how many categories (of which the category number existed in the runtime before reloading) have been modified in the alarm server. If a user has defined a new category record, but its category number was used by an existing category, the category reload will count this situation as a modification of the category with this category number, rather than an addition. A category is shown as modified if any content in the category record is modified, including the case that only client side settings are modified. In this case, the reload does not change the state of this category in the memory, but it is still counted as one modified category.
- **Deleted** — Indicates how many categories have been deleted from the alarm server. A category is defined as deleted if none of the other categories has the same category number as this category. In the case that two

category records have the same category number, deleting the latter one in the table will instead increment the modify count, while delete the former one would not increment any count.

Two category records with the same category number are treated as one category in runtime, and the later one would be loaded.

When [\[Alarm\]EnableErrorLogging](#) is set to 1, an alarm server adds a syslog entry for errors from adding, modifying or deleting an alarm record or a category record. The errors can be due to category number out of range or category priority number out of range. Two new hardware alarms (520 and 523) have been introduced for those errors. See Cicode and General Errors for more details.

Reloading an alarm server will affect alarm property tags and may make them invalid (when an alarm is deleted) or make them available (when adding alarms).

#### To reload an alarm server:

1. Compile the project on main and standby servers.
2. Reload the standby alarm server using Cicode function [ServerReload](#) or the Runtime Manager. At this point the standby server loads the new RDB and indicates that the server is in "Prepared" state for synchronization. The database is not updated at this stage.
3. Reload the main alarm server. At this point the main server loads the new RDB and then updates the configuration data in its database. The database on the main alarm server then synchronizes with the database on the standby server.

Use the [\[Alarm\]UseConfigLimits](#) INI parameter to set whether on reload or restart the alarm property values are retrieved from the RDB or the alarm server side database.

## See Also

[Server Side Online Changes](#)

### Restart a Trends Server

The following data will be reloaded on a trend server when adding, removing and modifying the records of:

- Trend tags
- SPC tags.

Modifying the following archive properties will put a record in an error state and no acquisition will be done for corresponding records:

- Sample Period
- Type (from Periodic to non Periodic or vice versa)
- Filename
- Storage Method
- No. Files
- Time
- Period.

A hardware alarm, a syslog entry and a reload error message will be raised. The history files will not be deleted. For periodic to periodic Type changes, reload will not generate an error.

If you want to have a valid archive for the modified record, you need to delete the old history files (after archiving manually) and then trigger the reload again after forcibly compiling the Trend configuration. This is necessary because the trend system doesn't reload the trend rdb file if the compile time of in-memory file is same to the on disk file. Therefore changes to an existing trend record's archive properties is a two staged reload.

Due to reload the display clients will need to request the latest configuration data for trend records. The Process Analyst will periodically refresh configuration data from the trend server. The Trend server tag browsing is optimized to handle the more frequent requests.

For legacy trends including SPC, configuration data will be refreshed from the trend server when a page update is triggered.

## See Also

[Impact of a Server Reload on Historical Records](#)

[Server Side Online Changes](#)

## Restart a Reports Server

The following fields will be reloaded on a Report Server:

- Reports
- Accumulators

## See Also

[Impact of a Server Reload on Historical Records](#)

[Server Side Online Changes](#)

## Impact of a Server Reload on Historical Records

The following sections detail the effects on the history of reloaded Alarm, Trend and Report records. This is dependent on which fields have changed.

- A 'Yes' in the **New Alarm** column indicates changes to that field will make a new alarm and delete the old one and therefore the history is not kept.
- A 'Yes' in the **Invalid History** column means changes to that field will make the history of that record invalid.

## Alarm Servers

### Common Alarm fields

Field Name	New Alarm	Invalid History
Alarm Tag	Yes	No
Cluster Name	Yes	No
Alarm Description	No	No
Alarm Category	No	No
Help	No	No
Delay	No	No
Comment	No	No
Privilege	No	No
Area	No	No
Custom 1..8	No	No
Paging	No	No
Paging Group	No	No

**Analog and Time Stamped Analog Alarm fields**

Field Name	New Alarm	Invalid History
Variable Tag	No	Yes
High High	No	Yes
High	No	Yes
Low Low	No	Yes
Low	No	Yes
SetPoint	No	Yes
Deviation	No	Yes
High High Delay	No	No
High Delay	No	No
Low Low Delay	No	No
Low Delay	No	No
Deviation Delay	No	No

Field Name	New Alarm	Invalid History
Rate	No	Yes
Deadband	No	Yes
Format	No	No

**Digital and Time Stamped Digital Alarm fields**

Field Name	New Alarm	Invalid History
Variable Tag A	No	Yes
Variable Tag B	No	Yes

**Time Stamped Alarm fields**

Field Name	New Alarm	Invalid History
Variable Tag	No	Yes
Timer	No	Yes

**Advanced Alarm fields**

Field Name	New Alarm	Invalid History
Expression	No	Yes

**Multi-Digital Alarm fields**

Field Name	New Alarm	Invalid History
Variable Tag A	No	Yes
Variable Tag B	No	Yes
Variable Tag C	No	Yes
Realarm	No	Yes
On Function	No	Yes
Off Function	No	Yes
State	No	Yes
State Description	No	Yes

**Trend Servers**

Changes to the following properties of trends make existing trend files invalid and result in the creation of a new

trend file (renaming the old one for backup):

- Sample Period
- Type
- Storage Method
- No of Files
- Time (File)
- Period (File)

Changing the scale of a tag will make the trend history invalid.

Effect of reload on a modified Trend:

Field Name	New Alarm	Invalid History
Trend Tag Name	Yes	No
Cluster Name	Yes	No
Expression	No	Yes
Trigger	No	Yes
Comment	No	No
Privilege	No	No
Area	No	No
Eng Units	No	No
Format	No	No

## Report Servers

Effect of reload on a modified Report:

Field Name	New Alarm	Invalid History
Report Name	Yes	No
Cluster Name	Yes	No
Time	No	Yes
Period	No	Yes
Trigger	No	No
Report File Format	No	No

Field Name	New Alarm	Invalid History
Output Device	No	No
Privilege	No	No
Area	No	No

## Accumulators

Effect of reload on a modified Accumulator:

Field Name	New Alarm	Invalid History
Name	Yes	No
Cluster Name	Yes	No
Trigger	No	Yes
Run Time	No	Yes
No. of Starts	No	No
Totaliser Inc	No	No
Totaliser	No	No
Comment	No	No
Privilege	No	No
Area	Yes	No

## Restarting the System Online

With the online restart facility, you can change your project configuration and examine the results in the runtime system without having to shut down Plant SCADA. You can update your system while it is running.

---

**Note:** If you've configured Plant SCADA to use multiprocessor support, it is recommended that you use the Runtime Manager instead of the restart facility. The runtime manager allows you to restart different processes individually, whereas the restart facility will restart every Plant SCADA processes on the machine.

---

The time taken for the system changeover depends on the size of the project and the extent of the changes to the project:

- If you only change graphics pages, Plant SCADA does a *partial restart* (changing only the pages in the runtime system). Changeover is instantaneous.
- If you change any databases (for example, add a new alarm tag, trend tag, or Cicode function), Plant SCADA does a *full restart* to run the updated project.

## See Also

[Restarting a Networked System Online](#)

### Restarting a Networked System Online

If you are using Plant SCADA on a network, and would like to restart Plant SCADA without undue impact on the control of the plant, use a structured restart procedure. You can use any Plant SCADA computer on the network to initiate the online restart.

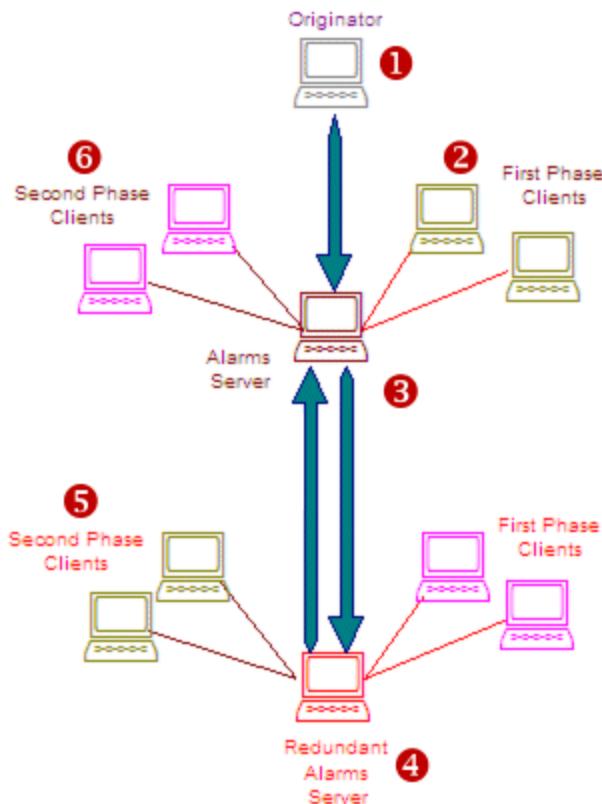
#### To phase shutdowns:

1. Set the citect.ini parameter [\[Shutdown\]Phase](#) on each client to either 1 or 2 to identify the phase of the client.
2. Issue a [Shutdown](#) command from the server.

When the Shutdown() command is issued, clients with phase set to "1" are shutdown first. When the first phase 1 client has reconnected to the server, clients with phase set to "2" are shutdown.

Plant SCADA automatically manages the online restart in the following sequence:

1. The Originator issues the Shutdown("Everybody") command.
2. The Alarm Server that the originator is connected to shuts down its first phase clients.
3. The Alarm Server that the originator is connected to advises the other Alarm Server then shuts itself down.
4. The second Alarm Server shuts down its first phase clients and waits for the other Alarm Server to restart (running the new project).
5. When the first Alarm Server is back on line, the second Alarm Server shuts down its second phase clients and then shuts itself down.
6. The first Alarm Server restarts its first phase clients and shuts down and restarts its second phase clients.



**Note:** If you've configured Plant SCADA to use multiprocessor support, it is recommended that you use the Runtime Manager instead of the restart facility. The runtime manager allows you to restart different processes individually, whereas the restart facility will restart every Plant SCADA process on the machine.

## See Also

[\[Shutdown\]Phase](#)  
[\[Shutdown\]NetworkIgnore](#)  
[Shutdown](#)

## Using Multiple Projects

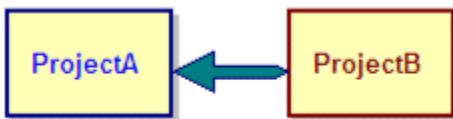
The most effective method of using the online restart facility is to use two projects. The first project becomes the current runtime system while the second project is in the development stage. You can manage both projects as follows:



Project A is currently the runtime system while Project B is under development.



When Project B is complete, you can use the online restart facility to change the runtime system to Project B.



You can then copy Project B to Project A for further development.

---

**Note:** If you've configured Plant SCADA to use multiprocessor support, it is recommended that you use the Runtime Manager instead of the restart facility. The runtime manager allows you to restart different processes individually, whereas the restart facility will restart every Plant SCADA process on the machine.

---

### Initiating the Online Restart

To initiate the online restart, the originator (any Plant SCADA computer on the network) issues a shutdown command with the Shutdown function, for example:

```
Shutdown("Everybody", "MyProject", 2);
```

Where possible, balance clients across both phases of the shutdown. The [Shutdown]Phase parameter defines the phase to which each Plant SCADA computer responds.

You can exclude selected computers (for example I/O Servers) from the online restart procedure with the [Shutdown]NetworkIgnore parameter.

For security, you can prevent selected computers from initiating the online restart procedure with the [Shutdown]NetworkStart parameter.

---

**Note:** If you've configured Plant SCADA to use multiprocessor support, it is recommended that you use the Runtime Manager instead of the restart facility. The runtime manager allows you to restart different processes individually, whereas the restart facility will restart every Plant SCADA process on the machine.

---

### Using a Callback Function

You can use a callback function (with the OnEvent function) to perform housekeeping tasks before the system shuts down. You would normally call OnEvent in the main startup function (defined with the [Code]Startup parameter). Each time a Shutdown() call is made, the callback function is run.

```
/* A user shutdown procedure. */
INT
FUNCTION
MyStartupFunction()
  ...
  OnEvent(25, MyShutdown);
  ...
END
INT
FUNCTION
MyShutdown()
```

```

STRING sPath;
// Perform housekeeping tasks
...
sPath = ProjectCurrentGet();
If sPath = "ProjectA" Then
    ProjectSet("ProjectB");
Else
    ProjectSet("ProjectA");
END
Shutdown("Everybody", sPath, 2);
END

```

## Firewall Settings and Plant SCADA

Plant SCADA networking and redundancy needs runtime to communicate through Windows Firewall. This means your Windows Firewall settings will need to be adjusted so that Plant SCADA and its components are included in the list of authorized programs.

The Plant SCADA installer can automatically adjust the required settings for you. If Windows Firewall is operational on a computer when installation occurs, the installer will display a **Firewall** page. To allow Plant SCADA to adjust these setting for you, select **Yes, please modify Windows Firewall settings**.

This will create Inbound Rules for each of the following components.

Name	Program	Local Port
Citect SCADA Runtime (x64)	C:\Program Files (x86)\AVEVA Plant SCADA\Bin\Bin (x64)\Citect.exe	All ports
Plant SCADA Runtime	C:\Program Files (x86)\AVEVA Plant SCADA\Bin\Citect32.exe	All ports
Configurator	C:\Program Files (x86)\Common Files\ArchestrA\configurator.exe	All ports
Configurator 443	All programs	443
Configurator 80	All programs	80
LicenseServerPort	All programs	55555
LicenseServerAgentPort	All programs	59200

**Note:** Microsoft Windows® distinguishes between **Public**, **Home** and **Work** networks. Each network has its own firewall profile. The Plant SCADA installer will automatically modify the Windows Firewall settings for the active network profile during installation. If you plan to change your network settings, you will need to manually modify the firewall settings for each profile within Windows.

If during installation you select **No, I will modify Windows Firewall settings later**, you will need to manually configure an Inbound Rule for each of the components listed above. You should confirm if a rule already exists, as Inbound Rules are also created under the following circumstances:

- Inbound Rules for the License Server ports are created when installation of License Server occurs.

- Inbound Rules for Configurator will be created when a System Management Server is selected within Configurator.
- Inbound Rules are created for Runtime when it is launched.

**Note:** If you postpone modifying the Windows Firewall settings during installation, when you launch Runtime for the first time a Windows Security Alert dialog will appear. When this occurs, click the **Allow access** button. The Inbound Rules for Plant SCADA Runtime will be updated and runtime will be launched.

If required, you can also manually modify the Inbound Rules created by Plant SCADA. For example, if the default "All ports" setting that is applied to Citect SCADA Runtime does not comply with your security requirements, you can manually set the **Local Port** property to a specific port.

You should also check that the required **Ports** are correctly configured.

## Inbound Rule settings

Inbound Rules are configured using Windows Firewall Advanced Settings (refer to Microsoft Windows documentation for further information about configuring Inbound Rules).

For each Inbound Rule, you need to configure the following properties:

### General Properties

- **Name** – see table above.
- **Enabled** – Yes.
- **Action** – Allow the connection.

### Programs and Services Properties

- **Programs** – see table above.

### Advanced Properties

- **Profile** – for Plant SCADA Runtime select **Domain**. For Configurator and License Server components select **Domain, Private and Public**.

### Protocols and Ports Properties

- **Protocol Type** – TCP (or UDP where required).
- **Local Port** – see table above.
- **Remote Port** – All Ports.

## Ports

You should confirm that the following ports are open (if they are required).

Port	Description
80	The default HTTP port used for web port sharing.
443	The default HTTPS port used for web port sharing.

Port	Description
808	Net.TCP Port Sharing Service. This is used by an Industrial Graphics Server or OPC UA Server.
1900	SSDP port for announcing the System Management Server.
2073	If a client acts as CTAPI server then port 2073 has to be added to the inbound rules.
2088	Time synchronization port. This service is not enabled by default.
3073	CTAPI (encrypted connections).
48031	The default for an OPC UA Server's Endpoint Connection setting.

On computers running a server process, you also need to open the server ports as inbound rules. See [Server Processes](#).

---

**Note:** If the alarm server is not functional, or the hardware alarm "No server could be found" is raised for a report server, trend server, I/O server or alarm server, you should check the firewall settings to see if communication between runtime and the network is blocked.

---

## Startup and Runtime Configuration

You can specify a Cicode function to execute automatically when Plant SCADA starts up. This Cicode gets executed as soon as the Cicode system comes online. use the Setup Wizard to specify the name of the startup function.

You can also run a report on startup. Plant SCADA searches for a default report called "Startup" when it starts up. If you have configured a report called "Startup", it is run automatically. You can change the name of the startup report (or disable it altogether) by using the Setup Wizard.

You can customize many elements of Plant SCADA's runtime and startup behavior. Only The Setup Wizard is necessary, but you can also use parameters for more control.

## Server Redirection Using Address Forwarding

Plant SCADA allows you to temporarily transfer the communications of a server to another computer to assist with hardware maintenance and system analysis.

By including an address forwarding section within a citect.ini file, you can override a server's project-configured address, redirecting network traffic to a different address and port.

For example, if you would like to test a new server before adding it to a project configuration, you could make the necessary address forwarding adjustments within the citect.ini file of a single client to direct it to the new hardware. The server could be tested within the context of a system without having to recompile the project.

You need to make the necessary adjustments within the citect.ini file for every computer that is likely to be

impacted by address forwarding.

For example, if you have a computer configured to run two I/O servers, and you would like to temporarily redirect one of them, then both server machines and every connecting client need to have the necessary address forwarding adjustments made to their citect.ini files, so that the servers are aware of each other.

If your system is configured to support network redundancy, a server may have multiple network cards. If this is the case, you need to consider the following when configuring a server redirection:

- Make sure the IP address configured for forwarding is the one that is currently being used by the destination server.
- If a hostname is used for an address forwarding target, make sure it is mapped to the IP address that is currently being used by the destination server. This mapping can be defined locally (via a hosts file) or globally (via DNS services).
- Make sure static IP addresses are used, not dynamic addresses.

Address forwarding is implemented using the following syntax within a citect.ini file:

```
[AddressForwarding]  
<ClusterName>.<ServerName>=<IPv4 ipaddress>:<port>
```

Or:

```
[AddressForwarding]  
<ClusterName>.<ServerName>=<IPv6 ipaddress>:<port>
```

Address forwarding is only interpreted and used during startup of Plant SCADA Runtime. It is recommended (but not necessary) that the Setup Wizard is run before running up a project to confirm your changes.

## WARNING

### UNINTENDED EQUIPMENT OPERATION

- Do not use address forwarding as the primary mechanism for specifying client/server communications. It bypasses the normal configuration checks enforced by the compiler.
- Carefully track any adjustments you make to the citect.ini file, as they need to be manually corrected once redirection is no longer necessary.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Virtualization Host Support

You can run components of your Plant SCADA system in a virtual environment.

The following virtualization environments are supported:

- Microsoft Hyper-V: based on the version of Windows
- VMware Workstation: basic virtualization without High Availability and Disaster Recovery
- VMware vSphere

For further information on virtualization, please refer to the [AVEVA™ Knowledge & Support Center](#).

## Anti-virus Software Setup

### **WARNING**

#### **SYSTEM PERFORMANCE DEGRADATION**

The "on access" scan in anti-virus products can lock files used by Plant SCADA, usually having the effect of slowing Plant SCADA down whilst it waits for the scan of that file to finish.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

### **CAUTION**

#### **INOPERABLE SYSTEM OR LOSS OF DATA**

In some extreme cases, anti-virus software may (incorrectly) detect certain patterns within data files as being viruses. Depending on the anti-virus configuration, this may result in files being relocated or deleted, resulting in data being lost or the system being inoperable.

**Failure to follow these instructions can result in injury or equipment damage.**

It is recommended that the following directories are excluded from scanning by any anti-virus products:

- Program Files installation directory (including files and sub directories)
- Data and Logs directories
- Any alarm server archive paths

The above exclusions are recommended for "on access" or "real time" scans that run continuously and scan each file that is read from or written to. Scheduled scans, which are usually daily or once a week, should also have these exclusions added with the exception of the Program Files exclusion.

## Using an Alternative INI File

When Plant SCADA starts up, it reads default values from the Citect.ini configuration file. Plant SCADA expects this file to exist in the config folder in the User and Data directory selected during installation. If the file is not found in this location, it will not search elsewhere and will instead display an error.

If you need to store your INI file elsewhere, specify the path to it on the command line when starting citect32.exe and CitectIDE.exe.

If you are running multiple projects, you can specify an alternative Citect.ini file for each project. The new INI file name needs to be passed to the Plant SCADA system applications (CitectIDE.exe and Citect32.exe) as a command line argument.

#### **To specify an alternative INI file for Plant SCADA:**

1. Click the icon that you use to start Plant SCADA Studio or Runtime.
2. Right-click to display the shortcut menu and select **Properties**.
3. Click the **Shortcut** tab.
4. Add the name of the INI file to the command line property of the appropriate icon using the -i option. For

example, to start Plant SCADA using the initialization file "my.ini", enter the following line:

```
"%PROGRAMFILES(X86)%\AVEVA Plant SCADA\bin\citect32.exe" -i"c:\citect\user\myproj\MY.INI"
```

**Note:** The Setup Wizard uses the same INI as specified for Plant SCADA Studio. You can specify different INI files for both the Runtime and Plant SCADA Studio programs. However, if you initiate Runtime from Plant SCADA Studio, it uses the INI that is specified for Plant SCADA Studio.

## Monitor Runtime

There are two types of runtime information you can use to identify and resolve operational problems:

- Hardware alarms
- Log files.

Hardware alarms are typically displayed on a dedicated alarm page to alert operators to current problems (see [Alarms](#)).

Log files are a record of time-stamped system data that can be analyzed to determine the cause of a problem (see [Log Files](#)).

This information can be analyzed directly to identify problems, or you can use the Plant SCADA Kernel to perform advanced debugging (see [Debug Runtime](#)).

## See Also

[Configure Logging](#)

[Adjust Logging During Runtime](#)

[The Crash Handler](#)

## Log Files

Log files are a record of time-stamped system data. Plant SCADA supports the log files described in the table below.

**Note:** Some common AVEVA components, such as Industrial Graphics and Enterprise Licensing, display log messages in the Log Viewer, an MMC snap-in that operates within AVEVA's Operations Control Management Console. For more information, see [Log Viewer](#).

Log file	Description
syslog.dat	<p>The syslog.dat file is the primary log file for Plant SCADA. It contains useful system information, from low-level driver traffic and Kernel messages, to user defined messages. Trace options (except some CTAPI traces) are sent to this file.</p> <p>The SysLog.dat file is a tab-delimited file with each field set to a minimum width. This enables the file to</p>

Log file	Description
	<p>be easily viewed in a text editor or imported into an application such as Excel. The file has the following format:</p> <ul style="list-style-type: none"> <li>• Level – 5 chars</li> <li>• Category – 11 chars</li> <li>• Thread Id – 4 byte hex</li> <li>• Driver Name – 16 chars</li> <li>• Unit – 16 chars</li> <li>• Function – 50 chars</li> <li>• File – 30 chars</li> <li>• Line – 4 byte decimal</li> <li>• Message – 1024 chars</li> </ul> <p>Plant SCADA locks syslog.dat while running. However, you can still view it by using the 'SysLog' command in the Kernel.</p> <p>There is a single SysLog file per IOServer process for both the Drivers and the IOServer (in earlier versions of Plant SCADA there was 1 file for the IOServer and 1 file for the Drivers).</p>
tracelog.dat	<p>The tracelog.dat file contains managed code logging, mainly in relation to data subscriptions and updates. Field traces and requests to native drivers go to the syslog.dat or a specific driver log file.</p> <p>In traces in the PSIClient and CSAToPSI categories, Sent and Received Notations (indicated with "&lt;=" and "&gt;") will be added to indicate tag data flow on the client side.</p> <ul style="list-style-type: none"> <li>• "&gt;" indicates that the log entry is tracing a message that the client has sent to clusters.</li> <li>• "&lt;=" indicates that the log entry is trace a message that the client has received from an IO Server.</li> </ul> <p>In addition, the session name will be appended in the message to identify the I/O server which handled the tag request.</p>
debug.log	<p>This file contains information about a crash or other serious internal issues. If a crash occurs, it will identify the version and path of each DLL being used at the time. It can be used to confirm you have the right</p>

Log file	Description
	version of files.
kernel.dat	Kernel.dat contains a copy of the kernel screens. It has the "dumpkernel(0x8000)" mode added to it on a crash, and is also available via Cicode calls to "dumpkernel".
ipc.log	This log is used for CTAPI communication traffic.
<driver>.dat	Driver logs relating to the operation of a particular driver and are named accordingly. For example, the OPC driver is logged in 'OPC.dat'.
Params.dat	The Params.dat file is an historical record of non default SCADA parameters. For example, 2010/12/01-14:42:07.847 [Code] Threads= 128 Default= 64.
tracelog.RuntimeManager.dat	This file contains logs relating to the operation of the Runtime Manager.

Log files may have additional suffixes included in their name, for example, an archived log file will include a timestamp (see [Log File Time Stamping](#)).

If a system uses separate processes, a log file is appended with the component name. An example of this could be:

`syslog.IOServer.Cluster1.dat`

Plant SCADA log files are stored in the Windows Program Data directory (see [Log File Locations](#)).

---

**Note:** If Plant SCADA suffers an unexpected shut down, the Crash Handler will create a compressed file containing a number of log and data files that may be useful in determining the cause. See [The Crash Handler](#).

## See Also

[Configure Logging](#)

[Perform Log File Calculations](#)

## Log File Time Stamping

Plant SCADA log file entries use the following timestamp format:

`yyyy-mm-dd<SPACE>HH:mm:ss.fff<TAB>TZD`

Where:

- `yyyy` = year (for example: 2008, 2009)
- `mm` = month (for example: 01, 05, 12)
- `dd` = date (for example: 01, 02, 31)
- `HH` = hour of the day (using 24 hour format, for example: 08, 13, 22)

- *mm* = minute (for example: 00, 02, 59)
- *ss* = seconds (for example: 00, 02, 20, 59)
- *fff* = milliseconds (for example: 000, 123, 999)
- *TZD* = local time offset from the UTC (for example: +10:00, -09:00)

---

**Note:** Time-stamps are always in local time.

---

For example:

2009-06-03 11:19:33.249 +01:00

## See Also

[Configure Logging](#)

## Log File Locations

The Plant SCADA log files are located in the Windows Program Data directory:

%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Logs

Driver log files will also appear in the Logs directory in a folder named after the particular driver.

## See Also

[Configure Logging](#)

## Perform Log File Calculations

If you wish to perform calculations on the time and dates recorded in Plant SCADA [Log Files](#), this can be accomplished by opening the files in Microsoft Excel.

Open the file in Excel by selecting settings in the Text Import wizard such that the date and time fields are NOT split into separate cells. Once opened in Excel, select the column containing the date and time and format the cells in the column with a custom format of yyyy-mm-dd hh:mm:ss.000. This will allow Excel to correctly parse the date/time, and store the value in days, as the number of days (and fractions of a day) since 1900-01-01 00:00:00.000 (this date/time has a value of 1).

Calculations between date/time values can be simply performed. However, it needs to be remembered that, since Excel treats times as days, if the difference in milliseconds between two date/times is necessary, the result of the difference calculation needs to be multiplied by (24\*60\*60\*1000).

## Parsing the UTC offset

Using the UTC offset in Excel is a little more complicated, as Excel will not accept negative times. Therefore format the UTC offset cell/s as a text string, and write a formula to convert it into a fraction of a day. For example, if the UTC offset is held in cell B4, the formula is:

=LEFT(B4,1)&TIME(MID(B4,2,2),RIGHT(B4,2),0)

This will convert the UTC offset into a correctly signed numerical value expressed as a fraction of one day.

For example:

- "+09:00" is converted to "+0.375"
- "-04:30" is converted to "-0.1875"

This UTC offset value can be subtracted from the local time to derive the UTC time.

UTC offset could be used with merge logs from two different time zones. In this case you may wish to make both logs based on UTC time. For example:

11:00:00.000 +01:00 is 10:00:00.000 in UTC (minus 1 hour)

09:01:00.000 -01:00 is 10:01:00.000 in UTC (plus 1 hour)

## See Also

[Configure Logging](#)

## Configure Logging

You can make adjustments to the way Plant SCADA logs data using Citect.ini parameters. This includes the ability to filter logs according to priority, category or severity.

## Configure syslog.dat

The syslog.dat file is the primary log file for Plant SCADA (see [Log Files](#)). The following list represents the basic set of parameters you can use to configure logging to syslog.dat.

- [\[Alarm\]EnableErrorLogging](#) — Enables or disables the logging of operational errors to syslog.dat.
- [\[Alarm\]EnableStateLogging](#) — Enables or disables the logging of major alarm system state changes to syslog.dat.
- [\[General\]Verbose](#) — enables a startup dialog box which shows the opening runtime databases and other information.
- [\[General\]VerboseToSysLog](#) — sends additional startup information generated by [General]Verbose to syslog.dat.

The syslog.dat file is restricted in size (to 2000 kb by default). When it reaches its size limit, Plant SCADA renames it "syslog.bak", and starts a new syslog.dat.

You can make this size restriction larger or smaller by using the [\[Debug\]SysLogSize](#) parameter. For example, the following lines in the Citect.ini will set the syslog.dat size to 30 Mb:

```
[DEBUG]
SysLogSize=30000
```

If you want to archive unlimited system log files, you can set [\[Debug\]SysLogArchive](#) to 1. This adds a time stamp to the file name.

## Configure tracelog.dat

The tracelog.dat file contains managed code logging, mainly in relation to data subscriptions and updates (see [Log Files](#)).

The following list represents the basic set of parameters you can use to configure logging to tracelog.dat.

- [\[Debug\]EnableLogging](#) — enables or disables logging to the tracelog.dat file.
- [\[Debug\]Priority](#) — allows you to filter messages logged to the tracelog.dat file according to their priority.
- [\[Debug\]SeverityFilter](#) — allows you to filter messages logged to the tracelog.dat file according to their severity.
- [\[Debug\]SeverityFilterMode](#) — indicates whether messages declared by [\[Debug\]SeverityFilter](#) will be included or excluded from the log.
- [\[Debug\]CategoryFilter](#) — allows you to filter messages logged to the tracelog.dat file by component category.
- [\[Debug\]CategoryFilterMode](#) — indicates whether messages declared by [\[Debug\]CategoryFilter](#) will be included or excluded from the log.

---

**Note:** By default, the setting for [\[Debug\]CategoryFilterMode](#) will exclude the categories specified by [\[Debug\]CategoryFilter](#). Be aware that this behavior is different to [\[Debug\]SeverityFilterMode](#), which includes the declared severities by default.

---

Archiving of the tracelog.dat file is enabled or disabled by the parameter [\[Debug\]ArchiveFiles](#). Archiving occurs once the maximum size specified by [\[Debug\]MaximumFileSize](#) is reached.

A full list of available parameters are listed on the Logging Parameters page of the Computer Setup Editor. From here, you can learn how each parameter will impact system logging, and make adjustments directly into the local Citect.ini file.

Some of the available logging parameters can be updated while your Plant SCADA system is running. For more information, see [Adjust Logging During Runtime](#).

---

**Note:** Logging can cause a strain on system resources. When configuring logging, be aware of the potential impact on normal operation. For example, a large number of traces can affect CPU performance, while log file archiving may use up disk space.

---

## See Also

[Log Files](#)

[The Crash Handler](#)

## Adjust Logging During Runtime

You can modify and query logging settings during runtime using the following Cicode functions:

- [SetLogging](#) - allows you to make changes to the current logging configuration without the need to restart a system that is already running. An optional parameter can be used to persist the settings to the INI file.
- [GetLogging](#) - allows you to view current logging settings.

The parameters you can modify and query include the following:

- [\[CtAPI\]Debug](#)
- [\[Debug\]DriverTrace](#)
- [\[Debug\]SysLogSize](#)

- [Debug]EnableLogging
- [Debug]Priority
- [Debug]CategoryFilterMode
- [Debug]CategoryFilter
- [Debug]SeverityFilterMode
- [Debug]SeverityFilter
- [Debug]LogShutdown
- [Debug]DebugAllTrans
- [IOServer]RedundancyDebug
- [General]Verbose
- [General]VerboseToSysLog

There is also a subset of the logging related INI parameters that can be modified online within the Citect.ini file, as the system will read their values periodically or on demand. The parameters you can modify in this way include the following:

- [PubSub]LogLevel
- [PubSub]LogDevice
- [General]ShowDriverError
- [Debug]SysLogArchive
- [Dial]Debug
- [Trend]TrendDebug

The parameters that support this functionality are identified on the Logging Settings page of the Computer Setup Editor.

## See Also

- [Log Files](#)
- [The Crash Handler](#)

## The Crash Handler

The Crash Handler can be used to help determine the cause of an unexpected program shut down.

If Plant SCADA suffers an unexpected shut down, the Crash Handler will create a compressed file containing a number of log and data files that may be useful in determining the cause. These files include:

- syslog.dat
- Citect.ini
- tracelog.dat
- kernel.dat
- params.dat

- reloadlog.dat
- debug.log
- user.dmp (see below)

In a multi-process system, these files may use extended names, for example:

syslog.IOServer.Cluster1.dat

You can configure Plant SCADA to save the Crash Handler zip file to the local Logs directory. If an unexpected shut down occurs, the path to the saved zip file will be recorded in the syslog.dat.

These features are not enabled by default; to enable and configure the Crash Handler you need to set the [CrashHandler Parameters](#).

---

**Note:** AVEVA may use the information included in an unexpected shutdown email to help resolve problems in future releases, but it cannot reply or follow up a problem with users directly. To discuss these, contact Technical Support.

---

## Configure User.dmp

User.dmp is a configurable file that logs exception details. When an unexpected shut down occurs, it captures objects and their variables in memory in a process referred to as a "mini-dump". This dump file can be analyzed by Technical Support.

The information included in this process is adjustable via the parameter [\[Debug\]MiniDumpType](#). It supports a "light" mini-dump (with local stack information for unmanaged code) through to a "heavy" dump (the default setting) which includes accessible memory for a process and thread information. You can also switch the mini-dump off.

To reduce the amount of disk space used by a dump file, it is compressed after it is created (along with other log files) into one of the following zip files:

- Citect32Exception\_[timestamp].zip (32-bit process)
- CitectException\_[timestamp].zip (64-bit process)

To help manage the amount of disk space used by this process, you can adjust the number of zip files that are stored using [\[Debug\]MaxMiniDumps](#).

You can also raise a hardware alarm to warn when disk space on the drive where these files are stored falls below a specified minimum amount using [\[Debug\]LogsDriveMinimumFreeSpace](#).

## See Also

[Log Files](#)

## Debug Runtime

Debugging Plant SCADA's runtime system involves two processes:

- Gathering information about Runtime
- Analyzing logged data to identify problems.

The information gathered from your system will include:

- Hardware alarms
- Log files.

Hardware alarms are typically displayed on a dedicated alarm page to alert operators to current problems.

[Log Files](#) are a record of time-stamped system data that can be analyzed to determine the cause of a problem.

This information can be analyzed directly to identify problems, or you can use the Plant SCADA Kernel to perform advanced debugging.

[The Kernel](#) can perform low-level diagnostic and debugging operations, and runtime analysis of your Plant SCADA system. Use it to display low-level data structures, runtime databases, statistics, debug traces, network traffic, I/O device traffic and so on.

You can also call built-in Cicode function or user-written Cicode functions, and use the Cicode Profiler to view how your Cicode is performing.

---

**Note:** The process involved in debugging device communications is described in the *Communicating With I/O Devices* section of the help. See [Troubleshooting Device Communications](#).

---

## See Also

[System Tuning](#)

## The Kernel

You use the Plant SCADA kernel to perform low-level diagnostic and debugging operations, and for runtime analysis of your Plant SCADA system. Use it to display low-level data structures, runtime databases, statistics, debug traces, network traffic, I/O device traffic and so on. You can also call built-in Cicode functions or user-written Cicode functions.

The Kernel includes the following areas:

- **General** - presents statistics and information on the overall performance of Plant SCADA. For example, this page shows memory usage, summaries of protocol and I/O Device statistics, as well as CPU usage. Access this by using the Page General command.
- **Table** - displays information about Plant SCADA runtime data structures. This area is extensive and is initially difficult to navigate. However, Page Table Stats is insightful. Access this by using the Page Table command.
- **Drivers** - displays statistics and information about the individual protocols running on the I/O Server. Each individual port has its own page of information. Access this by using the Page Driver command.
- **I/O Devices** - similar to the driver information, this shows specific statistics and information about each I/O Device. Access this by using the Page Unit command.

---

**Note:** You need to restrict access to the Kernel: anyone using the Kernel has total control of Plant SCADA (and subsequently your plant and equipment).

---

## ⚠ WARNING

### UNINTENDED EQUIPMENT OPERATION

- Do not use the kernel for normal Plant SCADA operation. The kernel is only for diagnostics and debugging purposes.
- Configure your security so that only approved personnel can view or use the kernel.
- Do not view or use the kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

### Access to Cicode and Cache commands

So that there is no unauthorized use of Cicode and Cache commands, users who access the Kernel need to be assigned to a **Role** that has the **Kernel Access** property set to "Full Access" otherwise they will not be able to execute these commands.

**Note:** If you change the **Kernel Access** setting for a Role and run the recompiled project, you will need to restart any server processes that you want to run the Kernel on.

See [Display the Kernel Window](#) for instructions on how to enable access to the Kernel.

### See Also

[Inside the Kernel](#)

[Kernel Commands](#)

### Display the Kernel Window

You can display the Kernel window in the following ways.

- Plant SCADA can provide a command option on the **Control** menu (see [Display the Kernel from the Control Menu](#)).
- You can define a runtime command to display the Kernel window when necessary (see [Define a Runtime Command](#)).
- You can launch the Kernel for a selected process from Runtime Manager (see [Use the Kernel with a Selected Process](#)).

To display the Kernel window for a display client, the user that is currently logged in now needs to be assigned to a **Role** that has the **Kernel Access** property set to "Full Access" or "Read Only". A Read Only user can access the Kernel, but cannot perform some privileged commands like running Cicode.

To display the Kernel window for a server process (including when [Running the System as a Windows Service](#)), you will need to log in using a dialog that is displayed when the Kernel window is launched.

You will need to enter a user that is assigned to a Role that has the **Kernel Access** property set to "Full Access" or "Read Only".

**Note:** After commissioning a system, it is recommended that you review the **Kernel Access** setting for each Role.

---

This will avoid accidental or unauthorized use of the Kernel.

---

## See Also

[Close the Kernel Window](#)

[The Kernel](#)

### Display the Kernel from the Control Menu

To add a **Kernel** option to the control menu of a display client, the Citect.ini parameter [\[Debug\]Menu](#) needs to be set to 1. By default, this parameter is set to zero (0).

You can also run the Setup Wizard, select **Custom** mode, and select the **Kernel on menu** option on the [Control Menu Security Configuration](#) page.

You can then display the Kernel window by selecting the Kernel option from the Control Menu (top-left corner) at runtime.

---

**Note:** The menu option will not be available if the logged in user is not assigned to a [Role](#) that has the **Kernel Access** property set to "Full Access" or "Read Only".

---

If you do not have a title bar displayed, you can access the Control Menu by pressing ALT-SPACE (verify that the **Alt-Space enabled** option is selected on the [Keyboard Security Configuration](#) page of the Setup Wizard).

## See Also

[Display the Kernel Window](#)

[Close the Kernel Window](#)

### Define a Runtime Command

To display the Kernel window, define a runtime command that calls the [DspKernel](#) function, passing 1 in the **iMode** argument. For example:

Command	DspKernel(1);
Comment	Displays (opens) the Kernel window

---

**Note:** The Kernel will not display if the logged in user is not assigned to a [Role](#) that has the **Kernel Access** property set to "Full Access" or "Read Only".

---

To close the Kernel window, call the DspKernel() function again, passing 0 in the **iMode** argument:

Command	DspKernel(0);
Comment	Closes the Kernel window

## See Also

[Display the Kernel Window](#)

## [Close the Kernel Window](#)

### **Close the Kernel Window**

You can close the Kernel window in the following ways:

- Select **File | Exit** from the main menu
- Select **Close** from the control menu
- Click the X button in the top-right corner of the main Kernel window.

### **See Also**

[Display the Kernel Window](#)

[Inside the Kernel](#)

### **Inside the Kernel**

When displayed, the Plant SCADA Kernel consists of a top-level window that contains several child windows:

- A **Main** window that displays diagnostic messages
- A **Shell** window for executing commands
- Additional windows opened in response to entered commands.

At startup, the **Main** window displays information about your Plant SCADA startup processes. The Main window also displays runtime system messages, providing a continuous operational history of your Plant SCADA system.

The **Shell** window contains a command line interface (similar to the Command prompt) where you can type in Kernel commands to perform a Kernel operation or to display other child windows.

---

**Note:** Access to the Cache and Cicode Kernel commands is restricted to specific users for security. For details, see the section *Access to Cicode and Cache Commands* in the topic [The Kernel](#).

If Plant SCADA runtime is configured to use multiple client or server processes, each process will have a separate Kernel window that displays specific system messages. For example, system messages related to a trend server process will appear in the Trend window in the Kernel.

For a default configuration, the Main window displays the following messages:

- **Initializing Sub Systems** - The primary parts of Plant SCADA are getting started.
- **Initializing Font System** - Creating fonts that have been defined within Plant SCADA. These are fonts used for displaying items such as alarms, and pre-V5.0 dynamic text.
- **Initializing Client System.**
- **Starting IO Server** - Only visible if the Plant SCADA computer is an I/O Server. If the computer is an I/O Server and this message does not display, the computer is improperly set up: run the Setup Wizard to check your configuration. **IO Server Started** - The server has started and is functioning correctly.
  - **Initializing I/O Server** - Starting to check what is necessary for the I/O Server to work, and initializing any cards that are necessary.
  - On a Client, these messages will be replaced with **Calling '<I/O Server>' Connected**.

- **Initializing Cicode System** - Cicode has been loaded into memory, and is prepared to run.
  - **Initializing com System** - Checking that ports and hardware are responding and functioning correctly.
  - **Initializing Request System** - The system that handles requests from the Client part of Plant SCADA to the Server parts of Plant SCADA.
  - **Initializing Trend Client System** - The Trend Client is slightly different than the normal client, so it needs separate initialization.
  - **Starting Trends Server** - You will only see these messages if the Plant SCADA computer is a Trends Server. If the computer is a Trends Server, and these messages do not display, most likely the computer is improperly set up. run the Setup Wizard to check your configuration.
    - **Trend Startup** - Plant SCADA is checking for the trend files, and making new ones if they can't be found.
    - **Initializing Trend Acq System** - Every trend you define has its own sample rate. Here Plant SCADA is setting up the system so it can poll the data at the correct rate for each trend pen.
- On a Client, these messages will be replaced with **Calling '<Trends Server>' Connected**.
- **Initializing Alarm System**.
  - **Loading Alarm Databases** - You will only see these messages if the Plant SCADA computer is an alarms server. This is loading alarm data into memory. If the computer is an alarms server, and these messages do not display, most likely the computer is improperly set up. run the Setup Wizard to check your configuration.
  - **Open Alarm Save File**

#### Loading Alarm Save File

**Alarm Save File Loaded** - Plant SCADA gets the alarm save file (that was created in the specified directory), and examines it in order to see the status of existing alarms. If you have a redundant alarms server, then this alarms server will interrogate the other instead of using the alarm save file - since the information on the other server will be newer than any file.

**Starting Alarm Processing** - The server is now processing (and serving) alarm data.

On a Client, these messages will be replaced with **Calling '<Alarm Server>' Connected**.

- **Initializing Report System**.
- **Starting Reports Server** - You will only see this message if the Plant SCADA computer is a reports server. If the computer is a reports server, and this message does not display, most likely the computer is improperly set up. run the Setup Wizard to check your configuration. On a Client, this message will be replaced with **Calling '<Reports Server>' Connected**.
- **Initializing Page System** - Plant SCADA will now display the Startup Page. At this time Plant SCADA will cover up the Kernel if it is displayed.
- **Initializing Functions** - Executing any Cicode functions that have been defined as running at start up.

The next line of information is the start up time and Plant SCADA version number.

- **Channel PORT# is Online**

#### Channel PORT# is Online

**Channel PORT# is Online** - You will only see these messages if the Plant SCADA computer is an I/O Server. These are messages telling you that any ports you have defined in the I/O Server have come online. If you get a messages saying that the port is not online, or could not be opened, check the configuration of your project. PORT# is the Port Name specified in the Ports form.

- **Unit 'UNIT#' Port PORT# is Online**

### Unit 'UNIT#' Port PORT# is Online

**Unit 'UNIT#' Port PORT# is Online** - Only visible if the Plant SCADA computer is an I/O Server. This indicates that the I/O Device with the Unit Number of UNIT# (as defined in the I/O Devices form), is connected to port PORT# .

- **Communication System Online** - Plant SCADA has completed startup operations and is now fully operational (running).

## See Also

[Use the Kernel to Commission a System](#)

### Use the Kernel to Commission a System

All systems in Plant SCADA start smoothly. When commissioning a system, check the Kernel. If any element does not initialize properly or reports errors at startup, your Plant SCADA system is not working correctly and requires investigation.

Common causes of startup errors are:

- Incorrect computer setup (usually solved by the Setup Wizard).
- Networking errors or bad hardware.
- Communication errors (usually just a configuration issue).

Use the Main window to check that your I/O devices come online correctly when starting. First the ports need to initialize, then the I/O device itself will come online. When a device does not initialize as expected, Plant SCADA displays a message such as "PLC not responding", "I/O Device Offline" or similar.

Some I/O devices might take two attempts to come online. If so, Plant SCADA waits (usually 30 seconds) and tries again. If the I/O device does not come online after the second attempt, check your configuration (at both ends) and cabling.

---

**Note:** The Kernel continues to report changes in the status of I/O devices to the Main window. This information might also be reported as alarms to the Hardware Alarms page.

---

## See Also

[Kernel Commands](#)

### The Cicode Profiler

The Cicode Profiler lets you gather information on how Cicode is performing in your project. This information includes the name of the function, the number of times the function was called (including from inside other functions), and the percentage of time the function took up during the profile period.

A high time and percentage value may indicate the Cicode function (and Cicode functions that may also call it) need to be reviewed and optimized.

---

**Note:** Contact Technical Support for assistance with optimizing your Cicode functions and those functions belonging to system include projects.

---

The information is presented in the following:

- A new kernel table, '[Page Table Profile](#)'.
- A CSV file. If in single process mode the file will be called "CicodeProfile.csv". For multi-process mode, the file name will contain the cluster and server name for server processes, for example, "CicodeProfile.Cluster1.TrendServer1.csv".

This file is created when the Profiler is disabled at the end of a profiling session (either through one of the methods for disabling or by shutting down Plant SCADA). This file can be found in the Log directory for the Plant SCADA installation.

To activate the Cicode Profiler, you can do one of the following:

1. Use the 'Profile' Kernel command).
2. Define type 4 when using Cicode function [CodeSetMode](#).
3. Set the INI parameter [\[Code\]ProfilerEnabled](#). Refer to the Parameters help for more information.

## See Also

[Kernel Commands](#)

## Kernel Commands

Commands are issued at a command line interface (similar to the Windows Command Prompt), from the Shell window. Some commands display their results in the main Kernel window; others open a child window for information display (or for further commands). You can open a maximum of eight windows at once (including the Main and Shell windows).

You can use several keyboard keys to scan and reuse commands from the command history, to speed up the issuing of Kernel commands. (The command history is a list of commands that you have previously issued). These keyboard keys are listed below:

Key	Description
Up arrow	Scans backward through the command history. (Commands are displayed in the command line.)
Down arrow	Scans forward through the command history. (Commands are displayed in the command line.)
F3	Puts the last command you issued in the command line.
Left arrow	Moves the cursor back one character at a time (in the command line).
Right arrow	Moves the cursor forward one character at a time (in the command line).
Delete	Deletes the character to the right of the cursor (in the command line).

Key	Description
Backspace	Deletes the character to the left of the cursor (in the command line).
Insert	Switches from over-strike mode to insert character mode (in the command line).

The table below describes the Kernel commands.

Command	Description
<a href="#">Cache</a>	Changes the cache timeout for each I/O Device.
<a href="#">Cicode</a>	Opens a child window that you can use to call Cicode functions.
<a href="#">Cls</a>	Clears text from the current window (the Shell window or Cicode window). You can also use "Cls Main" to clear text in the main window.
<a href="#">Ctapi</a>	Enables logging of server-side CtAPI debugging information.
<a href="#">Debug</a>	Enables the debugging of raw data transfer between Plant SCADA and a driver.
<a href="#">Diag Cicodestack</a>	Sends a report on the call stacks for all running Cicode to the relevant log file.
<a href="#">DriverTrace</a>	Controls the listing of driver control blocks (DCBs) between the I/O server and drivers.
<a href="#">Exit</a>	Closes a Cicode or Shell window.
<a href="#">Help</a>	Displays a list of the commands available in the Kernel.
<a href="#">Log</a>	Enables or disables the logging of I/O Device reads and writes.
<a href="#">Page General</a>	Displays general statistics information.
<a href="#">Page Driver</a>	Displays information about each driver in the Plant SCADA system.
<a href="#">Page Memory</a>	Displays the memory debug heap.
<a href="#">Page Queue</a>	Displays the queues that are active on the system.
<a href="#">Page RDB</a>	Displays information about Plant SCADA's Runtime Databases.

Command	Description
<a href="#">Page Table</a>	Displays information about Plant SCADA's internal data structures.
<a href="#">Page Unit</a>	Displays information about each I/O Device in the Plant SCADA system.
<a href="#">Pause</a>	Pauses debug output.
<a href="#">Profile</a>	Enter "profile 1" to enable the Cicode Profiler, or "profile 0" to disable it (see <a href="#">The Cicode Profiler</a> ).
<a href="#">Shell</a>	Opens a new command (shell) window. <b>Note:</b> The last shell window cannot be closed.
<a href="#">Stats</a>	Resets system statistics.
<a href="#">Tran</a>	Enables logging for tran activity messages in the syslog.dat file.
<a href="#">Write Log</a>	Sends a customized message to the relevant syslog.dat file.
<a href="#">Write Modules</a>	Lists all the loaded native process modules into the relevant debug.log file.

## See Also

[The Kernel](#)

[Display the Kernel Window](#)

## Cache

Changes the cache timeout for each I/O Device with which Plant SCADA is communicating.

---

**Note:** To use this command, you need to be running the Kernel under a user account that is assigned to a [Role](#) with the **Kernel Access** property set to "Full Access". For more information, see the section *Access to Cicode and Cache Commands* in the topic [The Kernel](#).

---

## Syntax

**Cache <I/O Device name> <Timeout>**

Where:

- **<I/O Device name>**

Any valid I/O Device defined in the project (using the I/O Devices form), or \* for I/O Devices.

- **<Timeout>**

Timeout in milliseconds, or 0 (zero) to disable timeout.

This command allows you to tune the cache timeout while the I/O Server is communicating with the I/O Devices. If you set the timeout to 0 (zero), the cache is disabled. If you specify a cache timeout for an I/O Device that has the cache disabled, the cache is enabled.

Any changes made to an I/O Device only apply while the I/O Server is running. If you restart the I/O Server, the cache timeout reverts to the value configured in the project. Once you have determined the optimum cache timeout, make the change persistent by setting the value in the I/O Devices form (for the particular I/O Device).

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

## Cicode

Opens a child window that you can use to call Cicode functions, on either a local or remote computer. Any built-in or user-written function can be called from this window.

---

**Note:** To use this command, you need to be running the Kernel under a user account that is assigned to a [Role](#) with the **Kernel Access** property set to "Full Access". For more information, see the section *Access to Cicode and Cache Commands* in the topic [The Kernel](#).

---

You do not need to specify all the optional arguments, as the optional arguments will be given runtime defaults (not compiler defaults) as follows:

Data type INT = 0

Data type REAL = 0.0

Data type STRING = ""

## Syntax

**Cicode [<Name>]**

Where:

- **<Name>**

Optionally the name of a Plant SCADA Server (for example, Alarms, Reports, Trends) or a client computer name.

If you enter the Cicode command with no Name argument, a local Cicode window is created. Cicode commands are executed on the local computer.

In a multi-cluster systems when connecting to a server, the following syntax needs to be specified:

<clustername>.<server>

For example, to connect to the Alarm Server on Cluster1, use:

**Cicode Cluster1.Alarm**

If you enter a server or computer name as the Name argument, you can create a Cicode window to the remote server or computer. Cicode commands entered in a remote Cicode window are executed on the remote

computer. For example, to create a Cicode window where commands execute on the Alarm Server, use:

`Cicode Alarm`

If you issue the command from a server, you can create a window to the "MyComputer" computer:

`Cicode MyComputer`

If the remote computer can be found, its name is displayed in the title of the Cicode window, otherwise a local window is created.

---

**Note:**

- Specifying a server name or computer name does not work in a Kernel window operating on a server running in service mode.
- You can only specify a computer name if you are issuing the command on a server. This function only supports client-to-server or server-to-client connections.

---

Each Cicode command is executed with its own Cicode task, so you can start tasks that take a long time to complete. The Cicode prompt returns immediately after the Cicode task has started and the task continues to run in the background. If the function is completed immediately, the return result of the function is displayed. If the function continues to run, the result is not displayed and cannot be returned - the message "Task still running" and the task handle is returned instead.

---

**Note:** Remember that there is no Privilege check on any command issued from this window, so you have full access to the system.

---

The Cicode prompt 0:> shows the current window number with which any object is associated. To change the current window, use the WinGoto() function (or any other Cicode function that affects the current window).

The Cicode window does not recognize any variable names, so when you call a Cicode function, you can only pass constants (for example numbers or strings). When you call a function that expects a string, pass a string constant, for example Prompt("Hello from the Kernel"). If the string is only a single word, you do not have to use delimiters, for example Prompt(Hello). The Cicode window tries to convert whatever you enter (as arguments) into the correct data type. If it cannot convert the arguments, it passes either 0 (zero) or an empty string to the function.

---

**Note:** Some Cicode functions are implemented as label macros by the compiler. These macros allow backward compatibility when the number of arguments to a function has been changed. Because the Cicode window does not expand macros, you cannot call these functions directly. You need to use the macro expansion. If the function you are trying to use cannot be found, try again by adding an underscore (\_) to the front of the function name, for example \_DevClose(1).

---

You can also shut down the Plant SCADA system from this window by using the Shutdown() function.

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

## Cls

Clears text from the current window (the Shell window or Cicode window), and moves the cursor to the top left-hand corner.

## Syntax

- **Cls**

Use this command to clear the current window when it has become cluttered (from displaying debug data or too many commands).

Use **Cls Main** to clear the Main window.

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

## Ctapi

This command enables logging of server-side CtAPI debugging information for the communication sessions to various CtAPI clients. Output is to the syslog.dat file. This command results in the same outcome as setting the parameter [\[CtAPI\]Debug](#) to 1 (true).

**Note:** This command is designed to assist qualified technical support personnel with system diagnosis.

---

## Syntax

**Ctapi <0 or 1>**

1 = CtAPI logging enabled

0 = CtAPI logging disabled

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

## Debug

Enables the debugging of raw data that is transferred between Plant SCADA and a selected driver. Protocol traffic is displayed in the Kernel window and logged to the SysLog.DAT file.

## Syntax

**Debug<Port> <Mode> [<Display>]**

Where:

- **<Port>**  
A port name configured in the project (using the Ports form)
- **<Mode>**

The mode:

- ALL - Trace low-level communication traffic to the Kernel window.
- READ - Trace the low-level communication traffic of read commands to the Kernel window.
- WRITE - Trace the low-level communication traffic of write commands to the Kernel window.
- ERROR - Trace any low-level communication traffic that contains a protocol error. This mode only generates traces when an error is detected, so you can leave this option on for long periods of time (to find difficult problems).
- OFF - Stop the debug trace of any type of command.

**Note:** The TO and FROM modes are now obsolete.

• <Display>

Optionally the display scenario:

- MONO - Send the debug to a monochrome monitor (if one is attached) as well as the Kernel.
- MONOONLY - Send the debug to the monochrome monitor only.

To place a port in debug mode, enter DEBUG followed by the port name and the mode you require. If you do not know the name of the port, enter DEBUG (without any arguments), and Plant SCADA displays a list of available ports.

Only the I/O Server communicates with the I/O Devices, so this command is generally used only on the I/O Server.

When you enable a debug trace mode, Plant SCADA displays protocol traffic in the Kernel window and logs it to the SysLog.DAT file. This tends to reduce Plant SCADA's performance (as there may be a lot of data), and therefore not enable debug trace on an I/O Server that is important to your current operation. Use this command only during commissioning or on a non-vital section. Excessive use of this command may cause the I/O Device to go offline.

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

Do not enable or use the Debug Trace mode in an operational environment. This mode is only for use during commissioning.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

When the debug trace is sent to a Monochrome monitor, it is displayed directly from the interrupt routine of the driver and therefore at a much faster rate. This trace (if MONOONLY is used) causes less CPU overload of Plant SCADA while the trace is active, and provides instantaneous output. This method is useful if you are developing your own driver and your driver crashes before the debug trace is displayed in the Kernel.

You can use the [Shell](#) command to create an extra command window while the trace is active. This allows you to enter more commands (in the new window).

You can use the [Pause](#) command to stop the debug output (to view the data).

## **See Also**

[Kernel Commands](#)

[Display the Kernel Window](#)

## Diag Cicodestack

This command sends a report on the call stacks for all running Cicode to the relevant debug.log file.

## Syntax

### Diag Cicodestack

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

## DriverTrace

Controls the listing of driver control blocks (DCBs) between the I/O server and drivers. It can be useful in debugging a driver, for example, it can help determine why a driver is unable to fully initialize.

This command supports the same functionality as the Citect.ini parameter [\[Debug\]DriverTrace](#), though the traces can be applied dynamically during a session. The Kernel command also provides an additional DUMP option when running on an I/O server (see below).

**Note:** Running syslog traces can draw heavily on a CPU usage. monitor the impact on CPU performance when implementing a large number of traces.

## Syntax

**DriverTrace** [OFF|CMDS|VER[BOSE]|ERR[OR]|PORT=<name>|MASK=<xxxxxxxx>|DUMP]

Where:

- **blank**

Returns the current DriverTrace settings. This can be useful to determine the current mask or port settings before implementing a new trace. Just key DriverTrace into the Kernel and hit the return key to view the current settings.

- **OFF**

Turns the DriverTrace off.

- **CMDS**

Turns the DriverTrace on, but does not display the contents of the data buffer.

**Note:** Specific driver commands need to be enabled with the Kernel command DriverTrace[Mask], or the DriverTraceMask INI parameter.

- **VER[BOSE]**

Turns the DriverTrace on, and displays the contents of the data buffer.

**Note:** Specific driver commands need to be enabled with the Kernel command DriverTrace[Mask], or the DriverTraceMask INI parameter.

- **ERR[OR]**

Turns the DriverTrace on, but only traces DCBs with errors (DCBs are the internal data structures used for communication between the I/O server and a driver).

- **PORT**

Allows traces to be limited to a particular driver port, by defining a port name. The default setting, PORT=\*, will trace all ports.

- **MASK**

A 4-byte hexadecimal number that represents a bit mask used to either include or exclude driver commands from the DriverTrace. The driver commands and their values are as follows:

Command	Bit Position
CTDRV_INIT	00000001
CTDRV_OPEN	00000002
CTDRV_INIT_CHANNEL	00000004
CTDRV_INIT_UNIT	00000008
CTDRV_READ	00000010
CTDRV_WRITE	00000020
CTDRV_CONVERT	00000040
CTDRV_CANCEL	00000080
CTDRV_CPU	00000100
CTDRV_DATABASE	00000200
CTDRV_STOP_UNIT	00000400
CTDRV_STOP_CHANNEL	00000800
CTDRV_CLOSE	00001000
CTDRV_FORMAT	00002000
CTDRV_STATS	00004000
CTDRV_DEBUG	00008000
CTDRV_INFO	00010000
CTDRV_STATUS_UNIT	00020000
CTDRV_INIT_CARD	00040000
CTDRV_UPDATE_INFO	00080000

Command	Bit Position
CTDRV_UI_READ	00100000
CTDRV_UI_WRITE	00200000
CTDRV_EXIT	00400000
CTDRV_UNITACTIVATES	01000000
CTDRV_SUBSCRIPTIONS	02000000
CTDRV_EVENTUPDATES	04000000
CTDRV_STATUS_DISCONNECT	40000000

For example, the value you would use to include only the CTDRV\_OPEN, CTDRV\_INIT\_UNIT and CTDRV\_READ commands would be: 0000001A.

Most users will want to exclude the CPU function call, as this happens often. Do this by setting a mask of 7fffffeff. The default <mask> is 7FFFFFFF.

- **DUMP**

Sends a list to the I/O server's syslog.dat file that provides a snapshot of the driver control blocks (DCBs) in the various I/O server queues.

## Examples

```
-> 07f96fdc Cmd: 03 CTDRV_INIT_UNIT ,MOCKOPC, Port: PORT1, Unit: IODev1
(UR0,N1)
| 07f96fdc UnitType: 0 (0x0), UnitAddr: 0 (0x0), BitWidth: 1, UnitCount: 16
<-07f96fdc Cmd: 03 CTDRV_INIT_UNIT ErrDriver 23 (0x17) and took 0ms
-> 07ff1584 Cmd: 04 CTDRV_READ ,DISKDRV, Port: DISKDRV, Unit: IODevDisk
(UR1,N2)
| 07ff1584 UnitType: 1 (0x1), UnitAddr: 1 (0x1), BitWidth: 16, UnitCount: 5,
RawType: INTEGER
<+07ff1584 Cmd: 04 CTDRV_READ ErrDriver 0 (0x0) and took 0ms
<+07ff1584 UnitType: 1 (0x1), UnitAddr: 1 (0x1), BitWidth: 16, UnitCount: 5,
RawType: INTEGER
<+07ff1584 000: 1 0 0 0 5
~> 00000000 Cmd: 25 CTDRV_SUBSCRIBE ,MOCKOPC, Port: PORT1, Unit: IODev1
(UR0,N1)
>-00000000 SUBSCRIBE Tag=Tag4 UpdateRate=500ms
~< 000003e9 Cmd: 26 CTDRV_EVENTUPDATES,MOCKOPC, Port: PORT1, Unit: IODev1
(UR0,N1)
<-000003e9 DS Event UPDATE_MODE=ALL Tag=Tag4 RawValue=0x00
Timestamp=<time_not_set> RawQual=0x0000 DSError=0 (0x0)
```

Explanation of arrows used above

-> Flags information going down to the driver

|

<- Flags information back from the driver straight away

<+ Flags information back from the driver asynchronously (i.e. after some small time delay)

~> Flags subscription based calls into the driver

>-

~< Flags subscription based updated from the driver

<-

#### (URx,Ny) addition to UNIT and Data driver traces

UR stands for Unit Record number and is 0 based, so a value of 3 would be the 4th device in the IODevice settings. N stands for the (Network) Number in the IODevice settings. The use of this is to distinguish between redundant units which may use the same unit name. Thus the trace will positively confirm which unit is in use.

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

## Exit

Closes the Cicode or Shell windows.

## Syntax

### Exit

---

**Note:** You cannot use this command to close the last Shell window. See [Close the Kernel Window](#).

---

To close every other window, select Close from the window's control-menu box, or press Esc, or double click the control-menu box.

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

## Help

Displays a list of some of the commands that are available in the Kernel.

## Syntax

### Help

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

## Log

Enables or disables the logging of I/O DEVICE reads and writes to the syslog.dat file.

## Syntax

**Log<I/O Device> <Mode>**

Where:

- **<I/O Device>**  
The name of the I/O Device to log
- **<Mode>**  
Either: READ, WRITE, or OFF

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

## Page General

## Syntax

[Page General](#)

### General Statistics

Alarm DB Status	The status of the alarm database within the current Plant SCADA process. The status can be one of the following: <ul style="list-style-type: none"><li>• Initializing - the alarm database is initializing.</li><li>• Main - the alarm database is running as a main node.</li><li>• Standby - the alarm database is running as a standby node.</li><li>• Invalid - the alarm database encounters an invalid runtime condition. For example, a connection interruption.</li><li>• Shutdown - the alarm database is shutting down.</li></ul>
Server Name	The server name of this computer or, if it is a client, the name of the primary server this client talks to.
Node Name	The computer name of this computer. You can set this through the Setup Wizard. This is only used if you have Plant SCADA in a networking configuration.

Time	The (current) time of day and the date.
CPU Index	An indication of the performance of the Computer. This number provides a rough indication of the performance of the computer. On a Compaq 486/25M it will be 25.
Running	The total time Plant SCADA has been running
Stat Reset	The time since the Cicode command to reset the statistics was run.
Memory Total	The total amount of free memory (including virtual memory).
Memory Physical	The total amount of free physical memory. The physical RAM that is free on the computer (not including virtual memory).
Memory Resources	The % of free Windows system resources.
Scheduler Cycles	<p>The number of times that the Plant SCADA scheduler is looking for a task to execute. A good indication of how fast the computer can respond to an event. This number depends on the speed of the computer and the CPU resources that Plant SCADA is using.</p> <p>This value is completely different depending on whether you are running 32-bit or 16-bit system. This is because Plant SCADA can use the 32-bit operating system to perform process scheduling, an operation that Plant SCADA needs to perform itself in the 16-bit environment.</p> <p>16-bit: The busier Plant SCADA is, the lower the number. Typical values of 10,000 to 50,000; a highly loaded system may drop to below 1000.</p> <p>32-bit: The busier Plant SCADA is, the higher the number. Typical values of 100 to 1000; a highly loaded system may rise above 5000.</p>
CPU Usage	<p>The percentage of the total available computer processing power that is being used on this computer.</p> <p>As the percentage increases to a high level, the performance of Plant SCADA may level off or become sluggish. If the CPU usage is consistently very high (greater than 70%), there could be a problem with your system or you may be overloading the CPU. For best performance, run your system between 0% and 40%.</p>

Tasks Per Sec	The number of tasks per second that the Plant SCADA scheduler is executing, to show how busy Plant SCADA is. This number will be between 30 and 200 tasks per second. If the computer is an I/O Server, the task number is higher (because each protocol uses many tasks).
Task Late Time	This field indicates how late tasks may be running. Plant SCADA uses a DeltaCheckTask to monitor long running tasks. If tasks are determined to be running later than the current displayed value, then the new value is displayed. If less than the previous time, then the displayed value will be an average of the values. If tasks frequently run late, it is likely that the system is overloaded and it will not be acting on values and information at the right time.
Lost Errors	Plant SCADA keeps track of internal errors in its local error buffers. The Lost Errors counter is incremented when an error is detected and there is no buffer in which to put the alert message. This number should normally be 0 (zero). If this field is incrementing, you may have a problem with your system.

### I/O Server Statistics

The following statistics are only incremented if the computer is configured as an I/O Server:

Read Requests	<p>The first number is the total number of read requests sent to the I/O Server from client Plant SCADA computers, including the local Plant SCADA client. This number continues to increment from 0 (to billions) depending on how fast clients are requesting data and how long the server has been running.</p> <p>The second number is more useful - it records the read requests per second. This value also depends on how fast the clients are requesting data from the I/O Server, and should be between 5 and 400 requests per second</p>
Physical Reads	<p>The first number is the total number of physical reads made to the I/O Devices. Because the I/O Server can optimize the number of read requests made to the I/O Devices, this number is usually smaller than the number of read requests. This number is similar to the read requests and increments forever.</p> <p>The second number is the number of physical reads per second (that the I/O Server is performing). The rate depends on the clients and how fast the server</p>

	can communicate to the I/O Devices (typically 5 to 200).
Blocked Reads	The total number of times a request was made for the same I/O Device address while the I/O Server was already reading that address. The server blocks the two requests together as an optimization.
Digital Reads	The total number of digital points that the I/O Server has read from I/O Devices.
Digital Reads per Sec	The number of digital reads per second that the I/O Server is processing. This provides a general indication of performance. It is dependent on the protocol.
Write Requests	<p>The first number is the total number of write requests sent to the I/O Server from client Plant SCADA computers, including the local Plant SCADA client. The second number is the write requests per second (that the I/O Server is performing).</p> <p>The number of requests depends on the rate at which clients are sending write requests to the I/O Server (typically 0 to 20). If the number of requests continually exceeds 10, then this may be causing performance to decline (usually caused by Cicode performing too many writes to the I/O Devices).</p>
Physical Writes	<p>The first number is the total number of physical writes made to the I/O Devices. Because the I/O Server can optimize the number of write requests made to the I/O Devices, this number is usually smaller than the number of write requests. The second number is the number of physical writes per second (that the I/O Server is performing).</p> <p>The number of writes depends on the rate at which clients are sending write requests to the I/O Server (typically 0 to 20). If the number of requests continually exceeds 10, then this may be causing performance to decline (usually caused by Cicode performing too many writes to the I/O Devices).</p>
Blocked Writes	The total number of times a request was made for the same I/O Device address while the I/O Server was already writing that address. The server blocks the two requests together as an optimization.
Register Reads	The total number of register points that the I/O Server has read from I/O Devices.

Register Reads per Sec	The number of register reads per second that the I/O Server is processing. This provides a general indication of performance. It is dependent on the protocol.
Cache Reads	The number of read requests serviced from the read cache.
Cache Reads(%)	<p>The percentage of reads serviced from the cache. If the cache read % is large, (for example greater than 40%), the I/O Device cache timeout could be set too high. Try reducing the I/O Device cache time to bring the % cache below 40%.</p> <p>The % of read cache depends on the configuration of your Plant SCADA system and the number of clients connected. If you have many clients looking at the same I/O Device data, the cache % may be very high, however this does not usually impact performance. For example, if you have 10 Plant SCADA clients viewing the same page, a cache read of 90% would be acceptable.</p>
Cache Flush	The number of times a cache buffer was flushed because a write request was directed to the same location. When you write to the I/O Device and that address is in the read cache, Plant SCADA flushes the data from the cache. The next time the data is read, it is reloaded from the I/O Device to reflect the new value.
Cache RD Ahead	The number of read-ahead updates made to the cache buffer. When read ahead caching is enabled, the I/O Server will try to read any I/O Device data which is coming close to cache 'timeout' time. The I/O Server will only read this data if the communication channel to the I/O Device is idle - to give higher priority to other read requests.
Cache Buffers	The number of active cache buffers allocated. Each block of data that is read from the I/O Device requires a cache buffer when stored in the cache. The number of cache buffers active at once depends on the dynamic operation of the Plant SCADA computers and their project configurations (typically 0 to 100).
Cache Short	The number of times the cache needed to allocate additional buffers but no buffers were available. When this happens the server cannot cache the data. This does not generate errors but it does lower performance. If this field increments too quickly,

	increase the available memory.
Response Times	<p>The time taken by the I/O Server to process read and write requests (i.e. the time from when a request arrives at the server to when it is sent back to the client). This time depends on the physical response time of the I/O Device and how long the request had to wait in a queue for other requests to be completed. This time increases as the server's loading increases, and is a good indication of the total system response.</p> <p>The average, minimum, and maximum times are displayed. Typical values depend on the I/O Device protocol, however Plant SCADA should have an average response of 500 to 2000 ms. Slower protocols or a heavily loaded system may make the response time higher, but be able to get response times of less than two seconds even for very large systems.</p>

**Licensing Statistics**

Points	The maximum number of I/O points that can exist in your Plant SCADA system. The combined Static and Dynamic count cannot exceed this number. The point limit is part of the Plant SCADA software protection, and is programmed into your license.
Max Full	The maximum number of fully functional Plant SCADA computers (full server licenses) that can exist in your Plant SCADA system. This number is part of the Plant SCADA software protection, and is programmed into your license.
Current Full	The current number of fully functional Plant SCADA computers (full server licenses) that are running in your Plant SCADA system.
Peak Full	The peak number of fully functional Plant SCADA computers (full server licenses) that your Plant SCADA system has experienced since it was re-started.
Max Mngr	The maximum number of View-only Clients that can exist in your Plant SCADA system. This number is part of the Plant SCADA software protection, and is programmed into your license.
Current Mngr	The current number of View-only Clients that are running in your Plant SCADA system.
Peak Mngr	The peak number of View-only Clients that your Plant SCADA system has experienced since it was re-started.

Max Dsp	The maximum number of Display-only Clients that can exist in your Plant SCADA system. This number is part of the Plant SCADA software protection, and is programmed into your license.
Current Dsp	The current number of Display-only Clients that are running in your Plant SCADA system.
Peak Dsp	The peak number of Display-only Clients that your Plant SCADA system has experienced since it was re-started.

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

## Page Driver

Displays information about each driver in the Plant SCADA system. This window is only displayed if the Plant SCADA computer is configured as an I/O Server with physical I/O Devices attached.

## Syntax

### Page Driver

Use Page Up and Page Down keys to scan the driver list.

### Driver Statistics

Port Name	The name of the physical port the driver is using for communication.
Protocol	The name of the protocol.
Title	The protocol title and version string. Protocol drivers for Plant SCADA Versions 3.xx, 4.xx, and 5.xx, are Version 2.0 type drivers.
Read Requests	The first number is the total number of read requests sent to the driver from every client Plant SCADA computer, including the local Plant SCADA client. The second number is the read requests per second (that the I/O Server is performing).
Physical Reads	The first number is the total number of physical reads made to the I/O Devices. Because the I/O Server can optimize the number of read requests made to the I/O Devices, this number is usually smaller than the

	number of read requests. The second number is the number of physical reads per second (that the I/O Server is performing).
Blocked Reads	The total number of times a request was made for the same I/O Device address while the driver was already reading or about to read that address. The driver blocks the two requests together as an optimization.
Digital Reads	The total number of digital points that the I/O Server has read from every I/O Device.
Digital Reads per Sec	The number of digital reads per second that the I/O Server is processing. This provides a general indication of performance. It is dependent on the protocol.
Write Requests	The first number is the total number of write requests sent to the driver from every client Plant SCADA computer, including the local Plant SCADA client. The second number is the write requests per second (that the I/O Server is performing).
Physical Writes	The first number is the total number of physical writes made to the I/O Devices. Because the I/O Server can optimize the number of write requests made to the I/O Devices, this number is usually smaller than the number of write requests. The second number is the number of physical writes per second (that the I/O Server is performing).
Blocked Writes	The total number of times a request was made for the same I/O Device address while the driver was already writing that address. The driver blocks the two requests together as an optimization.
Register Reads	The total number of register points that the I/O Server has read from every I/O Device.
Register Reads per Sec	The number of register reads per second that the I/O Server is processing. This provides a general indication of performance. It is dependent on the protocol.
Cache Reads	The number of read requests serviced from the read cache.
Cache Reads(%)	The percentage of reads serviced from the cache. If the cache read % is large (for example greater than 40%), the I/O Device cache timeout could be set too high. Try reducing the I/O Device cache time to bring

	<p>the % cache below 40%.</p> <p>The % of read cache depends on the configuration of your Plant SCADA system and the number of clients connected. If you have many clients looking at the same I/O Device data, the cache % may be very high, however this does not cause a problem. For example, if you have 10 Plant SCADA clients viewing the same page, a cache read of 90% would be acceptable.</p>
Error Count	The total number of errors encountered by the driver.
Short buffers	The number of times the server needed a buffer for the driver but none were available. This can reduce communication performance and result in loss of requests. Increase available memory if this field increments too quickly (that is, more than 10 per minute).
Driver Errors	The number of low-level driver errors encountered before retries were performed. This field may continue to increment even if no errors are reported because the driver retries and it may complete the command on the second attempt. If this field increments quickly (i.e. 10 or more per minute) without other errors being reported, you may have a low-level error that affects performance.
Out of Buffers	The number of times the driver requires a buffer but none were available. The driver needs to discard the data. If this field increments too quickly, increase the number of pending commands the driver can process. This field only increments with client drivers. I/O Device drivers do not require pending buffers.
Time Outs	The number of timeouts encountered by the driver during operations. This field may continue to increment with no visual errors as the driver retries the operation. If this field increments excessively, there may be a communication problem.
Retries	The number of retries executed by the driver. If this field is incrementing, there may be a communication problem.
Maximum Pending Commands	This is the number of requests that are kept in a buffer waiting to be serviced by the protocol. There is a pre defined default value for this for every protocol. If a protocol is capable of handling multiple requests or commands at a time then this number may be high. If

	<p>the protocol is capable of only handling 1 command at a time then this will probably be 1 or 2.</p>
Blocking Size (Bytes)	<p>This is the range that the I/O Driver will use when blocking read or write requests together at runtime. The default value for this is typically set after quite a lot of experimentation. The value is calculated as the optimum range of data that the I/O Device can respond to, to get the fastest response times from your I/O Device.</p> <p>For example, if the Blocking Size is 100 and you have a graphics page that has Address1 and Address 99 on it, Plant SCADA will read both of these addresses in one request. If you have Address 1 and Address 101 on the page Plant SCADA will issue 2 separate read requests. The block size may not be the maximum packet size for the protocol, since a particular type of I/O Device may respond faster to smaller requests of data than larger requests.</p> <p>There is one important consideration when using this method; many I/O Devices need to have their Memory tables created by the user. If the user does not define every memory address in a range, then Plant SCADA may try and read a block of memory from the I/O Device that does not exist (giving a hardware alarm). This is because Plant SCADA will ask for the whole range of addresses between the starting address and the ending address. So, if in our previous example the I/O Device did not have address 76, it would report back to Plant SCADA that it could not read address 76. The I/O Driver does not know that it doesn't need this address and will retry the command, and in some cases will eventually put the I/O Device offline.</p> <p>Confirm that you always have defined the memory addresses that Plant SCADA will need to read.</p>
Timeout Period (ms)	<p>This is the period of time that the I/O Driver will wait before re-requesting data, if no answer comes from the I/O Device.</p>
Maximum Retries	<p>The number of times the I/O Driver will attempt to get data from the I/O Device.</p> <p>Combining this with Timeout gives you the total period before Plant SCADA will put an I/O Device offline.</p> <p>If the Timeout=2000 and Retry=2 then Plant SCADA</p>

	will wait 2 seconds for a response, then retry, wait 2 seconds, retry, wait 2 seconds, Offline. Total time between losing communications and deciding it is offline is now 6 seconds. You can modify these parameters, but if you set them too low you will generate unneeded retries and possibly get I/O Device Offline messages.
Poll Time (ms)	This is the time in milliseconds that Plant SCADA will check the port for data or write data to the port. If this is 0 then the protocol is operating in Interrupt mode.
Transmit Delay (ms)	This is the time that Plant SCADA will hold a packet of data between receiving a response from the last request and sending the new request. This is usually 0, however some protocols can become saturated and start to misbehave. In these cases the default value has been calculated while the protocol was being tested, and modifying this value to something smaller will cause problems. However, making it bigger will only have a very slight impact on the overall response times in your system, but may make the communications more stable.
Watchtime (seconds)	This is the period of time that Plant SCADA will wait after deciding an I/O Device is offline before trying to re-establish communications. This is typically 30 seconds. It can be made smaller but not be made smaller than the period that Timeout and Retry will be - otherwise you will not be able to re-establish communications with an I/O Device.
Response Times	The time taken by the driver to process read and write requests (i.e. the time taken to process a single read or write operation to the I/O Device). This time depends only on the physical response time of the I/O Device, because no queue waiting time is included. This field reflects any tuning of the communication channel (for example increasing the baud rate will reduce the response time).  The average, minimum and maximum times are displayed.
Channel Usage	This field displays the percentage of the total capacity of the hardware channel that is currently being used. The I/O Server tries to keep the utilization as high as possible, however if the client Plant SCADA computers are requesting data slower than the channel can supply, the total will be below 100%. It is possible for

	the channel usage to rise above 100% as some I/O Device drivers can process more than one command at the same time (having two or more commands using the channel at the same time).
Bytes Per Second	The number of bytes transferred (each second) by the driver. This number provides a simple performance indication that is useful when tuning the driver.
Special Variables	By enabling verbose mode (press V) or by pressing the down arrow, twenty special variables are displayed. The meaning of these variables is driver-specific.

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

## Page Memory

Displays memory debugging information. This window is designed for use by Plant SCADA specialists, and requires a high degree of expertise to use.

## Syntax

[Page Memory](#)

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

## Page Queue

Queues are the main data structure used in Plant SCADA, and this allows the various queues active on the system to be displayed.

## Syntax

**Page Queue [<Name>]**

Where:

<Name>

The name of a specific queue to display (optional). If the name does not match an existing queue, the first queue is shown.

To view the different queues, press the page up and page down keys.

On each page, the handle of the queue being viewed, its name, and its length is displayed. The number of queues and entries within these depends entirely on the custom configuration of Citect32 and the individual project.

Queue formats can vary but several common formats exist.

### **Example**

Queue Sleep

Handle 5 Length 49

	Name	Hnd	State	Prty	CPU	Min	Max	Avg	Count
U	Anm.Animate	9	sleep	user	0.4	0.000	0.003	0.000	1134
U	Tran.Task.Delay	15	sleep	user	0.0	0.000	0.000	0.000	2231
U	DISKDRV \\ WatchDog	22	sleep	user	0.0	0.000	0.000	0.000	7
U	Alarm.Heart	57	sleep	user	0.0	0.000	0.000	0.000	1
U	Alarm.ServerMoni	69	sleep	user	0.0	0.000	0.000	0.000	3
U	Report.Heart	72	sleep	user	0.0	0.000	0.000	0.000	1
U	Alarm.HardRelease	70	sleep	user	0.0	0.000	0.000	0.000	3
U	Trend.Client	55	sleep	user	0.0	0.000	0.000	0.000	95
U	Spl.Task	45	sleep	user	0.0	0.000	0.000	0.000	96

This is the most common queue type. Column meanings are as follows:

- **Mode (Name hidden):** U for user-space task, and S for system tasks.
- **Name:** The name of the task.
- **Hnd :** The internal handle for the task.
- **State:** The state of the task, it can be one of: Free, Curr, Ready, Sleep, Wait, Susp and Dodgy.
- **Prty:** The priority of the item in the queue. It can be one of; Low, User and High.
- **CPU:** Shows the percentage of CPU time used by this task.

- **Min:** Minimum response time for the task (from statistics).
- **Max:** Maximum response time for the task (from statistics).
- **Avg:** Average response time.
- **Count:** The number of times this task has been run.

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

## Page RDB

Displays information about each of Plant SCADA's runtime databases (RDBs).

## Syntax

### Page RDB

Use the Page Up and Page Down keys to move between the tables.

The runtime databases contain your compiled project configuration information. There are two types of RDB; resident and non-resident.

Resident databases are loaded at Plant SCADA startup and remain in memory. Examples of resident databases are Alarms, Trends, Reports, and Functions. The names of resident databases start with the underscore character (\_).

Non-resident databases are those that are associated with pages. These databases are loaded (and unloaded) as necessary - when the page is displayed.

Each database is divided into 10 (or so) tables; <name>, TEXT, REQUEST, PIECE, CODE, SCALE, RUN, WRITE, FUNC, SYMB, and <name>.

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

## Page Table

Contains information about Plant SCADA's runtime data structures. Currently there are about 50 tables of information, most of which are only relevant in specific circumstances.

## Syntax

### Page Table [<Name>]

Where:

<Name>

The name of a specific table to display (optional). If the name does not match an existing table, the first table is shown.

The Table window that appears hosts a number of tables. Use the Page Up and Page Down keys to navigate through the table list. Use Up Arrow and Down Arrow keys to scroll the data for the table that is currently displayed.

The table MASTER\_TABLE displays a list of every table. The following tables are particularly useful:

- [Page Table Accum.ReloadError](#)
- [Page Table Alarm.ReloadError](#)
- [Page Table Broadcast](#)
- [Page Table Cicode](#)
- [Page Table CSAToPSI.Subs](#)
- [Page Table Data.Connection](#)
- [Page Table Data.Recordset](#)
- [Page Table OPCServerConnections](#)
- [Page Table Page](#)
- [Page Table PerformanceCounter](#)
- [Page Table Platform.Sessions](#)
- [Page Table Profile](#)
- [Page Table Report.ReloadError](#)
- [Page Table SQL](#)
- [Page Table Stats](#)
- [Page Table Tran](#)
- [Page Table Trend.ReloadError](#)
- [Page Table Users](#)

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

## Page Table Accum.ReloadError

Displays the accumulator records in the runtime system that did not reload properly with the latest server online change.

Field	Description
Name	Record name, truncated after 32 characters
Status	Showing the nature and the status of the record update in the format of

Field	Description
	"[Add   Modify   Delete] ( [succeeded   failed] )"
Time	The time when the record was updated.
Error code	<p>The error code of the record update.</p> <p>A positive value indicates that an issue was detected when updating the record during the latest server online change.</p>

**See Also**[Page Table](#)

## Page Table Alarm.ReloadError

Displays the alarm records in the runtime system that did not reload properly with the latest server online change.

Field	Description
Name	Alarm record name, truncated after 32 characters.
Type	Alarm type, i.e. Digital, Analog, Advanced, Multi-Digital, Time Stamped, TS-Digital or TS-Analog.
Status	Showing the nature and the status of the record update in the format of "[Add   Modify   Delete] ( [succeeded   failed] )"
Time	The time when the record was updated.
Error code	<p>The error code of the record update.</p> <p>A positive value indicates that an issue was detected when updating the record during the latest server online change.</p>
AcqError	<p>The acquisition error code.</p> <p>A positive value indicates that there is an issue with acquiring the alarm tag value from I/O Server.</p>
State	The current state of the alarm.

## See Also

[Page Table](#)

# Page Table Broadcast

Displays the number of broadcasts on a running instance of a client or server.

Field	Description
MsgType	The broadcast message type.
Broadcasts	The number of broadcasts from when the instance started.
BroadcastsLast5Mins	The number of broadcasts in the last 5 minutes.

## See Also

[Page Table](#)

# Page Table Cicode

This table contains a list of Cicode tasks currently running. It contains the task name, handle and running state, as well as some statistics. CPU\_Time is the total time that the task has run for - it is incremented each time the task runs. The CPU is the percentage of total available CPU that the task is using (fast tasks often have 00 CPU).

## See Also

[Page Table](#)

# Page Table CSAtoPSI.Subs

Displays a list of client tag subscriptions. The Page table CSAtoPSI.Subs contains the following columns:

Field	Description
TagName	String representing the tag or tag element.
Handle	The handle for the subscription.
Value	The current value of the tag.
Timestamp	The timestamp of the last change for the tag.

Field	Description
Quality	The quality of the tag value.
ValueChanges	The number of value changes the tag has had.

## See Also

[Page Table](#)

# Page Table Data.Connection

This table displays a list of current database connections. The Page table users contains the following columns:

Field	Description
Dbc	The handle to the connection.
St	The connection state.
Rn	The number of recordsets.
Qn	The number of executed queries.
Min	The min execution time(s).
Avg	The average execution time(s).
Max	The max execution time(s).
Connect	The connection string (shown only in verbose mode).
Select	The last executed query (shown only in verbose mode).

## See Also

[Page Table](#)

# Page Table Data.Recordset

This table displays a list of recordsets. The Page table users contains the following columns:

Field	Description
Rs	The handle to the recordset.
Dbc	The handle to the DB connection object.
IsDev	"is device" flag.
IsDef	"is default" flag.
Col	The number of columns.
Row	The number of rows.
Connect	The connection string (shown only in verbose mode).
Select	The executed query (shown only in verbose mode).

## See Also

[Page Table](#)

# Page Table OPCServerConnections

Displays information about the OPC DA Server.

Field	Description
OPC Handle	This is the handle the OPC server uses to reference a particular OPC connection. This handle can be cross referenced with log file entries.
Index	This is the server object index, which is incremented each time a new client connects.
#Groups	The number of groups created within the OPC server instance. This number includes both active and inactive groups.
#Items	The number of items creates within the OPC server instances. This number is the total of all active/inactive items in all groups.
#Reads	The number of reads that have been requested for items.
#Writes	The number of writes that have been request for items.

Field	Description
#Updates	The number of updates that have been received for subscribed items.

**See Also**[Page Table](#)

## Page Table Page

This table lists the windows currently displayed in the client.

**Note:** This table does not exist if the current process does not have a role of display client.

Field	Description
Window	The window number that can be obtained from the WinNumber() function. Be aware that this is not the same as the window handle, returned from the WndFind() function.
Parent	The window number of its parent.
Child	The first child window number.
Next	The number of next child (sibling) in the child link of its parent.
RdbPage	The handle of page RDB (Plant SCADA Runtime Database).
FPS	The frames attempted to be rendered per second of the window.
Page	The page name.

**See Also**[Page Table](#)

## Page Table PerformanceCounter

Displays the performance counters that are holding an instance in the current Plant SCADA runtime process.

Field	Description
Category	The performance counter category name (omitting the "Citect.Platform." prefix to save space).
Counter	The performance counter name.
Instance	The performance counter instance name (omitting the process signature, i.e. procid: #####).
Value	The current value of the performance counter.
State	The current state of the performance counter, either "Enabled" or "Disabled".

To enable some counters, you need to manually adjust the following parameters:

- [\[Debug\]EnablePSICounters](#)
- [\[Debug\]EnableTaskFrameworkCounters](#)
- [\[Debug\]EnableTransportCounters](#).

In most cases, this will only be required if you are following instructions from Technical Support.

## See Also

[Page Table](#)

# Page Table Platform.Sessions

This table shows a list of platform sessions supporting client/server communications.

There are two modes for viewing the table: Standard and Verbose. The Standard mode shows information for every session in a tabular format. It can be scrolled using the up/down arrow keys, the top row indicates the currently selected session. Extended information for the selected session can be viewed by toggling to Verbose mode by pressing the "v" key. Once in Verbose mode, the selected session can be changed using the Page Up/ Page Down keys.

In standard mode the Platform.Session table contains the following columns:

Field	Description
Name	The name of the session.
Type	This is either: <b>Client</b> - indicates a client session <b>Server</b> - indicates a server session.
Mode	This is either: <b>Named Pipe</b> - local session

Field	Description
	<b>Null</b> - in-proc <b>Host</b> - TCP/IP.
Remote	The remote address if it is a client session.
State	The current state of the session.
Security Info	If a secure TCP/IP session, a combination of the SSL Protocol, Cipher Algorithm and Hash algorithm used. If a secure local session, "Local ACL" is displayed. An in-process session will not show any information.

## See Also

[Page Table](#)

# Page Table Profile

Displays the information generated once.

**Note:** Use the +/- keys to sort the information. The name of the column by which the information is sorted is displayed in uppercase letters. Sort order details (ascending/descending) are mentioned in the table below.

Field	Description
Function / FUNCTION	The name of the function. Built-in functions are identified with "*", for example "AlarmCountEquipment *". Special functions are identified with "@", for example "Page Animation @". When the column title is uppercase, the rows are sorted by this column (ascending).
Called / CALLED	The number of times the function was called during the profiling period. When the column title is uppercase, the rows are sorted by this column (descending).
Incl(s) / INCL(S)	The number of seconds the function was running for during the profiling period. This includes any time spent inside other functions called by this function. When the column title is uppercase, the rows are sorted by this column (descending).
Excl(s) / EXCL(S)	The number of seconds the function was running for during the profiling period. This excludes any time

Field	Description
	spent inside other functions called by this function. When the column title is uppercase, the rows are sorted by this column (descending).
Incl(%)	The time spent in this function as a percentage of the profiling period. This includes any time spent inside other functions called by this function.
Excl(%)	The time spent in this function as a percentage of the profiling period. This excludes any time spent inside other functions called by this function.
Incl(ms)	The time spent in this function divided by the number of times this function is called during the profiling period. This gives an average time per call in milliseconds. This includes any time spent inside other functions called by this function. This value is capped at 9999.999 milliseconds.
Excl(ms)	The time spent in this function divided by the number of times this function is called during the profiling period. This gives an average time per call in milliseconds. This excludes any time spent inside other functions called by this function. This value is capped at 9999.999 milliseconds.

**Note:** The Inclusive values are useful for identifying an area of concern from the top down, but being inclusive, the values contain all time spent (including when the Cicode is dormant). Therefore this needs to be considered when analyzing the data. For background Cicode tasks that run indefinitely, the inclusive figures will likely be high, as the function does not stop running. If the same function is launched in multiple background tasks concurrently, the Incl(%) value will be higher than 100%, as it is being run in parallel.

## See Also

[Page Table](#)

# Page Table Report.ReloadError

Displays the report records in the runtime system that did not reload properly with the latest server online change.

Field	Description
Name	Record name, truncated after 32 characters
Status	Showing the nature and the status of the record

Field	Description
	update in the format of: "[Add   Modify   Delete] ( [succeeded   failed] )"
Time	The time when the record was updated.
Error code	The error code of the record update. A positive value indicates that an issue was detected when updating the record during the latest server online change.

**See Also**[Page Table](#)

## Page Table SQL

This table shows the following details for each SQL connection:

Field	Description
Dbc	The handle to the DB connection object.
Stmt	The handle to the statement.
Fmt	The handle to the format definition.
Connect	The connection string.
Select	The last executed query.
QryActv	Indication that the query is active (set to TRUE between SQLExec and SQLEnd).
TrnActv	Indication that the transaction is active (shown only in verbose mode).

**See Also**[Page Table](#)

## Page Table Stats

This table contains the cycle and execution times of every task that is running in Plant SCADA. The Execution time

is the time taken for the entire task to run. The Cycle time is the time between when a task starts and when it starts again. The CPU is the percentage of total available CPU that the task is using (fast tasks often have 00 CPU).

The Plant SCADA 0 entry is the display task (graphics page updates) for the main window. That is, the total time taken for the Client to request data from the I/O Server, the I/O Server to get the data and send it back to the Client, and the Client to update the display.

---

**Note:** Plant SCADA 0 corresponds to the display task for the main window. Plant SCADA 1 for the first child window, Plant SCADA 2 for the third, and so on.

---

The CodeX entries correspond to Cicode tasks, where X is the handle of the task. You can find out which task corresponds to which handle by viewing Cicode table.

---

**Note:** There will be a Trend Acq entry for every different trend sampling period you have defined in your project.

---

## See Also

[Page Table](#)

## Page Table Tran

This table shows a list of channels of communication between Plant SCADA components. A tran exists between exactly two separate components. A client tran initiates a connection, a server tran waits for a connection. Client and server in this context bears no relation to the type of component that owns the tran.

There are two modes for viewing the tran table: Standard and Verbose. The Standard mode shows information for every tran in a tabular format. It can be scrolled using the up/down arrow keys, the top row indicates the currently selected tran. Extended information for the selected tran can be viewed by toggling to Verbose mode by pressing the "v" key. Once in Verbose mode, the tran to display can be changed using the Page Up/Page Down keys.

In standard mode the tran table contains the following columns:

Field	Description
Name	<p>The name and type of communication channel truncated to 20 characters:</p> <ul style="list-style-type: none"><li>• Client Format: &lt;cluster name&gt;&lt;service name&gt; Examples: Cluster1Alarm, Cluster1Report, Cluster1Trend</li><li>• Server Format: &lt;server name&gt;&lt;cluster name&gt; Examples: AlarmServer1Cluster1, ReportServer1Cluster1, TrendServer1Cluster1</li><li>• Dedicated (Client and Server) Format: @@&lt;cluster name&gt;&lt;server name&gt; Examples: @@Cluster1.AlarmServer1, @@Cluster1.ReportServer1,</li></ul>

Field	Description
	<p>@@Cluster1.TrendServer1</p> <ul style="list-style-type: none"> <li>• Platform (Client and Server) Format: &lt;server name&gt;&lt;cluster name&gt; Examples: AlarmServer1Cluster1, IOServerCluster1</li> </ul>
Node	<p>Either the node to which the tran is connected, or the status of the connection:</p> <ul style="list-style-type: none"> <li>• <b>&lt;call&gt;</b> The client tran is attempting to connect.</li> <li>• <b>&lt;listen&gt;</b> The server tran is waiting to be connected.</li> <li>• <b>&lt;disabled&gt;</b> The tran is currently disabled.</li> <li>• <i>node name</i> The tran has connected and is online. The value is the name of the node/computer to which the tran is connected. This could be the current computer or a different computer depending on the TCP/ IP configuration of the project.</li> </ul>
Type	<p>This is either:</p> <ul style="list-style-type: none"> <li>• <b>Client</b> - indicates a client tran. This tran actively attempts to connect to a server tran.</li> <li>• <b>Server</b> - indicates a server tran. This tran passively waits for a connection attempt from a client tran.</li> <li>• <b>SerRnd</b> - indicates a server-to-server redundant tran. This is the connection between the primary and standby servers when a project is configured for server redundancy.</li> </ul>
Mode	<p>This is either:</p> <ul style="list-style-type: none"> <li>• <b>Local</b> This tran connects two components which exist in the same process. For instance, when Plant SCADA is run in single process mode, every server component runs in the same process as the client. In this case, the connections between these components would be marked as Local.</li> </ul>

Field	Description
	<ul style="list-style-type: none"> <li>• <b>OutPro</b> (Out of Process) This tran either has an established connection or is attempting to establish a connection between two components via TCP/IP. The components exist in different processes. They may exist on the same computer or on different computers.</li> <li>• <b>OutPrD</b> (Out of Process Dedicated) A dedicated connection exists between the client and each server process on a single machine when Plant SCADA is run in multi-process mode. These trans use named pipe connections to verify that a communication path exists between every Plant SCADA component running on one computer regardless of the project's TCP/IP configuration. Dedicated connections do not exist between different computers.</li> <li>• <b>Platfo</b> (Platform Tran) This tran uses the Platform networking module as the transport layer between components. For version 7.0, it is only used to connect to the I/O Server. <b>Note:</b> There are client-side Alarm Server platform trans which will show up in the Tran Table. The server-side Alarm Server platform trans were not necessary or implemented for version 7.0 so these client-side trans will remain in a state of Connecting. They can be ignored.</li> <li>• <b>Remote</b> This server tran is waiting for a connection attempt via TCP/IP. Upon connection, this mode changes to OutPro.</li> </ul>
Hnd	This value represents the handle number in the tran table of the record.
Cnt	<p>This value indicates the number of times the tran has established a connection. Specifically it counts the number of times the tran receives the MSG_OPEN message. This value has no meaning for local trans for which it remains 0.</p> <p>If the number is high, it can indicate that your network is dropping and then re-establishing connections. However, it could also mean that the server has been running for a long time and many clients have started</p>

Field	Description
	and stopped, thereby closing and opening sessions.
Send	This value displays the number of messages that have been sent by the tran.
Rec	This value displays the number of messages that have been received by the tran.
Wait	This value represents the number of times the tran had to wait to get a buffer in order to send a message.
Stack	<p>This value indicates the protocol number. Historically, this incorporated the NetBIOS LanA numbers with the TCP/IP protocol stacks. However, NetBIOS support was removed for version 7.0 while TCP/IP support was enhanced to include redundant Network Interface Cards (NIC).</p> <p>The Stack value displays an index (1-based) which indicates on which redundant IP address the server tran is listening or connected. Its value is only used for out-of-process TCP/IP server trans; it has no meaning for every other tran.</p>
Service	This column displays the type of the service used by the tran (regardless of the tran mode). The valid services are Alarm, IO, Report and Trend.
State	<p>This is the current state of the tran. Valid states are:</p> <ul style="list-style-type: none"> <li>• <b>Online</b> - The tran is online.</li> <li>• <b>Offline</b> - The tran is offline.</li> <li>• <b>Connecting (client trans only)</b> - The client tran is connecting to the listed component.</li> <li>• <b>Disconnecting</b> - The tran is disconnecting from the listed component.</li> <li>• <b>Listening (server trans only)</b> - The server tran is listing to a client tran.</li> <li>• <b>Disabled</b> - The tran connection is disabled.</li> </ul>
Access	<p>This is the current status of the connection. It can be:</p> <ul style="list-style-type: none"> <li>• <b>No Access</b> - This tran connection does not have access to the listed component. This indicates that either the component is not running, or the listed component does not have permissions to access</li> </ul>

Field	Description
	<p>it.</p> <ul style="list-style-type: none"> <li>• <b>Trusted</b>- The tran connection with the listed component has been authenticated using "Server Password".</li> <li>• <b>Authenticated</b> - The tran connection with the listed component has been authenticated using the logged-on user credentials.</li> </ul>
Security Info	<p>Displays the description of the security certificate applied.</p> <p>More information about security is available via <a href="#">Page Table Platform.Sessions</a>.</p>

## See Also

[Page Table](#)

## Page Table Trend.ReloadError

Displays the alarm records in the runtime system that did not reload properly with the latest server online change.

Field	Description
Name	Trend record name, truncated after 32 characters.
Type	Trend type, i.e. Periodic, Event or Periodic Event.
Status	Showing the nature and the status of the record update in the format of "[Add   Modify   Delete] ( [succeeded   failed] )"
Time	The time when the record was updated.
Error code	The error code of the record update. A positive value indicates that an issue was detected when updating the record during the latest server online change.
AcqError	The acquisition error code. A positive value indicates that there is an issue with acquiring the trend tag value from I/O Server or reading from trend archive.

Field	Description
Archive	The trend archive file name.

## See Also

[Page Table](#)

# Page Table Users

This table displays a list of users currently logged into the system either locally or remotely. The Page table users contains the following columns:

Field	Description
Name	The user name for a particular user.
Last Login	Time at which user last logged in.
Last Logout	Time at which user last logged out.
Num connected	Number of references, both internal and external.
Default	Indicates whether user is the default user for the Plant SCADA process. Only one user can be a default user.
Logged In	Indicates if user is currently logged in.
Local User	Indicates if user is the local user of the current process. Only one user can be the current processes local user.

## See Also

[Page Table](#)

## Page Unit

Displays information about I/O devices in a Plant SCADA system. This information is only displayed if the Plant SCADA computer is configured as an I/O server. Only the I/O devices connected to the I/O server are included.

## Syntax

### Page Unit

Use the Page Up and Page Down keys to scan the I/O devices for this I/O server. To view redundant I/O devices on other I/O servers, use Verbose mode.

**I/O Device Information**

Unit	The name of the I/O device defined in the project.
IO Server	The name of the I/O server that is servicing this I/O device.
Comment	A description of the I/O device.
Unit Network No	The I/O device number defined in the project.
PLC Number	The physical I/O device address defined in the project.
Port Name	The communication port to which the I/O device is connected.
Protocol	The protocol used for communication with the I/O device.
Unit Status	<p>The status of the I/O device. Only I/O devices that are serviced by this I/O Server are shown, and their redundant I/O devices if page is shown in Verbose mode.</p> <p>The unit status can be one of the following:</p> <ul style="list-style-type: none"> <li>• RUNNING - Indicates that the communication link with the I/O device is good.</li> <li>• STANDBY - Indicates that the communication link with the I/O device is good, but communication with that I/O device is currently being performed by another port. This port is in standby mode.</li> <li>• STARTING - Indicates that the server is currently establishing a communication link (with the I/O device).</li> <li>• STOPPING - Indicates that the server is currently relinquishing control of the communication link (with the I/O device).</li> <li>• OFFLINE - Indicates that the server cannot establish a communication channel with the I/O device. If a standby port or server is available, Plant SCADA tries to communicate to the I/O device using that port.</li> <li>• REMOTE - Indicates that the status of the I/O device is OK, but it is not currently connected.</li> </ul> <p><b>Note:</b> If running the Kernel on the I/O server, you will get the current status of the I/O device. If running the Kernel on a client elsewhere, you will get the last known status.</p>

Primary	<p>Indicates if the I/O device is configured to be in primary mode.</p> <ul style="list-style-type: none"> <li>• Yes = Primary</li> <li>• No = Standby.</li> </ul> <p>If the I/O device is in primary mode, the server starts a communication channel with the I/O device as soon as the server is activated. If an I/O device is in standby mode, the I/O device remains inactive when the server starts (until a primary I/O device becomes inoperative).</p>
Priority	<p>The relative priority configured for this I/O device. This field is only relevant for redundant I/O devices.</p> <p>The standby I/O device with the highest priority will take over from the primary when required, the next highest priority will take over from that device if required. Be aware that the highest priority will have the lowest priority value.</p>

All of the following fields are not shown when the I/O device is running in Memory Mode.

Generic Error	The last generic error code returned by the driver. Because most protocol drivers have their own special errors, they cannot be recognized by the I/O server. The drivers convert their special errors into generic errors that can be identified by the server.
Error Handle	This field is for internal diagnostics.
Driver Error	The driver-specific error code. Each driver has its own special error codes. Refer to the driver specific errors (for the particular protocol) for an explanation of each of the error codes.
Error Message	The alert message associated with the generic error code.
Error Count	The total number of errors from the I/O device.
Restarts	The number of times the server has tried to establish a connection with the I/O device. This number is normally 1, because the server establishes a connection at startup. If this field displays a number greater than 1, there is a problem with the communication channel.
Response Times	The time taken by the driver to process read and write requests (i.e. the time taken to process a single read

	<p>or write operation to the I/O device). This time depends only on the physical response time of the I/O device, because no queue waiting time is included. This field reflects any tuning of the communication channel (for example doubling the baud rate will half the response time). The average, minimum, and maximum times are displayed.</p> <p><b>Note:</b> One I/O device with a slow response may slow down your entire system. For example, if you have an I/O device with a response of 2000 ms, any pages in your system that use data from that device, will have a minimum update time of 2000 ms.</p>
Cached	This field indicates if the I/O device data is cached. Caching may impact communications with the physical device.
Cache Timeout	If the I/O device is cached, this field displays the cache timeout value. Data is held in the cache for this timeout period before being discarded and re-read from the I/O device. Only read data is cached.
Blocking Constant	The current blocking constant value for this I/O device, as specified in the protocol. This value influences how requested I/O is blocked together.

The following fields are only shown when the I/O device is not in memory mode and configured for scheduled dial-up remote mode.

Dial-up Connection	<p>The status and history of the dial-up connection.</p> <ul style="list-style-type: none"> <li>• SUCCESS - The number of successful dial-up attempts.</li> <li>• FAIL - The number of unsuccessful dial-up attempts.</li> <li>• TOTAL - The total number of dial-up attempts.</li> <li>• NEXT - The time of the next scheduled dial-up attempt.</li> </ul>
Total Connect Time	Accumulated time over all connections.
Average Connect Time	Average time per connection.
Subscription Management	<In Progress>
Unit State	An I/O device can have <n> level redundancy with a Primary and multiple Standbys on one or more I/O servers. The Unit State for a specific I/O device can be

	<p>one of the following:</p> <ul style="list-style-type: none"><li>• GLOBAL ACTIVE - The highest priority redundant online I/O device, across all I/O servers. Be aware that the highest priority will have the lowest priority value.</li><li>• LOCAL ACTIVE - The highest priority redundant online I/O device, on this I/O server.</li><li>• INACTIVE - The I/O device is either offline, or online and not Local Active.</li><li>• FORCE ACTIVE - A Local Active unit forced into operation when receiving subscription requests.</li></ul>
Active Subscriptions	The number of client tag subscriptions serviced for this I/O device.
Poll Reads Total	The accumulated number of poll reads issued to physical device to service tag subscriptions.

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

## Pause

Pauses debug output in the Kernel window.

This is a toggle command. Using it a second time will restart debug output.

## Syntax

**Pause**

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

## Shell

Opens a new command (shell) window.

## Syntax

### Shell

You can use shell windows in a similar manner to a Main window. Shell windows are useful for displaying debugging information, or entering commands when the Main window is displaying debug trace data. You can close the shell windows by selecting Close from the window's system icon or with the [Exit](#) command.

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

### Stats

Resets system statistics (used in the page general, page drivers, page table (stats), and page I/O Device windows) to 0 (zero).

## Syntax

### Stats

This command allows you to reset the statistics after Plant SCADA has been running for a long time, and therefore provides an indication of the statistics now (instead of an average over the total time that Plant SCADA has been running).

**Note:** Some I/O Server statistics are automatically reset every few minutes.

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

### Tran

Enables logging for tran activity messages in the syslog.dat file. Tran refers to a connection between one Plant SCADA server/client component and another. A client tran initiates a connection, a server tran waits for a connection. Client and server in this context bears no relation to the type of component that owns the tran.

This command operates as a toggle. Using it the first time will turn logging on, using it again will turn it off.

You can use [Page Table Tran](#) to view the details for existing trans, including the name and handle.



### LOSS OF CONTROL

It is possible to overload your system if tran logging is used excessively. Do not enable tran logging except on the advice of Technical Support.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Syntax

**Tran <Name>**

Or

**Tran <Handle>**

Where:

- <Name> = a component name and type of communication channel (truncated to 20 characters).

For example:

Client

Format: <cluster name><service name>

Examples: Cluster1Alarm, Cluster1Report, Cluster1Trend

Server

Format: <server name><cluster name>

Examples: AlarmServer1Cluster1, ReportServer1Cluster1, TrendServer1Cluster1

Dedicated (Client and Server)

Format: @@<cluster name><server name>

Examples: @@Cluster1.AlarmServer1, @@Cluster1.ReportServer1, @@Cluster1.TrendServer1

Platform (Client and Server)

Format: <server name><cluster name>

Examples: AlarmServer1Cluster1, IOServerCluster1

- <Handle> = a numerical value for a tran available from [Page Table Tran](#).

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

## Write Log

This command sends a customized message to the syslog.dat file associated with the current process.

It allows real-time activity to be recorded in a log file. For example, you could add the message "Disconnecting PLC123 now".

## Syntax

**Write Log <Message>**

<Message> = User-defined text.

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

## Write Modules

This command lists all the loaded native process modules into the debug.log file associated with the current instance of Plant SCADA.

This information can be used to confirm which DLL versions are being used and their current location (as a path).

For example:

```
(00020000) - (00102FFF) v8.50.0.xxx C:\Program Files (x86)\AVEVA Plant SCADA\Bin\  
Citect32.exe  
(06430000) - (066C8FFF) v8.50.0.xxx C:\Program Files (x86)\AVEVA Plant SCADA\Bin\  
ctEng32.dll
```

## Syntax

**Write Modules** or **Write Mod**

## See Also

[Kernel Commands](#)

[Display the Kernel Window](#)

## System Tuning

Plant SCADA is designed for optimal performance, so it is not necessary for most users to tune their system. However, special circumstances might require that you adjust your system for optimal performance. The Kernel allows you to locate areas that need tuning, and the tuning itself is usually done through parameters. For example, you can improve performance of the client by using the [\[Page\]ScanTime](#) and [\[Alarm\]ScanTime](#) parameters.

## See Also

[Cache Tuning](#)

## Cache Tuning

Tune the cache large enough so that unnecessary reads are not generated, and small enough that old data is not returned while keeping the communication channel busy. If the cache is too large, the communication channel might become idle for a while and so waste its bandwidth. Also if the cache is too large, a Plant SCADA client might start to short cycle on reads request, which will generate unnecessary network or internal traffic load.

Read short cycling occurs when a client requests data from the I/O Server, and the data is returned from the

cache, so it is returned quickly. The client will process the data (display it on screen) then ask for the same data again. If the I/O Server again returns the same data from the cache, the client will process the same data again which is redundant and a waste of CPU and the network (to transmit the request and response). When short cycling starts to occur, the CPU and network loading will rise while the PLC communication traffic will start to fall.

To tune the cache you need to balance the cache time between unnecessary reads and short cycling. The method described below assumes you know how to use the Plant SCADA debugging [The Kernel](#).

1. Turn off unit caching, use the CACHE command in the Kernel so you don't have to re-compile your project.
2. Run one Plant SCADA client only on the network, use the Client in the I/O Server for the test.
3. Display a typical page to generate normal PLC loading for your system.
4. In the Kernel use the STATS command to reset the Plant SCADA statistics.
5. In the Kernel display the page 'PAGE TABLE STATS'. This page shows the cycle and execution time of various Plant SCADA tasks, some of which consume PLC data. The tasks called 'Citect  $n$ ' where  $n$  is a number are the tasks which get data from the PLC and display on screen. Look at the Avg Cycle time, this is the third column from the left. Assume that the Avg cycle time is 1200 ms. This will mean that the current page is gathering PLC data and displaying its data on the screen in 1200 ms.
6. Always set the cache time below this average cycle time to minimize short cycling. On average set it to less than half this time, that is 600 ms.
7. Set the cache time to half the cycle time (600 ms). You might not see any improvement in performance with a single client, as caching will only improve performance with multiple clients. You might see improvements if you are also running trends, alarms or reports which are requesting the same data.
8. Add another Plant SCADA client that is displaying the same data. Reset the STATS and check the Average cycle time. Each new client will not increase the cycle time, it will drop slightly. Also look at PAGE GENERAL, to see that each new client services its reads from the cache; i.e., the % cache reads increases.
9. If the average cycle time drops to less than half the original time then short cycling is occurring and you need to decrease the cache time until this stops.

Tuning the cache is a trial and error process - as you change it, the read cycle time will also change. The cache time will also depend on what the current PLC traffic is. The current traffic is dynamic as Plant SCADA will only read what is needed depending on the current page, trend, alarm and reports running. Monitor the average cycle time under lower loading conditions and set the cache as low as necessary to stop or help prevent short cycling.

## Log Viewer

The Log Viewer allows you to view logged messages for AVEVA components that have the Operations Control Logger enabled. For Plant SCADA, this includes the following components:

- Plant SCADA Studio
- Plant SCADA Runtime (see also [Monitor Runtime](#))
- Deployment Server/Deployment Client
- OPC UA Server (Connectivity Service)
- AVEVA Industrial Graphics (Connectivity Service)
- AVEVA Enterprise Licensing
- AVEVA Platform Common Services.

You can use filter, mark, find, time range and other functions to specify which messages are shown in the Log Viewer.

The Log Viewer is an MMC snap-in that operates within AVEVA's Operations Control Management Console.

#### To display Log Viewer messages:

1. From the Windows **Start** menu, select **Operations Control Management Console** from the **AVEVA** program group.
2. Locate the **Log Viewer** within the console tree.
3. To view the messages for any locally installed components, select **Local** within the **Default Group** branch.

The Operations Control Management Console is supported by its own documentation, which includes information explaining how to use the Log Viewer. To access this information, open **AVEVA Help** from the **AVEVA** program group in the Windows **Start** menu.

#### See Also

[Log Files](#)

## Running the System as a Windows Service

Plant SCADA can be run as a Windows™ service, allowing for unattended operation of a system's servers.

This offers the following benefits:

- Automated server operation on hardware startup
- Continuous server operation across multiple user logins.

For example, you may want to operate Plant SCADA as a distributed system across a number of server computers that are locked in a secure room. Each server could be configured to automatically run as a service on startup, meaning access to the server room is not necessary for normal system operation. Each server will also be able to run continuously without the need for a user to log in.

To run Plant SCADA as a service, you need to configure the service named "Plant SCADA Runtime Manager" in Microsoft Management Console on each server computer. For more information, see [Configure a System to Run as a Windows Service](#).

By default, Plant SCADA operates under the "NT SERVICE\Citect Runtime Manager" virtual service account when running as a service. In some circumstances, this may not be suitable. For example, if your system includes an OPC DA Server, you will need to specify a particular user account to run the service (see [Running as a Service Under a Specific User Account](#)).

---

**Note:** If you have configured a Plant SCADA OPC DA server on a computer that is running Plant SCADA as a Windows service, you will need to make the additional configuration changes. See [Running an OPC DA Server as a Service](#).

When Plant SCADA is operating as a service, the Runtime Manager will operate in "service mode". For more information about service mode and its limitations, see [Operate Runtime Manager in Service Mode](#).

You cannot debug Cicode for a process that is running as a Windows service.

---

**Note:** When you launch a display client on a computer that is running Plant SCADA as a Windows service, the

---

client will request an additional control client license from the system. If this occurs on a machine with a valid license that has been obtained locally, you will be granted the additional license automatically.

---

## Configure a System to Run as a Windows Service

When you install Plant SCADA on a computer, a service named "Plant SCADA Runtime Manager" is also installed. Operation of this service is managed using the Computer Management console, a component of the Windows™ operating system.

You can operate the Plant SCADA Runtime Manager in two ways:

- Manual operation - where the service is controlled by a user via the Computer Management console
- Automatic operation - where the service is set to run whenever the host computer starts.

Typically you would configure the service to run automatically to allow a dedicated server to become operational without any user interaction.

---

**Note:** If you plan to operate Plant SCADA as a service on multiple computer across a distributed system, you need to configure the Plant SCADA Runtime Manager service on each computer.

---

### To configure the Plant SCADA Runtime Manager service:

1. Open **Services** in the Microsoft™ Computer Management console (for more information, refer to the operating system documentation provided by Microsoft).
2. Locate Plant SCADA Runtime Manager in the list of included services. The **Status** column will indicate the current operational state of the service.
3. To manually start the service, right click on it and select **Start** from the menu that appears.

Or:

Double-click on the service to open the Plant SCADA Runtime Manager Properties dialog, and select the **Start** button on the **General** tab.

You can also use these methods to **Stop** and **Restart** the service.

4. To set the service to start automatically, open the Plant SCADA Runtime Manager Properties and set the **Startup Type** field to **Automatic**. The service will automatically start the next time the computer is turned on.

---

**Note:** If you attempt to switch a computer between service mode and normal operation, you will need to close down any local display clients and the Runtime Manager before you attempt to restart Plant SCADA in a different mode of operation.

---

## See Also

[Running as a Service Under a Specific User Account](#)

## Running as a Service Under a Specific User Account

By default, Plant SCADA operates under the "NT SERVICE\Citect Runtime Manager" virtual service account when running as a service.

In some circumstances, this will not be appropriate as a specific user (or type of user) may be required to

support particular functionality at runtime. For example:

- An OPC DA server uses specific DCOM settings that are not compatible with the local system accounts or the "NT SERVICE\Citect Runtime Manager" virtual service account.  
An OPC DA client will also be impacted by this incompatibility with DCOM.
- The PS Direct driver requires that the runtime user is part of the administrators group.

If you are faced with this type of situation, you may need to specify a particular user account under which the Plant SCADA service will operate.

#### To specify the user account that operates the Plant SCADA service:

1. Open **Services** in the Microsoft™ Computer Management console (for more information, refer to the operating system documentation provided by Microsoft).
2. Locate Plant SCADA Runtime Manager in the list of services.
3. Double-click on the service to open the Plant SCADA Runtime Manager Properties dialog.
4. Go to the **Log On** tab, and select the **This Account** option.
5. Enter the name and password for the account that will operate the service.

#### See Also

[Running an OPC DA Server as a Service](#)

### Running an OPC DA Server as a Service

By default, Plant SCADA operates under the "NT SERVICE\Citect Runtime Manager" virtual service account when running as a service. This is not suitable for a system that includes an OPC DA server, as this type of server uses specific DCOM settings that are not compatible with the virtual service account.

To address this situation, you need to make the following configuration changes on the computer where an OPC DA server will run as a service:

- Specify a particular user account under which the Plant SCADA service will operate (see [Running as a Service Under a Specific User Account](#)).
- In the DCOM settings for the OPC DA server, confirm that the same user account is used as the identity that operates the server.

#### To specify the DCOM identity that operates the OPC DA server:

1. Open the **Component Services** console (for more information, refer to the operating system documentation provided by Microsoft™).
2. In the list of Component Services, open the **My Computer** branch, then the **DCOM Config** directory.
3. Right-click on the SCADA OPC DA Server and select **Properties**.
4. Go to the **Identity** tab, and select the **This User** option.
5. Enter the name and password for the account that is configured to operate Plant SCADA as a service.

The OPC DA server will now operate as a service with appropriate DCOM credentials.

**Note:** If you need to return Plant SCADA to normal operation (that is, you will no longer run it as a service), you will need to set the DCOM identity for the OPC DA Server back to **Interactive user**.

## See Also

[Using an OPC DA Server](#)

[OPC DA Server DCOM Settings](#)

## Time Synchronization

To maintain time synchronization, Plant SCADA installs a Windows service, **TimeSyncService**. It runs on a virtual service account. This service maintains the time on the local computer associated with one or more time sources. A time source is a computer on which the time service runs.

A time synchronization utility is provided by Plant SCADA to assist you to configure time synchronization, and control the service as part of your administration environment. The dialog stores and reads settings in the **TimeSyncConfig.xml** file, which is installed in the Plant SCADA Config directory by default (see [View a Project's Folders](#)).

You can use the configuration utility to specify an alternative path to the config file, such as a network share. This can be useful where you have multiple computers using the same configuration data as you only need to make changes on one machine.

To display the Time Synchronization Configuration dialog box, navigate to Windows™ **Start** menu | Programs | AVEVA Plant SCADA, and run **Time Synchronization Config**.

## Time Synchronization Dialog Box

The fields available on the Time Synchronization dialog are described in the following table.

Field	Description
Current status	Displays the status of the TimeSync Windows service, as displayed in service properties under computer management. You may click the Start Service button if the service is stopped, or Stop Service if it is running. If the service is identified as being disabled, the button is also disabled. To enable the service use the Windows administrative tools as either automatic or manual startup.
Startup type	Identifies if the service is started manually, or automatically. If the service is disabled, use the Windows administrative tools to enable the service as either automatic or manual startup.
TCP/IP Port	The network port the service uses to listen for connections from clients.
Last synchronization	Displays the value of the <b>LastSyncTime</b> registry setting.

Field	Description
	This is the Local time at which the last successful synchronization occurred.
Current local time	Displays the current time on the local computer, updating in every second.
Log information events	Controls whether the service writes events of type 'Information' to the event log. The default is unchecked so that only alerts (called "warnings" in the software) and errors are recorded.
Keep this computer's time synchronized	Select this check box to enable the computer to be a time client. This allows you to enter the poll time and list of time servers with which to synchronize.
Synchronize every	Enter a number in hours and minutes between 000:01 and 168:59 (inclusive) to specify the period between synchronization that needs to occur. The default value is 24:00.
Synchronize Now	Click to synchronize immediately.
Synchronize with first available	Displays a list of computers, and the current time on those computers if available. The display is updated in every second.
Add	Displays a dialog box to enter the Computer Name, IPv4 address or IPv6 address of a server. Click <b>OK</b> to add the server to the "Synchronize with first available" list.
Remove	Select a computer from the list above, and click "Remove" to remove it from the list.

**Note:** When you add a time source to the list, the current time on that machine will be displayed, provided the service is running on that remote machine and listening on the same port number. If "Not available" then the service is not running, or is running and using a different port number, or that port number is being blocked by a firewall. The column in the list box is provided as a diagnostics function to confirm that the machine names entered can be synchronized against. The time displayed in this box is an approximate only.

## Runtime - Frequently Asked Questions

### Q: How do I run a Cicode function on startup?

**A:** Specify the Cicode function with the Setup Wizard (in Custom mode). Use the **Startup Functions Setup** page. See [Startup Functions Configuration](#) in the Plant SCADA documentation for details.

**Q: How do I run a report on startup?**

A: Plant SCADA searches for a report called "Startup" when it starts up. If Plant SCADA locates this report, it is run automatically. You can change the name of the default startup report with the Setup Wizard (in Custom mode). Use the **Startup report** field on the Reports Setup page. See [Reports Configuration](#) in the Plant SCADA documentation for details.

**Q: How do I remove the Cancel button from the Startup Message Box?**

A: Use the Setup Wizard (in Custom mode). De-select the **Display Cancel button on startup** option on the Security Setup - Miscellaneous page. See [Miscellaneous Security Configuration](#) in the Plant SCADA documentation for details.

**Q: How do I remove the Shutdown command from the Control menu?**

A: Use the Setup Wizard (in Custom mode). De-select the **Shutdown on menu** option on the Security Setup - Control Menu page. See [Control Menu Security Configuration](#) in the Plant SCADA documentation for details.

**Q: How do I remove Plant SCADA Studio/Graphics Builder commands from the Control Menu?**

A: Use the Setup Wizard (in Custom mode). De-select the **Plant SCADA Studio on menu** option on the Plant SCADA Runtime control menu page. See [Control Menu Security Configuration](#) in the Plant SCADA documentation for details.

# Runtime Client Tools

Plant SCADA includes a set client tools that are designed to operate in the runtime environment. They include:

- **Process Analyst** - displays trend and/or alarm tag data (both real-time and historical) for system analysis during runtime (see [Process Analyst](#))
- **Scheduler** — allows you to view and manage schedule entries for a selected piece of equipment to enable automated operation of a Plant SCADA system (see [Schedules](#)).

## See Also

[Runtime Manager](#)

## Process Analyst

Process Analyst allows operators to view trend and/or alarm tag data (both real-time and historical) for comparison and analysis during runtime through their existing Plant SCADA server architecture.

**Note:** You can access the Process Analyst documentation from the Process Analyst main toolbar and the Process Analyst Properties dialog box.

## See Also

[The Process Analyst Interface](#)

[Pens](#)

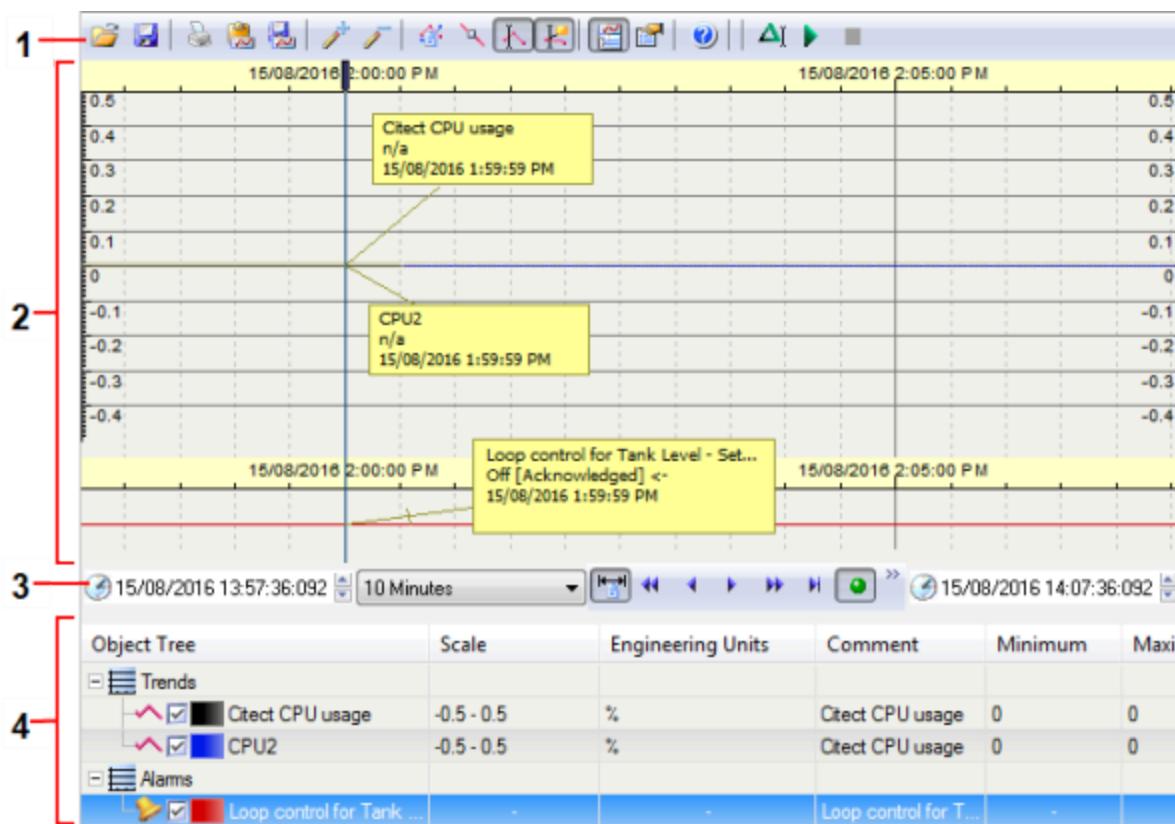
[Cursors](#)

[Interacting with the Process Analyst](#)

[Configure the Process Analyst at Runtime](#)

## The Process Analyst Interface

The Process Analyst interface comprises the following components:



#### 1. Main toolbar

Contains commands for performing general operations in the Process Analyst, such as opening views, printing reports, and so on. You can customize this toolbar. See [The Main Toolbar](#).

#### 2. Chart view

The chart view displayed runtime and historical data using a set of pens that draw sample values against time. Pens can be grouped within the chart view using panes.

#### 3. Navigation toolbar

Contains commands that allow an operator to travel forward or backward through trends, as well as other navigation-related tasks. You can customize this toolbar.

#### 4. Object view

When displayed, the object view presents information about your Process Analyst pens, such as name, color, scale, and so on.

## See Also

[The Chart View](#)

[The Object View](#)

[The Navigation Toolbar](#)

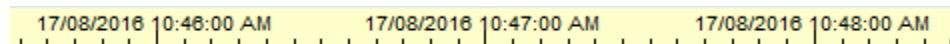
[The Main Toolbar](#)

[Process Analyst Properties Dialog Box](#)

## The Chart View

The chart view is where Process Analyst pens draw sample values against time. Pens can be grouped in the chart view using one or multiple **panes**.

The chart view displays a date/time axis at the top of each pane.



A vertical (value) axis may also appear at the left-hand edge of the pane. (The value axis is shown only for analog pens.)



### About the date/time axis

The date/time axis displays time using the current locale format specified in your computer date/time settings. If the millisecond component is necessary, it is appended to the end in the format "<xxx>ms." Since the local time zone is determined from the current computer settings, these settings need to be configured accurately. The date/time axis will accommodate daylight savings transitions.

**Note:** In order for the Process Analyst to indicate that Daylight Savings is in effect, the **Automatically adjust clock for daylight saving changes** option on the Windows™ Date and Time properties needs to be enabled.

The date/time axis can also display data using the universal time coordinate (UTC) format. You can switch between local or UTC time as you like (see [Configure Pen Axes](#)).

The date/time axis is divided into major and minor time intervals, which change dynamically depending upon the time span. In the illustration above, the major intervals are 1 minute apart, and the minor are 5 seconds apart.

Note the following:

- When the axis time span is 1 minute or less, the format of the axis labels includes milliseconds and the date is removed.
- When the axis time span is 1 week or above, the time is removed and only the date is displayed.

By default, the date/time axis displays a time span of 10 minutes; the major intervals represent 5 minutes, and the minor intervals 30 seconds.

### About the vertical (value) axis

Like the date/time axis, the value axis consists of major and minor intervals, but they represent value intervals

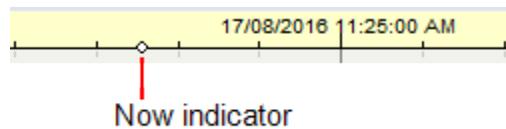
rather than date and time. The intervals are calculated automatically by the Process Analyst.

The axis displayed reflects the values for the primary selected pen.

By default the vertical axis will use the engineering scale from the tag of the selected pen. The vertical axis also supports autoscaling. When autoscaling is enabled, the vertical axis automatically adjusts its limits to accommodate new samples as they are added to each individual pen.

You can scroll and scale the vertical axis; for details, see [Scroll the Chart](#) and [Scale the Chart](#). You can also configure the appearance of the vertical axis; for details, see [Configure Pen Axes](#).

The Now Indicator is a small white circle on the date/time axis that indicates the current time based on the computer's time settings.



The position of the Now indicator is refreshed according to the value specified in the [Configure General Properties](#) field in the Process Analyst Control Properties dialog box. See Also [Synchronize to Now](#).

Process Analyst pens also use gridlines as a visual guide to help an operator determine the value of trends. Major gridlines are solid lines; minor gridlines are dashed lines. Analog pens have vertical and horizontal gridlines; alarm and digital pens only have vertical gridlines. The display of gridlines changes dynamically according to the selected time span. See [Configure Pen Gridlines](#).

## The Object View

The Object View provides a tabular view of the pens displayed in the Process Analyst. You can use the Object View to access information about the pens on the chart, along with information about associated tags. When displayed, the Object View is located under the navigation toolbar.

Object Tree	Scale	Engineering Units	Comment	Min
- Trends				
Citect CPU usage	-0.5 - 0.5	%	Citect CPU usage	0
CPU2	-0.5 - 0.5	%	Citect CPU usage	0
- Alarms				
Loop control for Tank ...	-	-	Loop control for T...	

The **Object Tree** column displays a hierarchically arranged view of the panes and pens on the chart. The column uses the following icons:

Icon	Object
	Pane
	Analog pen
	Digital pen
	Alarm pen

The check box next to a pen icon controls whether the pen is visible on the chart. The gradient-filled color box to the left of the pen name indicates the pen's line color as it appears on the chart.

Clicking a pen in the Object View selects that pen. There is always one pen selected in each pane. In the example above, the primary selected pen (Loop Control for Tank...) is highlighted in blue; the pen selected in the Trends pane (CPU2) is highlighted in gray. See [Pen Selection](#).

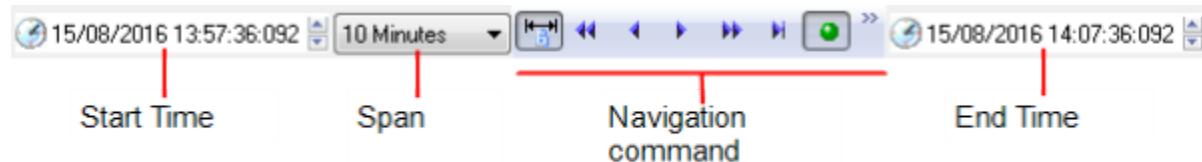
The columns that appear in the Object View table are configurable. An operator can select the columns they require from a predefined list. For more information, see [Configuring the Object View](#).

## See Also

[Use the Object View](#)

## The Navigation Toolbar

The Navigation Toolbar contains commands to allow an operator to travel forward or backward through trends, as well as other navigation-related tasks.



A **Start Time Picker** is located on the left-hand side of the navigation toolbar, an **End Time Picker** is located on the right. You can use these to [Specify a Start Time and End Time](#).

If a time picker control is using Daylight Savings time, the clock to the left of the control will have a shaded segment.



The **Span Picker** contains commonly used predefined time spans that you can select from a drop-down menu. The time span of the trend display represents the difference between the start time and the end time. Selecting a time span adjusts the start time and leaves the end time as it is. To set a time span that is not included in the list of predefined times, see [Set a Nonstandard Time Span](#). See also [Lock/Unlock the Time Span](#).

The Navigation Toolbar also includes a set of commands buttons. You can use these buttons to perform the following tasks:

- [Navigate Time](#)
- [Synchronize to Now](#)
- [Toggle Autoscrolling](#)
- [Zoom In/Zoom Out](#)
- [Undo Last Zoom](#)
- [Toggle Box Zoom](#)
- [Edit Vertical Scale](#).

You can also [Configuring Toolbars](#) to contain different items.

## The Main Toolbar

The Process Analyst main toolbar is located above the chart view. The main toolbar contains commands that allow you to perform general operations, such as save and load Process Analyst [Working with Views](#), print trend reports, add or remove pens, display or hide cursors and labels, and so on.

Toolbar commands can be customized; for details, see [Configuring Toolbars](#).

The table below describes the items that are included on the main toolbar by default.

Item	Description
	<b>Load View.</b> Loads a saved view from file. For details, see <a href="#">Load a View</a> .
	<b>Save View.</b> Saves a view to file. For details, see <a href="#">Saving a View</a> .
	<b>Print.</b> Displays the standard Windows Print dialog box for printing trend reports. For details, see <a href="#">Printing and Exporting</a> .
	<b>Copy to Clipboard.</b> Copies visible pens to the Windows Clipboard. For details, see <a href="#">Copying Data to the Clipboard</a> .
	<b>Export to File.</b> Exports visible pens to an Excel-compatible file. For details, see <a href="#">Copying Data to File</a> .
	<b>Add Pen.</b> Displays the Add New Pen(s) dialog box for adding a pen. For details, see <a href="#">Add Pens</a> .
	<b>Remove Pen.</b> Deletes the currently selected pen from the trend display. For details, see <a href="#">Delete Pens</a> .
	<b>Lock/Unlock Pens.</b> Toggles the locking of pens. For details, see <a href="#">Lock/Unlock Pens</a> .
	<b>Show/Hide Points.</b> Toggles the display of points representing where sample data was recorded in the archive. For details, see <a href="#">Pens</a> .
	<b>Show/Hide Cursors.</b> Toggles the display of cursors. For details, see <a href="#">Cursors</a> .
	<b>Show/Hide Cursor Labels.</b> Toggles the display of cursor labels. For details, see <a href="#">Using Cursor Labels</a> .
	<b>Toggle Object View.</b> Toggles the display of the Object View. For details, see <a href="#">The Object View</a> .
	<b>Properties.</b> Displays the Properties dialog box for

Item	Description
	configuring the Process Analyst control. For details, see <a href="#">Configure the Process Analyst at Runtime</a> .
	<b>Help.</b> Displays the Process Analyst online Help.

## See Also

[The Navigation Toolbar](#)

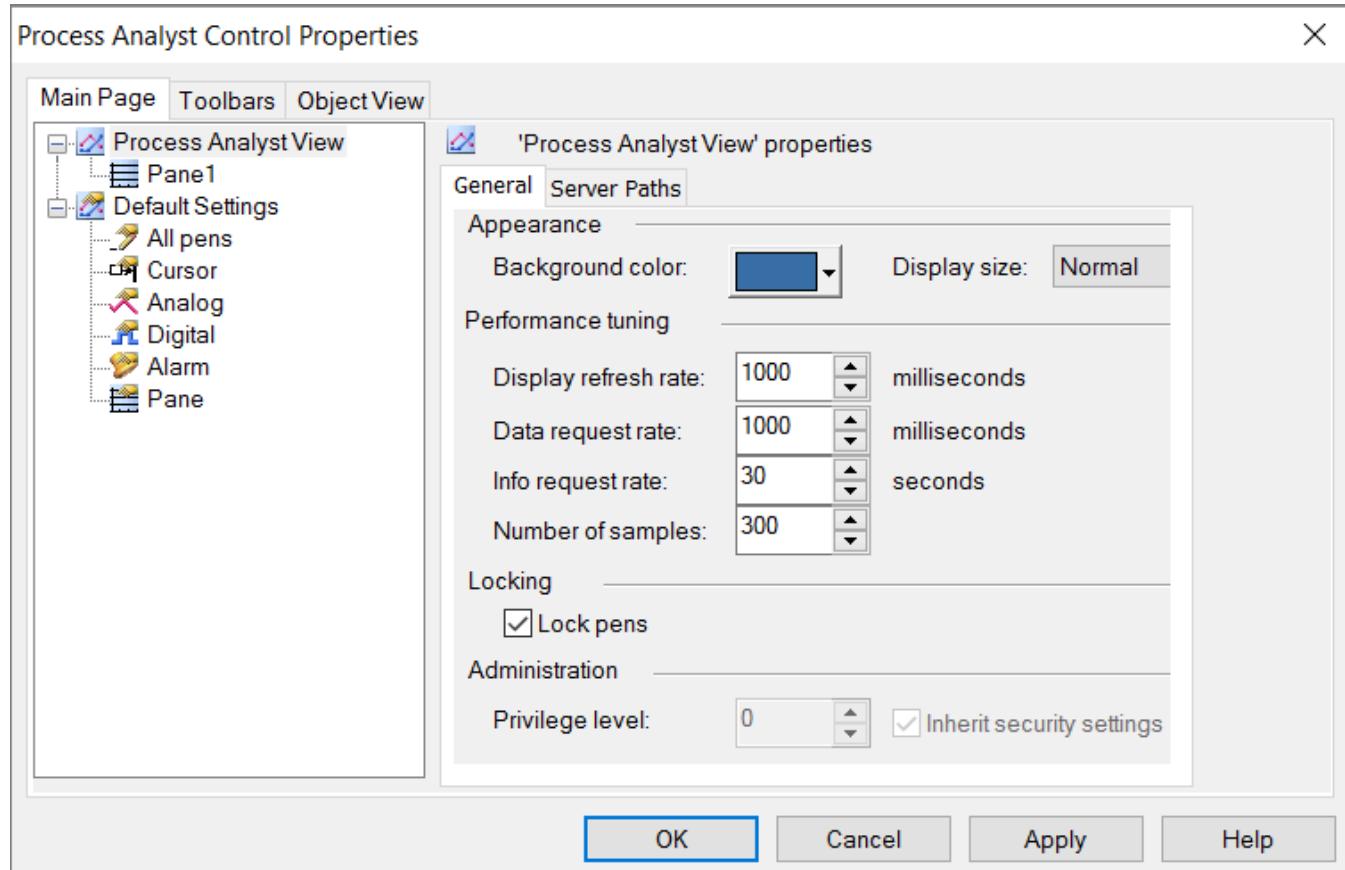
## Process Analyst Properties Dialog Box

You use the Process Analyst Properties dialog box to configure Process Analyst views. You can also configure chart-wide properties.

The Properties dialog box has three tabs: Main page, Toolbars, and Object View.

### Main page

You use the Main page of the Properties dialog box to configure general properties and access the server path properties. The Main page looks like this:



The list on the left-hand side contains the property tree, a hierarchical list of Process Analyst interface components. Selecting an item displays the property controls for that component on the right. The pens in the property tree indicate the information that the pen is trending. See [Configure General Properties](#) and [Configure Server Paths](#).

## Toobars

You use the Toolbars page to configure the main toolbar and navigation toolbar. Operators and Users can configure the toolbars at run time and design time. Use the Toolbars page to configure the toolbars; for details, see [Configuring Toolbars](#).

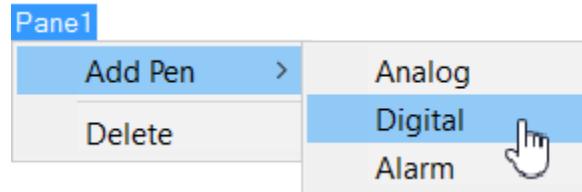
## Object View

You use the Object View page to configure the Object View. Operators and Users can select (at run time and design time) the columns they want to display, as well as change the column width and display order. Users can define new columns and edit existing columns at design time.

See [Configuring the Object View](#).

## Using the Property Tree Right-click Menu

Right-clicking an item in the property tree displays the shortcut menu for that item, as shown below.



The tasks you can perform vary depending on your privilege level: if you don't have the necessary privilege at run time to perform an action, that control is disabled/removed. For example, the right-click menu removes the Add Pen option at run time if you don't have the privilege to add a pen. Commands that are unavailable appear "grayed-out." The right-click menu contains the following options:

Right-click this item...	Actions
Chart	Add Pane - add a new pane. Add Cursor - add a new cursor.
Pane	Add Digital - adds a new digital pen. Add Analog - adds a new analog pen. Add Alarm - adds a new alarm pen. Note: After adding a pen from this menu, configure the data connection by clicking the Connection tab and typing the name of the tag into the Tag text box. Delete - deletes the pane.
Pen	Delete - deletes the pen.

Right-click this item...	Actions
Cursor	Delete - deletes the cursor.

Use the Main page for the following:

- [Configure Chart-wide Properties](#)
- [Configure Chart Panes](#)
- [Configure Pens](#)
- [Configure Cursors](#)
- [Configure Defaults](#)

## Pens

Process Analyst pens draw sample values against time. Each pen has its own colored line in the chart view (and can contain other graphical elements).

Sample markers (or points) are drawn on the line to indicate where data was recorded in the archive. The style of the line indicates the quality of the data (see [Data Quality](#)). The style of the sample marker indicates the compaction of the sample (see [Data Compaction](#)).

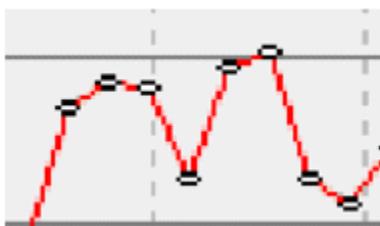
### The Process Analyst supports three types of pen:

- **Analog pens**

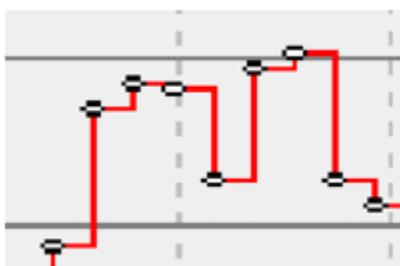
The Process Analyst control typically uses analog pens to represent variable or numerical data. Only analog pens have a value (vertical) axis, which the data is plotted against.

Analog pens have two types of interpolation that allow you to specify how to connect data samples on a trend line:

- **Straight** - a line is drawn directly between the points or sample values like this:



- **Stepped** - the lines drawn always maintain the value of the previous sample until a sample with a different value arrives, in which case a vertical line is drawn:



See Also [Interpolated Samples](#).

- **Digital pens**

The Process Analyst control typically uses digital pens to represent binary data. Values on the pen are restricted to 0 or 1. Any value equal to or greater than 0.5 is set to 1; other values are set to 0. A fill color is used to indicate where the data is 1, as shown here:



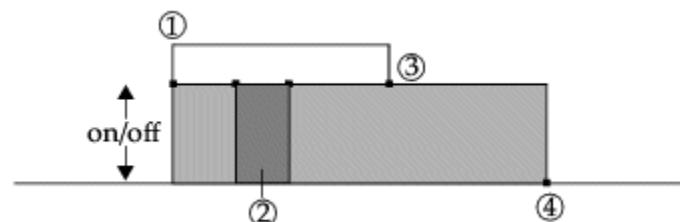
By default, the layout of digital pens is stacked. For details, see [Pen Layout](#).

- **Alarm pens**

The Process Analyst uses alarm pens to graphically display the history of Plant SCADA alarms over time. The alarm's on/off transition state changes and acknowledgment are represented graphically in the alarm pen display.

When an alarm is on, it is represented by a bar filled with color. The color indicates different states. The line above represents operator acknowledgment of the alarm.

The diagram below illustrates how an alarm pen displays the information of an alarm tag:



1. The alarm is turned on in its initial state and is unacknowledged.
2. The alarm changes to a different state, but is still unacknowledged.
3. The alarm is acknowledged.
4. The alarm is turned off.

The Process Analyst allows the appearance of pens to be configured during run time and design time. You can configure the line color, width, and fill color. For details, see:

- [Configure Analog and Digital Pens](#)
- [Configure Alarm Pens](#).

---

**Note:** To configure default appearance settings for a particular type of pen, go to the Process Analyst Properties dialog and select the pen type in the property tree under **Default Settings**.

---

## See Also

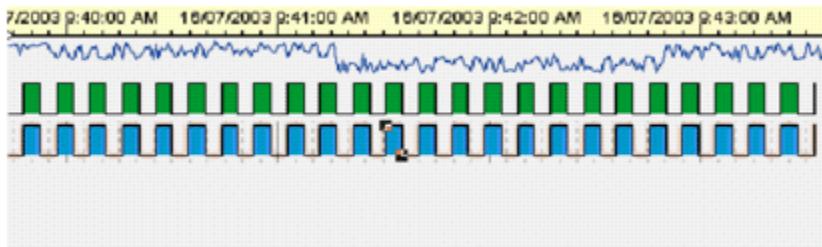
[Pen Layout](#)

[Pen Selection](#)

## Pen Layout

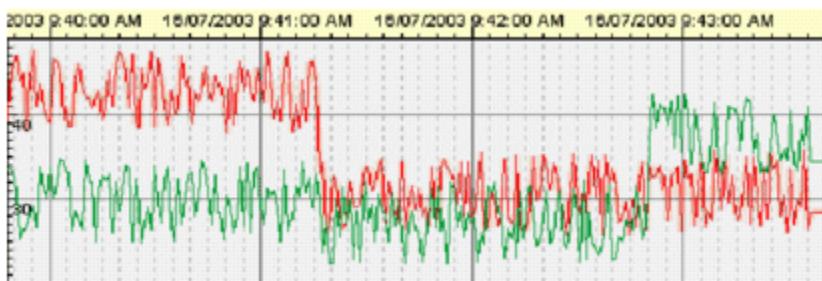
You can display pens in the Process Analyst by stacking or overlaying.

- In **stacked** mode, a user-specified amount of vertical real-estate is allocated to the pen, and with this, stacked pens are laid out under each other on the pane, starting from the top of the pane under the date/time axis, like this:



Here, three pens (one analog and two digital) are stacked under each other. Stacking applies to every type of pens.

- In **unstacked** mode, pens are drawn on top of each other. The order in which the pens were added to the pane governs the drawing order: the last pen added is the topmost pen drawn. When a pen is selected, it is brought to the front of any other pens displayed



Here, two analog pens are overlaid. You can also overlay digital and alarm pens.

You can have any mix of stacked and unstacked pens on a pane.

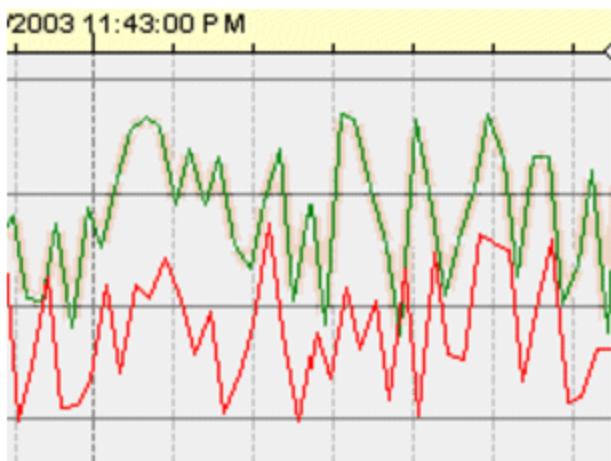
## Pen Selection

Each pane on the Process Analyst can have one selected pen. The axes that are displayed on a pane are that of the selected pen. The last pen selected across every pane is referred to as the *primary* selected pen.

You can select a Process Analyst pen in several ways:

- By clicking on the pen's graphical elements (i.e., the pen line).
- If the pens are stacked, by clicking the background under the pen line.
- By selecting the pen in the Object View.

The selection of a pen is indicated by a subtle halo effect surrounding the pen line. In the example shown here, the top (green) pen is selected, indicated by the halo surrounding the pen:

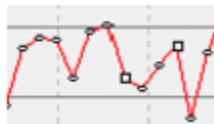


The halo does not appear if there is only one pen on the pane. Selecting a pen on a pane also causes the same pen to be highlighted in the Object View. Selecting a pen causes that pen to be drawn in front of other pens on the pane.

## Data Compaction

Data compaction is the visual representation of multiple data points that are not distinguishable due to their rate of occurrence across the currently selected time span.

Data is compacted by grouping raw samples together and visually representing them as one sample. Sample compaction is indicated on the graph by using different sample markers. For example, in the illustration below, the two sample markers that appear as squares actually represent multiple raw samples.



The following illustration zooms in on the second multiple sample. It shows that what appeared to be a single sample actually consists of several raw samples:



The Process Analyst uses the following default point styles for single and multiple samples:

Sample compaction	Point Type
Single	● Ellipse
Multiple	□ Rectangle
Interpolated	▲ Triangle (see <a href="#">Interpolated Samples</a> ).

## See Also

[Request Modes](#)

## Interpolated Samples

Normally samples are only single or multiple. But there is a specific situation in which an interpolated sample is used to correct a graph that only occurs with event trends.

The frequency of the data stored in an event trend can vary dramatically; for example, where several samples are within one display period, followed by no samples for a long time.

A multiple sample will be drawn with a value calculated from the samples within the period. But the value after that period will be whatever the last sample in the period was. So an interpolated sample is added at the start of the next display period to correct the graph.

## See Also

[Request Modes](#)

## Request Modes

When [Data Compaction](#) is used to create a graphical representation of multiple values, a calculation needs to take place to determine the value and time stamp of the value displayed. The Process Analyst provides the following options for this calculation:

- **Average** - The value will be an average of the individual samples within the multiple sample, as will the timestamp. This is the default calculation method.
- **Maximum**: The value will be the maximum value out of the individual samples within the multiple sample. The timestamp will be that of the individual sample that was the maximum.
- **Minimum**: The value will be the minimum value out of the individual samples within the multiple sample. The timestamp will be that of the individual sample that was the minimum.
- **Newest**: The value will be the latest arrived value out of the individual samples within the multiple sample. The timestamp will be that of the individual sample that was the newest.

## Data Quality

Process Analyst pens use the same sample quality values as Plant SCADA trend and alarm samples. There are four sample quality values:

- **Good** - Samples were recorded in the trend archive as good.
- **NA** - When Plant SCADA is unable to obtain a sample or the data retrieved was invalid, an N/A sample will be recorded in the trend archive.
- **Gated/Disabled** - For trends, when the trigger of a trend is off, a value of "Gated" is recorded in the trend archive. For alarms, this sample quality value indicates that the alarm has been disabled.

The Process Analyst uses the following default line styles to indicate data quality:

Quality	Line style
Good	Solid ———
NA	None
Gated	Dot ······

Consider the following examples:

Data sample	Description
	This example shows several single samples. The third sample has a quality of N/A, indicated by the break in the trend line.
	Here the quality of the third sample is gated, indicated by the broken line connecting these samples.

With multiple samples, the quality of the last sample in the group determines how the line is drawn. Consider the following examples:

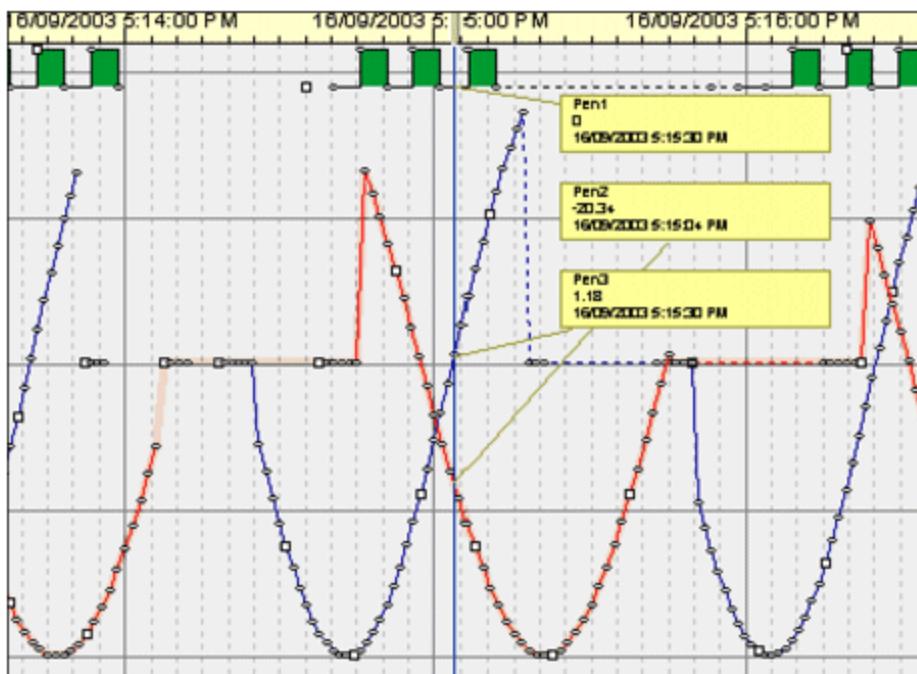
	This example shows that the third sample is a multiple sample. The quality of the third (multiple) sample and the next sample are N/A, indicated by the break in the trend line.
	Here the quality of the third multiple sample is gated, indicated by the broken line connecting the samples.

The line style indicating the data quality is configurable during run time and design time; for details, see [Configure Pen Quality](#).

## Cursors

A cursor enables an operator to determine the value of a pen at a given point in time by dragging the cursor to the specific point on the pen line. A [Using Cursor Labels](#) is used to display the value.

An Operator can define many of the properties of cursors and cursor labels. For details, see [Configure Cursors](#).



In this example the cursor intersects three pens; the cursor labels (the yellow rectangles) display the corresponding pen values.

To move a cursor, drag the cursor line left or right. As the cursor moves, the cursor labels move with the cursor and are updated continuously, reflecting the position of the cursor.

**Note:** The cursor extends across every configured pane.

A line connects the cursor label to the associated pen line. The line has three main states:

State	Style	Example
Intersection within pen data	Line	
Intersection before or after pen data	Line with indicator	
No intersection and no data	Invisible line	

#### To show/hide a cursor:

- Click **Show/Hide Cursor** on the main toolbar. You can display additional cursors by using the Properties dialog box.



You can display as many cursors as you want. To add a cursor, right-click the root item (**Process Analyst View**) in the property tree in the Properties dialog box, and choose **Add Cursor**.

## Using Cursor Labels

Each cursor has one cursor label for each pen displayed. The cursor label displays the value of the pen at the point where the cursor intersects with the pen data.

An alarm pen label value has the following format:

*State [acknowledgement]*

*State* refers to the alarm state at the point of intersection, and *acknowledgement* refers to the acknowledged state of the alarm at the point of intersection.

### To display cursor labels:

- Click **Show/Hide Cursor Labels** on the main toolbar.



This table summarizes how to use cursor labels:

Task	Description
Move a cursor label	Click the cursor label and drag the label to a new location.
Change the size of cursor labels	Click the cursor label you want to re-size. Place the mouse cursor on one of the sizing boxes, and drag the label to the new size. If you drag the corner of the label, the label text resizes to an optimal size for the label.
Lock or unlock the cursor labels	Click the <b>Lock/Unlock Cursor Labels</b> . When on, this command causes cursor labels to be "frozen" in the position.

The cursor label displays the following information:

Cursor field	Applies to	Description
Pen Name	All Pen types	Displays the non-unique Process Analyst pen name
Value/Quality	All Pen types	Displays the value of the pen at the point the cursor intersects with the pen data
Date-Time Stamp	All Pen types	Displays the date/time stamp at the point the cursor intersects with the date/time axis.
Alarm Sample Comment	Alarms	Comment bound to an alarm sample.

The fields are displayed in the cursor label using the order defined above using the format specified for the

vertical axis. For example, if your vertical axis format is "km/h", the label reads "<value> km/h".

The label displays the following values when the quality of the data is not good:

<b>Cursor value</b>	<b>Description</b>
NA	At the point of intersection the pen has no available data for display.
Gated	At the point of intersection the pen's data has been gated.
Disabled	At the point of intersection the alarm tag of the pen was disabled.

The label value can also contain a directional indicator that functions as follows:

<b>Cursor value</b>	<b>Description</b>
<value> ->	The cursor is to the left of the first available sample for this pen.
<- <value>	The cursor is to the right of the last available sample for this pen.

## Mouse Pointers

When using the Process Analyst, the mouse pointer changes shape to indicate the operations you can perform at that time.

<b>Mouse pointer</b>	<b>Region</b>	<b>Description</b>
	Pen line	The mouse pointer looks like this when the pointer is on a pen. Clicking the mouse at this point selects the pen.
	Pen line/pen background	The mouse pointer looks like this when the mouse is over a pen's background and both horizontal and vertical scrolling are enabled. Clicking and dragging at this point results in the free movement of the pen. Scrolling the mouse wheel results in horizontal-only movement.
	Horizontal axis	The mouse pointer looks like this when the pointer is on the horizontal axis and horizontal scaling is enabled. Clicking and

Mouse pointer	Region	Description
		dragging (or scrolling the mouse wheel) will result in the axis being scaled.
	Pen line/pen background	The mouse pointer looks like this when the pointer is on the horizontal axis and only horizontal scrolling is enabled. Clicking and dragging (or scrolling the mouse wheel) will result in the axis being scrolled.
	Vertical axis	The mouse pointer looks like this when the pointer is on the vertical axis and vertical scaling is enabled. Clicking and dragging (or scrolling the mouse wheel) will result in the axis being scaled.
	Vertical axis	The mouse pointer looks like this when the pointer is on the vertical axis and only vertical scrolling is enabled. Clicking and dragging (or scrolling the mouse wheel) will result in the axis being scrolled.
	Box Zoom mode	The mouse pointer looks like this when Box Zoom mode is enabled. See <a href="#">Toggle Box Zoom</a> .

## Interacting with the Process Analyst

This section describes the ways in which an operator can interact with the Process Analyst during runtime.

It includes the following topics:

- [Add Pens](#)
- [Delete Pens](#)
- [Lock/Unlock Pens](#)
- [Specify a Start Time and End Time](#)
- [Shift and Fit Time Units](#)
- [Scroll the Chart](#)
- [Scale the Chart](#)
- [Lock/Unlock the Time Span](#)
- [Navigate Time](#)

- [Synchronize to Now](#)
- [Toggle Autoscrolling](#)
- [Zoom In/Zoom Out](#)
- [Undo Last Zoom](#)
- [Toggle Box Zoom](#)
- [Set a Nonstandard Time Span](#)
- [Edit Vertical Scale](#)
- [Reset to Default Span](#)
- [Use the Object View](#)
- [Using Instant Trends with Process Analyst.](#)

You can use the right-click (context) menu to quickly access frequently used commands. This menu is context-sensitive, providing relevant commands for different regions of the display. The right-click menu appears when you click any of the following regions:

- Horizontal axis
- Vertical axis
- Background
- Pen

## Add Pens

The Process Analyst allows operators to search the trend tags and alarm tags that are defined on a Plant SCADA trends or alarms server. You can use the search results to add pens to the current trend display.

You use the Add New Pens dialog box to add a new pen to your trend display. To display the Add Pens dialog box, click **Add Pens** on the main toolbar.

### Adding a new pen:

To identify the item you would like a pen represent, you need to initially define a search to determine what is available. You can then select a particular item from the search results.

1. Search the available data on the SCADA server:
  - a. Select the **Type** of data you want to search for. The options are **Alarms** or **Trends**.
  - b. Type in a **Tag**, **Cluster** or **Comment** filter to help refine the search results. Be aware that if the type selected is Alarm, the Comment filter will not be available.

The filters have basic wildcard and Boolean search functionality. You can use the keywords AND, OR and NOT with wildcard strings, as well as group Boolean terms using parentheses. Searches are not case sensitive.

For example, entering "L\*" in the Tag Filter returns tags beginning with the letter "L" in every cluster. Entering "L\* OR H\*" will find tags beginning with "L" or "H". More complex examples include "L\* OR (H\* AND NOT \*G)". This would return tags that start with "L" or any that start with "H", but do not end in "G". If you leave a field blank, each item of the selected type will be retrieved.
- c. Click **Search**. The results are returned in the **Search results** list.

The results are not sorted: the tags appear in the order they were configured in Plant SCADA. The cluster associated with each tag is also displayed. The Process Analyst displays only the tags in clusters that this client has access to. In a system with more than one cluster, if a tag is not configured with a cluster, it is listed once for each cluster.

2. The Search Result list displays up to 100 entries at a time. If your search returns more than 100 results, use the **First**, **Prev**, **Next**, and **Last** buttons to navigate your search results.
3. Select one or more items from the **Search Results** list. You can use the **Ctrl** and/or **Shift** keys to select multiple tags.
4. Select the destination pens to Add **Add pens to**. Pens can be added to any existing pane, or to a new pane.
5. Select a **Pen Type**. A trend tag can be represented by an analog or digital pen. An alarm tag can be represented by an alarm pen only.
6. Select how to resolve the pen name:
  - **Comment**- applies the tag comment as the pen name. If the tag does not have a comment specified, a name is automatically generated.
  - **Tag**- applies the tag name as the pen name.
  - **Auto**-applies an automatically generated name to the pen using the format *Pen<X>* where *X* is an incremented number, starting with the first available number.
7. Click **Add**. This moves the selected items in the Search Results list into the Selected Items list. The Selected Items list contains the tags that will be added as pens to the Process Analyst. You can perform multiple searches to add tags into the Selected Items list.

---

**Note:** To remove a tag from the Selected Items list, highlight the item you want to move, and then click **Remove**.

---

8. To view details about a selected tag, click **Show Detail**. A Pen Details dialog box appears, showing defined information for the selected tag.
9. Click **OK**. Your selected tags appear on the trend display as pens.

## See Also

[Delete Pens](#)

## Delete Pens

Operators can delete pens from the trend display at any time.

---

**Note:** Deleting a pen is different than hiding the pen from display by using the Visibility check box in the Object View. For details, see [Use the Object View](#).

---

### To remove a pen:

1. [Pen Selection](#) you want to delete.
2. Click **Remove Pen** in the main toolbar.



The pen is deleted from the display.

## See Also

[Add Pens](#)

## Lock/Unlock Pens

By default, the Process Analyst locks together the time span and position in time (horizontal axis) of every pen. However, you can unlock the pens, allowing the pens to be displayed across different positions in time and/or time spans.

For example, you could unlock pens to compare a previous month's data for a tag with the data for this month. You would do this by adding two pens to a pane that represent the same tag, then unlocking the pens, and adjusting the time positions for each pen as necessary.

To control pen locking and unlocking, you use the **Lock/Unlock Pens** button on the main toolbar.



This option is also available on the right-click (context) menu.

Locking and unlocking has the following behavior:

- When pens are locked, every time-related operation is applied to every pen.
- When pens are unlocked, every time-related operation is applied to the primary selected pen.
- Synchronization applies every pen regardless of their being locked or unlocked.

When transitioning from locked to unlocked, the time span and position in time of every pen are synchronized to match that of the primary selected pen.

## See Also

[Add Pens](#)

[Delete Pens](#)

## Specify a Start Time and End Time

You can specify a start time and an end time for the trend display by using the date/time pickers. The start time picker is located on the left-hand side of the navigation toolbar, the end time picker on the right.



The date/time picker formats the date and time using the settings obtained from your computer for the currently logged in user. The date/time picker displays time in 24-hour format (*dd/mm/yyyyhhmm:ssnnn*) where:

- *dd* represents days
- *mm* represents months
- *yyyy* represents years
- *hh* represents hours
- *mm* represents minutes

- *ss* represents seconds
- *nnn* represents milliseconds (added automatically to the time)

### To change the date or time in the date/time picker:

1. Click the element of the date or time you want to change in the start time picker or the end time picker.
2. Do either of the following:
  - Type in a time explicitly.
  - Press the **Up arrow key** or **Down arrow key** to increment or decrement the value respectively.

**Note:** You can use the **Left arrow** and **Right arrow** keys to move between time elements.

---

## Shift and Fit Time Units

You can manipulate the start time and end time by using special keyboard shortcuts. Using these shortcuts, you can do the following:

- Shift by unit
- Fit to unit.

### Shift by unit

Shifting date or time by unit allows you to change the opposite date/time element to the one selected by the corresponding date or time component. For example, if you shift by unit the month time element in the start time, the month time element in the end time increments by one month exactly, including days, minutes, and seconds. This also works for months that have different end days.

#### To shift by unit:

1. Press and hold down the **Shift** key.
2. Click a date or time element in the date/time picker. The opposite time picker changes by the base time amount of the selected time element.

### Fit to unit

Fitting date or time to unit allows you to synchronize the selected time element to the zero position of that time element in the start time and end time. For example, an operator clicks on the *hh* time element of the Start picker, which shows 19:30:05.123. After **Ctrl + click**, the Start hour time element shows 19:00:00.000, and the End time element shows 20:00:00.000. Now the time span represents exactly one hour, synchronized on the hour.

#### To fit to unit:

1. Press and hold the **Ctrl** key.
2. Click a date or time element in the date/time picker. Both the start time and end time element are synchronized to zero based on the date/time element selected.

## Scroll the Chart

The Process Analyst allows you to scroll through data in both the horizontal and vertical directions by dragging the mouse or spinning the mouse wheel.

### To scroll by dragging:

1. Click and hold down the left mouse button on the pen (or background) that you want to scroll.
2. Drag the mouse in the direction you want to scroll:
  - **Horizontal axis:** drag right to move backward in time, drag left to move forward.
  - **Vertical axis:** drag up to scroll down the axis, drag down to scroll up the axis.
3. Release the left mouse button to complete the scrolling.

### To scroll by using the mouse wheel:

1. Click the pen or background that you want to scroll.
2. Spin the mouse wheel in the direction you want to scroll:
  - **Horizontal axis:** spin up to move backward, spin down to move forward.
  - **Vertical axis:** spin up to scroll up the axis, spin down to scroll down.

You can disable scrolling in the horizontal direction, the vertical direction, or both by using the Property dialog box or the right-click (context) menu (see [Configure Pen Axes](#)).

The Process Analyst indicates whether scrolling is enabled or disabled by displaying a different-shaped mouse pointer; for details, see [Mouse Pointers](#).

## Scale the Chart

The Process Analyst allows you to change the scale of the data in both the horizontal and vertical direction by dragging the mouse or spinning the mouse wheel.

### To scale the data by dragging:

1. Click and hold down the left mouse button on the axis that you want to scale.
2. Drag the mouse in the direction you want to scale:
  - **Horizontal axis:** drag left to expand the scale, drag right to shrink.
  - **Vertical axis:** drag up to expand the scale, drag down to shrink.
3. Release the left mouse button to complete the scaling.

### To scale by using the mouse wheel:

1. Click the axis that you want to scale.
2. Spin the mouse wheel in the direction you want to scale:
  - **Horizontal axis:** spin up to shrink the axis, spin down to expand.
  - **Vertical axis:** spin up to expand the axis, spin down to shrink.

You can disable scrolling in the horizontal direction, the vertical direction, or both by using the Property dialog box or the right-click (context) menu (see [Configure Pen Axes](#)).

The Process Analyst indicates whether scaling is enabled or disabled by displaying a different-shaped mouse pointer (see [Mouse Pointers](#)).

## Lock/Unlock the Time Span

When the time span is locked and the start time and/or end time picker changed, the current time span is maintained. If the time span is unlocked, the time span is not maintained when any of the time pickers are changed.



By default, the span is locked. You can toggle span locking on or off by using the **Span Lock** button.

## See Also

[The Navigation Toolbar](#)

## Navigate Time

The navigation controls allow an operator to navigate backwards or forwards through time. The amount of time moved depends upon the time currently selected in the Span Picker. For example, if 10 minutes is selected in the Span Picker and **Back One Span** is clicked, the display moves back 10 minutes into the pen's history.

The following navigation controls are available:

Navigation control	Description
	<b>Back One Span</b> - moves back one time span.
	<b>Back Half a Span</b> - moves back half a time span.
	<b>Forward Half a Span</b> - moves forward half a time span.
	<b>Forward One Span</b> - moves forward one time span.

## See Also

[The Navigation Toolbar](#)

## Synchronize to Now

The Synchronize to Now command synchronizes every pen such that the date/time reflects "Now," which is positioned on the right-hand edge of the screen. "Now" is calculated using the current system time.



The Synchronize to command is also available from the right-click (context) menu.

## See Also

[The Navigation Toolbar](#)

## Toggle Autoscrolling

When Autoscroll is turned on, as time passes the position in time of pens moves by the same amount to keep pace; by default, the display is updated every second. The refresh rate of the display can be controlled by using the [Configure General Properties](#) property.

When Autoscroll is turned off, as time passes the position in time of pens remain fixed.

By default, Autoscroll is on. You can toggle Autoscrolling on or off by using the **Toggle Autoscrolling** button.



Using the navigation controls, including the Time Span picker, causes Autoscrolling to be turned off

The Autoscroll command is also available from the right-click (context) menu.

## See Also

[The Navigation Toolbar](#)

## Zoom In/Zoom Out

Use the Zoom In 50% and Zoom Out 50% command buttons on the [The Navigation Toolbar](#) like this:

Command	Icon	Description
Zoom In 50%		Zooms in on the displayed data, halving the span of both axes.
Zoom Out 50%		Zooms out of the displayed data, doubling the span of both axes.

**Note:** The midpoint of each axis is maintained during these zoom operations.

## See Also

[Toggle Box Zoom](#)

[Undo Last Zoom](#)

## Undo Last Zoom

The Undo Last Zoom button on the [The Navigation Toolbar](#) allows you to undo the last zoom operation, returning the display to the previous state.



## See Also

[Zoom In/Zoom Out](#)

[Toggle Box Zoom](#)

## Toggle Box Zoom

The **Toggle Box Zoom** button on the [The Navigation Toolbar](#) switches between Box Zoom mode and normal interaction mode. In Box Zoom mode, you can define an area of the chart to zoom in on for more detail.

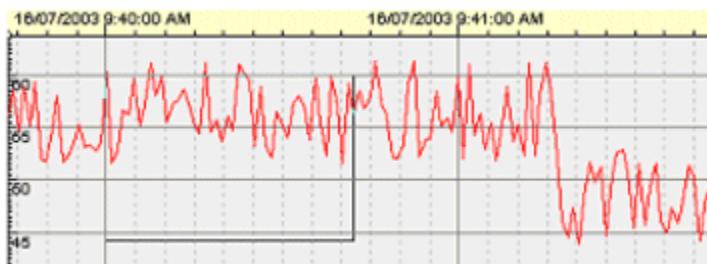
### To use Box Zoom:

1. Select the pen to zoom in on.
2. Click **Toggle Box Zoom** on the navigation toolbar.

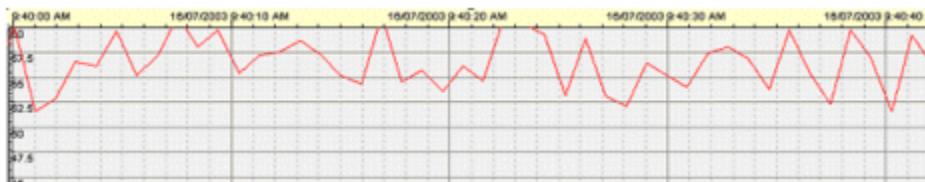


The cursor changes to a cross.

3. Click and drag the bounding box to enclose the part of the data you want to zoom in on, as shown below.



4. Release the mouse button. The display changes to a close-up of the selected data.



5. To exit Zoom mode, click the **Toggle Box Zoom** button.

Depending on whether the pens are locked or unlocked, the Toggle Box Zoom commands works differently:

- For locked pens, the zoom is applied to pens in the horizontal date/time axes. If an analog pen is being zoomed, the zoom is applied to the vertical (value) axis of non-autoscaled analog pens in the pane in which the zoom box was initiated.

- For unlocked pens, the zoom is applied only to the selected pen in both the date/time and vertical (value) axes. The value axis is only affected if autoscale is off.

**Note:** Vertical zoom is only applied to analog pens, since it has no effect with alarm or digital pens.

## See Also

[Zoom In/Zoom Out](#)

[Undo Last Zoom](#)

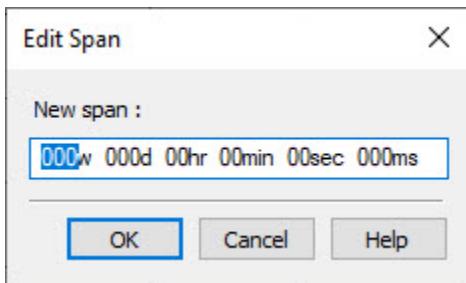
## Set a Nonstandard Time Span

Click the **Edit Span** button to display the Edit Span dialog box, which allows you to set non-standard time spans.



**To edit a nonstandard time span:**

- Click **Edit Span** on the [The Navigation Toolbar](#). The Edit Span dialog box appears.



The fields provided are: **w** = weeks, **d** = days, **hr** = hours, **min** = minutes, **sec** = seconds, and **ms** = milliseconds.

- Enter a **New span**. Click the element of the time span that you want to change, then either type in a new value, or use the **Up arrow** or **Down arrow** to specify a new value. You can use the **Right arrow** and the **Left arrow** key to move between the time elements.
- Click **OK**. The new time span is applied.

## Edit Vertical Scale

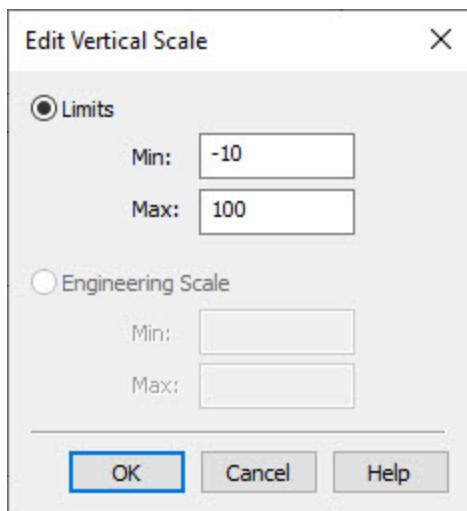
The Process Analyst allows operators to edit the vertical scale of a selected analog pen to display more appropriate values, if necessary. The vertical scale for digital or alarm pens cannot be edited.

**To edit the vertical scale:**

- Click the **Edit Vertical Scale** button on the [The Navigation Toolbar](#).



The Edit Vertical Scale dialog box appears.



2. Click the **Limits** or **Engineering Scale** option. The Limits values displayed are the current values used by the vertical scale. The Engineering Scale values are obtained from the trend tag.
3. Enter a new **Minimum** value and **Maximum** value, and then click **OK**.

## Reset to Default Span

Use the **Reset to Default Span** button to reset the time span to the default time span of the primary selected pen.



The default span can be configured by using the Property dialog box. For details, see [Configure Pen Axes](#).

## See Also

[Configuring Toolbars](#)

## Use the Object View

The table describes how to perform basic functions with Object View.

Task	Description
Toggle the display of Object View on or off	Click <b>Toggle Object View</b> on the main toolbar.
Change the size of Object View	Drag the splitter bar that is located between the chart area and the Object View up or down.
Expand or collapse a tree node in the <b>Object Tree</b> column	Either click the (+) box to expand the node or the (-) box to collapse the node; or double-click the item to toggle between expanded and collapsed states. This does not affect the display of panes in the chart.

Task	Description
Select a pen	Click the pen in the Object View table. Selecting a pen in the Object View gives the focus in the chart to the selected pen, and vice versa. You can only select one pen per pane at a time (you cannot select a pane).
Display or hide a pen	Click to clear the check box to hide the pen; click the check box again to display the pen.
Dynamically change the width of a column during display	<p>Drag the column divider left or right.</p> <p><b>Note:</b> You can quickly re-size a column to fit the size of the widest item in a column by double-clicking a column separator. To re-size the column back to its original size, double-click the separator again. You can also configure the width of a column via the Process Analyst Properties dialog; for details, see <a href="#">Configuring the Object View</a>.</p>

## See Also

[Configuring the Object View](#)

## Using Instant Trends with Process Analyst

At runtime from the Process Analyst you can browse variable tags or local variables directly, and subscribe for instant trending.

When you select a pen, in addition to Alarm and Trend Tags, you have the option of selecting Variable Tags or Local Variables for subscription and display on the trend graph.

---

**Note:** Displayed values will only be temporarily cached and will not be persisted.

### To use instant trends at runtime in the Process Analyst:

1. Display a page at runtime containing a Process Analyst.
2. Click on the 'Add Pens' button in the tool bar.
3. For the 'Type' field, select 'Variable Tags' or 'Local Variables'. This will allow you to browse for the tag you wish to instant trend.
4. Click the search button.
5. From the search results select which tag you wish to Instant Trend and click 'Add', as per usual.
6. Click 'OK' to add the pen to the Process Analyst.

## Configure the Process Analyst at Runtime

Many of the Process Analyst control's properties can be configured at run time to allow an operator to customize the control to suit their working preferences. To configure the Process Analyst, you use the [Process Analyst](#)

Properties Dialog Box.

## See Also

[Configure Chart-wide Properties](#)

[Configure Chart Panes](#)

[Configure Pens](#)

[Configure Cursors](#)

[Configure Defaults](#)

[Configuring Toolbars](#)

[Configuring the Object View](#)

[Working with Views](#)

## Configure Chart-wide Properties

You use the Main page of the [Process Analyst Properties Dialog Box](#) to configure chart-wide properties. Select **Process Analyst** at the top of the property tree to display the Process Analyst properties page. This page contains two tabs, **General** and **Server Paths**, used to modify the following configurations:

- [Configure General Properties](#)
- [Configure Server Paths](#)

## Configure General Properties

You can configure general properties such as the background color of the chart, the refresh rate, data request rate, number of samples for pens, and specify whether chart pens are to be locked. The Administration area indicates the privilege setting for the current Operator and whether security settings are inherited from the graphics page containing the Process Analyst.



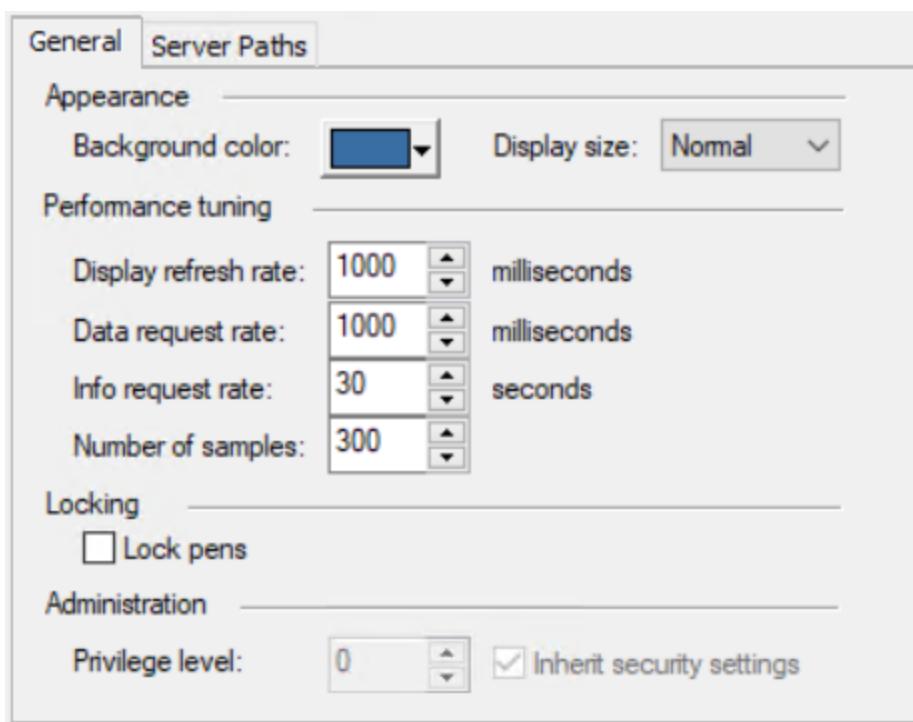
### UNACCEPTABLY SLOW PROGRAM EXECUTION

- Do not specify a Display refresh rate less than 500 ms.
- Do not specify a Number of Samples greater than 500.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**To configure general properties:**

1. Click the **General** tab on the Main page.



2. Click the color swatch and select a **Background color**.
3. Select a **Display size**: Use this setting to set the size of the fonts and other user interface elements to suit your screen size.
4. Specify a **Display refresh rate**.

This value determines the rate at which the display data is refreshed on the display; it also controls how often the position of the Now indicator is refreshed. This control is disabled if you do not have appropriate privilege. The default value is 1000 milliseconds. The permitted range is between 10 milliseconds to 60,000 milliseconds. Specifying a rate below 500 is not recommended if your chart contains many pens, since this may negatively affect performance.

5. Specify a **Data request rate**.

This value determines the maximum frequency of data requests. The Process Analyst internally determines when a request is necessary, but you can use this property to cap the Process Analyst's performance.

This control is disabled if you do not have appropriate privilege. The default value is 1000 milliseconds. The permitted range is between 10 to 60,000 milliseconds. This property affects Trends Server performance.

6. Specify a **Info request rate**.

The Process Analyst will periodically refresh the configuration data for alarm and trend pens to reflect any changes caused by a reload on the server. This property defines (in seconds) how often the server is checked for pen configuration changes.

You can use any value between 0 and 600 (seconds); the default value is 30 (seconds). A setting of 0 (zero) means no automatic refresh will occur.

The functionality of Info Request Rate is associated with online changes.

The behavior is that Process Analyst periodically checks for changes in properties of the underlying tag of the trended pens and update them.

The periodic updates cause change in the vertical scales of analog trends and update the ENG\_ZERO and ENG\_FULL in "Engineering Scales" column as well as other available properties in the Object View.

However, once changed manually by the user, the vertical scale of an analog pen is not subsequently altered by periodic Info Requests.

When saved and restored from PAV file, the vertical scales will remain as manually changed by the user and not updated as a result of the periodic Info Requests.

#### 7. Specify a Number of Samples.

This specifies the date/time axis span of each pen in number of samples. This control is disabled if you do not have the appropriate privilege. The default value is 300. The permitted range is between 10-5000. This number is dependent on the [Trend]MaxRequestLength parameter in the Citect.ini file. The default value of [Trend]MaxRequestLength is set to 4000, this will only allow a maximum of 3330 samples to display. If you wish to display the maximum number of 5000, set the [TREND]MaxRequestLength parameter to 6004 or greater. This is calculated by the number of samples (5000) plus 20% plus an additional 4 samples.

---

**Note:** This value is closely tied to your display resolution. The default setting is ideal for screen resolutions from 1024x768 to 1280x1024. The association between Number of Samples and the display resolution occurs because for each sample shown on screen the Process Analyst attempts to leave a small gap to allow for sample markers. Because the Process Analyst shows samples when they occur, it requires less data than a traditional trend client. Retrieving data is expensive and the more data you retrieve the more time the request takes. *It is recommended that this parameter not exceed 500.*

---

The chart has a minimum resolution of one millisecond per sample. If the time span is reduced enough so that the number of samples exceeds the number of milliseconds in the time span, the number of milliseconds in the time span is used instead of the number of samples.

8. Select the **Lock pens** check box to lock your pens, or clear the check box to turn off pen-locking. For details on pen locking, see .
9. Set the **Administration Privilege Level** necessary for an operator to use this object/group. Select the **Inherit Security Settings** check box if you want security settings to be inherited from the page containing the PA, or clear the check box to overwrite security settings when a new .pav file is loaded.
10. Click **Apply**.

### **WARNING**

#### **UNACCEPTABLY SLOW PROGRAM EXECUTION**

Do not set the Administration Privilege level to zero on a running system.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## See Also

[Exporting Pen Data](#)

## Configure Server Paths

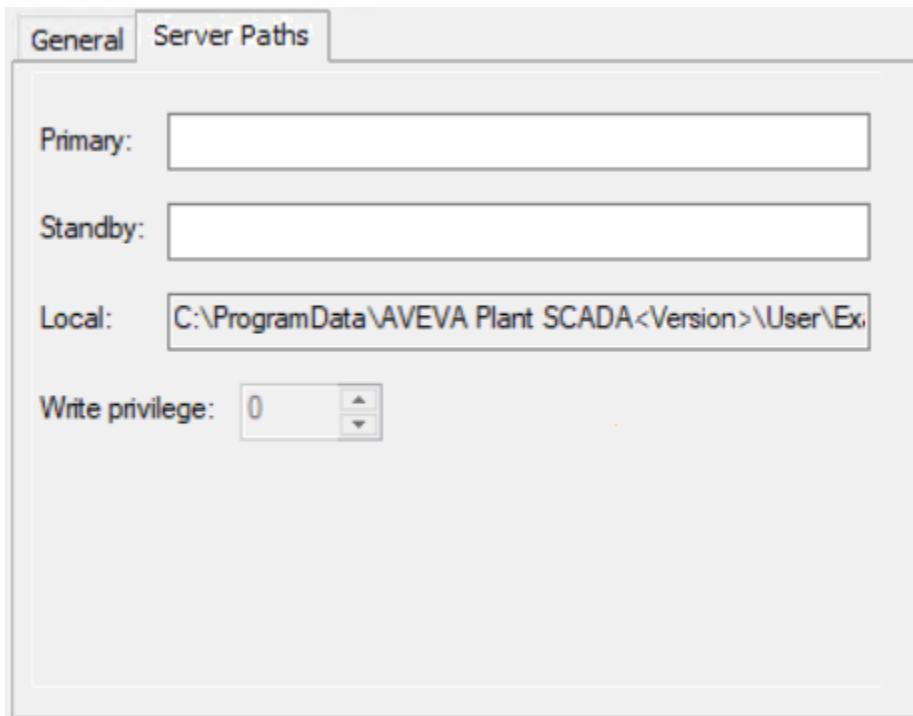
You can configure the file server locations that the Process Analyst uses to load and save Process Analyst views, and displays the current Plant SCADA run path if the Process Analyst is embedded in a running Plant SCADA system. This command is disabled at run time if you do not have the appropriate privilege. For details about saving and loading views, see [Working with Views](#) and [Process Analyst View Synchronization](#).

The Process Analyst uses four possible storage locations:

- **User** - maps to the client machine's logged-in user's **My Documents** folder. This option is available for any possible privilege and Plant SCADA mode.
- **Primary** - User-definable.
- **Secondary** - User-definable.
- **Local** - displays the current Plant SCADA run path (read-only). This text box only gets populated when the Process Analyst is running in Plant SCADA V6.0 or higher. This path is an Analyst Views subdirectory under the Plant SCADA current Run directory.

### To configure server paths:

1. Click the **Server Paths** tab on the Main page.



2. Enter the location of the **Primary** file server.
3. Enter the location of the **Standby** file server (optional). This specifies the file server to use if the primary file server is unavailable.
4. Click **Apply**.

## Configure Chart Panes

You use the Properties dialog box to configure chart panes. After adding a pane, you can configure its size relative to other panes, as well as select a different color. Pane properties can be configured during run time.

### To add a pane:

- In the property tree of the Properties dialog box, right-click the **Process Analyst view** item at the top of the tree, and then select **Add Pane**. (To remove a pane, right-click a pane in the tree and choose **Delete**.)

### To configure the chart pane:

1. In the property tree of the Properties dialog box, select the pane you want to configure. The properties for that pane appear.



**Note:** To configure defaults for your panes, select the **Pane** item in the **Default Settings** node of the property tree, not a specific pane.

2. Click the color swatch and select a new **Background color**.
3. Select a **Height** option:
  - **Variable** - Automatically calculates the pane height based on the value in the **Size** control. For example, if the chart contains two panes, selecting this option and using a **Size** value of **110** will set this pane to 110% of the size of the other pane in the chart. Fixed height panes have precedence of variable-size panes.
  - **Fixed** - Sets the pane height to the value specified in the **Size** control.
4. Specify a **Size** for the pane.
5. Click **Apply**.

## Configure Pens

The Process Analyst allows you to configure your pens to suit your preferences. Pen configuration tasks are performed by using the Properties dialog box, which is used to:

- [Configure Pen Appearance](#)
- [Configure Pen Gridlines](#)
- [Configure Pen Axes](#)
- [Configure Pen Quality](#)
- [Configure the Pen Data Connection](#)
- [Configure Cursor Labels](#).

### Configure Pen Appearance

You use the Process Analyst Properties dialog box to configure the appearance of pens. Pen appearance can be configured at run time by Operators and Users (and at design time by Users).

For details about pen appearance, see [Pens](#).

**Note:** To configure default settings for pen appearance, select **Analog**, **Digital**, or **Alarm** in the property tree under **Default Settings**, and then complete the procedure below for the type of pen you want to configure.

## See Also

[Configure Analog and Digital Pens](#)

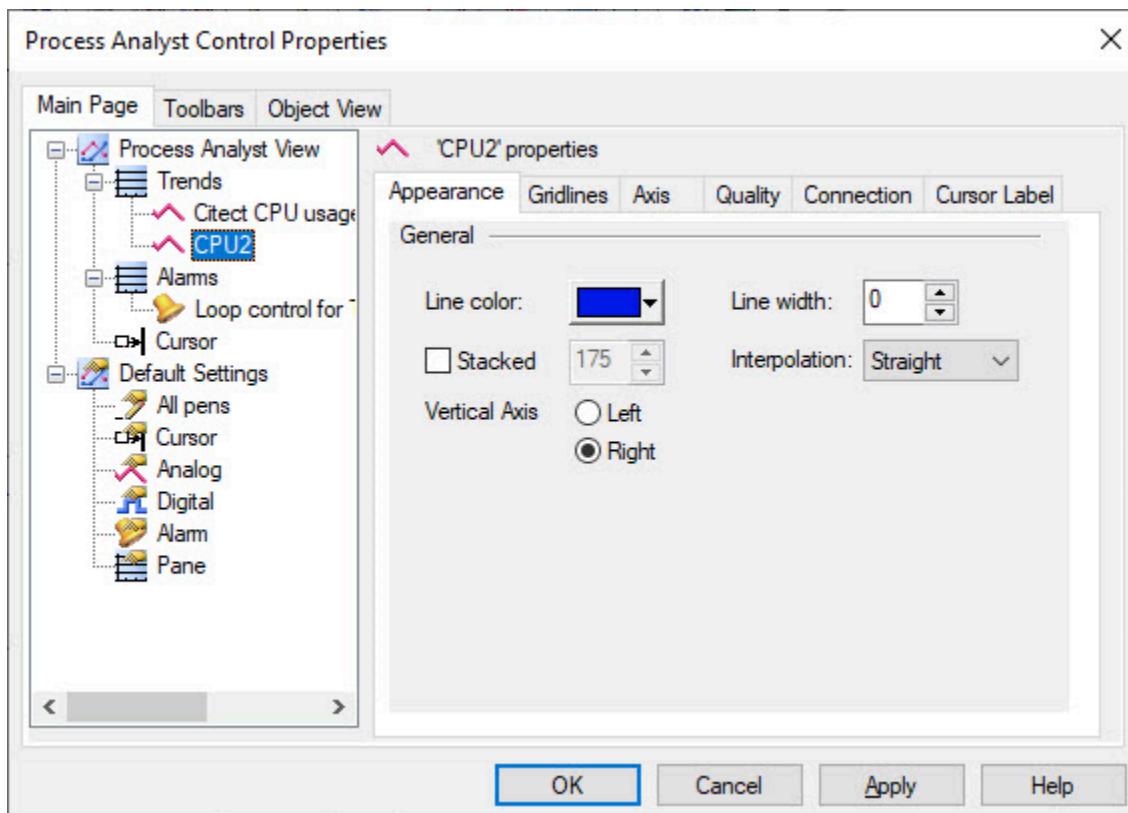
[Configure Alarm Pens](#)

### Configure Analog and Digital Pens

Configuring the appearance of analog or digital pens involves selecting the line color, stack property, line width, and either the method of interpolation (analog pens) or fill color (digital pens).

#### To configure pen appearance:

1. Select the pen you want to configure.
2. Click the **Appearance** tab to display the appearance property controls. Properties vary depending on type of pen.



3. Select a **Line color** using the color swatch.
4. Specify a **Line width**.
5. To stack a pen, select the **Stacked** option and then specify a **Height** in pixels for the stack.
6. Do one of the following:
  - For **analog pens**, choose an **Interpolation** method. **Straight** causes a line to be drawn directly between two data points. **Stepped** causes a line to be drawn between points maintaining the value of the previous sample until a sample with a different value arrives, in which case a vertical line is drawn.
  - For **analog pens**, select if the pen will display on the left or right **Vertical Axis**. You can now compare trend tag data side by side using a dual axis.

**Note:** For overlaid pens, all pen vertical axis will be visible regardless of the selection. For this reason up to two pens can be overlaid with axis on left and on right. If all axes are on one side, only the selected one will be visible

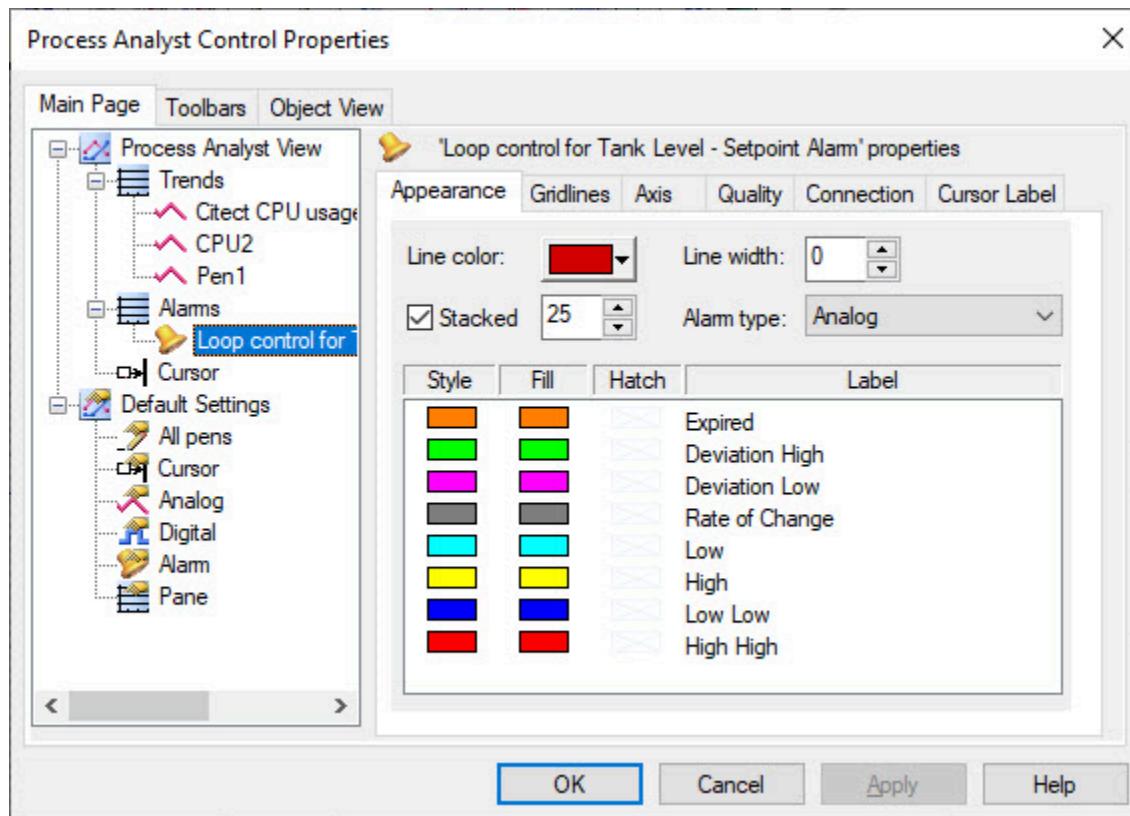
- For digital pens, select the **Filled** check box, and then select a fill color from the color swatch.
7. Click **Apply**.

## Configure Alarm Pens

Configuring the appearance of alarm pens involves selecting the line color, stack property, line width, alarm type, and the properties for that alarm type.

### To configure alarm pen appearance:

1. Select the pen you want to configure.
2. Click the **Appearance** tab to display the appearance property controls for the selected alarm pen.



3. Select a **Line color** using the color swatch.
4. Specify a **Line width**.
5. To stack a pen, select the **Stacked** option and then specify a **Height** in pixels for the stack.
6. Select an **Alarm type**. For details about the different types of alarm pen available, see [Pens](#).
7. For each **Label** for the alarm type you selected, select a **Style**, a **Fill** color, and/or a **Hatch** color by using the swatches.
8. Click **Apply**.

## Configure Pen Gridlines

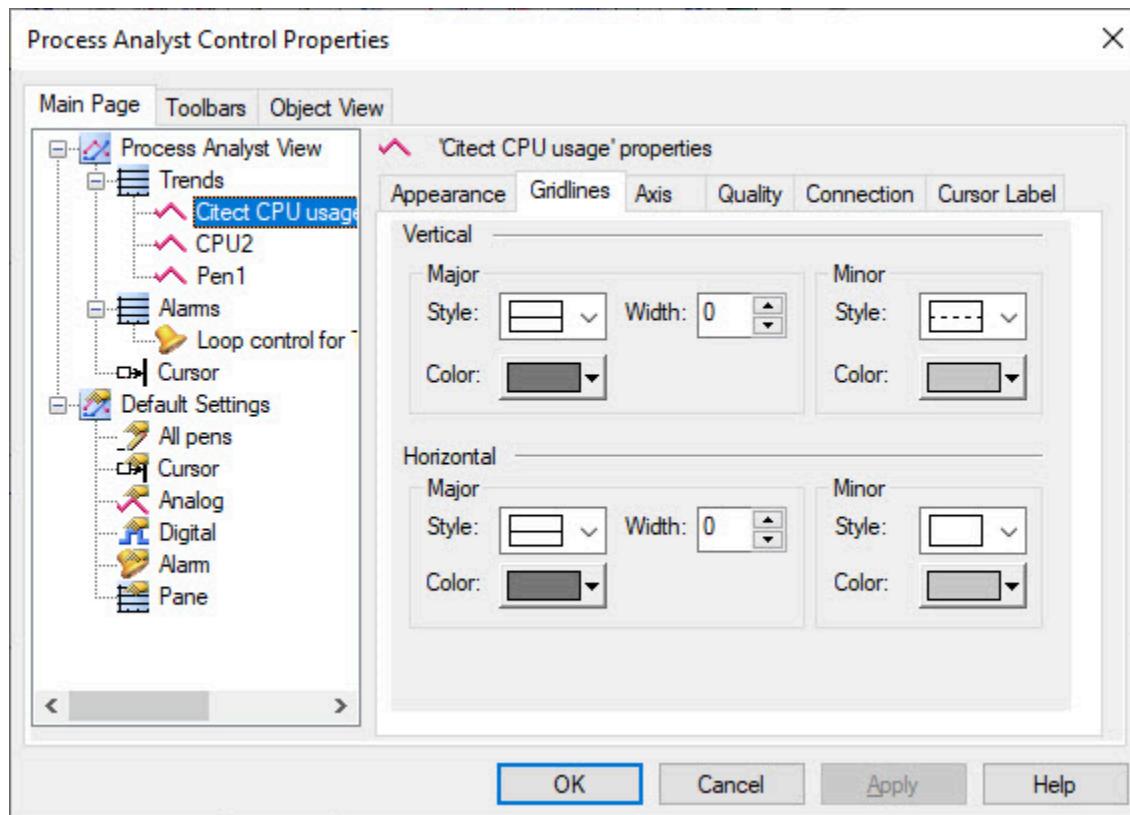
You use the Process Analyst Properties dialog box to configure the gridlines for a selected pen. Pen gridlines can be configured at run time by Operators, and at design time by Users.

For more information about pen gridlines, see [The Chart View](#).

**Note:** To configure defaults for pen gridlines, select the **All pens** item in the property tree under **Default Settings**, and then complete the procedure below.

### To configure pen gridlines:

1. Click the **Main Page** tab.
2. From the property tree list, select the pen you want to configure gridlines for.
3. Click the **Gridlines** tab to display the gridlines property controls.



4. In the **Vertical: Major** area, select a **Style**, specify a **Width**, and then select a **Color**.
5. In the **Vertical: Minor** area, select a **Style**, specify a **Width**, and then select a **Color**.
6. In the **Horizontal** area (analog pens only), select a **Style** for the minor gridline, specify a **Width**, and then select a **Color** for the major gridline.
7. Do the same if necessary for the minor gridline.
8. Click **Apply**.

## Configure Pen Axes

You use the Process Analyst Properties dialog box to configure the axis of the selected pen. A pen axis can be

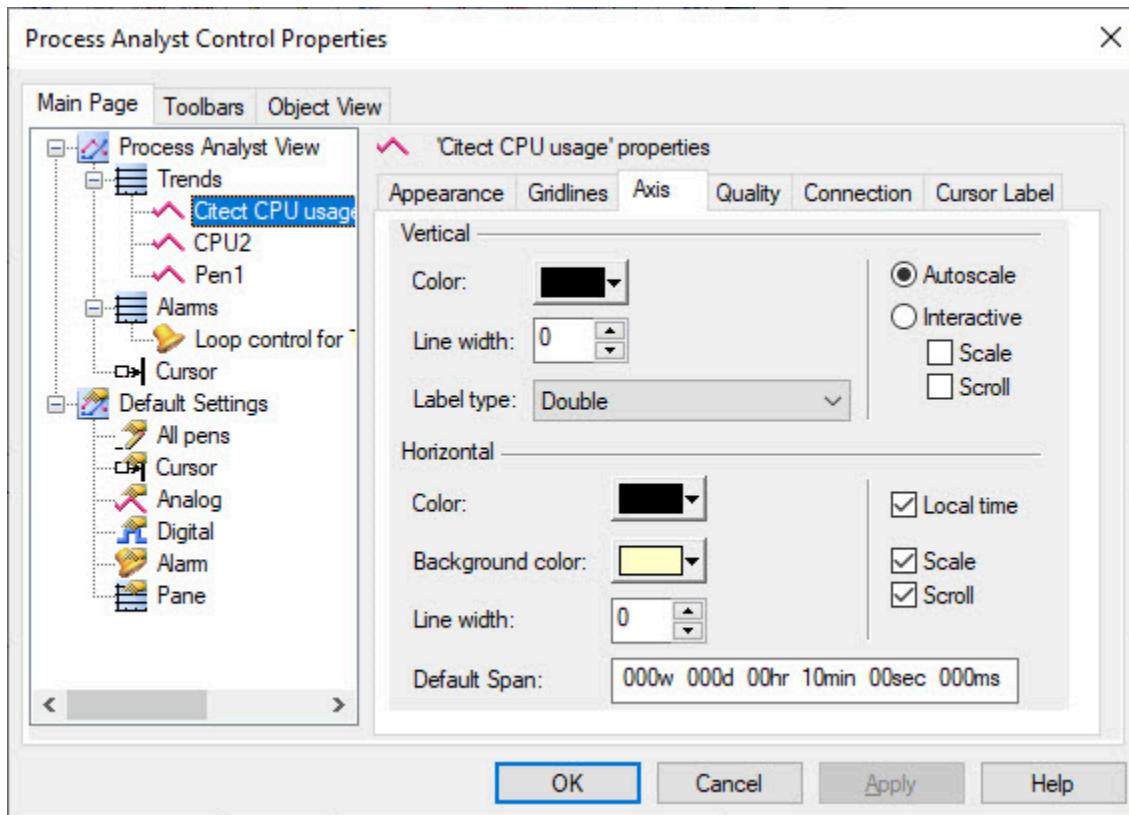
configured at run time by Operators, and at design time by Users.

You can configure the color, line width, label type, scroll and scale properties for the date/time and value axes. You can also choose whether to display time on the date/time axis using local or UTC format.

**Note:** To configure defaults for pen axes, select the **All pens** item in the property tree under **Default Settings**, and then complete the procedure below.

### To configure a pen axis:

1. Click the **Main Page** tab.
2. From the property tree list, select the pen you want to configure axes for.
3. Click the **Axis** tab to display the axis property controls.



4. In the **Vertical** area, select a **Color** by using the color swatch.
5. Enter a new **Line width**.
6. Select a **Label type**. This specifies the format to use for axis values.
7. Do one of the following (analog pens only):
  - Select the **Autoscale** option to autoscale the vertical axis.
  - Select the **Interactive** option, and then select **Scale** to be able to interactively scale the vertical axis; and/or select **Scroll** to be able to scroll the axis.

**Note:** These options are also available on the right-click (context) menu.

8. In the **Horizontal** area, select a **Color** by using the color swatch.
9. Select a **Background color** by using the color swatch.
10. Enter a new **Line width**.

11. Enter a **Default Span** to define the span you want to use for a new pen.

The default span is used by the Process Analyst when the Operator or User clicks the **Reset to Default Span** button, or if the pen is added in pen unlocked mode, or if the pen is the first one added to a display.

If you are setting the span value as a default setting for new pens, the new span value is inherited by new pens created.

12. Select the **Local Time** option to display the date/time axis in local time using your machine settings. If this option is not selected, the time is displayed in UTC format. For details about time display on the date/time axis, see [The Chart View](#).
13. Select **Scale** to be able to interactively scale the vertical axis.
14. Select **Scroll** to be able to scroll the axis.

---

**Note:** The **Scale** and **Scroll** options are also available on the right-click (context) menu.

---

15. Click **Apply**.

## Configure Pen Quality

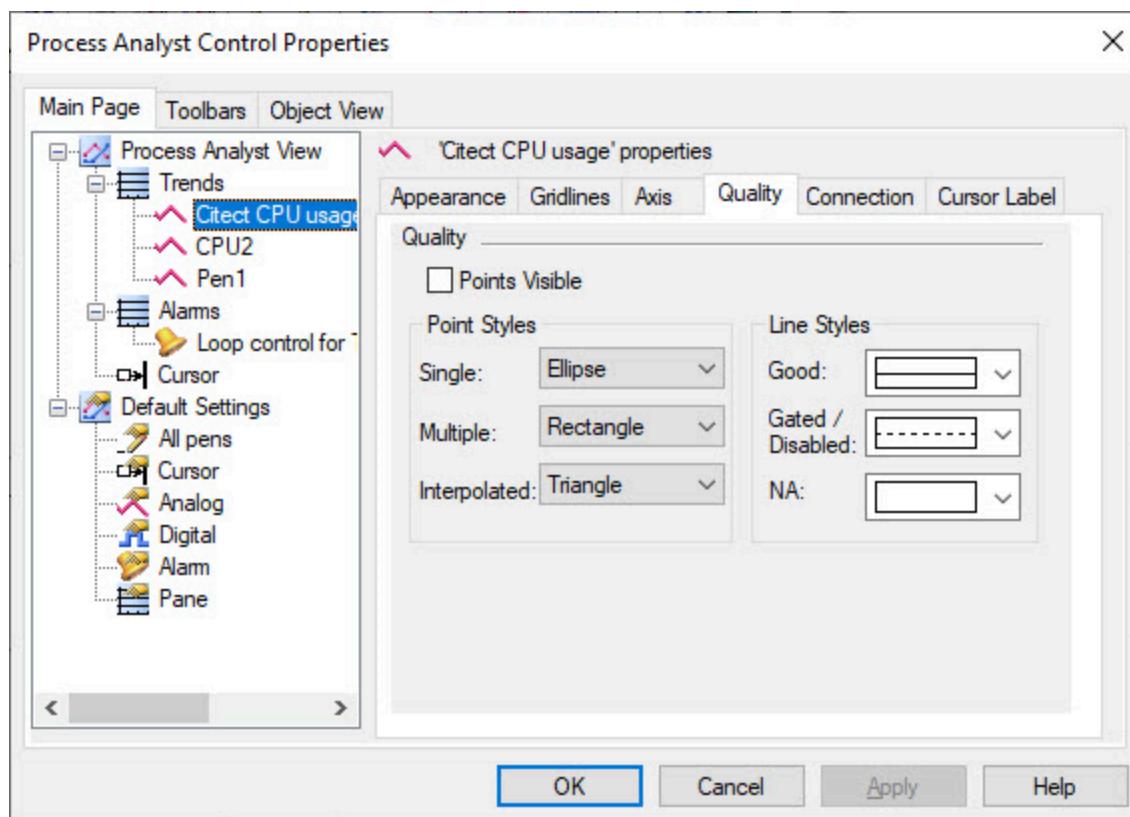
You use the Process Analyst Properties dialog box to configure the quality of the selected pen. Pen quality can be configured at run time by Operators, and at design time by Users.

Configuring the pen quality allows you to define the appearance of sample markers on a selected pen, as well as the line styles of the pen, based upon the quality of the data being trended by the Process Analyst.

For details about how the Process Analyst represents data quality, see [Data Quality](#).

### To configure pen quality:

1. Click the **Main Page** tab.
2. Select the pen you want to configure.
3. Click the **Quality** tab to display the quality property controls.



4. To enable points for the pen to be visible, select the **Points Visible** option.
5. In the **Point Styles** area, select a **Single** point style to represent a single data sample.
6. Select a **Multiple** point style to represent multiple data samples.
7. Selected an **Interpolated** point style for interpolated data samples.
8. In the **Line Styles** area, select a line style to represent a **Good** sample.
9. Select a line style to represent a **Gated/Disabled** sample.
10. Select a line style to represent an **NA** sample.
11. Click **Apply**.

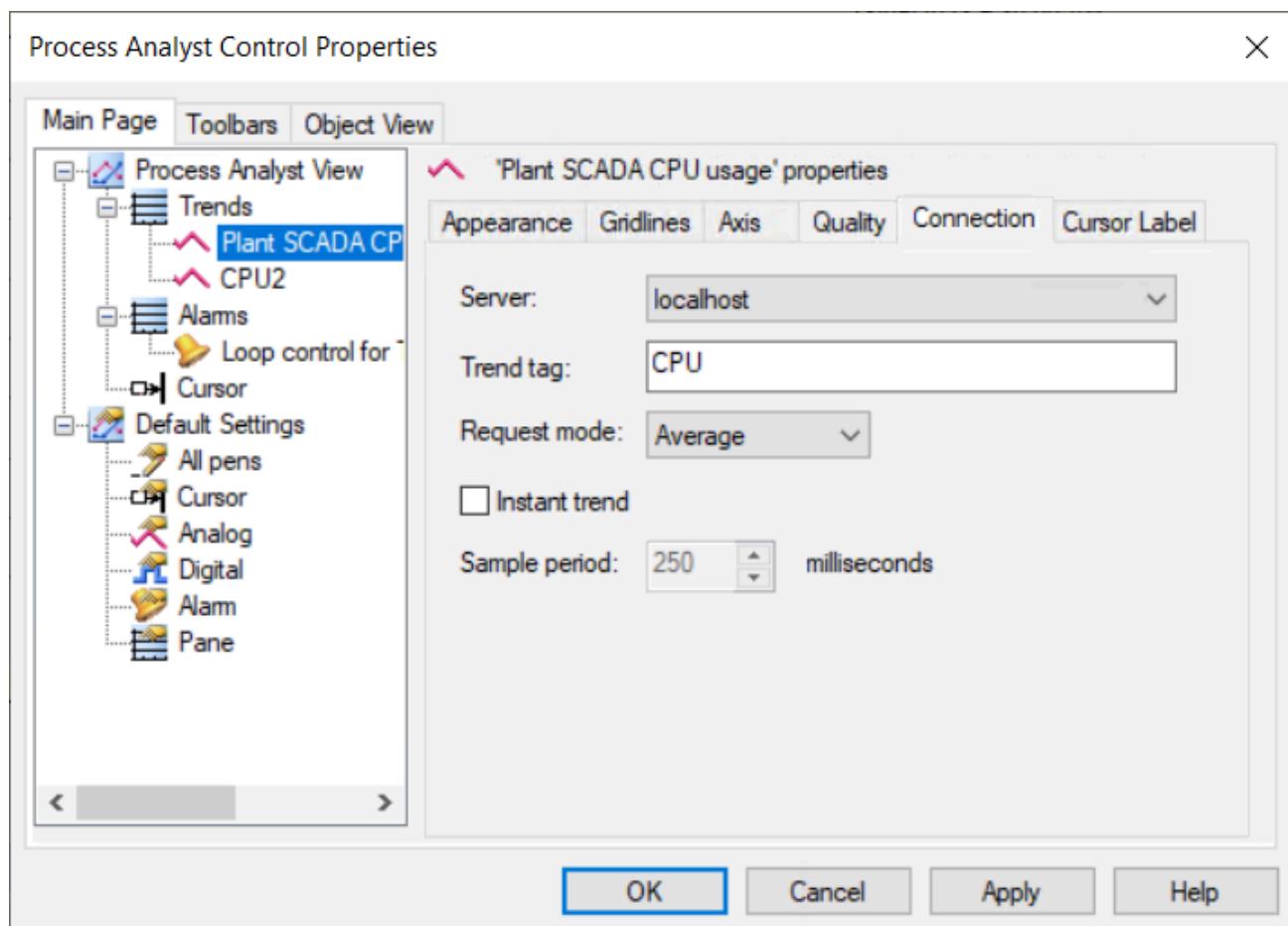
## Configure the Pen Data Connection

You use the Process Analyst Properties dialog box to configure the pen data connection. This allows you to define the server, trend tag, and request mode for the selected pen.

Pen connection can be configured at run time by Operators and Users that have the appropriate privileges.

### To configure pen data connection:

1. Select the pen you want to configure.
2. Click the **Connection** tab to display the connection property controls.



3. For the **Server** data connection, a <localhost> connection is selected by default, indicating that the Process Analyst will connect to the Plant SCADA run time client running on the same computer, and pass its requests through to the client, which will pass them onto the server.

You can also select <Unbound> for the **Server** field from the drop-down menu. An engineer can select this option to manually add samples to analog or digital pens using [IPen.AddSample](#). For example, if you are reading information from an external source, you can add samples to the display.

4. In the **Trend tag** field, enter the trend tag for the pen. In a system with more than one cluster, specify both the cluster and tag using the format <cluster name.tag name>. Omitting the cluster name will cause an error. If the system has only one cluster configured, you can just enter the tag name. The configured cluster will be assumed.
5. Select a **Request mode**. The default is Average.

The request mode defines how multiple samples are treated by the Process Analyst. Regardless of the request mode used, the timestamp for a sample is always averaged.

6. In selecting the **Instant trend** box - the process analyst requests the pen's data using the new instant trend system instead of the standard trend system, as a configured trend tag. If the Instant Trend box is selected the **Sample Period** field becomes available. The sample period is used in the instant trend cache to subscribe to the IO tag on the IO server, and sets how often the IO Device is polled to check the value of the tag on the IO server.

**Note:** A new sample will only appear in the Process Analyst for an instant trend if the tag value changes. So even if your sample period is very low, you may only get the occasional new sample displayed for when the value changes. The faster the sample period, the more precise the samples will be in the instant trend, but at

a cost of a higher load placed on the IO Device (more frequent polling).

7. Click **Apply**.

## See Also

[Data Compaction](#)

### Configure Cursor Labels

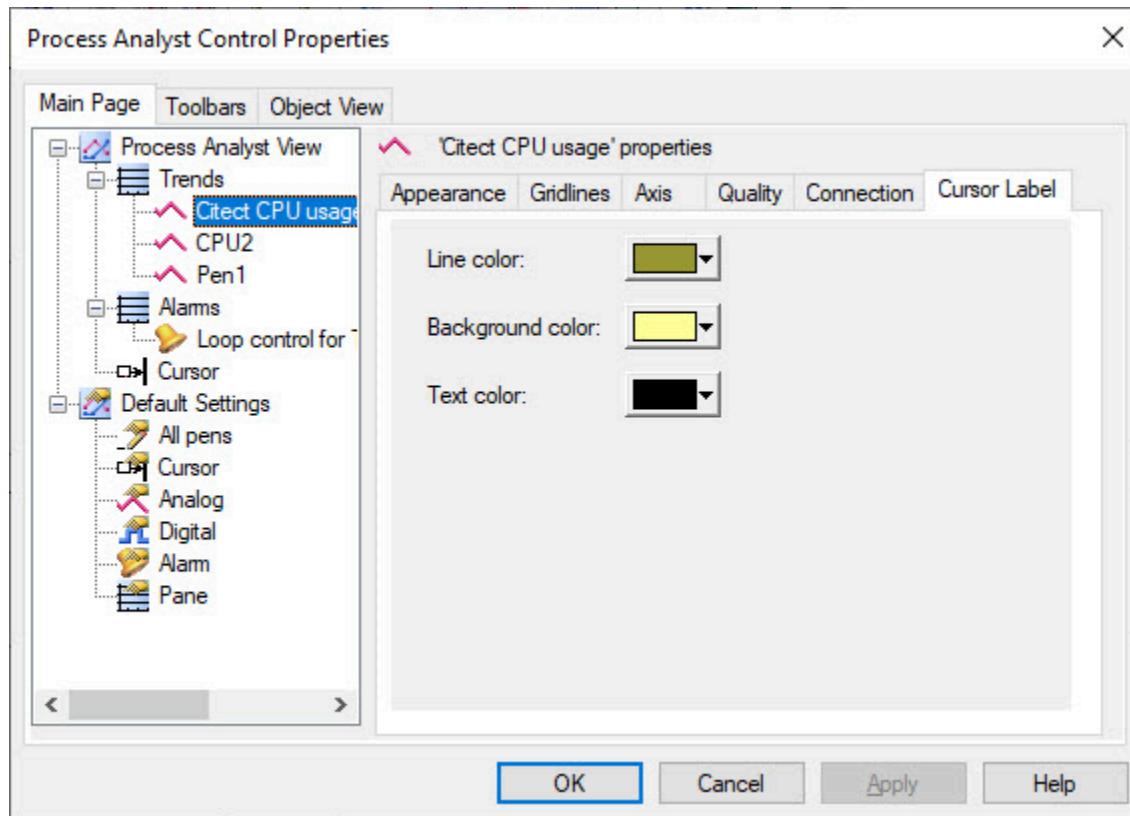
You use the Process Analyst Properties dialog box to configure the pen cursor labels. Configuring the pen cursor labels allows you to specify the color used for the lines, background, and text on the cursor label. The information shown on a cursor label is predefined and cannot be changed.

For details about cursor labels, see [Using Cursor Labels](#).

Pen cursor labels can be configured at run time by both Operators and Users that have the appropriate privileges.

#### To configure cursor labels:

1. Select the pen you want to configure.
2. Click the **Cursor Label** tab to display the connection property controls.



3. Select a **Line color** from the color swatch.
4. Select a **Background color** from the color swatch.
5. Select a **Text color** from the color swatch.

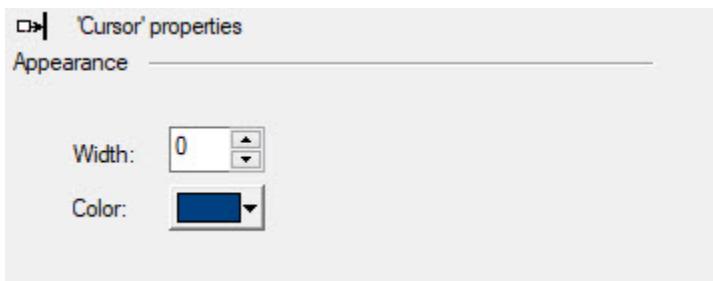
6. Click **Apply**.

## Configure Cursors

You can configure the line width and line color of a selected cursor. Changes to the cursor line color apply only to the currently selected cursor. For details on cursors, see [Cursors](#).

### To configure the cursor:

1. In the property tree of the Process Analyst Properties dialog box, click the cursor you want to configure. The **Appearance** property controls appear.



2. Type in a new **Width** value, and/or select a new **Color**.
3. Click **Apply**.

## Configure Defaults

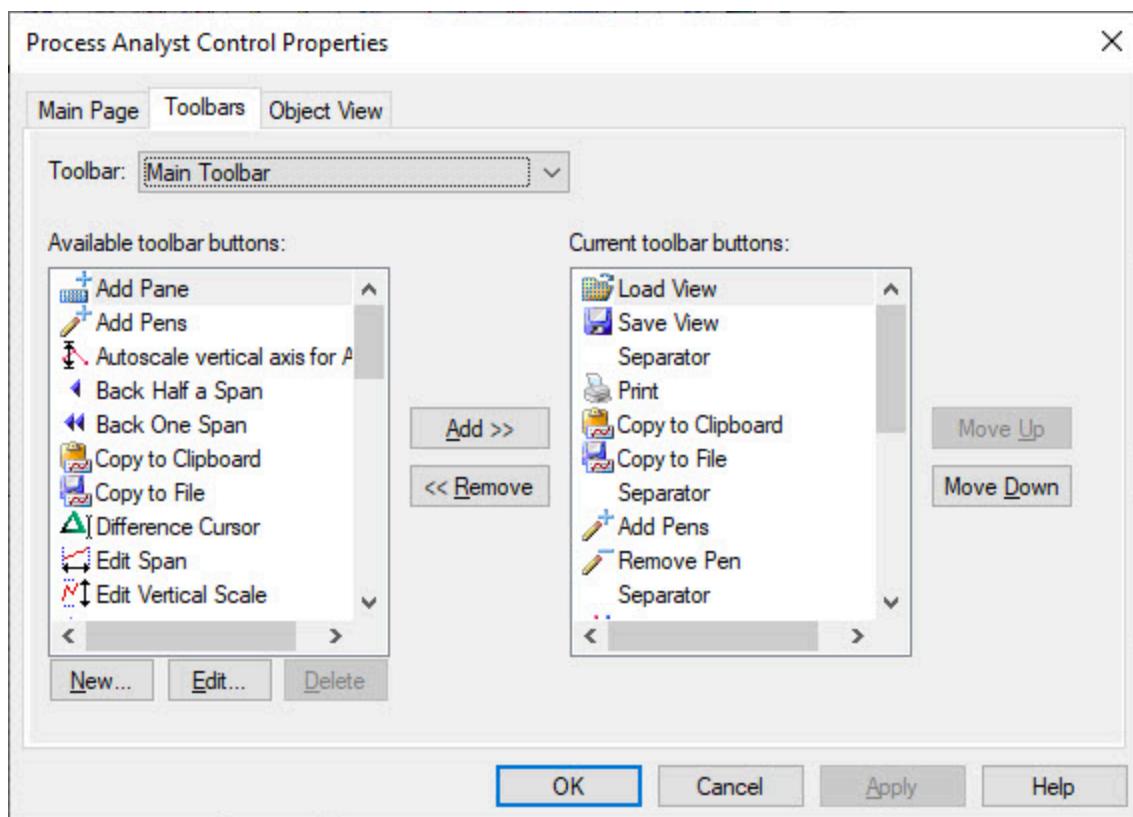
The defaults are a collection of properties that are inherited by each item (pane, pen, cursor, and so on) when that item is created. These default properties are maintained for the lifetime of the item until its properties are modified.

You configure these defaults in the same way as you configure the individual components. The Default Settings node on the property tree contains the following items:

- **All pens** - configure the gridlines, axis, quality, connection, and cursor label properties for pen types. See [Configure Pen Gridlines](#), [Configure Pen Axes](#), [Configure Pen Quality](#), [Configure the Pen Data Connection](#), and [Configure Cursor Labels](#).
- **Cursor** - configure cursor defaults. See [Configure Cursors](#).
- **Analog** - configure the appearance of analog pens. See [Configure Pen Appearance](#).
- **Digital** - configure the appearance of digital pens. See [Configure Pen Appearance](#).
- **Alarm** - configure the appearance of alarm pens. See [Configure Pen Appearance](#).
- **Pane** - configure the pane height and appearance defaults. See [Configure Chart Panes](#).

## Configuring Toolbars

The Process Analyst has two toolbars, the main toolbar and the navigation toolbar. You use the Properties dialog box to configure the toolbars.



Operators can configure the Process Analyst toolbars by:

- [Adding or Removing Toolbar Commands](#)
- [Changing the Order of Toolbar Commands.](#)

Users can perform additional tasks such as:

- [Adding New Commands](#)
- [Editing Existing Custom Commands.](#)

## Adding or Removing Toolbar Commands

Operators can add or remove toolbar commands during run time.

### To add or remove commands from a toolbar:

1. From the **Toolbar** menu, choose the toolbar you want to customize (**MainToolbar** or **Navigation Toolbar**).
2. **To add a command to the toolbar:** In the **Available toolbar buttons** list, select the command you want to add to the toolbar, and then click **Add**. The selected command moves to the **Current toolbar buttons** list.  
The **Available toolbar buttons** list contains the command buttons available in your system, including predefined as well as user-defined commands.
3. **To remove a command from the toolbar:** In the **Current toolbar buttons** list, select the command you want to remove from the toolbar, and then click **Remove**. The selected command moves to the **Available toolbar buttons** list.

## Changing the Order of Toolbar Commands

Operators can change the order of toolbar commands during run time.

### To change the order of commands:

- Select a command in the **Current toolbar buttons** list and click **Move Up** or **Move down** to move the selected command up or down the list as necessary.

## Configuring the Object View

Operators can configure the Object View to display additional pen information to the columns that are displayed by default. You configure the Object View by using the Properties dialog box.

Operators can select which columns to display, as well as change the size of existing columns and the column display order. Users can define new columns, or edit or delete existing columns; for details, see [Creating or Editing Object View Columns](#).

You can configure the Object View to display these predefined columns:

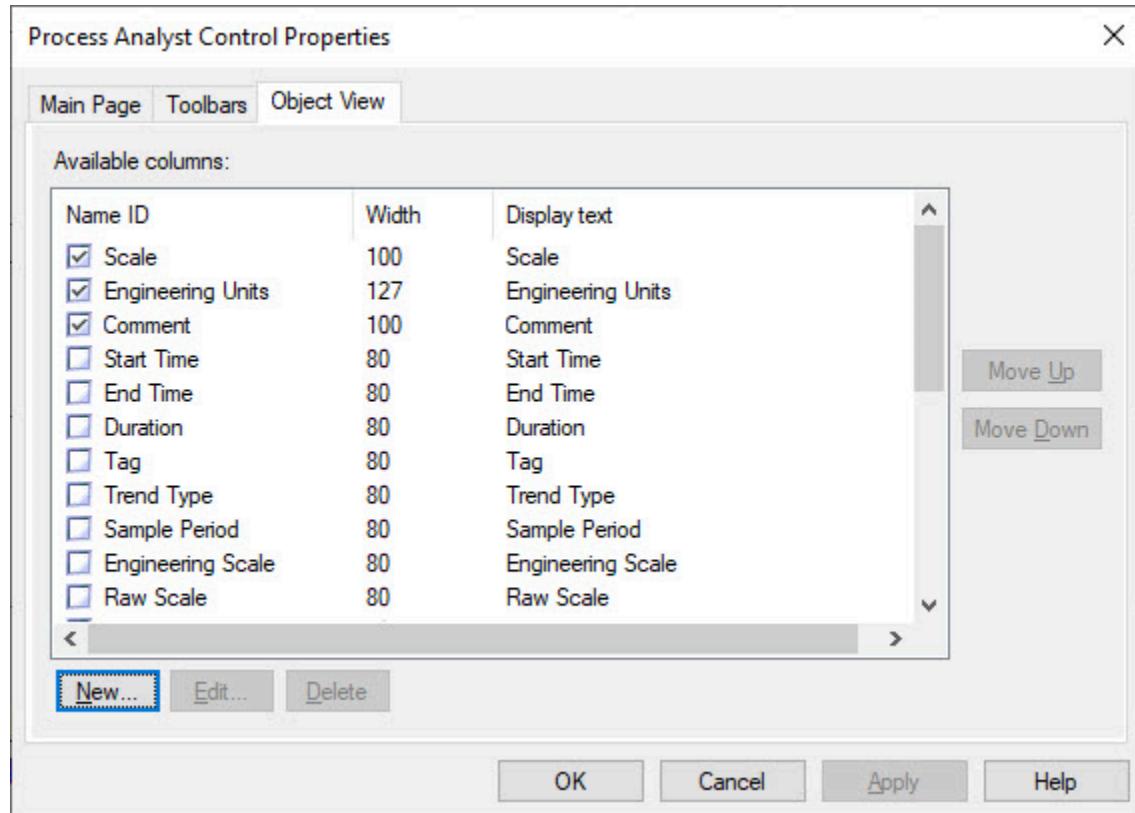
Column	Description
Scale	Vertical axis start and end position of the pen.
Engineering Units	Engineering units associated with the pen.
Comment	The trend/alarm comment defined for the pen.
Start Time	Date/time axis start position of the pen.
End Time	Date/time axis end position of the pen.
Duration	Difference between the start time and the end time.
Tag	Pen's associated trend or alarm tag.
Trend Type	Trend type of associated tag.
Sample Period	Sampling period of the associated trend tag.
Engineering Scale	Engineering scale for associated trend tag.
Raw Scale	Raw scale for associated trend tag.
Alarm Category	Category of associated alarm tag.
Alarm Description	Description of associated alarm tag.
Alarm Area	Area of associated alarm tag.
Alarm Name	Name of associated alarm tag.
Alarm Type	Alarm type of associated alarm tag.

Column	Description
Error	Displays the error of the last data request. Blank if last data request succeeded.
Minimum	Lowest displayed value (trend tags only).
Maximum	Highest displayed value (trend tags only).
Average	Average of every displayed value (trend tags only).

For information on columns that are displayed by default, see [The Object View](#).

## Object View Properties Page

The Object View properties page allows you to show or hide existing columns, create custom columns, edit existing columns, and re-order columns.



The Properties page displays the available columns for the Object View and their properties:

- **NameID** - Internal identifier, which needs to be unique.
- **Width** - Default width of the column in pixels.
- **Display Text** - Title displayed in the column header.

The check boxes in the NameID column are bound to a column's visibility: a column is visible only if the associated check box is selected.

The **Move Up** and **Move Down** buttons to the right of the **Available Columns** list box allow you to reorder columns. The order of the columns from top to bottom in the list dictates their display order from left to right in the Object View. Clicking **Move Up** or **Move Down** shifts the currently selected item up or down respectively.

## See Also

[Creating or Editing Object View Columns](#)

## Working with Views

An Operator can save the visual setup of a Process Analyst control by saving a *view*, which is saved as a Process Analyst View (.pav) file. They can also load views that have been created previously. A view saves the state of every command, as well as properties for every the Process Analyst components (pane, pen, axes, background, and so on).

To save a view or to load a view, you use the **Save View** and **Load View** commands, respectively, on the main toolbar.

## See Also

[Saving a View](#)

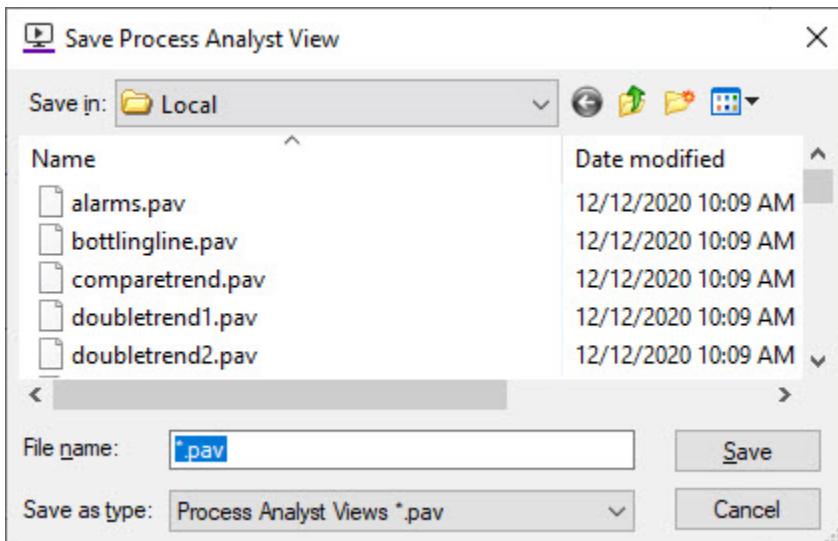
[Load a View](#)

## Saving a View

A Process Analyst view stores the trends and alarms that are being displayed, the columns being viewed in the Object View, the toolbar buttons that are available, as well as the "look and feel" of the view.

### To save a view:

1. On the main toolbar, click **Save View**. The Save Process Analyst View dialog box appears, showing the location where you can save views.



2. Choose the location where you want to save your view.

**Note:** It is your administrator's responsibility to set up the correct directories for saving views.

3. Enter a **File name** for your view, and then click **OK**.

To support redundancy, if the **Local** option is available and selected, Plant SCADA attempts to save the view to the primary, standby *and* local locations.

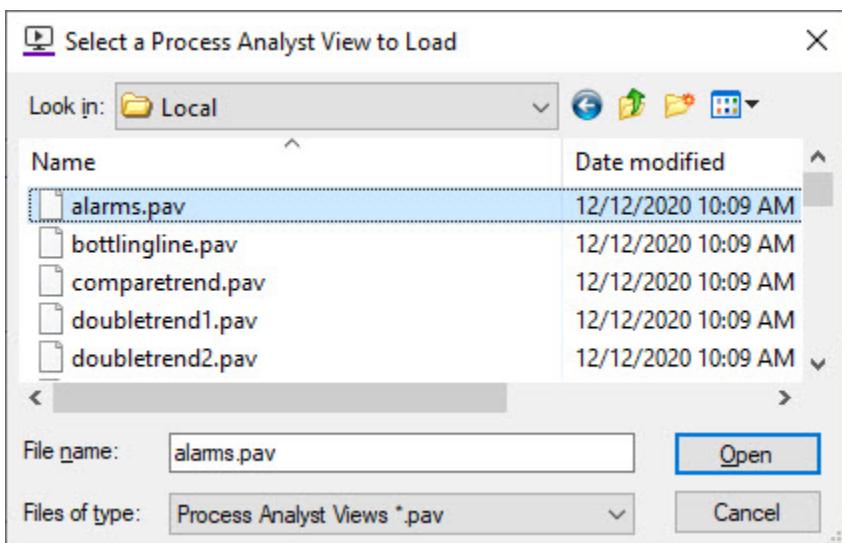
## Load a View

When loading a view, the start time and end time of a view is restored only if autoscroll is off. If autoscroll is on, pens are synchronized to "Now."

When loading a view, the only locations that are available (**My Documents**, **Primary**, and **Standby**) are those that have been configured by your administrator.

### To load a view:

1. On the main toolbar, click **Load View**. The Load dialog box appears.



2. Select a view to load, and then click **Open**. The view is loaded.

## Operator Command Reference

You use the toolbar commands on the main toolbar and navigation toolbar to perform commonly used functions for viewing and interacting with Process Analyst data, such as adding or removing pens, displaying cursors, and so on.

Process Analyst has predefined commands, grouped into the following categories:

- [View Commands](#)
- [Zoom Commands](#)
- [Navigation Commands](#)
- [Export Commands](#)

- [Interface Commands](#)
- [General Commands](#)

The toolbars in your run time environment might have been customized during implementation, so not every command might appear on your toolbars. Additionally your toolbars might have custom commands not described here. The tables describe the *default* set of commands delivered with the Process Analyst.

## View Commands

The Process Analyst has the following view commands by default:

Icon	Tooltip	Description
	Save View	Displays the Save File dialog box allowing an Operator to save a Process Analyst view to a specified location. For details, see <a href="#">Saving a View</a> .
	Load View	Displays the Load View dialog box allowing the operator to specify a view to load. For details, see <a href="#">Load a View</a> .

## See Also

- [Zoom Commands](#)
- [Navigation Commands](#)
- [Export Commands](#)
- [Interface Commands](#)
- [General Commands](#)

## Zoom Commands

The Process Analyst has the following zoom commands by default:

Icon	Tooltip	Description
	Toggle Box Zoom	Toggles the Process Analyst into box zoom mode. The mouse cursor changes to a crosshair used to define an area to zoom in on. Zoom may be canceled by right-clicking or toggling the Zoom command off. For details, see <a href="#">Toggle Box Zoom</a> .
	Zoom in 50%	Executes a horizontal and vertical zoom in of 50% of the current span(s) of the pen(s). For details, see <a href="#">Zoom In/Zoom Out</a> .
	Zoom out 50%	Executes a horizontal and vertical zoom out of 50% of the current span(s) of the pen(s). For details, see <a href="#">Zoom In/Zoom Out</a> .
	Undo Last Zoom	Undoes the last zoom operation. For details, see <a href="#">Undo Last Zoom</a> .
	Reset to Default Span	Restores the pen(s) spans to their original default settings. For details, see <a href="#">Reset to Default Span</a> .
	Edit Span	Displays the Edit Span dialog box allowing an operator to explicitly enter a time span to apply to the display. For details, see <a href="#">Set a Nonstandard Time Span</a> .
	Edit Vertical Scale	Enabled when an analog pen is selected. For details, see <a href="#">Edit Vertical Scale</a> .

## See Also

[View Commands](#)

[Navigation Commands](#)

[Export Commands](#)

[Interface Commands](#)

[General Commands](#)

## Navigation Commands

The Process Analyst has the following navigation commands by default. For details about these commands, see

## Navigate Time.

Icon	Tooltip	Description
	Toggle Span Lock	Toggles the locking of the time span. A time span is the "distance" in time between the start time and end time of the chart. For details, see <a href="#">Lock/Unlock the Time Span</a> .
	Back One Span	Moves the pen(s) back in time exactly one time span. For details, see <a href="#">Navigate Time</a> .
	Back Half a Span	Moves the pen(s) back half a span. For details, see <a href="#">Navigate Time</a> .
	Forward One Span	Moves the pen(s) forward in time exactly one span. For details, see <a href="#">Navigate Time</a> .
	Forward Half a Span	Moves the pen(s) forward half a span. For details, see <a href="#">Navigate Time</a> .
	Synchronize to Now	Synchronizes pen(s) such that the end date time reflects "now" which is positioned on the right-hand edge of the screen. "Now" is calculated using the current system time. For details, see <a href="#">Synchronize to Now</a> .
	Toggle Auto-Scrolling	Toggles the automatic scrolling off and on for every pen. For details, see <a href="#">Toggle Autoscrolling</a> .

**See Also**

- [View Commands](#)
- [Zoom Commands](#)
- [Export Commands](#)
- [Interface Commands](#)
- [General Commands](#)

**Export Commands**

The Process Analyst has the following export commands by default:

Icon	Tooltip	Description
	Export to File	Copies visible pens to an Excel compatible file. For details, see <a href="#">Copying Data to File</a> .
	Copy to Clipboard	Copies visible pens to the clipboard. Interface Commands. For details, see <a href="#">Copying Data to the Clipboard</a> .

## See Also

[View Commands](#)

[Zoom Commands](#)

[Navigation Commands](#)

[Interface Commands](#)

[General Commands](#)

## Interface Commands

The Process Analyst has the following interface commands by default:

Icon	Tooltip	Description
	Show/Hide Cursor	Toggles the display of cursors. For details, see <a href="#">Cursors</a> .
	Show/Hide Cursor Labels	Toggles the display of cursor labels. Enabled only when a cursor is visible and when a pen exists. For details, see <a href="#">Using Cursor Labels</a> .
	Show/Hide Points	Toggles the display of points representing where sample data was recorded in the archive. For details, see <a href="#">Pens</a> .
	Lock/Unlock Cursor Labels	Toggles the locking/unlocking of cursor labels. Enabled only when a cursor is visible and when a pen exists. For details, see <a href="#">Using Cursor Labels</a> .
	Lock/Unlock Pens	Toggles the locking/unlocking of pens. For details, see <a href="#">Lock/Unlock Pens</a> .

Icon	Tooltip	Description
	Add Pane	Adds a new pane to the view. For details, see <a href="#">Configure Chart Panes</a> .
	Remove Pane	Removes the pane of the primary selected pen. A dialog confirms the delete. For details, see <a href="#">Configure Chart Panes</a> .
	Autoscale Vertical Axis for Analog Pens	Toggles autoscaling for the selected pen on a per-pen basis. For details, see <a href="#">Scale the Chart</a> .
	Lock/Unlock Vertical Axis Scrolling	Toggles interactive scrolling of the vertical axis and disables autoscaling. For details, see <a href="#">Scroll the Chart</a> .

## See Also

[View Commands](#)

[Zoom Commands](#)

[Navigation Commands](#)

[Export Commands](#)

[General Commands](#)

## General Commands

The Process Analyst has the following general commands by default:

Icon	Tooltip	Description
	Add Pen	Displays the add pen dialog. For details, see <a href="#">Add Pens</a> .
	Remove Pen	Removes the selected pen from the display. For details, see <a href="#">Delete Pens</a> .
	Toggle Object View	Toggles the display of the Object View. For details, see the <a href="#">The Object View</a> .
	Print	Displays the print dialog, allowing the user to print the current state of the Process Analyst. For details, see <a href="#">Printing and Exporting</a> .

Icon	Tooltip	Description
	Refresh Data	Refreshes the data for the selected pen, or every pen (if locked).
	Show Properties	Displays the Process Analyst Properties dialog box. For details, see <a href="#">Process Analyst Properties Dialog Box</a> .
	Help	Displays the Process Analyst Help.

## See Also

[View Commands](#)

[Zoom Commands](#)

[Navigation Commands](#)

[Export Commands](#)

## Printing and Exporting

You can print detailed reports of your Process Analyst trends for management reports and other purposes. You can configure Process Analyst reports to include other print options designed to maximize the business value of your reports. You can also export pen data to the Windows Clipboard or to Microsoft Excel.

---

**Note:** For details about general print options in Windows, refer to your Windows documentation.

---

## See Also

[Process Analyst Reports](#)

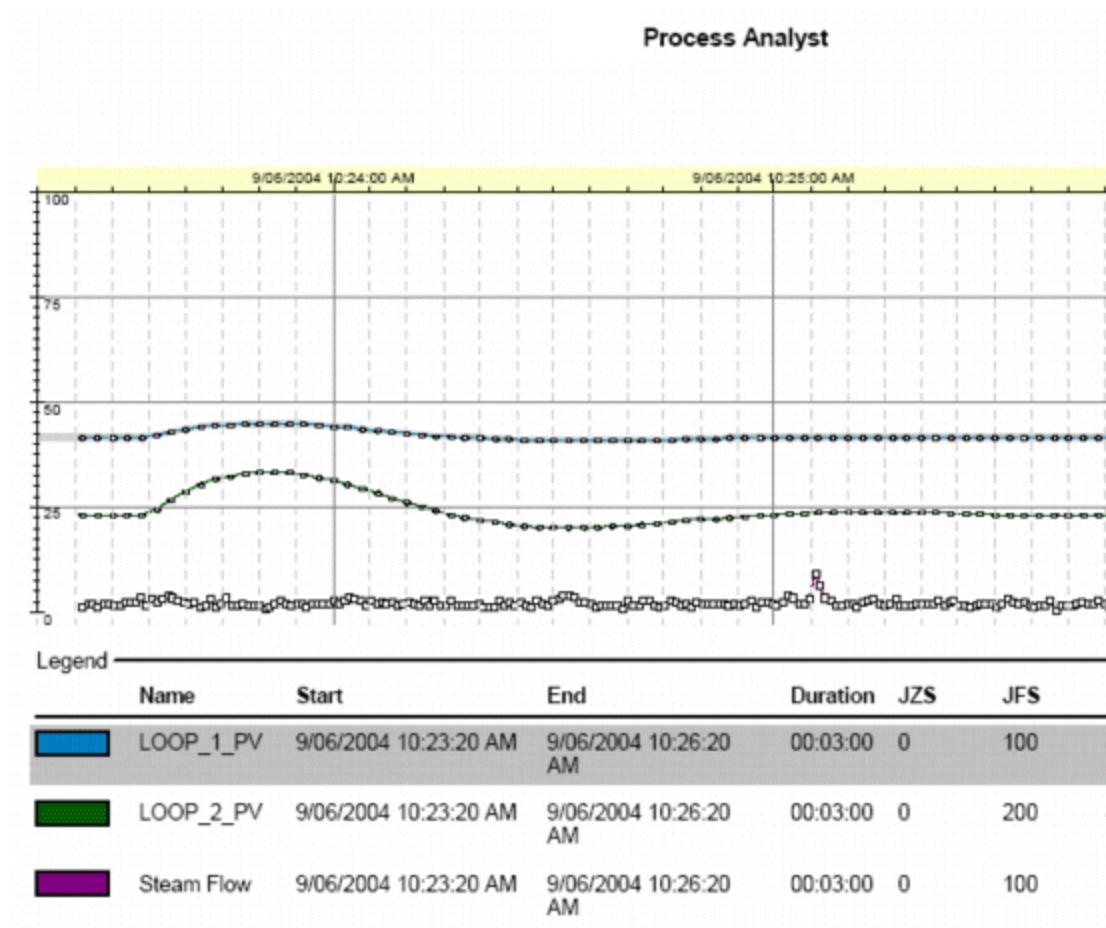
[Configure Process Analyst Report Options](#)

[Exporting Pen Data](#)

## Process Analyst Reports

Process Analyst reports are formatted automatically by the system to make optimal use of the paper size and orientation. For example, if the page is small and the report contains a lot of information, the reports will use a smaller font to try to fit the information to the page. For larger pages, a larger font will be used. Reports use an Arial font between 8-14 points.

A typical Process Analyst report looks like this:



This example shows a report of a chart titled *Process Analyst*; the chart has only one pane, which contains three analog pens. The topmost pen in the report legend is highlighted, indicating that this pen is selected; consequently, the axes shown in the report are associated with this pen. You can see that this pen is selected in the chart by the "halo" effect surrounding the pen. The color boxes on the left-hand side of the legend help you to distinguish between the pens.

#### To print a report:

- Click **Print**. The Print dialog box appears. Click the **Print** button, or choose **Print** from the right-click (context) menu.

## Configure Process Analyst Report Options

You can configure Process Analyst reports to contain such things as specific items on a report legend (pen names, durations, engineering units, for example). You can also include header information and page numbers.

You use the Print dialog box to configure Process Analyst reports. To display the Print dialog box, click **Print** on the main toolbar. After configuring your reports, click **Print** on the **General** panel of the Print dialog box to print your report.

## See Also

- [Setting up Report Legends](#)
- [Setting up Report Options](#)

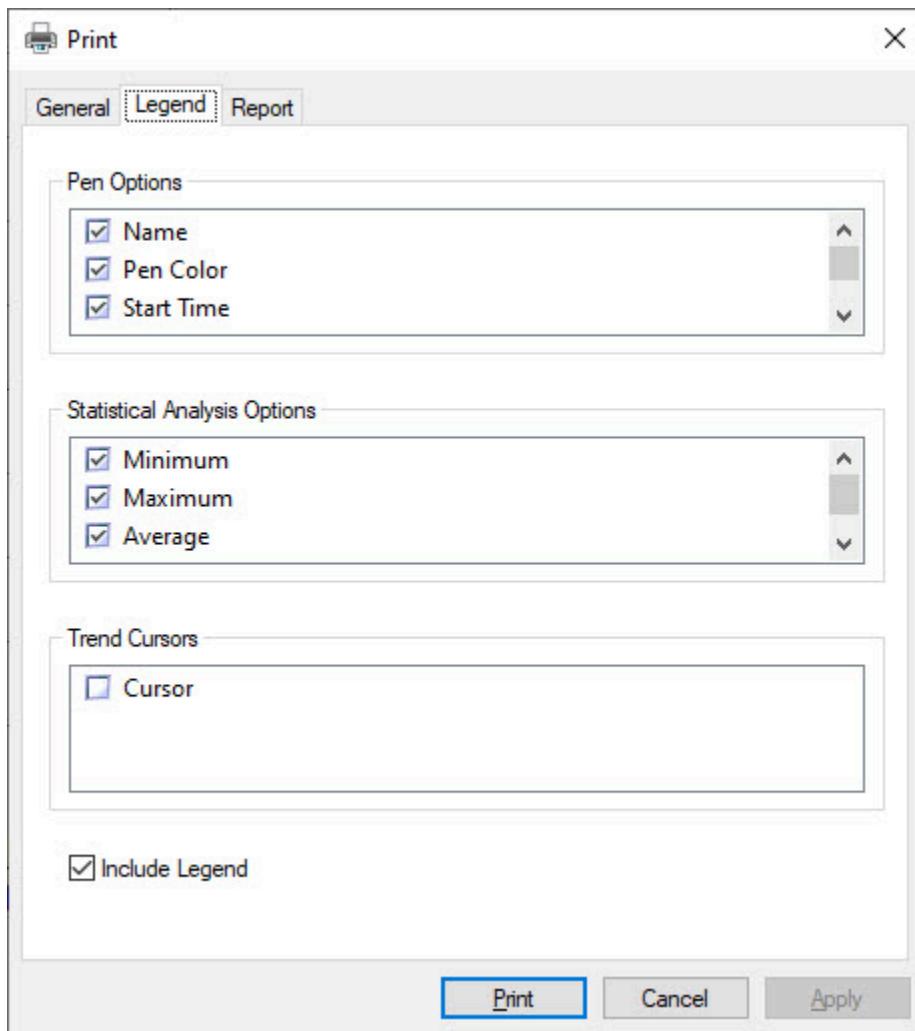
### Setting up Report Legends

You can configure your reports to include *report legends*. The information in the report legend is derived from the information properties of the underlying tag that is associated with a pen. If there are no information properties defined for a tag, this information isn't available for a legend.

You set up your report legends by using the Legend panel of the Print dialog box.

#### To set up a report legend:

1. In the Print dialog box, click the **Legend** tab. The Legend panel appears.



2. The panel shows, by default, the **Pen Options**, **Statistical Analysis Options**, and **Cursors** lists (if there is a cursor currently displayed on the chart). The options available to you might differ from the ones shown here.
3. Select the check box of the **Pen Options** you want to include in your report. For details about these options, see [Configuring the Object View](#).

4. Select the **Statistical Analysis Options** you want to include. This section is available only if the chart contains at least one analog or digital pen.
  - **Minimum** - causes the minimum value from cache to be returned. Note that this value might not be a real logged sample if the sample found is a multiple calculated sample.
  - **Maximum** - causes the maximum value from cache to be returned. Note that this value might not be a real logged sample if the sample found is a multiple calculated sample.
  - **Average** - uses time-weighted averaging to determine the average for both stepped and interpolated lines. This means that if a trend stays at a value of 10 for 1 hour and then spikes quickly at a value of 50 for a minute, the average will not be significantly affected.
5. Select the **Cursors** you want to include.
6. If you want to include a report legend, make sure the **Include Legend** check box is selected.
7. Click **Apply**.

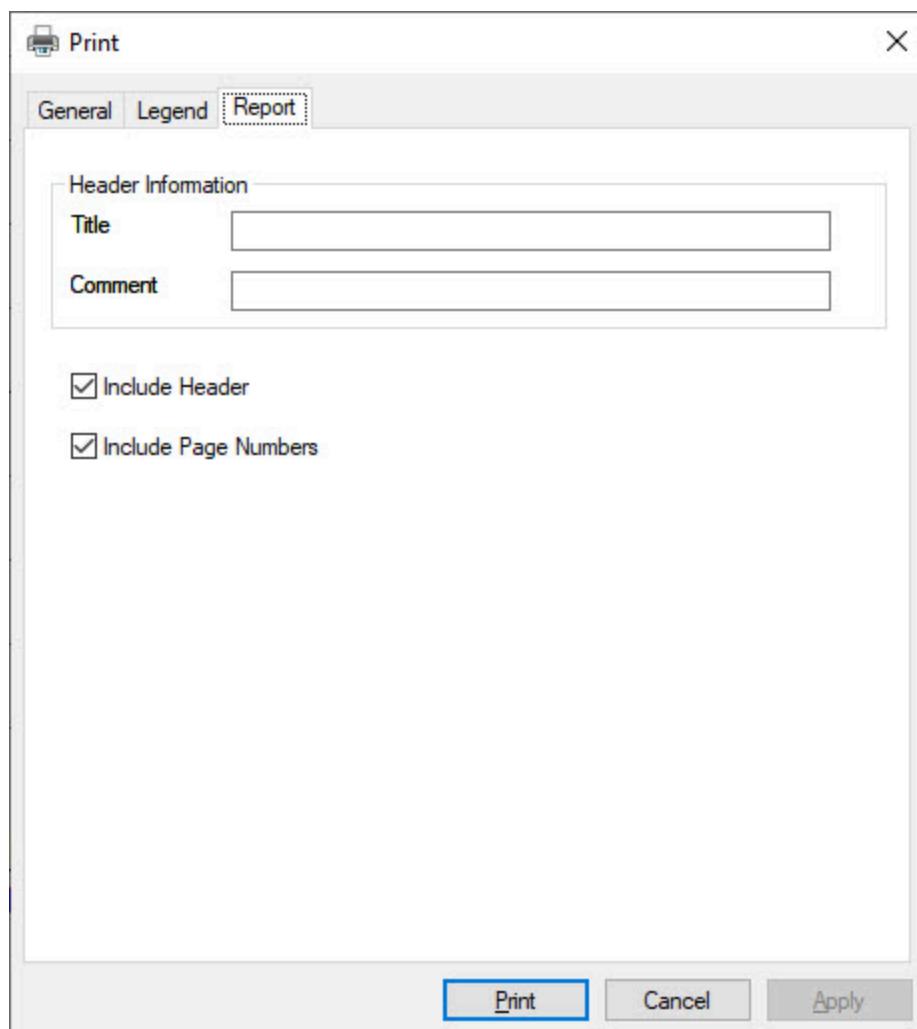
## Setting up Report Options

You can configure your reports to include a report header, which can include a report title and comment. For multiple-page reports, you can include page numbers, which appear at the bottom of each report page.

You set up your report options by using the Report panel of the Print dialog box.

### To set up report options:

1. In the Print dialog box, click the **Report** tab. The Report panel appears.



2. In the **Header Information** area, type a **Title** for the report. If necessary, include a **Comment**. Comments are printed under the report title on each report page.
3. To include a header, make sure the **Include Header** check box is selected.
4. To include page numbering, make sure the **Include Page Numbers** check box is selected.
5. Click **Apply**.

## Exporting Pen Data

You can export Process Analyst data for pens that are visible to either the Windows Clipboard (by using the **Copy to Clipboard** command) or to an Microsoft Excel-compatible file (**Copy to File**).

When you export data, it is exported using a standard format of columns that represent time, milliseconds, and then a column per pen, as shown here:

Time	Milliseconds	Pane1-Pen1	Pane1-Pen2	Pane1-Pen3
15/06/2004 01:17:25	100	NA	10	Off
15/06/2004 01:17:26	100	1	20	Low [Unacknowledged]
15/06/2004 01:17:27	100			Low [Acknowledged]
15/06/2004 01:17:28	100	3	25	Low [Acknowledged]

Export functionality doesn't simply return the sample markers displayed on the graph. Instead, it exports an interpolated value per display period from the start time to the end time of the pen. The display period can be calculated by dividing the time span of the pen by the [IProcessAnalyst.NumberofSamples\[Property\]\[Get/Set\]](#) property.

Before exporting the data, the Process Analyst sorts the timestamps for pens from the earliest to the latest sample. When the pens are unlocked and have different time spans, the data for each pen might have different timestamps. As each entry is added to a row in the table, the value of the pen at that particular timestamp is exported. If a pen does not have a sample for that timestamp, the column for that pen is left blank.

An export will also write values of NA, GATED and alarm states as localized text when necessary.

Pen columns use the format *<pane>-<pen>* where *pane* is the name of the pane that contains the pen, and *pen* is the name of the pen.

## See Also

[Copying Data to the Clipboard](#)

[Copying Data to File](#)

### Copying Data to the Clipboard

Copying pen data to the Clipboard allows you to paste the data into another application, such as an Excel spreadsheet.

#### To copy data to the Clipboard:

1. Select the pen(s) you want to copy data for.
2. Click **Copy to Clipboard**, or select **Copy** from the right-click (context) menu.

### Copying Data to File

Copying pen data to Microsoft Excel allows you to manipulate the data using spreadsheet application capabilities.

#### Note

- The Time column is an encoded (OLEDATE) double value, which holds the date and time in seconds in local

time. When exporting pen data to Excel, change the format of the Time column to dd/mm/yyyy hh:mm:ss so that the time is displayed correctly. Because the OLEDATE data type excludes milliseconds, a separate column is provided, which exports the millisecond component for each timestamp.

- The results exported are in Unicode format. use Excel 2000 and later, which support this format.

#### To copy data to file:

1. Select the pen(s) you want to copy data for.
2. Click **Copy to File**. The Save As dialog box appears.
3. Enter a filename and click **Save**. The data is exported in a delimited format.
4. Open the file you just created, and complete the Text Import Wizard.

## Customize Process Analyst

The Process Analyst integrates into the Plant SCADA system and is designed to work primarily with the Plant SCADA Graphics Builder and the run time environment. But the Process Analyst can also be embedded in custom Visual Basic and .NET applications. In these situations Plant SCADA is still necessary.

### See Also

[Configuring the Process Analyst Control from Graphics Builder](#)

[Security and Permissions](#)

[Multi-language Support](#)

[Persistence](#)

[Backing up Projects](#)

## Configuring the Process Analyst Control from Graphics Builder

Being an ActiveX control, you can insert the Process Analyst onto a Plant SCADA graphics page. To do this, do one of the following:

- In Graphics Builder, choose **Edit | Insert ActiveX Control**. The Insert ActiveX dialog box appears. Double-click the **Citect Process Analyst Control** item in the **ActiveX Controls** list box. The control is inserted onto the graphics page and the Properties dialog box appears.
- Click the **Process Analyst** button in the Graphics Builder toolbox.



After inserting the Process Analyst into a page, you can re-size it into position. To view the configuration pages for the Process Analyst, double-click the Process Analyst control. For details on configuring the design time properties for the Process Analyst, see [Configuring Design Time Properties](#).

### See Also

[Persistence](#)

## Security and Permissions

The Process Analyst integrates into the Plant SCADA security model by allowing access to certain Process Analyst features based on the privilege level of the currently logged in Operator.

The Process Analyst has nine privilege levels. Privilege level zero (0) indicates there is no security and any user can perform the function. Levels 1-8 map directly to the eight (8) privilege levels of security provided by Plant SCADA. The Process Analyst, by default, assumes the area of the page that it is situated on; this can be changed in the Graphics Builder. So if an operator has area access for the page and has privilege level 1, and the function they want to use is level 2, the function will be unavailable. If the operator had level 2, the function would then become available. The Process Analyst also supports the Plant SCADA Hierarchical Privilege security option.

Security can be applied to the following features:

- Administration; for details, see [Administration Privilege](#).
- Commands; for details, see [Command Privilege](#).
- Saving Process Analyst views (write privilege); for details, see [Write Privilege](#).

### Administration Privilege

The Process Analyst also uses an Administration privilege level to disable engineer-oriented features at run time. For example, the ability to add new custom commands and so on are disabled if the Operator does not meet the necessary privilege level. The Administration privilege level has never to be zero on a running system as this would expose properties to an Operator, which could adversely affect the performance of the client and/or server (for example, Number Of Samples property).



#### UNACCEPTABLY SLOW PROGRAM EXECUTION

Do not set the Administration Privilege level to zero on a running system.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

The features that are disabled when an operator does not meet the Administration privilege level include:

- Add Pen context menu in Property dialog box.
- New/Edit/Delete command (includes changing the privilege of commands).
- New/Delete Column.
- Data Refresh Rate property.
- Display Refresh Rate property.
- Number Of Samples property.
- Server Paths tab.
- Server field on Connection tab.
- Tag field on Connection tab.

To modify the Administration privilege level, see [Configure Chart-wide Properties](#).

## Command Privilege

The Process Analyst allows a privilege level to be assigned to each command (standard or custom command). If an Operator does not have the necessary privilege level to use that command, the associated toolbar button is disabled and cannot be executed.

## See Also

[Editing Existing Custom Commands](#)

## Write Privilege

The Process Analyst uses a concept of "write" privilege level to control whether an operator can save Process Analyst views to a location other than "My documents". These views can then be loaded into the Process Analyst later by any Operator. The write privilege is set at design time on the Server Paths property page located on the root Process Analyst node in the Property pages dialog box. If the write privilege level is set to zero (0), any operator can save to any location. If the write privilege is any other level, the Operator needs to have that privilege level to be able to save an Analyst view to a location other than "My Documents".

## See Also

[Configure Server Paths](#)

[Working with Views](#)

[Process Analyst View Synchronization](#)

## Multi-language Support

The Process Analyst supports the Plant SCADA multilanguage ability of changing the user interface language dynamically at run time. If the language is changed in Plant SCADA, the Process Analyst will change its language to match.

The process of configuring the Process Analyst for multiple languages is different from that of Plant SCADA. This section describes how to localize the Process Analyst user interface and get it to work with Plant SCADA.

## See Also

[About Process Analyst Resources](#)

[Creating Your Own Process Analyst Resource.dll](#)

[Localizing the Process Analyst Resource DLL](#)

[Using Plant SCADA to Switch the Process Analyst Language](#)

[Changing the Input Language](#)

## About Process Analyst Resources

The Process Analyst uses a special file, called Resources.dll, to store its display strings and dialog boxes. This file

holds the native translations for your version of the Process Analyst; these native translations are considered the default language. For example, the Japanese version of the Process Analyst will contain Japanese resources inside the Resources.dll file.

A separate Resources.dll file is required for each individual language that you want to support in your system. Each file uses a special format to indicate the language. The Process Analyst expects the file to be named:

Resources\_<LanguageCode>.dll.

Where <LanguageCode> is the unique identifier of your dll.

For example, a French resources file will be named Resources\_fr.dll. Plant SCADA uses the RFC 1766 standard for specifying culture names.

The following files are installed with Plant SCADA:

- Resources.dll — Default
- Resources\_de.dll — German
- Resources\_en.dll — English
- Resources\_es.dll — Spanish
- Resources\_fr.dll — French
- Resources\_it.dll — Italian
- Resources\_ja.dll — Japanese
- Resources\_ko.dll — Korean
- Resources\_no.dll — Norwegian
- Resources\_pt.dll — Portuguese
- Resources\_pt-br.dll — Brazilian Portuguese
- Resources\_ru.dll — Russian
- Resources\_sv.dll — Swedish
- Resources\_zh-cn.dll — Chinese PRC

They are located by default in the following directory:

C:\Program Files (x86)\Common Files\AVEVA Plant SCADA

You can also [Creating Your Own Process Analyst Resource.dll](#).

All language Resources\*.dll files need to be placed in the same directory as the Analyst.dll file.

The example below shows a system that contains English as the default, and has alternative languages of French, German and Chinese.

- Resources.dll (default - any language, for example English)
- Resources\_fr.dll (French standard)
- Resources\_zh-CN.dll (Chinese PRC)
- Resources\_de.dll (German)

## **Creating Your Own Process Analyst Resource.dll**

To create your own resources dll, you need to do the following:

1. Install the specific languages you are localizing on your Windows system.
2. Set your system to use that specific language.

---

**Note:** To create your own resources.dll file, you will need to use Microsoft Developer Studio 6 or an equivalent tool.

---

## See Also

[Localizing the Process Analyst Resource DLL](#)

### Localizing the Process Analyst Resource DLL

Once you have set up your system to cope with multiple languages, you can begin localizing. Do the following:

1. Open Microsoft Visual Studio.
2. Choose **File | Open**.
3. Browse to the location of the Process Analyst's Resources.dll file. By default it is located at:  
%PROGRAMFILES(X86)%\Common Files\AVEVAPlant SCADA
4. Verify that the **Files of type** menu has **Executable Files (.exe;.dll;\*.ocx)** selected.
5. Verify that the **Open as** menu has **Resources** selected.
6. Select **Resources.dll** and click **Open**.
7. Save the file under a new name. For example, if you are localizing for Japanese, use **Resources\_ja-JP.dll**. See [About Process Analyst Resources](#) for naming conventions.
8. Before changing any string, you need to change the language code for each dialog box and the string table by doing the following:
  - a. Expand the **String Table** folder in the tree.
  - b. Right-click the **String Table** entry.
  - c. Choose **Properties** from the right-click (context) menu (see below).
  - d. From the **Language** menu, select the language that you are localizing for.
  - e. Click **Close** in the top-right corner of the dialog.
  - f. Repeat these steps for each of the dialogs inside the **Dialog** folder.

Once the language code has been set for every dialog and the string table, you are ready to begin changing the text.

### Localizing dialog boxes

To localize a dialog box, do the following:

1. Expand the **Dialogs** folder in the tree.
2. Double-click a dialog to edit.
3. Select an item of text and right-click to display the properties for that item.
4. Enter your replacement text into the **Caption** field.

5. Click the **Close** button in the top-right corner of the dialog box.

Note the following:

- Controls can be repositioned or re-sized if necessary to fit your replacement text.
- Never re-size a dialog box. The size of a dialog box is set to an optimum size so that it integrates into Graphics Builder correctly.
- Dialogs 3028 and 3050 do not require translation.

## Localizing the String Table

To localize the string table, do the following:

1. Expand the **String Table** folder in the tree.
2. Double-click the **String Table** entry. This will display a table showing you the strings of the Process Analyst.
3. Double-click an entry to display the Properties dialog box.
4. Type in the replacement text in the **Caption** box.
5. Click the Close button in the top-right corner of the dialog box.

---

**Note:** When translating strings, if a string contains "%s", "%x", "%d" and so on, *do not* remove or replace those symbols as they are important to the Process Analyst.

---

## Using Plant SCADA to Switch the Process Analyst Language

Plant SCADA uses the Cicode function Login() or UserLogin(). To allow the Process Analyst to determine the language to display, you need to map your Plant SCADA language databases to the Process Analyst resource files.

To do this, add a new .ini section called [ProcessAnalyst] to the Citect.ini file on each of your Plant SCADA clients and servers, and create a mapping for each language. (This section might already exist in your Citect.ini file.) The mapping needs to use this format:

LanguagePath.<Language>=<ProcessAnalystLanguage>

Where <Language> is the name of a specific Plant SCADA language as defined in Plant SCADA Studio's **Setup** activity.

For example,

```
[ProcessAnalyst]
LanguagePath.French=fr LanguagePath.French(France)=fr
LanguagePath.Chinese=zh-CN
LanguagePath.German=de
```

The last step is to verify each of your machines contains the necessary language fonts. Windows provides facilities to add the necessary languages to your machine via the Regional and Language Options dialog box, accessible from the Control Panel. This step is necessary if you want to use Asian languages on an English operating system. See [Creating Your Own Process Analyst Resource.dll](#) for details on adding languages to your system.

With the .ini file now configured, languages installed, and the Resource.dll files in place, when either the Login() or UserLogin() Cicode functions are called, Plant SCADA and the Process Analyst will automatically change into the selected language.

## See Also

[Manually Switching Languages using IProcessAnalyst.Language](#)

### Manually Switching Languages using IProcessAnalyst.Language

The Process Analyst can also switch languages by itself using the IProcessAnalyst.Language property. You can call this property directly from Cicode.

**Note:** Using this method will only switch the Process Analyst language and not the one used by Plant SCADA. See [IProcessAnalyst.Language \[Property\] \[Get/Set\]](#).

### Changing the Input Language

When your system has been configured to use multiple languages, you will find a new icon in the system tray displayed as "EN" or similar.

**To change input language:**

1. Click EN to display the input language option menu.



2. Select the language you want to use (to work correctly with Visual Studio, to match the language you selected in Windows setup). This might display a language-specific IME editor, which allows you to select characters to use in your translations.

## Persistence

Persistence refers to saving the state (properties, pens, and so on) of the Process Analyst to disk. Plant SCADA and the Process Analyst provide the following methods of persistence:

- Saving as part of a Plant SCADA Graphics Builder page (design time)
- Save View toolbar button on the Process Analyst (run time)
- SaveToFile automation method on the Process Analyst (run time)
- Saving between Plant SCADA page transitions (run time)

### Saving while Using Graphics Builder

This feature allows you to configure the default look and/or what pens will be displayed on the Process Analyst at design time while you are designing your graphics pages in Plant SCADA's Graphics Builder.

Design time is the appropriate time to configure the appearance properties, toolbar buttons and, most importantly, the security of the Process Analyst since these will become the default settings of the Process Analyst when your page is displayed at run time.

When a page containing the Process Analyst is saved in the Graphics Builder the properties you configured on the Process Analyst will be stored within the Graphics Builder page.

---

**Note:** When defining new custom toolbar buttons, any icon you assign will be copied and also stored within the Graphics Builder page. This allows your custom toolbar buttons to work on any machine.

---

## Using the Save View Toolbar Button

This feature is valid only at run time and allows operators to save the current state of the Process Analyst (called a *view*) to a standalone file. These files can be loaded during run time, and are an efficient way to store commonly used pen configurations.

## Using the SaveToFile Automation Method

This feature is valid only at run time and allows a user to write Cicode to save the current state of the Process Analyst to a standalone file, referred to as an Analyst view. These files can be loaded during run time using the LoadFromFile automation method (or the **Load View** toolbar button).

Views are an efficient way to store commonly used pen configurations.

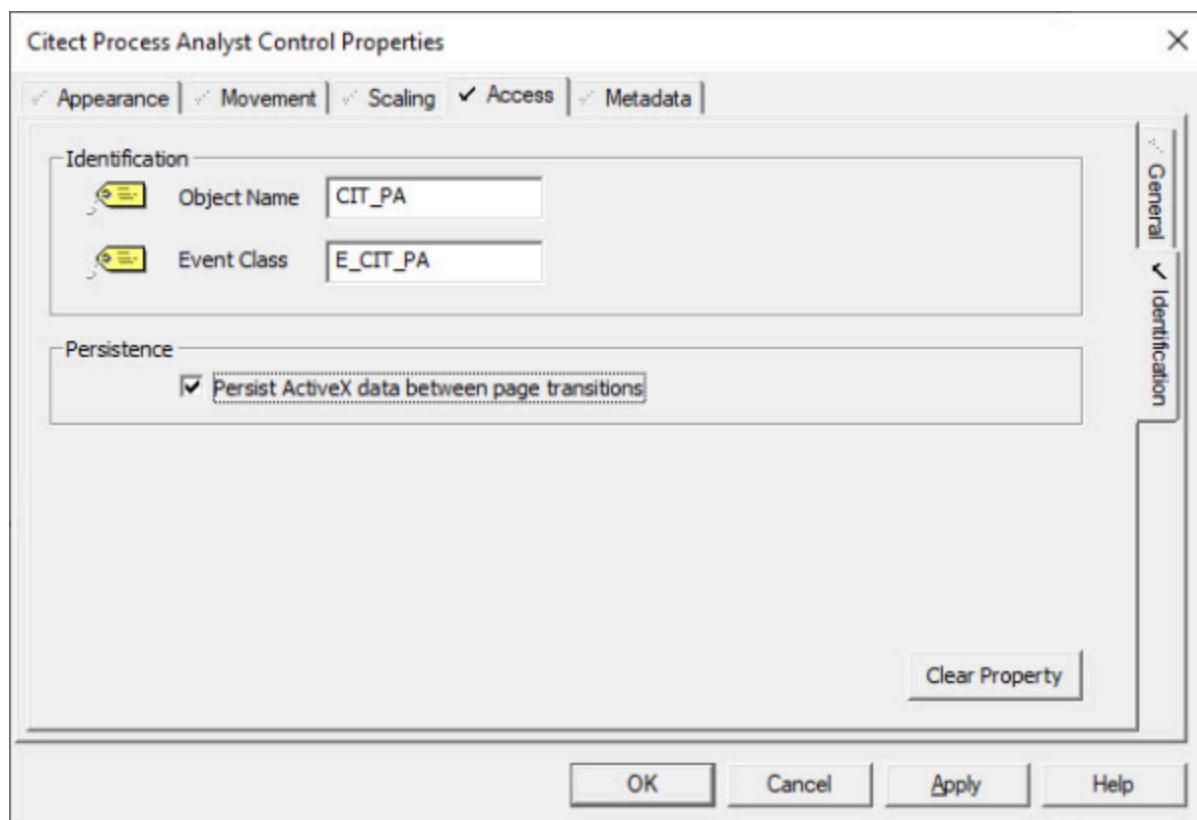
## Saving between Plant SCADA Page Transitions (Runtime)

Using Plant SCADA run time, if you modify the Process Analyst (for example, changing the timespan of a pen) and move off the page, your changes will be lost. This behavior is not always what you want, so Graphics Builder provides an option **Persist ActiveX data between page transitions** to save the state of an ActiveX control when you switch between pages.

Enabling this option causes Plant SCADA to write a temporary file to the Plant SCADA Data directory in the format of <Event class>.stg whenever you leave a page that contains an ActiveX object (for example, the Process Analyst). When you reenter the page, Plant SCADA looks for that same file and, if found, will load the settings from it. These files only exist while Plant SCADA run time is running. When you shut down Plant SCADA, the temporary \*.stg files are deleted.

### To save between page transitions:

1. Double-click the Process Analyst ActiveX control you want to change. The Properties dialog box appears.
2. Click the **Access** tab.
3. Click the **Identification** tab. The Identification panel appears.



4. In the Persistence area, select the **Persist ActiveX data between page transitions** check box, and then click **Apply**.

## Resetting Back to the Default State

You can reset the original configuration of the Process Analyst control by calling the Cicode function `ObjectResetState`. This function takes the object handle of the Process Analyst control, which you retrieve by using the Cicode function `ObjectByName`.

## Backing up Projects

When you save views to the Local storage location, the Process Analyst will create a \*.pav file in an Analyst Views subfolder under your project directory. If your project contains Analyst views, verify that the **Save sub-directories** option is selected in the Backup Project dialog box before backing up your project.

## Configuring Design Time Properties

Most Process Analyst properties can be defined or modified during run time and design time. This section describes properties that can be configured only during design time, usually by a User.

For information about configuring run time properties, see [Process Analyst Properties Dialog Box](#).

## See Also

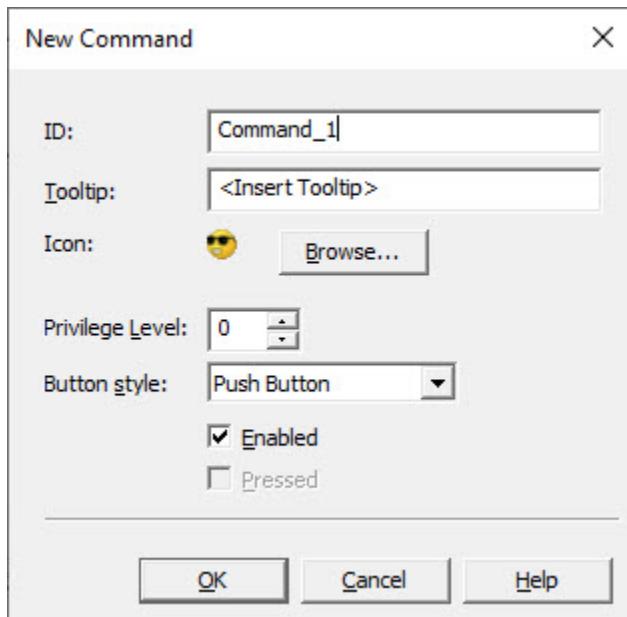
- [Adding New Commands](#)
- [Editing Existing Custom Commands](#)
- [Creating or Editing Object View Columns](#)
- [Process Analyst View Synchronization](#)

### Adding New Commands

Users can define new toolbar commands during design time if they have the appropriate privilege level.

#### To add a new command:

1. On the Toolbars page of the Properties dialog box, click **New**. The New Command dialog box appears.



2. The dialog shows the unique, system-generated ID for the new command. If necessary, enter a new **ID** for the command. This ID can be used in Cicode to determine which command has been triggered or to find a specific command in the Plant SCADA system.
3. Enter the **Tooltip** text for the new command. You are limited to 64 characters. Tooltip text appears when the mouse pointer is over the toolbar command.
4. Click **Browse** and navigate to the icon to represent the new command. The icon image appears on the toolbar command button.
5. To define how the command behaves, choose a button style from the **Button style** menu:
  - **Push Button** - click the **Enabled** check box to set the default appearance of the button when the button is enabled or disabled.
  - **Toggle Button** - click **Enabled** or **Pressed** to specify the "on" appearance.

## See Also

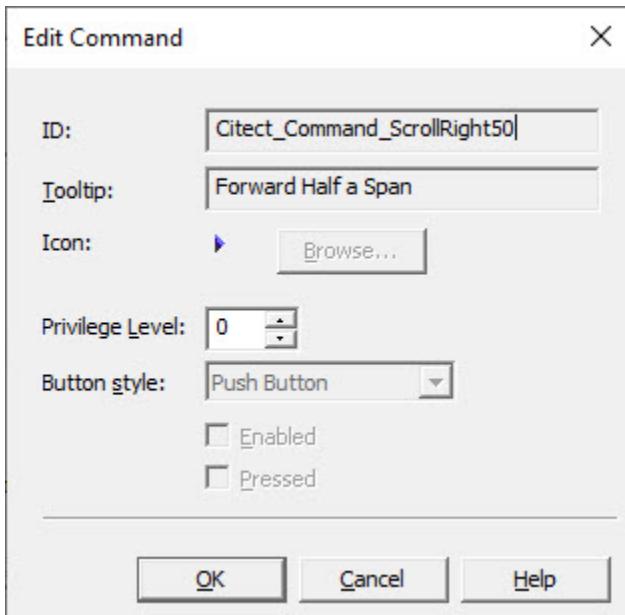
[Editing Existing Custom Commands](#)

### Editing Existing Custom Commands

Users can edit existing toolbar commands if they have the appropriate privilege level. Commands can only be edited during design time, and only fields for custom commands can be edited.

#### To edit an existing custom command

1. Open the Properties dialog box and click the **Toolbars** tab.
2. Select the command you want to edit in the **Available toolbar buttons** list box, and then click **Edit**. The Edit Command dialog box appears.



3. If necessary, click **Browse** to navigate to a new icon to use for the command.
4. If necessary, edit the **Tooltip** text. The maximum length for Tooltip text is 64 characters.
5. If necessary, choose a new button style from the **Button style** menu.

## See Also

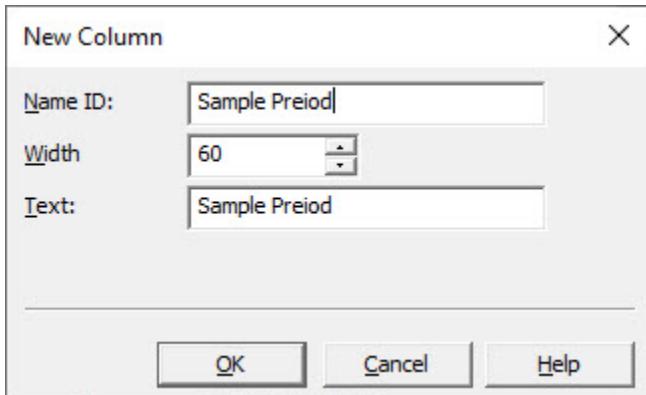
[Adding New Commands](#)

### Creating or Editing Object View Columns

Users can create or delete Object View columns (during design time), as well as edit existing columns (run time or design time). Object View columns display information about your pens. These configuration tasks are performed by using the Citect Process Analyst Properties dialog box.

**To create an Object View column:**

1. Click the **Object View** tab. The Object View panel appears.
2. Click **New**. The New Column dialog box appears.



3. Enter a **Name ID** for the column. The value is used to reference the column in code.
4. Specify a **Width**.
5. Enter the **Text** to use for the column in the Object View display.

**To delete an Object View column:**

- Select the column you want to delete and click **Delete**.

**To edit an Object View column:**

1. Select the column you want to edit, and then click **Edit**. The Edit Column dialog box appears.
2. Modify the information as necessary, and then click **OK**.

**See Also**

[Configuring the Object View](#)

**Process Analyst View Synchronization**

The Process Analyst implements a basic level of file synchronization for Process Analyst views (.pav files). This feature causes the Process Analyst to try and obtain the latest version of a .pav file before displaying it to the operator.

To achieve this, an engineer needs to first configure the Process Analyst to support Primary and Standby server locations for Analyst Views; for details, see [Configure Server Paths](#).

With these file servers in place, the Process Analyst now has a central location from which to obtain Process Analyst views. If one of the locations is unavailable, the operator can try the alternate location. When a client saves or loads a Process Analyst view, only that view on the Primary and Standby server locations will be synchronized to verify they are the same.

The table below outlines the rules of synchronization and privilege for the storage locations and client modes when loading and saving Process Analyst views.

Action	Mode	Privilege	Available Storage Locations*
Load	Normal client	Both**	The Primary and Standby options appear as configured as well as My Documents. If either are invalid or unavailable paths, they do not appear. If both are invalid or unavailable, the Local option appears. Default order is Primary, Standby, My Documents, and Local, My Documents respectively. Synchronisation occurs when loading from a Primary or Standby location.
Save	Normal client	Privileged	The Local and My Documents options are the only ones available. Local however will attempt to save to all server locations as well as the project directory. The pav file will be saved to all available locations from primary, standby and project directories. Default order: Local, My Documents.
Save	Normal client	Unprivileged	The My Documents option is the only one available.
Save	Normal client	Both**	The My Documents option is the only one available.

\* Refers to the **Look in** menu on SaveView and Load View dialog boxes.

\*\* Means both privileged and un-privileged.

When setting up file-servers to store Process Analyst views, verify that each client machine has privileges enabling it the desired read/write access to those locations.

## See Also

[Working with Views](#)

[Write Privilege](#)

## Using the Process Analyst Command System

This section describes how to use the Process Analyst command system.

## See Also

[Command System Overview](#)

[Custom Commands](#)

[Icons](#)

### Command System Overview

The Process Analyst provides an extensive command system allowing manipulation of common Process Analyst features, as well as providing the framework for creating custom user-defined commands.

The command system is configurable via the Toolbar property page and the automation model.

To access the command system via the automation model, call the property `GetCommandSystem()` from the [IProcessAnalyst interface](#). For details, see [IProcessAnalyst Interface](#).

### Custom Commands

Custom commands are defined in the Process Analyst, but needs to be implemented in Cicode. You define commands by using the `ICommandSystem->Create` method, or by using the **New** button on the Toolbar property page.

To implement the command, you need to respond to the event `CommandExecuted` (and optionally `UpdateCommand`). Both of these events notify you of the ID of the command which needs to be handled.

---

**Note:** Do not call commands from Cicode which will require input from the operator such as Add Pen, Open View, Save View, etc because they will hang the Plant SCADA runtime until the user closes the dialog.

---

### CommandExecuted

When an operator presses the toolbar button representing your command, it will trigger this event. This is your opportunity to execute the desired functionality of the command. This will not be triggered if the logged-in user does not meet the necessary privilege level. Be aware that this is an asynchronous operation.

### UpdateCommand

When the Process Analyst requires the Enable state or pressed states of its toolbar buttons to be refreshed, this event will be triggered. This will not be triggered if the logged-in user does not meet the necessary privilege level. This is asynchronous operation.

The state of commands (custom and pre-defined) will be saved to disk whenever the Process Analyst configuration is saved.

## See Also

[Persistence](#)

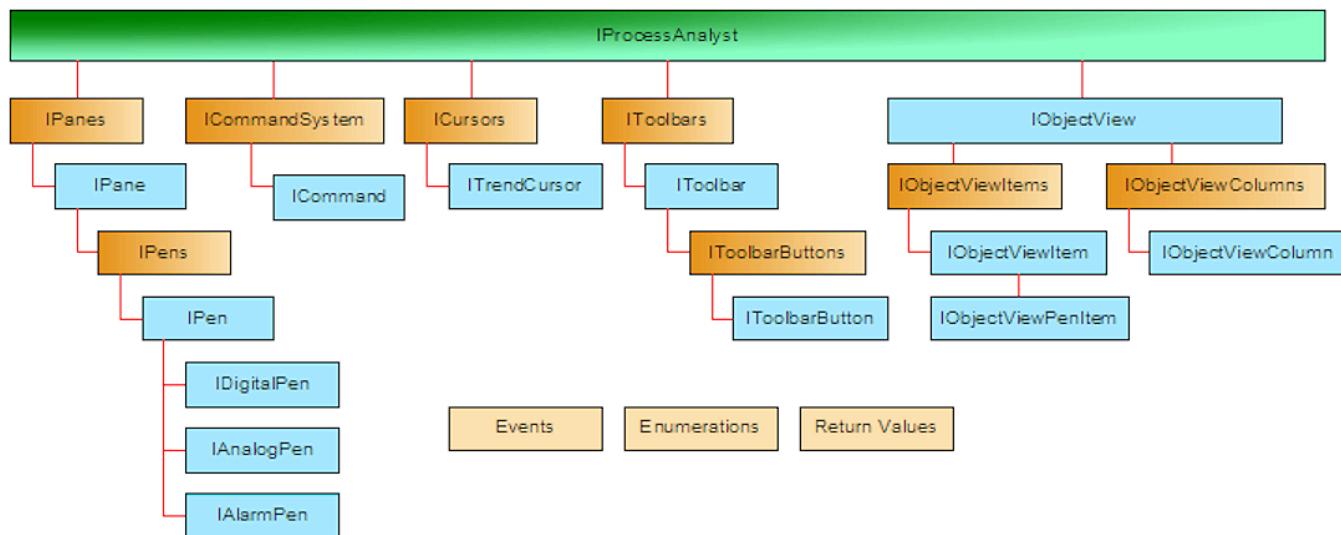
## Icons

For custom commands, the user can specify their own custom icons by pointing to a file on their hard drive. As these files may be deleted or moved over time, the Process Analyst makes an instant copy of the icon into memory when the command is added. This removes any dependence on the original icon file. When the Process Analyst configuration is saved, the icon data is also saved.

## Automation Model

The automation model allows applications or solutions to programmatically configure the Process Analyst control's appearance, performance, and behavior. The automation model also allows code, via automation events, to be attached to events fired from the Process Analyst Control and perform custom behavior.

The automation model allows almost every visual aspect of the control to be configured, as well as performance. It is simple and follows a traditional object-oriented approach (see below).



## See Also

[IPanes Interface](#)

[IPens Interface](#)

[ICommandSystem Interface](#)

[ICursors](#)

[IToolbars Interface](#)

[IToolbarButtons Interface](#)

[IObjectViewItems Interface](#)[Events](#)[Enumerations](#)[Return Values](#)

## Execution Results

Each property and method listed in the automation model will return one of the following results upon execution. The exact meaning is described in the Execution Result section for each property or method.

Execution Result	Cicode	VBA	C++
InvalidArgument	274	5	E_INVALIARG
GeneralFailure	356	2147500037	E_FAIL
PathNotFound	356	76	STG_E_PATHNOTFOUND
Success	0	-	S_OK

Errors are captured differently in Cicode and VBA. The following code examples show how to trap and handle errors in VBA and Cicode.

### [VBA]

```
Sub VBATest(myObject As Object)
    On Error Goto errHandler
    myObject.<function>
    Exit Sub
    errHandler:
    Print Err.Number, Err.Description
    Resume Next
End Sub
```

### [Cicode]

```
FUNCTION Test1(OBJECT hObject)
    ErrSet(1); // Enable User error checking (disabled HW alarm)
    IF ObjectIsValid(hObject) THEN
        _ObjectCallMethod(hObject, "<function>");
        error = IsError();
        errorMessage = IntToStr(error)
        IF (error <> 0) THEN
            Message("An error occurred", errorMessage, 0);
        END
    END
    ErrSet(0); // Enable hardware alarm reporting of errors
END
```

## Enumerations

- [AlarmType \[Enumeration\]](#)
- [AxisLabelType \[Enumeration\]](#)
- [DisplaySize\[Enumeration\]](#)
- [ErrorNotifyCode \[Enumeration\]](#)
- [FileLocation \[Enumeration\]](#)
- [HatchStyle \[Enumeration\]](#)
- [LineStyle \[Enumeration\]](#)
- [LineType \[Enumeration\]](#)
- [PenNameMode \[Enumeration\]](#)
- [PenType \[Enumeration\]](#)
- [PointType \[Enumeration\]](#)
- [QualityCompactionType \[Enumeration\]](#)
- [QualityType \[Enumeration\]](#)
- [RequestMode \[Enumeration\]](#)
- [ToolbarButtonType \[Enumeration\]](#)

### AlarmType [Enumeration]

Specifies the visual representation for an alarm pen.

#### Defined As

- [VBA] Integer
- [Cicode] INT
- [C++] AlarmType

#### Members

Member Name	Description	Value
AlarmType_Digital	The tag is digital alarm	0
AlarmType_Analog	The tag is an analog alarm	1
AlarmType_Advanced	The tag is an advanced alarm	2
AlarmType_TimeStamped	The tag is a time-stamped alarm	3
AlarmType_MultiDigital	The tag is a multi-digital alarm	4
AlarmType_ArgyleAnalog	The tag is a legacy Argyle analog alarm	5
AlarmType_TimeStampedDigital	The tag is a digital time-stamped	6

Member Name	Description	Value
	alarm	
AlarmType_TimeStampedAnalog	The tag is a analog time-stamped alarm	7

## See Also

[IAuditPen.AlarmType \[Property\]\[Get/Set\]](#)

### AxisLabelType [Enumeration]

Specifies how the labels are drawn on the vertical axis.

#### Defined As

- [VBA] Integer
- [Cicode] INT
- [C++] AxisLabelType

#### Members

Member Name	Description	Value
AxisLabelType_NONE	No labels will be visible on axis	0
AxisLabelType_DOUBLE	Displays in decimal format	1
AxisLabelType_INTEGER	Displays in integer format	2
AxisLabelType_PERCENT	Displays as "%"	3
AxisLabelType_AMPS	Displays as "A"	4
AxisLabelType_DEGREES	Displays as "deg"	5
AxisLabelType_FEET	Displays as "ft"	6
AxisLabelType_FEETPERMIN	Displays as "ft/min"	7
AxisLabelType_FEETPERSEC	Displays as "ft/s"	8
AxisLabelType_GALLONS	Displays as "gal"	9
AxisLabelType_GALLONSPERHR	Displays as "gal/h"	10
AxisLabelType_GALLONSPERMIN	Displays as "gal/min"	11
AxisLabelType_GALLONSPERSEC	Displays as "gal/s"	12

Member Name	Description	Value
AxisLabelType_HERTZ	Displays as "Hz"	13
AxisLabelType_KILOGRAMS	Displays as "kg"	14
AxisLabelType_KILOGRAMSPERHR	Displays as "kg/h"	15
AxisLabelType_KILOGRAMSPERMIN	Displays as "kg/min"	16
AxisLabelType_KILOGRAMSPERSEC	Displays as "kg/s"	17
AxisLabelType_KILOMETRESPERHR	Displays as "kg/h"	18
AxisLabelType_KILOPASCALS	Displays as "kPa"	19
AxisLabelType_KILOWATTS	Displays as "kW"	20
AxisLabelType_LITRES	Displays as "l"	21
AxisLabelType_LITRESPERHR	Displays as "l/h"	22
AxisLabelType_LITRESPERMIN	Displays as "l/min"	23
AxisLabelType_LITRESPERSEC	Displays as "l/s"	24
AxisLabelType_METRES	Displays as "m"	25
AxisLabelType_METRESPERMIN	Displays as "m/min"	26
AxisLabelType_METRESPERSEC	Displays as "m/s"	27
AxisLabelType_REVNS	Displays as "Rev"	28
AxisLabelType_REVSPERHR	Displays as "Rev/h"	29
AxisLabelType_REVSPERMIN	Displays as "RPM"	30
AxisLabelType_TONNES	Displays as "t"	31
AxisLabelType_TONNESPERHR	Displays as "t/h"	32
AxisLabelType_VOLTS	Displays as "V"	33
AxisLabelType_WATTS	Displays as "W"	34
AxisLabelType_LOOKUP	Displays user-defined text for label	35

**DisplaySize[Enumeration]**

Specifies the size of graphical elements in the control's design and runtime surface.

**Defined As**

- [VBA]Integer
- [Cicode]INT
- [C++]DisplaySize

**Members**

Member Name	Description	Value
DisplaySize_Normal	Refers to a normal size	0
DisplaySize_Large	Refers to a large size for use with high resolution monitors	1

**See Also**

[IProcessAnalyst.DisplaySize\[Property\]\[Get/Set\]](#)

**ErrorNotifyCode [Enumeration]**

Defines known errors that can occur during operation.

**Defined As**

- [VBA] Integer
- [Cicode] INT
- [C++] ErrorNotifyCode

**Members**

Member Name	Description	Value
ErrorNotifyCode_None	No error.	0
ErrorNotifyCode_InvalidTag	Occurs when the tag specified for the pen does not exist.	1
ErrorNotifyCode_CtapiConnectionOffline	Occurs when connections cannot be made to the Trends and/or Alarm Servers.	2
ErrorNotifyCode_Unknown	Occurs when an unknown Plant SCADA or Windows error occurs.	3
ErrorNotifyCode_NoServer	Occurs when Plant SCADA cannot find a server to get the data from.	4
ErrorNotifyCode_InvalidArgument	Occurs when an invalid argument is specified.	5
ErrorNotifyCode_OutOfMemory	Occurs when a memory error is	6

Member Name	Description	Value
	detected.	
ErrorNotifyCode_BadVersion	Occurs when the Trends and/or Alarm Servers do not match the client version.	7
ErrorNotifyCode_NoPrivilege	Occurs when the current user does not have the necessary privileges to view the data.	8

## See Also

[Error \[Event\]](#)

### FileLocation [Enumeration]

Specifies the location to save and write Process Analyst views to.

#### Defined As

- [VBA] Integer
- [Cicode] INT
- [C++] FileLocation

#### Members

Member Name	Description	Value
FileLocation_Local	Refers to the project folder	0
FileLocation_Server	Refers to the both the primary/standby server paths	1
FileLocation_User	Refers to the My Documents folder	2

## See Also

[IProcessAnalyst.LoadFromFile \[Method\]](#)

[IProcessAnalyst.SaveToFile \[Method\]](#)

### HatchStyle [Enumeration]

Defines the filling style for Alarm pens.

#### Defined As

- [VBA] Integer
- [Cicode] INT
- [C++] HatchStyle

## Members

Member Name	Description	Value
HatchStyle_None	No pattern	0
HatchStyle_Horizontal	Horizontal line pattern	1
HatchStyle_Vertical	Vertical line pattern	2
HatchStyle_ForwardDiagonal	Forward diagonal line pattern	3
HatchStyle_BackwardDiagonal	Backward diagonal line pattern	4
HatchStyle_Cross	Cross pattern	5
HatchStyle_DiagonalCross	Diagonal cross pattern	6

## See Also

[IAlarmPen.SetHatchStyle \[Method\]](#)

[IAlarmPen.GetHatchStyle \[Method\]](#)

## LineStyle [Enumeration]

Defines the drawing style for a line.

### Defined As

- [VBA] Integer
- [Cicode] INT
- [C++] LineStyle

### Values

Member Name	Description	Value
LineStyle_SOLID	Draws a solid line (all line widths)	0
LineStyle_DASH	Draws a dashed line (line width 1 only)	1
LineStyle_DOT	Draws a dot line (line width 1 only)	2
LineStyle_DASHDOT	Draws a dash dot (line width 1 only)	3

Member Name	Description	Value
LineStyle_DASHDOTDOT	Draws a dash dot dot (line width 1 only)	4
LineStyle_NONE	Draws no line	5

**LineType [Enumeration]**

Defines the visual representation of the lines between samples of an analog pen.

**Defined As**

- [VBA] Integer
- [Cicode] INT
- [C++] LineType

**Members**

Member Name	Description	Value
LineType_STRAIGHT	A single line is drawn from point A to point B.	0
LineType_STEPPED	The line drawn will maintain the value of the previous sample. When the samples differ, a vertical line will be drawn to the new sample value.	1

**See Also**

[IAnalogPen.LineInterpolation \[Property\]\[Get/Set\]](#)

**PenNameMode [Enumeration]**

Defines how the pen name will be generated. It is used in conjunction with the IPens.Create method.

**Defined As**

- [VBA] Integer
- [Cicode] INT
- [C++] PenNameMode

**Members**

Member Name	Description	Value
PenNameMode_Comment	The comment field obtained from the Plant SCADA trend/alarm tag will be used as the pen name.	1
PenNameMode_TagPenNameMode_Tag	The value of the IPen.DataPoint property will be used as the pen name.	2
PenNameMode_Custom	Indicates that you will be setting the name using the IPen.Name property.	3

## See Also

[IPens.Create \[Method\]](#)  
[IPen.DataPoint \[Property\]\[Get/Set\]](#)  
[IPen.Name \[Property\]\[Get/Set\]](#)

## PenType [Enumeration]

Defines the plotting style of a Process Analyst pen.

### Defined As

- [VBA] Integer
- [Cicode] INT
- [C++] PenType

### Members

Member Name	Description	Value
PenType_ANALOG	A pen with an analog range	4097
PenType_DIGITAL	A pen with a range of 0 and 1	4098
PenType_ALARM	A pen represented as states	4099

## See Also

[IPens.Create \[Method\]](#)

## PointType [Enumeration]

Defines the visual cue applied to samples of a pen.

**Defined As**

- [VBA] Integer
- [Cicode] INT
- [C++] PointType

**Members**

Member Name	Description	Value
PointType_NONE	No marker	0
PointType_RECT	A rectangular marker	1
PointType_CIRCLE	A circular marker	2
PointType_PLUS	A plus marker	3
PointType_CROSS	A cross marker	4
PointType_TRIANGLE	A triangular marker	5
PointType_ELLIPSE	A elliptical marker	6

**See Also**

[IPen.SetQualityCompactionPointType \[Method\]](#)

**QualityCompactionType [Enumeration]**

Specifies the different types of presentation used for a sample.

**Defined As**

- [VBA] Integer
- [Cicode] INT
- [C++] QualityCompactionType

**Members**

Member Name	Description	Value
QualityCompactionType_Single	Representation when the marker represents a single sample	0
QualityCompactionType_Multiple	Representation when the marker represents a calculation of two or more samples	1
QualityCompactionType_Interpolat	Representation when the marker	2

Member Name	Description	Value
ed	represents interpolated samples	

## See Also

[IPen.SetQualityCompactionPointType \[Method\]](#)

### QualityType [Enumeration]

Defines the known quality states of data.

#### Defined As

- [VBA] Integer
- [Cicode] INT
- [C++] QualityType

#### Members

Member Name	Description	Value
QualityType_Good	The sample is good	0
QualityType_NA	Indicates a loss of connection	1
QualityType_Gated	Indicates the data was marked as unwanted	2

#### Remarks

An alarm pens "disabled" state is treated as QualityType\_Gated.

## See Also

[IPen.SetQualityLineStyle \[Method\]](#)

### RequestMethod [Enumeration]

Defines the data acquisition method for a pen.

#### Defined As

- [VBA] Integer
- [Cicode] INT
- [C++] RequestMethod

#### Members

Member Name	Description	Value
RequestMode_Average	The value will be an average of the individual samples within the multiple sample, as will the timestamp	0
RequestMode_Minimum	The value will be the minimum value out of the individual samples within the multiple sample. The timestamp will be the average of the individual samples.	1
RequestMode_Maximum	The value will be the maximum value out of the individual samples within the multiple sample. The timestamp will be the average of the individual samples.	2
RequestMode_Newest	The value will be the latest arrived value out of the individual samples within the multiple sample. The timestamp will be the average of the individual samples.	3

## See Also

[IPen.RequestMode \[Property\]\[Get/Set\]](#)

### ToolbarButtonType [Enumeration]

Defines the type of a toolbar button.

#### Defined As

- [VBA] Integer
- [Cicode] INT
- [C++] ToolbarButtonType

#### Members

Member Name	Description	Value
ToolbarButtonType_Push	Standard push button behavior.	0
ToolbarButtonType_Toggle	The button has two states: On and Off.	1
ToolbarButtonType_Separator	A visual marker used to group	2

Member Name	Description	Value
	buttons.	

## See Also

[ICommandSystem.Create \[Method\]](#)  
[ICommand.ButtonType \[Property\]\[Get\]](#)

## Events

- [CommandExecuted \[Event\]](#)
- [CursorMoved \[Event\]](#)
- [Error \[Event\]](#)
- [HorizontalAxisChanged \[Event\]](#)
- [MouseClick \[Event\]](#)
- [MouseDoubleClick \[Event\]](#)
- [OVColumnAdded \[Event\]](#)
- [OVColumnRemoved \[Event\]](#)
- [OVItemAdded \[Event\]](#)
- [OVItemChecked \[Event\]](#)
- [OVItemRemoved \[Event\]](#)
- [OVItemSelected \[Event\]](#)
- [PenCreated \[Event\]](#)
- [PenDeleted \[Event\]](#)
- [PenRenamed \[Event\]](#)
- [PenSelectionChanged \[Event\]](#)
- [PropertyChanged \[Event\]](#)
- [UpdateCommand \[Event\]](#)
- [VerticalAxisChanged \[Event\]](#)
- [ViewLoadedSaved\[Event\]](#)

### **CommandExecuted [Event]**

This event is raised when a command is executed.

#### **Defined As**

- [VBA] CommandExecuted(commandId As String)
- [Cicode] CommandExecuted (OBJECT processAnalyst, STRING commandId)
- [C++] CommandExecuted (BSTR commandId)

**Parameters****commandId**

[in] Contains the unique identifier of the command that was executed.

**processAnalyst**

[in] Indicates the Process Analyst object which raised the event. (Cicode only)

**Remarks**

Each toolbar button is associated with a command so when they are pressed this event will be raised with the unique identifier of that command. You can then use that identifier to determine which command was executed.

By using this event you can implement your own custom commands.

**Calling Syntax**

Assumes you have a Process Analyst on a page with an event class defined as myPage\_AN35.

**[VBA]**

```
Sub myPage_AN35_CommandExecuted(commandId As String)
End Sub
.....
```

**[Cicode]**

```
FUNCTION myPage_AN35_CommandExecuted(OBJECT processAnalyst, STRING commandId)
END
```

**See Also**

[UpdateCommand \[Event\]](#)  
[ICommandSystem.Execute \[Method\]](#)

**CursorMoved [Event]**

This event is raised whenever the cursor position changes.

**Defined As**

- [VBA] CursorMoved(cursor As Object, position As Integer)
- [Cicode] CursorMoved (OBJECT processAnalyst, OBJECT cursor, INT position)
- [C++] CursorMoved (IPen\* pen, int position)

**Parameters****cursor**

[in] Refers to the cursor that has moved.

**position**

[in] Indicates the new position of the cursor.

**processAnalyst**

[in] Indicates the Process Analyst object which raised the event. (Cicode only).

#### Calling Syntax

Assumes you have a Process Analyst on a page with an event class defined as myPage\_AN35.

### [VBA]

```
Sub myPage_AN35_CursorMoved(pen As Object, position As Integer)
End Sub
```

### [Cicode]

```
FUNCTION myPage_AN35_CursorMoved(OBJECT processAnalyst, OBJECT cursor, INT position)
END
```

### Error [Event]

This event is raised whenever an error is generated from the Process Analyst.

#### Defined As

- [VBA] Error(errorCode As Integer, errorMessage As String)
- [Cicode] Error(OBJECT processAnalyst, INT errorCode, STRING errorMessage)
- [C++] Error(ErrorNotifyCode errorCode, BSTR errorMessage)

#### Parameters

errorCode

[in] Indicates the error that occurred. See the ErrorNotifyCode enumeration.

errorMessage

[in] Contains the message associated with the error code.

processAnalyst

[in] Indicates the Process Analyst object which raised the event. (Cicode only)

#### Calling Syntax

Assumes you have a Process Analyst on a page with an event class defined as myPage\_AN35.

### [VBA]

```
Sub myPage_AN35_Error(errorCode As Integer, errorMessage As String)
End Sub
```

### [Cicode]

```
FUNCTION myPage_AN35_Error(OBJECT processAnalyst, INT errorCode, STRING errorMessage)
END
```

## See Also

[ErrorNotifyCode \[Enumeration\]](#)

### HorizontalAxisChanged [Event]

This event is raised when the date/time axis position or scale of a pen is changed.

#### Defined As

- [VBA] HorizontalAxisChanged(pen As Object)
- [Cicode] HorizontalAxisChanged(OBJECT processAnalyst, OBJECT pen)
- [C++] HorizontalAxisChanged(IPen\* pen)

#### Parameters

pen

[in] Refers to the pen that has changed. This will be invalid if pens are locked.

processAnalyst

[in] Indicates the Process Analyst object that raised the event (Cicode only).

#### Remarks

When the LockedPens property is True, this event is fired only once with the pen parameter marked as invalid.

#### Calling Syntax

Assumes you have a Process Analyst on a page with an event class defined as AN35\_E.

### [VBA]

```
Sub AN35_E_HorizontalAxisChanged (pen As Object)
End Sub
```

### [Cicode]

```
FUNCTION AN35_E_HorizontalAxisChanged (OBJECT processAnalyst, OBJECT pen)
END
```

## See Also

[IProcessAnalyst.LockedPens \[Property\]\[Get/Set\]](#)

[VerticalAxisChanged \[Event\]](#)

### MouseClick [Event]

This event is raised whenever a single mouse click occurs on the graphical chart area of the Process Analyst.

#### Defined As

- [VBA] MouseClick(pen As Object, button As Integer)

- [Cicode] MouseClick(OBJECT processAnalyst, OBJECT pen, INT button)
- [C++] MouseClick(IPen\* pen, int button)

#### Parameters

pen

[in] Indicates which pen the click occurred on. This object will be invalid if no pen was clicked.

button

[in] Indicates which button was clicked: 0 = Left, 1 = Right.

processAnalyst

[in] Indicates the Process Analyst object that raised the event (Cicode only).

#### Calling Syntax

Assumes you have a Process Analyst on a page with an event class defined as myPage\_AN35.

### [VBA]

```
Sub myPage_AN35_MouseClick(pen As Object, button As Integer)
End Sub
```

### [Cicode]

```
FUNCTION myPage_AN35_MouseClick(OBJECT processAnalyst, OBJECT pen, INT button)
END
```

### MouseDoubleClick [Event]

This event is raised whenever a mouse double-click occurs on the graphical chart area of the Process Analyst.

#### Defined As

- [VBA] MouseDoubleClick(pen As Object, button As Integer)
- [Cicode] MouseDoubleClick(OBJECT processAnalyst, OBJECT pen, INT button)
- [C++] MouseDoubleClick(IPen pen, int button)

#### Parameters

pen

[in] Indicates which pen the double-click occurred on. This object will be invalid if no pen was double-clicked.

button

[in] Indicates which button was double-clicked: 0 = Left, 1 = Right.

processAnalyst

[in] Indicates the Process Analyst object that raised the event (Cicode only).

#### Calling Syntax

Assumes you have a Process Analyst on a page with an event class defined as myPage\_AN35.

**[VBA]**

```
Sub myPage_AN35_MouseDoubleClick(pen As Object, button As Integer)
End Sub
```

**[Cicode]**

```
FUNCTION myPage_AN35_MouseDoubleClick(OBJECT processAnalyst, OBJECT pen, INT button)
END
```

**OVColumnAdded [Event]**

This event is raised whenever a column is added to the ObjectView.

**Defined As**

- [VBA] OVColumnAdded(name As String)
- [Cicode] OVColumnAdded(OBJECT processAnalyst, STRING name)
- [C++] OVColumnAdded(BSTR name)

**Parameters**

item

[in] The name of the column that has been added

processAnalyst

[in] Indicates the Process Analyst object which raised the event. (Cicode only)

**Calling Syntax**

Assumes you have a Process Analyst on a page with an event class defined as myPage\_AN35.

**[VBA]**

```
Sub myPage_AN35_OVColumnAdded(name As String)
End Sub
```

**[Cicode]**

```
FUNCTION myPage_AN35_OVColumnAdded(OBJECT processAnalyst, STRING name)
END
```

**OVColumnRemoved [Event]**

This event is raised whenever a column is removed to the ObjectView.

**Defined As**

- [VBA] OVColumnRemoved(name As String)
- [Cicode] OVColumnRemoved(OBJECT processAnalyst, STRING name)
- [C++] OVColumnRemoved(BSTR name)

**Parameters****item**

[in] The name of the column that has been removed.

**processAnalyst**

[in] Indicates the Process Analyst object which raised the event. (Cicode only)

**Calling Syntax**

Assumes you have a Process Analyst on a page with an event class defined as myPage\_AN35.

**[VBA]**

```
Sub myPage_AN35_OVColumnRemoved(name As String)
End Sub
```

**[Cicode]**

```
FUNCTION myPage_AN35_OVColumnRemoved(OBJECT processAnalyst, STRING name)
END
```

**OVItemAdded [Event]**

This event is raised whenever an item is added to the ObjectView.

**Defined As**

- [VBA] OVItemAdded(item As Object)
- [Cicode] OVItemAdded (OBJECT processAnalyst, OBJECT item)
- [C++] OVItemAdded (IObjectViewItem\* item)

**Parameters****item**

[in] A reference to the item that was added to the ObjectView.

**processAnalyst**

[in] Indicates the Process Analyst object which raised the event. (Cicode only)

**Calling Syntax**

Assumes you have a Process Analyst on a page with an event class defined as myPage\_AN35.

**[VBA]**

```
Sub myPage_AN35_OVItemAdded(item As Object)
End Sub
```

**[Cicode]**

```
FUNCTION myPage_AN35_OVItemAdded(OBJECT processAnalyst, OBJECT item)
END
```

## OVItemChecked [Event]

This event is raised whenever an item is checked in the ObjectView.

### Defined As

- [VBA] OVItemChecked(item As Object)
- [Cicode] OVItemChecked(OBJECT processAnalyst, OBJECT item)
- [C++] OVItemChecked(IObjectViewItem\* item)

### Parameters

item

[in] A reference to the item that was checked in the ObjectView.

processAnalyst

[in] Indicates the Process Analyst object which raised the event. (Cicode only)

### Calling Syntax

Assumes you have a Process Analyst on a page with an event class defined as myPage\_AN35.

## [VBA]

```
Sub myPage_AN35_OVItemChecked(item As Object)
End Sub
```

## [Cicode]

```
FUNCTION myPage_AN35_OVItemChecked(OBJECT processAnalyst, OBJECT item)
END
```

## OVItemRemoved [Event]

This event is raised whenever an item is added to the ObjectView.

### Defined As

- [VBA] OVItemRemoved(item As Object)
- [Cicode] OVItemRemoved(OBJECT processAnalyst, OBJECT item)
- [C++] OVItemRemoved(IObjectViewItem\* item)

### Parameters

item

[in] A reference to the item that was removed from the ObjectView.

processAnalyst

[in] Indicates the Process Analyst object which raised the event. (Cicode only)

### Calling Syntax

Assumes you have a Process Analyst on a page with an event class defined as myPage\_AN35.

**[VBA]**

```
Sub myPage_AN35_OVItemRemoved(item As Object)
End Sub
```

**[Cicode]**

```
FUNCTION myPage_AN35_OVItemRemoved(OBJECT processAnalyst, OBJECT item)
END
```

**OVItemSelected [Event]**

This event is raised whenever an item is selected in the ObjectView.

**Defined As**

- [VBA] OVItemSelected(item As Object)
- [Cicode] OVItemSelected(OBJECT processAnalyst, OBJECT item)
- [C++] OVItemSelected(IObjectViewItem\* item)

**Parameters**

item

[in] A reference to the item that was selected in the ObjectView.

processAnalyst

[in] Indicates the Process Analyst object which raised the event. (Cicode only)

**Calling Syntax**

Assumes you have a Process Analyst on a page with an event class defined as myPage\_AN35.

**[VBA]**

```
Sub myPage_AN35_OVItemSelected(item As Object)
End Sub
```

**[Cicode]**

```
FUNCTION myPage_AN35_OVItemSelected(OBJECT processAnalyst, OBJECT item)
END
```

**PenCreated [Event]**

This event is raised whenever a pen is either created via the automation model, or added through the Add Pen dialog at run time.

**Defined As**

- [VBA] PenCreated(pen As Object)
- [Cicode] PenCreated(OBJECT processAnalyst, OBJECT pen)

- [C++] PenCreated(IPen\* pen)

**Parameters**

pen

[in] Refers to the pen that was created.

processAnalyst

[in] Indicates the Process Analyst object which raised the event. (Cicode only).

**Calling Syntax**

Assumes you have a Process Analyst on a page with an event class defined as myPage\_AN35.

**[VBA]**

```
Sub myPage_AN35_PenCreated(pen As Object)
End Sub
```

**[Cicode]**

```
FUNCTION myPage_AN35_PenCreated(OBJECT processAnalyst, OBJECT pen)
END
```

**PenDeleted [Event]**

This event is raised whenever a pen is deleted either by automation or via the interface.

**Defined As**

- [VBA] PenDeleted(penName As String)
- [Cicode] PenDeleted(OBJECT processAnalyst, STRING penName)
- [C++] PenDeleted(BSTR penName)

**Parameters**

penName

[in] Contains the name of the pen that was deleted.

processAnalyst

[in] Indicates the Process Analyst object which raised the event. (Cicode only)

**Calling Syntax**

Assumes you have a Process Analyst on a page with an event class defined as myPage\_AN35

**[VBA]**

```
Sub myPage_AN35_PenDeleted(penName As String)
End Sub
```

## [Cicode]

```
FUNCTION myPage_AN35_PenDeleted(OBJECT processAnalyst, STRING penName)
END
```

## PenRenamed [Event]

This event is raised whenever a pen is renamed via automation or through the user interface.

### Defined As

- [VBA] PenRenamed(pen As Object)
- [Cicode] PenRenamed(OBJECT processAnalyst, OBJECT pen)
- [C++] PenRenamed(IPen\* pen)

### Parameters

pen

[in] Refers to the pen that was renamed.

processAnalyst

[in] Indicates the Process Analyst object which raised the event. (Cicode only).

### Calling Syntax

Assumes you have a Process Analyst on a page with an event class defined as myPage\_AN35.

## [VBA]

```
Sub myPage_AN35_PenRenamed(pen As Object)
End Sub
```

## [Cicode]

```
FUNCTION myPage_AN35_PenRenamed(OBJECT processAnalyst, OBJECT pen)
END
```

## PenSelectionChanged [Event]

This event is raised whenever the selection changes in the Process Analyst.

### Defined As

- [VBA] PenSelectionChanged (pen As Object)
- [Cicode] PenSelectionChanged (OBJECT processAnalyst, OBJECT pen)
- [C++] PenSelectionChanged (IPen\* pen)

### Parameters

pen

[in] Refers to the pen that now has primary selection. This maybe invalid if the last pen was deleted from the

view.

**processAnalyst**

[in] Indicates the Process Analyst object which raised the event. (Cicode only)

#### Remarks

Selection can change via user interaction (such as clicking on pens, deleting/adding pens) and automation.

#### Calling Syntax

Assumes you have a Process Analyst on a page with an event class defined as myPage\_AN35.

### [VBA]

```
Sub myPage_AN35_PenSelectionChanged(pen As Object)
End Sub
```

### [Cicode]

```
FUNCTION myPage_AN35_PenSelectionChanged(OBJECT processAnalyst, OBJECT pen)
END
```

### PropertyChanged [Event]

This event is raised whenever a property that has been subscribed to has changed.

#### Defined As

- [VBA] PropertyChanged(interfaceName As String, propertyName As String)
- [Cicode] PropertyChanged (OBJECT processAnalyst, STRING interfaceName, STRING propertyName)
- [C++] PropertyChanged (BSTR interfaceName, BSTR propertyName)

#### Parameters

**interfaceName**

[in] Indicates which interface the property which has changed belongs to.

**propertyName**

[in] Indicates which property has changed.

**processAnalyst**

[in] Indicates the Process Analyst object which raised the event. (Cicode only)

#### Remarks

For this event to be raised you need to subscribe to one or more properties.

#### Calling Syntax

Assumes you have a Process Analyst on a page with an event class defined as myPage\_AN35.

### [VBA]

```
Sub myPage_AN35_PropertyChanged(interfaceName As String, propertyName As String)
End Sub
```

## [Cicode]

```
FUNCTION myPage_AN35_PropertyChanged(OBJECT processAnalyst, STRING interfaceName,  
STRING propertyName)  
END
```

## See Also

[IProcessAnalyst.SubscribeForPropertyChange \[Method\]](#)  
[IProcessAnalyst.UnsubscribePropertyChange \[Method\]](#)

## UpdateCommand [Event]

This event is raised whenever the Process Analyst needs to refresh the state of its toolbars.

### Defined As

- [VBA] UpdateCommand(commandId As String)
- [Cicode] UpdateCommand(OBJECT processAnalyst, STRING commandId)
- [C++] UpdateCommand(BSTR commandId)

### Parameters

commandId

[in] Contains the unique identifier of the command that needs to be refreshed.

processAnalyst

[in] Indicates the Process Analyst object which raised the event. (Cicode only)

### Remarks

This event is only raised for custom commands. use this event as an opportunity to update the enable and/or the pressed state of the toolbar button associated with the command.

This event will be raised frequently so limit the amount of code executed in response to this event.

An Update will be triggered in at least the following scenarios:

- Selection changes
- Command execution

### Calling Syntax

Assumes you have a Process Analyst on a page with an event class defined as myPage\_AN35.

## [VBA]

```
Sub myPage_AN35_UpdateCommand(commandId As Integer)  
End Sub
```

## [Cicode]

```
FUNCTION myPage_AN35_UpdateCommand(OBJECT processAnalyst, INT commandId)
END
```

## VerticalAxisChanged [Event]

This event is raised whenever the vertical axis position or scale of a pen is changed.

### Defined As

- [VBA] VerticalAxisChanged(pen As Object)
- [Cicode] VerticalAxisChanged(OBJECT processAnalyst, OBJECT pen)
- [C++] VerticalAxisChanged(IPen\* pen)

### Parameters

pen

[in] Refers to the pen that has changed.

processAnalyst

[in] Indicates the Process Analyst object which raised the event. (Cicode only)

### Calling Syntax

Assumes you have a Process Analyst on a page with an event class defined as myPage\_AN35.

## [VBA]

```
Sub myPage_AN35_VerticalAxisChanged (pen As Object)
End Sub
```

## [Cicode]

```
FUNCTION myPage_AN35_VerticalAxisChanged (OBJECT processAnalyst, OBJECT pen)
END
```

## See Also

### ViewLoadedSaved[Event]

This event is raised whenever a view is either loaded or saved via the automation model, or through the Load/Save dialog at run time.

### Defined As

- [VBA] ViewLoadedSaved(sViewLocation As String, bIsLoaded As Boolean)
- [Cicode] ViewLoadedSaved(OBJECT processAnalyst, STRING sViewLocation, INT bIsLoadedSaved)
- [C++] ViewLoadSaved(BSTR sViewLocation, VARIANT\_BOOL bIsLoaded)

### Parameters

sViewLocation

[in] Refers to the new view location.

bIsLoadedSaved

[in] Refers to whether this event is triggered by Load (-1, or VARIANT\_TRUE) or Save (0, or VARIANT\_FALSE).

processAnalyst

[in] Indicates the Process Analyst object which raised the event. (Cicode only).

#### Calling Syntax

Assumes your Process Analyst's Event Class Name has been defined as myPage\_AN35.

### [VBA]

```
Sub myPage_AN35_ViewLoadedSaved(pen As Object, isLoading as Boolean)
End Sub
```

### [Cicode]

```
FUNCTION myPage_AN35_ViewLoadedSaved(OBJECT processAnalyst, STRING sViewLocation, INT
bIsLoaded)
END
```

### Interfaces

- [IAlarmPen Interface](#)
- [IAalogPen Interface](#)
- [ICommand Interface](#)
- [ICommandSystem Interface](#)
- [ICursors Interface](#)
- [IDigitalPen Interface](#)
- [IOBJECTView Interface](#)
- [IOBJECTViewColumn Interface](#)
- [IOBJECTViewColumns Interface](#)
- [IOBJECTViewItem Interface](#)
- [IOBJECTViewItems Interface](#)
- [IOBJECTViewPenItem Interface](#)
- [IPane Interface](#)
- [IPanes Interface](#)
- [IPen Interface](#)
- [IPens Interface](#)
- [IProcessAnalyst Interface](#)
- [IToolbar Interface](#)
- [IToolbars Interface](#)

- [IProcessAnalyst.SynchroniseToNow \[Method\]](#)
- [IToolbarButtons Interface](#)
- [ITrendCursor Interface](#)

## IAlarmPen Interface

### Defined As

- [VBA] Object
- [Cicode] OBJECT
- [C++] IAlarmPen

### Methods (6)

- [IAlarmPen.GetFillColor \[Method\]](#)
- [IAlarmPen.SetFillColor \[Method\]](#)
- [IAlarmPen.GetHatchColor \[Method\]](#)
- [IAlarmPen.SetHatchColor \[Method\]](#)
- [IAlarmPen.GetHatchStyle \[Method\]](#)
- [IAlarmPen.SetHatchStyle \[Method\]](#)

### Properties (3)

- [IAlarmPen.LineColor \[Property\]\[Get/Set\]](#)
- [IAlarmPen.LineWidth \[Property\]\[Get/Set\]](#)
- [IAlarmPen.AlarmType \[Property\]\[Get/Set\]](#)

## IAlarmPen.AlarmType [Property][Get/Set]

Gets or Sets the display type of this alarm pen.

### Defined As

- [VBA] Long AlarmType
- [Cicode] INT AlarmType
- [C++] AlarmType AlarmType

### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

### Remarks

This AlarmType also dictates the number of alarm states, and their descriptions.

### Calling Syntax

This example assumes there is a valid alarm pen object to be passed into the example methods.

### [VBA]

```
Sub Example(alarmPen As Object)
.....Dim alarmType As Long
.....Getting Property value
.....alarmType = alarmPen.AlarmType
.....`Setting Property value to Analog
.....alarmPen.AlarmType = 1
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hAlarmPen)
.....// Getting property value
.....INT eAlarmType = _ObjectGetProperty(hAlarmPen, "AlarmType");
.....// Setting property to Analog
....._ObjectSetProperty(hAlarmPen, "AlarmType", 1);
END
```

### See Also

[IAlarmPen.AlarmType \[Property\]\[Get/Set\]](#)

## IAlarmPen.GetFillColor [Method]

Gets the color used to fill the pen for the specified state.

### Defined As

- [VBA] GetFillColor(state as Long) as Long
- [Cicode] INT GetFillColor(INT state)
- [C++] HRESULT GetFillColor(int state, OLE\_COLOR\* color)

### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

### Parameters

state

[in] The state for which fill color to retrieve (0 to 8).

### Remarks

The color value can be calculated using the following formula: color = (65536 \* Blue) + (256 \* Green) + (Red). Where red, green and blue are 0-255.

### Calling Syntax

This example assumes there is a valid AlarmPen object to be passed into the example methods.

**[VBA]**

```
Sub Example(alarmPen As Object)
.....Dim fillColor As Long
.....fillColor = alarmPen.GetFillColor(0)
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hAlarmPen)
.....INT nFillColor = _ObjectCallMethod(hAlarmPen, "GetFillColor" , 0);
END
```

## IAlarmPen.GetHatchColor [Method]

Gets the color used to draw the outline and hatching for the specified state.

**Defined As**

- [VBA] GetHatchColor(state as Long) as Long
- [Cicode] INT GetHatchColor(INT state)
- [C++] HRESULT GetHatchColor(int state, OLE\_COLOR\* color)

**Execution Result**

If the function succeeds, the return value will be Success. If the state is out of range, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

**Parameters**

state

[in] The state for which hatch color to retrieve (0 to 8).

**Remarks**

The color value can be calculated using the following formula:

color = (65536 \* Blue) + (256 \* Green) + (Red)

where red, green, and blue are 0-255.

**Calling Syntax**

This example assumes there is a valid AlarmPen object to be passed into the example methods.

**[VBA]**

```
Sub Example(alarmPen As Object)
.....Dim hatchColor As Long
.....hatchColor = alarmPen.GetHatchColor(0)
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hAlarmPen)
....INT nHatchColor = _ObjectCallMethod(hAlarmPen, "GetHatchColor" , 0);
END
```

## IAlarmPen.GetHatchStyle [Method]

Gets the hatch style used when drawing the boxes for the specified state.

**Defined As**

- [VBA] GetHatchStyle(state as Long) as Long
- [Cicode] INT GetHatchStyle(INT state)
- [C++] HRESULT GetHatchStyle(int state, HatchStyle\* color)

**Parameters**

state

[in] The state for which you would like to retrieve a hatch style (0 to 8).

**Execution Result**

If the function succeeds the return value will be Success. If the state is out of range then the return value will be InvalidArgument. If the pen is deleted the return value will be GeneralFailure.

**Remarks**

The color value can be calculated using the following formula: color = (65536 \* Blue) + (256 \* Green) + (Red). Where red, green and blue are 0-255.

**Calling Syntax**

This example assumes there is a valid AlarmPen object to be passed into the example methods.

**[VBA]**

```
Sub Example(alarmPen As Object)
....Dim hatchStyle As Long
....hatchStyle = alarmPen.GetHatchStyle(0)
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hAlarmPen)
....INT nHatchStyle = _ObjectCallMethod(hAlarmPen, "GetHatchStyle", 0);
END
```

**See Also**

[IAlarmPen.GetHatchStyle \[Method\]](#)

## IAlarmPen.LineColor [Property][Get/Set]

Gets or Sets the color that will be used to draw the pen line.

### Defined As

- [VBA] Long LineColor
- [Cicode] INT LineColor
- [C++] OLE\_COLOR LineColor

### Execution Result

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument. If the pen is deleted the return value will be GeneralFailure.

### Remarks

The color value can be calculated using the following formula:

$$\text{color} = (65536 * \text{Blue}) + (256 * \text{Green}) + (\text{Red})$$

where red, green, and blue are 0-255.

### Calling Syntax

This example assumes there is a valid alarm pen object to be passed into the example methods.

### [VBA]

```
Sub Example(alarmPen As Object)
.....Dim lineColor As Long
.....`Getting Property value
.....lineColor = alarmPen.LineColor
.....`Setting Property to red
.....alarmPen.LineColor = 255
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hAlarmPen)
.....// Getting property value
.....INT nLineColor = _ObjectGetProperty(hAlarmPen, "LineColor");
.....// Setting property to red
....._ObjectSetProperty(hAlarmPen, "LineColor", 255);
END
```

## IAlarmPen.LineWidth [Property][Get/Set]

Gets or sets the width in pixels of the pen line when it is drawn.

### Defined As

- [VBA] Long LineWidth

- [Cicode] INT LineWidth
- [C++] int LineWidth

#### Execution Result

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument. If the pen is deleted the return value will be GeneralFailure.

#### Limits

- Minimum = 0
- Maximum = 8

#### Calling Syntax

This example assumes there is a valid alarm pen object to be passed into the example methods.

### [VBA]

```
Sub Example(alarmPen As Object)
.....Dim lineWidth As Long
.....`Getting Property value
.....lineWidth = alarmPen.LineWidth
.....`Setting Property value
.....alarmPen.LineWidth = 5
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hAlarmPen)
.....// Getting property value
.....INT nLineWidth = _ObjectGetProperty(hAlarmPen, "LineWidth");
.....// Setting property value
....._ObjectSetProperty(hAlarmPen, "LineWidth", 5);
END
```

## IAlarmPen.SetFillColor [Method]

Sets the color used to fill the pen for the specified state.

#### Defined As

- [VBA] SetFillColor(state as Long, color as Long)
- [Cicode] INT SetFillColor(INT state, INT color)
- [C++] HRESULT SetFillColor(int state, OLE\_COLOR color)

#### Parameters

##### state

[in] The state for which you would like to assign a fill color (0 to 8).

##### color

[in] The fill color that you would like used to for this specific state.

#### Execution Result

If the function succeeds, the return value will be Success. If the state is out of range, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

#### Remarks

The color value can be calculated using the following formula:

color = (65536 \* Blue) + (256 \* Green) + (Red)

where red, green, and blue are 0-255.

#### Calling Syntax

This example assumes there is a valid AlarmPen object to be passed into the example methods.

### [VBA]

```
Sub Example(alarmPen As Object)
.....Dim fillColor As Long
.....`Setting FillColor to Red
.....alarmPen.SetFillColor(0, 255)
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hAlarmPen)
.....// Setting FillColor to Red
....._ObjectCallMethod(hAlarmPen, "SetFillColor" ,0, 255);
END
```

## IAlarmPen.SetHatchColor [Method]

Sets the color used to draw the outline and hatching for the specified state.

#### Defined As

- [VBA] SetHatchColor(state as Long, color as Long)
- [Cicode] INT SetHatchColor (INT state, INT color)
- [C++] HRESULT SetHatchColor (int state, OLE\_COLOR color)

#### Parameters

state

[in] The state for which you would like to assign a hatch color (0 to 8).

color

[in] The color that you would like to be used for a specified states hatch.

#### Execution Result

If the function succeeds the return value will be Success. If the state is out of range then the return value will be InvalidArgument. If the pen is deleted the return value will be GeneralFailure.

**Remarks**

The color value can be calculated using the following formula:

$$\text{color} = (65536 * \text{Blue}) + (256 * \text{Green}) + (\text{Red})$$

where red, green, and blue are 0-255.

**Calling Syntax**

This example assumes there is a valid AlarmPen object to be passed into the example methods.

**[VBA]**

```
Sub Example(alarmPen As Object)
.....Dim hatchColor As Long
.....`Setting HatchColor to Red
.....alarmPen.SetHatchColor(0, 255)
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hAlarmPen)
.....// Setting HatchColor to Red
....._ObjectCallMethod(hAlarmPen, "SetHatchColor",0, 255);
END
```

## IAlarmPen.SetHatchStyle [Method]

Sets the hatch style used for drawing the specified state.

**Defined As**

- [VBA] SetHatchStyle(state as Long, HatchStyle as Long)
- [Cicode] INT SetHatchStyle (INT state, INT hatchStyle)
- [C++] HRESULT SetHatchStyle (int state, HatchStyle hatchStyle)

**Parameters****state**

[in] The state for which you would like to assign a hatch style.

**hatchStyle**

[in] The hatch style that will be used for the specified state.

**Execution Result**

If the function succeeds the return value will be Success. If the state is out of range then the return value will be InvalidArgument. If the pen is deleted the return value will be GeneralFailure.

**Remarks**

The color value can be calculated using the following formula:  $\text{color} = (65536 * \text{Blue}) + (256 * \text{Green}) + (\text{Red})$ . Where red, green and blue are 0-255.

**Calling Syntax**

This example assumes there is a valid AlarmPen object to be passed into the example methods.

### [VBA]

```
Sub Example(alarmPen As Object)
.....`Setting HatchStyle
.....alarmPen.SetHatchStyle(0, 1)
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hAlarmPen)
.....// Setting HatchStyle
....._ObjectCallMethod(hAlarmPen, "SetHatchStyle" ,0, 1);
END
```

### See Also

[IAlarmPen.GetHatchStyle \[Method\]](#)

[IAnalogPen Interface](#)

#### Defined As

- [VBA] Object
- [Cicode] OBJECT
- [C++] IAnalogPen

#### Methods (0)

#### Properties (3)

- [IAnalogPen.LineColor \[Property\]\[Get/Set\]](#)
- [IAnalogPen.LineInterpolation \[Property\]\[Get/Set\]](#)
- [IAnalogPen.LineWidth \[Property\]\[Get/Set\]](#)

## IAnalogPen.LineColor [Property][Get/Set]

Gets or Sets the color that will be used to draw the pen line.

#### Defined As

- [VBA] Long LineColor
- [Cicode] INT LineColor
- [C++] OLE\_COLOR LineColor

#### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

#### Remarks

The color value can be calculated using the following formula:

color = (65536 \* blue) + (256 \* green) + (red)

where red, green, and blue are 0-255.

#### Calling Syntax

This example assumes there is a valid AnalogPen object to be passed into the example methods.

### [VBA]

```
Sub Example(analogPen As Object)
.....Dim lineColor As Long
.....`Getting Property value
.....lineColor = analogPen.LineColor
.....`Setting Property to red
.....analogPen.LineColor = 255
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hAnalogPen)
.....// Getting property value
.....INT nLineColor = _ObjectGetProperty(hAnalogPen, "LineColor");
.....// Setting property to red
....._ObjectSetProperty(hAnalogPen, "LineColor", 255);
END
```

## IAnalogPen.LineInterpolation [Property][Get/Set]

Gets or sets the drawing style used for drawing the connecting lines between points for this analog pen.

#### Defined As

- [VBA] Long LineInterpolation
- [Cicode] INT LineInterpolation
- [C++] LineType LineInterpolation

#### Execution Result

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument. If the pen is deleted the return value will be GeneralFailure.

#### Remarks

The LineInterpolation mode dictates how the two points of a line are joined when drawn. If Stepped, there will be two lines joining each point, one horizontal and one vertical. If Straight, only one line is used to directly connect the two points.

#### Calling Syntax

This example assumes there is a valid Analog Pen object to be passed into the example methods.

### [VBA]

```
Sub Example(analogPen As Object)
.....Dim lineInterpolation As Long
.....`Getting Property value
.....lineInterpolation = analogPen.LineInterpolation
.....`Setting Property value
.....analogPen.LineInterpolation = 1
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hAnalogPen)
.....// Getting property value
.....INT nInterpolation = _ObjectGetProperty(hAnalogPen, "LineInterpolation");
.....// Setting property value
....._ObjectSetProperty(hAnalogPen, "LineInterpolation", 1);
END
```

### See Also

[LineType \[Enumeration\]](#)

## IAnalogPen.LineWidth [Property][Get/Set]

Gets or sets the width in pixels of the pen line when it is drawn.

#### Defined As

- [VBA] Long LineWidth
- [Cicode] INT LineWidth
- [C++] int LineWidth

#### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

#### Limits

- Minimum = 0
- Maximum = 8

#### Calling Syntax

This example assumes there is a valid Analog Pen object to be passed into the example methods.

**[VBA]**

```
Sub Example(analogPen As Object)
.....Dim lineWidth As Long
.....`Getting Property value
.....lineWidth = analogPen.LineWidth
.....`Setting Property value
.....analogPen.LineWidth = 5
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hAnalogPen)
.....// Getting property value
.....INT nLineWidth = _ObjectGetProperty(hAnalogPen, "LineWidth");
.....// Setting property value
....._ObjectSetProperty(hAnalogPen, "LineWidth", 5);
END
```

**ICommand Interface**

Inbuilt Commands are listed below:

ID	Description
Citect_Command_LoadConfig	Displays the "Load File" dialog. The user can then specify a configuration file to load into the Process Analyst Control.
Citect_Command_SaveConfig	Displays the "Save File" dialog allowing the user to specify a location to save the current Process Analyst Configuration to.
Citect_Command_ToggleZoom	Toggles the Process Analyst into box-zoom mode. The mouse cursor will be changed into the cross hair. The Zoom is cancelled with a right mouse click, or toggling this command button again
Citect_Command_ZoomIn50	Executes a horizontal and vertical zoom in of 50% of the current spans(s) of the pen(s).
Citect_Command_ZoomOut50	Executes a horizontal and vertical zoom out of 50% of the current spans(s) of the pen(s).
Citect_Command_ZoomUndo	Undoes the last zoom operation
Citect_Command_SpanReset	Restores the pen(s) spans to their original default settings.
Citect_Command_SpanEdit	Displays a dialog allowing the operator to explicitly enter a span to apply to the pen(s).

<b>ID</b>	<b>Description</b>
Citect_Command_ScaleEdit	Edit Vertical Scale
Citect_Command_ScrollLeft100	Moves the pen(s) back in time exactly one span. 'Span' refers to the distance in time between the 'Start' and 'End' of the time-axis.
Citect_Command_ScrollLeft50	Moves the pen(s) back half a span.
Citect_Command_ScrollRight100	Moves the pen(s) forward in time exactly one span. 'Span' refers to the distance in time between the 'Start' and 'End' of the time-axis
Citect_Command_ScrollRight50	Moves the pen(s) forward half a span.
Citect_Command_SynchroniseToNow	Synchronises pens such that the end date time reflects "now" which is positioned on the right hand edge of the screen. "Now" is calculated using the current system time
Citect_Command_ToggleAutoScroll	Toggles the automatic scrolling off and on for pens
Citect_Command_CopyToFile	Exports visible pens to an Excel compatible file.
Citect_Command_CopyToClipboard	Exports visible pens to the clipboard.
Citect_Command_ToggleCursorVisibility	Shows or Hides cursors. If no cursor exists a cursor will be created. Note: Showing a cursor will automatically make the labels visible.
Citect_Command_ToggleCursorLabels	Shows or Hides cursor labels.
Citect_Command_ToggleCursorLabelsLock	Locks or Unlocks the movement of the labels with the cursor.
Citect_Command_TogglePensLock	Allows or disallows the independent movement of pens against each other.
Citect_Command_AddPane	Adds a new pane to the view
Citect_Command_RemovePane	Removes the pane of the primary selected pen. A dialog will confirm the delete.
Citect_Command_TogglePenAutoScale	Toggles the auto-scaling behaviour for the selected pen. This is on a per pen basis. Applying this will remove the Vertical axis scrolling.
Citect_Command_ToggleVerticalScrollLock	Toggles interactive scrolling of the vertical axis. Applying this will remove the autoscale behaviour.
Citect_Command_AddPen	Displays the add pen dialog

ID	Description
Citect_Command_PenRemove	Removes the selected pen from the display.
Citect_Command_ToggleObjectView	When pressed, the Object View will be displayed at the bottom of the Process Analyst Control. When unpressed, the Object View will be hidden and the Process Analyst Control will expand to take up the vacant space
Citect_Command_Print	Displays the print dialog, allowing the user to print the current state of the Process Analyst Control
Citect_Command_RefreshData	Refreshes the selected pen, or pens (if locked)
Citect_Command_Properties	Displays the properties dialog

**Defined As**

- [VBA] Object
- [Cicode] OBJECT
- [C++] ICommand

**Methods (0)****Properties (6)**

[ICommand.CommandId \[Property\]\[Get\]](#)  
[ICommand.ButtonType \[Property\]\[Get\]](#)  
[ICommand.Enabled \[Property\]\[Get/Set\]](#)  
[ICommand.Pressed \[Property\]\[Get/Set\]](#)  
[ICommand.Tooltip \[Property\]\[Get\]](#)

# ICommand.Tooltip [Property][Get]

Gets this commands Tooltip text.

**Defined As**

- [VBA] String Tooltip
- [Cicode] STRING Tooltip
- [C++] VARIANT\_BOOL Tooltip

**Execution Result**

If the property get succeeds, the return value will be Success. If the return variable is invalid, the return value will be InvalidArgument. If the command has been deleted, the return value will be GeneralFailure.

**Remarks**

This returns the text that is displayed in a tooltip window when the mouse pointer hovers over the command's

button.

#### Calling Syntax

This example assumes there is a valid Command object as retrieved from a Process Analyst's CommandSystem.  
(for example, VBA: ProcessAnalyst.CommandSystem.Item(1))

#### [VBA]

```
Sub Example(Command As Object)
Dim tooltip As String
`Getting Property value
tooltip = Command.Tooltip
End Sub
```

#### [Cicode]

```
FUNCTION Example(OBJECT hCommand)
// Getting property value
STRING sTooltip = _ObjectGetProperty(hCommand, "Tooltip");
END
```

## ICommand.ButtonType [Property][Get]

Gets this commands button type.

#### Defined As

- [VBA] Long ButtonType
- [Cicode] INT ButtonType
- [C++] ToolbarButtonType ButtonType

#### Execution Results

If the property get succeeds, the return value will be Success. If the return value is invalid, the return value will be InvalidArgument. If the command has been deleted, the return value will be GeneralFailure.

#### Remarks

The return value meaning is as follows:

- ToolbarButtonType\_Push = 0
- ToolbarButtonType\_Toggle = 1
- ToolbarButtonType\_Separator = 2

#### Calling Syntax

This example assumes there is a valid Command object as retrieved from a Process Analyst's CommandSystem.  
(for example, VBA: ProcessAnalyst.CommandSystem.Item(1))

**[VBA]**

```
Sub Example(Command As Object)
Dim buttonType As Long
`Getting Property value
buttonType = Command.ButtonType
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hCommand)
// Getting property value
INT nButtonType = _ObjectGetProperty(hCommand, "ButtonType");
END
```

## ICommand.CommandId [Property][Get]

Gets the CommandId of this command.

**Defined As**

- [VBA] String CommandId
- [Cicode] STRING hCommandId
- [C++] BSTR CommandId

**Calling Syntax**

This example assumes there is a valid Command object as retrieved from a Process Analyst's CommandSystem (for example, VBA: ProcessAnalyst.CommandSystem.Item(1)).

**[VBA]**

```
Sub Example(Command As Object)
Dim commandId As String
`Getting Property value
commandId = Command.CommandId
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hCommand)
// Getting property value
STRING sCommandId = _ObjectGetProperty(hCommand, "CommandID");
END
```

**See Also**

[ICommand Interface](#)

# ICommand.Enabled [Property][Get/Set]

Gets this commands enabled state.

## Defined As

- [VBA] Boolean Enabled
- [Cicode] INT Enabled
- [C++] VARIANT\_BOOL Enabled

## Execution Result

If the property get succeeds, the return value will be Success. If the return value is invalid, the return value will be InvalidArgument.

## Limits

- True (-1): Enabled
- False (0): Disabled

## Remarks

The setting of this property is only valid for custom commands.

## Calling Syntax

This example assumes there is a valid Command object as retrieved from a Process Analyst's CommandSystem. (for example, VBA: ProcessAnalyst.CommandSystem.Item(1))

## [VBA]

```
Sub Example(Command As Object)
Dim enabled As Boolean
`Getting Property value
enabled = Command.Enabled
`Setting Property value
Command.Enabled = True
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hCommand)
// Getting property value
INT nEnabled = _ObjectGetProperty(hCommand, "Enabled");
// Setting property value
_ObjectSetProperty(hCommand, "Enabled", -1);
END
```

# ICommand.Pressed [Property][Get/Set]

Gets and Sets this command's Pressed state.

**Defined As**

- [VBA] Boolean Pressed
- [Cicode] INT Pressed
- [C++] VARIANT\_BOOL Pressed

**Execution Result**

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

**Limits**

- True (-1): Pressed
- False (0): Impressed

**Remarks**

This is only useful for toggle buttons, indicating whether or not the button is in a pressed down state. The setting of this property is only valid for custom commands.

**Calling Syntax**

This example assumes there is a valid Command object as retrieved from a Process Analyst's CommandSystem.  
(for example, VBA: ProcessAnalyst.CommandSystem.Item(1))

**[VBA]**

```
Sub Example(Command As Object)
Dim pressed As Boolean
`Getting Property value
pressed = Command.Pressed
`Setting Property value
Command.Pressed = True
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hCommand)
// Getting property value
INT Pressed = _ObjectGetProperty(hCommand, "Pressed");
// Setting property value
_ObjectSetProperty(hCommand, "Pressed", -1);
END
```

## ICommand.Privilege [Property][Get]

Gets the privilege necessary to gain access to this command.

**Defined As**

- [VBA] Integer Privilege

- [Cicode] INT Privilege
- [C++] int Privilege

### Execution Result

If the property get succeeds, the return value will be Success. If the return variable is invalid, the return value will be InvalidArgument. If the command has been deleted, the return value will be GeneralFailure.

### Remarks

This is the necessary privilege level of the currently logged in Plant SCADA user to enable the state of this command, and hence allow access through the user interface. If the currently logged in Plant SCADA user doesn't have this privilege, any buttons tied to this command will be disabled.

### Calling Syntax

This example assumes there is a valid Command object as retrieved from a Process Analyst's CommandSystem.  
(for example, VBA: ProcessAnalyst.CommandSystem.Item(1))

## [VBA]

```
Sub Example(Command As Object)
Dim privilege As Integer
`Getting Property value
privilege = Command.Privilege
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hCommand)
// Getting property value
INT Privilege = _ObjectGetProperty(hCommand, "Privilege");
END
```

## ICommandSystem Interface

### Defined As

- [VBA] Object
- [Cicode] OBJECT
- [C++] ICommandSystem

### Methods (3)

- [ICommandSystem.Create \[Method\]](#)
- [ICommandSystem.Execute \[Method\]](#)
- [ICommandSystem.Remove \[Method\]](#)

### Properties (4)

- [ICommandSystem.Count \[Property\]\[Get\]](#)

- ICOMMANDSYSTEM.ITEM [PROPERTY][GET]
- ICOMMANDSYSTEM.\_NEWENUM [PROPERTY][GET]
- ICOMMANDSYSTEM.ITEMBYID [PROPERTY][GET]

## ICommandSystem.\_NewEnum [Property][Get]

This allows "For... Each... Next" integration in VB.

### Calling Syntax

This example assumes there is a valid CommandSystem object as retrieved from a Process Analyst. (for example, VBA: ProcessAnalyst.CommandSystem). This property is not applicable to Cicode.

### [VBA]

```
Sub Example(CommandSystem As Object)
Dim command As Object
Dim count Object
`Using Property
For Each command In CommandSystem
count = count + 1
Next command
End Sub
```

## ICommandSystem.Count [Property][Get]

Gets the number of commands in the command system.

### Defined As

- [VBA] Long Count
- [Cicode] INT Count
- [C++] int Count

### Execution Result

If the property get succeeds, the return value will be Success.

### Calling Syntax

This example assumes there is a valid CommandSystem object as retrieved from a Process Analyst. (for example, VBA: ProcessAnalyst.CommandSystem).

### [VBA]

```
Function Example(CommandSystem As Object)
Dim count As Long
`Getting Property value
count = CommandSystem.Count
End Function
```

**[Cicode]**

```
FUNCTION Example(OBJECT hCommandSystem)
// Getting property value
INT nCount = _ObjectGetProperty(hCommandSystem, "Count");
END
```

# ICommandSystem.Create [Method]

Creates a new Command object that is added to the CommandSystem.

**Defined As**

- [VBA] object Create (commandID As String, buttonType As Integer, tooltip As String, iconPath As String, privilege As Integer)
- [Cicode] OBJECT Create (STRING commandID, INT buttonType, STRING tooltip, STRING iconPath, INT Privilege)
- [C++] HRESULT Create (BSTR commandID, ToolbarButtonType ButtonType, BSTR tooltip, BSTR iconPath, int privilege, ICommand\*\* Val)

**Parameters****commandID**

[in] A unique identifier for this command (1-64 characters).

**buttonType**

[in] A value representing a button type.

- ToolbarButtonType\_Push = 0
- ToolbarButtonType\_Toggle = 1
- ToolbarButtonType\_Separator = 2

**tooltip**

[in] The text to be displayed as a tooltip for this command (1-64 characters).

**iconPath**

[in] The path to an icon file that will be used as this command's picture.

**privilege**

[in] A privilege value necessary by the Plant SCADA user to gain access to this command (0-8).

**Execution Result**

If the method succeeds, the return value is Success. If an argument is invalid or out of range, the return value is InvalidArgument. If the command was not created, the return value is GeneralFailure.

**Remarks**

The commandID cannot begin with the prefix "Citect\_".

**Calling Syntax**

This example assumes there is a valid CommandSystem object as retrieved from a Process Analyst. (for example,

VBA: ProcessAnalyst.CommandSystem).

## [VBA]

```
Sub Example(CommandSystem As Object)
Dim command As Object
Set command = CommandSystem.Create("Custom_CommandID1", 0, "Some tooltip text",
"c:\someicon.ico", 5)
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hCommandSystem)
OBJECT hCommand = _ObjectCallMethod(hCommandSystem, "Create", "Custom_CommandID1", 0, "Some
tooltip text", "c:\someicon.ico", 5);
END
```

# ICommandSystem.Execute [Method]

Executes the specified command's action.

### Defined As

- [VBA] Execute (commandId As String)
- [Cicode] Execute (STRING commandId)
- [C++] HRESULT Execute(BSTR commandId)

### Parameters

commandId

[in] The unique ID of the command whose action is to be executed.

### Execution Result

If this method succeeds, the return value will be Success. If the command is invalid, the return value will be InvalidArgument.

### Remarks

If the current Operator does not have the correct privilege, the command will not execute.

### Calling Syntax

This example assumes there is a valid CommandSystem object as retrieved from a Process Analyst. (for example, VBA: ProcessAnalyst.CommandSystem).

## [VBA]

```
Sub Example(CommandSystem As Object)
CommandSystem.Execute("Citect_Command_AddPen")
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hCommandSystem)
    _ObjectCallMethod(hCommandSystem, "Execute", "Citect_Command_AddPen");
END
```

**See Also**

[ICommand Interface](#)

## ICommandSystem.Item [Property][Get]

Gets the Command at a supplied index location in this collection.

**Defined As**

- [VBA] Item(index As Long) as Object
- [Cicode] OBJECT Item(INT index)
- [C++] Item(int index, ICommand\* Item)

**Parameters**

index

[in] Indicates the index location of the command to return from this collection. (One based)

**Execution Result**

If the property get succeeds, the return value will be Success. If the index is out of range, the return value will be InvalidArgument.

**Calling Syntax**

This example assumes there is a valid CommandSystem object as retrieved from a Process Analyst. (for example, VBA: ProcessAnalyst.CommandSystem).

**[VBA]**

```
Sub Example(CommandSystem As Object)
Dim command As Object
`Getting Property value
Set command = CommandSystem.Item(1)
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hCommandSystem)
// Getting property value
OBJECT hCommand = _ObjectCallMethod(hCommandSystem, "get_Item", 1);
END
```

# ICommandSystem.ItemById [Property][Get]

Gets the Command at a supplied index location in this collection.

## Defined As

- [VBA] Object Command
- [Cicode] OBJECT hCommand
- [C++] ICommand\* Command

## Parameters

commandId

[in] Indicates command ID of the command to return from this collection.

## Calling Syntax

This example assumes there is a valid CommandSystem object as retrieved from a Process Analyst. (for example, VBA: ProcessAnalyst.CommandSystem).

## [VBA]

```
Sub Example(CommandSystem As Object)
Dim command As Object
`Getting Property value
Set command = CommandSystem.ItemById(Citect_Command_AddPen)
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hCommandSystem)
// Getting property value
OBJECT hCommand = _ObjectCallMethod
(hCommandSystem, "get_ItemById", Citect_Command_AddPen);
END
```

## See Also

[ICommand Interface](#)

# ICommandSystem.Remove [Method]

Removes the specified command.

## Defined As

- [VBA] Remove (commandId As String)
- [Cicode] Remove (STRING CommandId)
- [C++] HRESULT Remove(BSTR CommandId)

**Parameters****commandId**

[in] The ID of the command to be removed.

**Calling Syntax**

This example assumes there is a valid CommandSystem object as retrieved from a Process Analyst. (for example, VBA: ProcessAnalyst.CommandSystem).

**[VBA]**

```
Sub Example(CommandSystem As Object)
CommandSystem.Remove("MyCommand1")
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hCommandSystem)
    _ObjectCallMethod(hCommandSystem, "Remove", "MyCommand1");
END
```

**ICursors Interface****Defined As**

- [VBA] Object
- [Cicode] OBJECT
- [C++] ICursors

**Methods (2)**

- [ICursors.Create \[Method\]](#)
- [ICursors.RemoveAll \[Method\]](#)

**Properties (4)**

- [ICursors.Item \[Property\]\[Get\]](#)
- [ICursors.\\_NewEnum \[Property\]\[Get\]](#)
- [ICursors.Count \[Property\]\[Get\]](#)
- [ICursors.ItemByName \[Property\]\[Get\]](#)

## ICursors.\_NewEnum [Property][Get]

Retrieves an enumerator for the cursors collection.

**Defined As**

- [VBA] Object \_NewEnum()
- [C++] HRESULT get\_\_NewEnum(LPUNKNOWN \*pVal)

#### Execution Result

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the collection is deleted, the return value will be GeneralFailure.

#### Remarks

Provided for the implementation of For Each...Next loops in VBA (See Calling Syntax, below). This property cannot be used in Cicode.

#### Calling Syntax

This example assumes you have a valid reference to the cursors collection and that there are cursors in the collection.

### [VBA]

```
Sub Example(cursors As Object)
Dim cursor As Object
Dim count As Integer = 0
For Each cursor In cursors
Set count = count + 1
Next
End Sub
```

## ICursors.Count [Property][Get]

Returns the number of cursors in the collection.

#### Defined As

- [VBA] Integer Count()
- [Cicode] INT Count()
- [C++] HRESULT get\_Count (long \*pCount)

#### Execution Result

If the property get succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument. If the collection is deleted the return value will be GeneralFailure.

#### Remarks

This property may be used in conjunction with the Item property to iterate through the collection in Cicode.

#### Calling Syntax

This example assumes you have a valid reference to the cursors collection.

### [VBA]

```
Sub Example(cursors As Object)
Dim cursorCount As Integer
```

```
cursorCount = cursors.Count
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hCursors)
INT cursorCount;
cursorCount = _ObjectGetProperty(hCursors, "Count");
END
```

# ICursors.Create [Method]

Creates a new TrendCursor at the given location.

### Defined As

- [VBA] Object Create(name As String, position As Integer)
- [Cicode] OBJECT Create(STRING name, INT position)
- [C++] HRESULT Create(BSTR name, int position, ITrendCursor\*\* ppTrendCursor)

### Parameters

name

[in] The desired unique name of the new cursor. This needs to be between 1 and 250 characters.

position

[in] The initial position of the new cursor. This value is given as the number of pixels from the left of the Process Analyst graph view.

### Execution Result

If the function succeeds, the return value will be Success. If the name is out of range, the return value will be InvalidArgument. If the name is not unique, the return value will be InvalidArgument.

If an unexpected error occurs, the return value will be GeneralFailure.

### Remarks

The cursor name needs to be unique. Attempting to create a cursor with a name that is already in use will result in error and the new cursor will not be created.

### Calling Syntax

## [VBA]

```
Sub Example(cursors As Object)
Dim newCursor As Object
newCursor = cursors.Create("Cursor1", 100)
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hCursors)
```

```
OBJECT hNewCursor = _ObjectCallMethod(hCursors, "Create", "Cursor1", 100);
END
```

## ICursors.Item [Property][Get]

Retrieves the Cursor from the collection at the specified index.

### Defined As

- [VBA] Object Item(index As Integer)
- [Cicode] OBJECT get\_Item(INT index)
- [C++] HRESULT get\_Item (long index, ITrendCursor \*\*cursor)

### Parameters

index

[in] The index of the necessary cursor.

### Execution Result

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

If the index is out of range, the return value will be InvalidArgument. If the collection is deleted, the return value will be GeneralFailure.

### Remarks

The index for the collection is 1 based. The valid range for this parameter is between 1 and the total number of cursors.

### Calling Syntax

This example assumes you have a valid reference to the cursors collection and that there are two items in the collection.

### [VBA]

```
Sub Example(hCursors As Object)
Dim hSecondCursor As Object
Set hSecondCursor = hCursors.Item(2)
End Sub
```

### [Cicode]

```
Sub Example(OBJECT hCursors)
OBJECT hSecondCursor = _ObjectCallMethod(hCursors, "get_Item", 2);
END
```

## ICursors.ItemByName [Property][Get]

Retrieves the Cursor at the specified index.

**Defined As**

- [VBA] Object ItemByName(name As String)
- [Cicode] OBJECT get\_ItemByName(STRING name)
- [C++] HRESULT get\_ItemByName (BSTR name, ITrendCursor \*\*cursor)

**Parameters**

name

[in] The name of the necessary cursor.

**Execution Result**

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the cursor is not found, the return value will be InvalidArgument.

If the collection is deleted, the return value will be GeneralFailure.

**Calling Syntax**

This example assumes you have a valid reference to the cursors collection, and that there is a cursor in the collection named "MyCursor".

**[VBA]**

```
Sub Example(cursors As Object)
Dim cursor As Object
Set cursor = cursors.ItemByName("MyCursor")
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hCursors)
OBJECT hCursor = _ObjectCallMethod(hCursors, "get_ItemByName", "MyCursor");
END
```

## ICursors.RemoveAll [Method]

Removes every cursor from the collection.

**Defined As**

- [VBA] RemoveAll()
- [Cicode] RemoveAll()
- [C++] HRESULT RemoveAll()

**Execution Result**

If the function succeeds the return value will be Success. If an unexpected error occurs, the return value will be GeneralFailure.

**Calling Syntax**

This example assumes there is a valid Cursors object to be passed into the example methods.

**[VBA]**

```
Sub Example(cursors As Object)
cursors.RemoveAll
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hCursors)
    _ObjectCallMethod(hCursors, "RemoveAll");
End Sub
```

**IDigitalPen Interface****Defined As**

- [VBA] Object
- [Cicode] OBJECT
- [C++] IDigitalPen

**Methods (0)****Properties (4)**

- [IDigitalPen.FillColor \[Property\]\[Get/Set\]](#)
- [IDigitalPen.LineColor \[Property\]\[Get/Set\]](#)
- [IDigitalPen.LineWidth \[Property\]\[Get/Set\]](#)
- [IDigitalPen.Fill \[Property\]\[Get/Set\]](#)

## IDigitalPen.Fill [Property][Get/Set]

Gets or sets whether the pen fill is displayed.

**Defined As**

- [VBA] Boolean Fill
- [Cicode] INT Fill
- [C++] VARIANT\_BOOL Fill

**Execution Result**

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument. If the pen is deleted the return value will be GeneralFailure.

**Remarks**

If the pen is filled, the area under the pen line will be filled with the color specified by the FillColor property.

**Limits**

- True (-1): = Fill is displayed
- False (0): = Fill is hidden

#### Calling Syntax

This example assumes there is a valid digital pen object to be passed into the example methods.

#### [VBA]

```
Sub Example(digitalPen As Object)
    Dim fill As Boolean
    'Getting Property value
    fill = digitalPen.Fill
    'Setting Property value
    digitalPen.Fill = True
End Sub
```

#### [Cicode]

```
FUNCTION Example(OBJECT hDigitalPen)
    // Getting property value
    INT nFill = _ObjectGetProperty(hDigitalPen, "Fill");
    // Setting property value
    _ObjectSetProperty(hDigitalPen, "Fill", -1);
END
```

#### See Also

[IDigitalPen.FillColor \[Property\]\[Get/Set\]](#)

## IDigitalPen.FillColor [Property][Get/Set]

Gets or Sets the color that will be used to fill the area under the line when the value is 1.

#### Defined As

- [VBA] Long FillColor
- [Cicode] INT FillColor
- [C++] OLE\_COLOR FillColor

#### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

#### Remarks

The color value can be calculated using the following formula:

$$\text{color} = (65536 * \text{blue}) + (256 * \text{green}) + (\text{red})$$

where red, green, and blue are 0-255. The area under the line is filled with this color if the value of the Fill

property is True (-1).

#### Calling Syntax

This example assumes there is a valid DigitalPen object to be passed into the example methods.

#### [VBA]

```
Sub Example(digitalPen As Object)
    Dim fillColor As Long
    `Getting Property value
    fillColor = digitalPen.FillColor
    `Setting Property to red
    digitalPen.FillColor = 255
End Sub
```

#### [Cicode]

```
FUNCTION Example(OBJECT hDigitalPen)
    // Getting property value
    INT nFillColor = _ObjectGetProperty(hDigitalPen, "FillColor");
    // Setting property to red
    _ObjectSetProperty(hDigitalPen, "FillColor", 255);
END
```

#### See Also

[IDigitalPen.Fill \[Property\]\[Get/Set\]](#)

## IDigitalPen.LineColor [Property][Get/Set]

Gets or Sets the color that will be used to draw the pen line.

#### Defined As

- [VBA] Long LineColor
- [Cicode] INT LineColor
- [C++] OLE\_COLOR LineColor

#### Execution Result

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument. If the pen is deleted the return value will be GeneralFailure.

#### Remarks

The color value can be calculated using the following formula:  $color = (65536 * Blue) + (256 * Green) + (Red)$ . Where red, green and blue are 0-255.

#### Calling Syntax

This example assumes there is a valid DigitalPen object to be passed into the example methods.

**[VBA]**

```
Sub Example(digitalPen As Object)
Dim lineColor As Long
`Getting Property value
lineColor = DigitalPen.LineColor
`Setting Property to red
digitalPen.LineColor = 255
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hDigitalPen)
// Getting property value
INT nLineColor = _ObjectGetProperty(hDigitalPen, "LineColor");
// Setting property to red
_ObjectSetProperty(hDigitalPen, "LineColor", 255);
END
```

## IDigitalPen.LineWidth [Property][Get/Set]

Gets or sets the width in pixels of the pen line when it is drawn.

**Defined As**

- [VBA] Long LineWidth
- [Cicode] INT LineWidth
- [C++] int LineWidth

**Execution Result**

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument. If the pen is deleted the return value will be GeneralFailure.

**Limits**

- Minimum = 0
- Maximum = 8

**Calling Syntax**

This example assumes there is a valid Digital Pen object to be passed into the example methods.

**[VBA]**

```
Sub Example(digitalPen As Object)
Dim lineWidth As Long
`Getting Property value
lineWidth = digitalPen.LineWidth
`Setting Property value
digitalPen.LineWidth = 5
```

```
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hDigitalPen)
// Getting property value
INT nLineWidth = _ObjectGetProperty(hDigitalPen, "LineWidth");
// Setting property value
_ObjectSetProperty(hDigitalPen, "LineWidth", 5);
END
```

## IObjectView Interface

### Defined As

- [VBA] Object
- [Cicode] OBJECT
- [C++] IObjectView

### Methods (0)

### Properties (7)

- IObjectView.Visible [Property][Get/Set]
- IObjectView.Height [Property][Get/Set]
- IObjectView.BackgroundColor [Property][Get/Set]
- IObjectView.ForeColor [Property][Get/Set]
- IObjectView.Columns [Property][Get]
- IObjectView.Items [Property][Get]
- IObjectView.SelectedItem [Property][Get]

## IObjectView.BackgroundColor [Property][Get/Set]

Gets or Sets the background color of the ObjectView. This number is treated as an OLE\_COLOR inside the Process Analyst.

### Defined As

- [VBA] Long BackgroundColor
- [Cicode] INT BackgroundColor
- [C++] OLE\_COLOR BackgroundColor

### Execution Result

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

### Remarks

The color value can be calculated using the following formula: color = (65536 \* Blue) + (256 \* Green) + (Red). Where red, green and blue are 0-255.

#### Calling Syntax

This example assumes that there is an IObjectView object being passed in as a parameter.

#### [VBA]

```
Sub Example(objectView As Object)
Dim backgroundColor As Long
`Getting Property value
backgroundColor = objectView.BackgroundColor
`Setting Property value to red
objectView.BackgroundColor = 255
End Sub
```

#### [Cicode]

```
FUNCTION Example(OBJECT hObjectView)
// Getting property value
INT nBackgroundColor =
_ObjectGetProperty(hObjectView, "BackgroundColor");
// Setting Property to Red
_ObjectSetProperty(hObjectView, "BackgroundColor", 255);
END
```

## IObjectView.Columns [Property][Get]

Gets the automation object representing the collection of columns currently visible in the ObjectView.

#### Defined As

- [VBA] Object Columns
- [Cicode] OBJECT Columns
- [C++] IObjectViewColumns\* Columns

#### Execution Result

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

#### Calling Syntax

This example assumes that there is an IObjectView object being passed in as a parameter.

#### [VBA]

```
Sub Example(objectView As Object)
Dim columns As Object
`Getting Property value
Set columns = objectView.Columns
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hObjectView)
// Getting property value
OBJECT hColumns = _ObjectGetProperty(hObjectView, "Columns");
END
```

## IOBJECTVIEW.ForeColor [Property][Get/Set]

Gets or Sets the Fore color (text and color box outlines) of the ObjectView. This number is treated as an OLE\_COLOR inside the Process Analyst.

**Defined As**

- [VBA] Long ForeColor
- [Cicode] INT ForeColor
- [C++] OLE\_COLOR ForeColor

**Execution Result**

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

**Remarks**

The color value can be calculated using the following formula: color = (65536 \* Blue) + (256 \* Green) + (Red). Where red, green and blue are 0-255.

**Calling Syntax**

This example assumes that there is an IObjectView object being passed in as a parameter.

**[VBA]**

```
Sub Example(objectView As Object)
Dim foreColor As Long
`Getting Property value
foreColor = objectView.ForeColor
`Setting Property value to red
objectView.ForeColor = 255
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hObjectView)
INT nForeColor = 0;
// Getting property value
nForeColor = _ObjectGetProperty(hObjectView, "ForeColor");
// Setting Property to red
_ObjectSetProperty(hObjectView, "ForeColor", 255);
END
```

# IObjectView.Height [Property][Get/Set]

Gets or Sets the height in pixels of the Object View window.

## Defined As

- [VBA] Long Height
- [Cicode] INT Height
- [C++] int Height

## Execution Result

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. A height value less than 0 will be InvalidArgument.

## Limits

- Height needs to be 0 or greater.

## Remarks

As the ObjectView and chart both share the same window, by enlarging the ObjectView, you make the Chart smaller and vice versa.

## Calling Syntax

This example assumes that there is an IObjectView object being passed in as a parameter.

## [VBA]

```
Sub Example(objectView As Object)
Dim height As Long
`Getting Property value
height = objectView.Height
`Setting Property value
objectView.Height = 25
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hObjectView)
INT nHeight = 0;
// Getting property value
nHeight = _ObjectGetProperty(hObjectView, "Height");
// Setting Property to false
_Object SetProperty(hObjectView, "Height", 25);
END
```

# IObjectView.Items [Property][Get]

Gets the automation object representing the collection of items at the root of the ObjectView tree.

**Defined As**

- [VBA] Object Items
- [Cicode] OBJECT Items
- [C++] IObjectViewItems\* Items

**Execution Result**

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

**Remarks**

The method will provide a list of the pane items in tree. Each pane item has an Items property which allows access to the pen items listed under it.

**Calling Syntax**

This example assumes that there is an IObjectView object being passed in as a parameter.

**[VBA]**

```
Sub Example(objectView As Object)
Dim items As Object
`Getting Property value
Set items = objectView.Items
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hObjectView)
// Getting property value
OBJECT hItems = _ObjectGetProperty(hObjectView, "Items");
END
```

## IObjectView.SelectedItem [Property][Get]

Gets the current primary selection in the ObjectView. This is the pen item that was last selected.

**Defined As**

- [VBA] Object SelectedItem
- [Cicode] OBJECT SelectedItem
- [C++] IObjectViewItem\* SelectedItem

**Execution Result**

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

**Calling Syntax**

This example assumes that there is an IObjectView object being passed in as a parameter.

**[VBA]**

```
Sub Example(objectView As Object)
Dim selectedItem As Object
`Getting Property value
Set selectedItem = objectView.SelectedItem
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hObjectView)
// Getting property value
OBJECT hSelectedItem = _ObjectGetProperty(hObjectView, "SelectedItem");
END
```

## IObjectView.Visible [Property][Get/Set]

Gets or Sets the visibility of the Object View window.

**Defined As**

- [VBA] Boolean Visible
- [Cicode] INT Visible
- [C++] VARIANT\_BOOL Visible

**Execution Result**

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

**Limits**

- True (-1): Visible
- False (0): Hidden

**Remarks**

By hiding the ObjectView, the chart gains the real estate previously held by it, and likewise the chart loses real estate when the ObjectView gets shown.

**Calling Syntax**

Assume that there is an IObjectView object being passed in as a parameter.

**[VBA]**

```
Sub Example(objectView As Object)
Dim visible As Boolean
`Getting Property value
visible = objectView.Visible
`Setting Property value
objectView.Visible = False
```

End Sub

**[Cicode]**

```
FUNCTION Example(OBJECT hObjectView)
INT bVisible = 0
// Getting property value
bVisible = _ObjectGetProperty(hObjectView, "Visible");
// Setting Property to false
_ObjectSetProperty(hObjectView, "Visible", 0);
END
```

**IObjectViewColumn Interface****Defined As**

- [VBA] Object
- [Cicode] OBJECT
- [C++] IObjectViewColumn

**Methods (0)****Properties (3)**

- IObjectViewColumn.Name [Property][Get]
- IObjectViewColumn.Text [Property][Get]
- IObjectViewColumn.Width [Property][Get/Set]

# IObjectViewColumn.Name [Property][Get]

Retrieves the unique identifier of this column.

**Defined As**

- [VBA] String Name
- [Cicode] STRING Name
- [C++] BSTR Name

**Execution Result**

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

**Calling Syntax**

This example assumes there is a valid column as retrieved from an ObjectView. (for example, VBA: objectView.Columns.Item(1)).

**[VBA]**

```
Sub Example(objectViewColumn As Object)
Dim name As String
`Getting Property value
name = objectViewColumn.Name
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hObjectViewColumn)
// Getting property value
STRING name = _ObjectGetProperty(hObjectViewColumn, "Name");
END
```

## IOBJECTVIEWCOLUMN.TEXT [Property][Get]

Gets the Text that is being displayed for this columns header.

**Defined As**

- [VBA] String Text
- [Cicode] STRING Text
- [C++] BSTR Text

**Execution Result**

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

**Calling Syntax**

This example assumes there is a valid column as retrieved from an ObjectView. (for example, VBA: objectView.Columns.Item(1)).

**[VBA]**

```
Sub Example(objectViewColumn As Object)
Dim text As String
`Getting Property value
text = objectViewColumn.Text
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hObjectViewColumn)
// Getting property value
STRING text = _ObjectGetProperty(hObjectViewColumn, "Text");
END
```

# IObjectViewColumn.Width [Property][Get/Set]

Gets or Sets the width in pixels of this column.

## Defined As

- [VBA] Long Width
- [Cicode] INT Width
- [C++] int Width

## Execution Result

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the width is out of range, the result will be InvalidArgument.

## Limits

A valid width is 0-1000.

## Calling Syntax

This example assumes there is a valid column as retrieved from an ObjectView. (for example, VBA: objectView.Columns.Item(1)).

## [VBA]

```
Sub Example(objectViewColumn As Object)
Dim width As Long
`Getting Property value
width = objectViewColumn.Width
`Setting Property value
objectViewColumn.Width = 150
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hObjectViewColumn)
// Getting property value
INT width = _ObjectGetProperty(hObjectViewColumn, "Width");
_ObjectSetProperty(hObjectViewColumn, "Width", 150);
END
```

## IObjectViewColumns Interface

### Defined As

- [VBA] Object
- [Cicode] OBJECT
- [C++] IObjectViewColumns

### Methods (4)

- IObjectViewColumns.Add [Method]
- IObjectViewColumns.Hide [Method]
- IObjectViewColumns.Remove [Method]
- IObjectViewColumns.Show [Method]

#### Properties (4)

- IObjectViewColumns.Count [Property][Get]
- IObjectViewColumns.Item [Property][Get]
- IObjectViewColumns.ItemByName [Property][Get]
- IObjectViewColumns.\_NewEnum [Property][Get]

## IObjectViewColumns.\_NewEnum [Property][Get]

This allows "For... Each... Next" integration in VB.

#### Calling Syntax

This example assumes there is a valid Columns collection as retrieved from an ObjectView. (for example, VBA: objectView.Columns). This property is not applicable in Cicode.

#### [VBA]

```
Sub Example(Columns As Object)
Dim column As Object
Dim count Object
`Using Property
For Each column In Columns
count = count + 1
Next column
End Sub
```

## IObjectViewColumns.Add [Method]

Adds a visible custom column to the ObjectView.

#### Defined As

- [VBA] Add(name As String, DisplayText As String, Width As Long)
- [Cicode] Add(STRING name, STRING DisplayText, INT Width)
- [C++] HRESULT Add (BSTR name, BSTR text, int width)

#### Parameters

name

[in] The string ID uniquely identifying this column (1-64).

text

[in] The title to be displayed in the column header (0-256).

width

[in] The width of this column in pixels (0-1000).

#### Execution Result

If the method succeeds, the return value will be Success. If an argument is out of range, the return value will be InvalidArgument. If the column cannot be added, the return value is GeneralFailure.

#### Calling Syntax

This example assumes there is a valid Columns collection as retrieved from an ObjectView. (for example, VBA: objectView.Columns).

### [VBA]

```
Sub Example(Columns As Object)
Columns.Add "NameID", "New Column", 120;
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hColumns)
_ObjectCallMethod(hColumns, "Add", "NameID", "New Column", 120);
END
```

### See Also

[OVColumnAdded \[Event\]](#)

## IObjectViewColumns.Count [Property][Get]

Gets the number of Columns in this columns collection.

#### Defined As

- [VBA] Long Count
- [Cicode] INT Count
- [C++] int Count

#### Execution Result

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

#### Calling Syntax

This example assumes there is a valid Columns collection as retrieved from an ObjectView. (for example, VBA: ObjectView.Columns).

**[VBA]**

```
Function Example(Columns As Object)
Dim count As Long
`Getting Property value
count = Columns.Count
End Function
```

**[Cicode]**

```
FUNCTION Example(OBJECT hColumns)
// Getting property value
INT nCount = _ObjectGetProperty(hColumns, "Count");
END
```

## IObjectViewColumns.Hide [Method]

Makes the specified column hidden within the Object View.

**Defined As**

- [VBA] Hide(columnName As String)
- [Cicode] Hide(STRING columnName)
- [C++] HRESULT Hide(BSTR columnName)

**Parameters**

columnName

[in] The string ID uniquely identifying the column you want to hide

**Execution Result**

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the field cannot be set, GeneralFailure is returned.

**Calling Syntax**

This example assumes there is a columns Collection from an ObjectView. (for example, VBA: objectView.Columns).

**[VBA]**

```
Sub Example(columns As Object)
columns.Hide "Error"
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hColumns)
_ObjectCallMethod(hColumns, "Hide", "Error");
END
```

## See Also

[IObjectViewColumns.Show \[Method\]](#)

# IObjectViewColumns.Item [Property][Get]

Gets the ObjectViewItem at a supplied index location in this collection.

### Defined As

- [VBA] Item(index As Long) as Object
- [Cicode] OBJECT Item(INT index)
- [C++] Item(int index, IObjectViewColumn\* Item)

### Parameters

index

[in] Indicates the index location of the column to return from this collection. (One based)

### Calling Syntax

This example assumes there is a valid Columns collection as retrieved from an ObjectView (for example, VBA: objectView.Columns).

## [VBA]

```
Sub Example(Columns As Object)
Dim column As Object
`Getting Property value
Set column = Columns.Item(1)
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hColumns)
// Getting property value
OBJECT hColumn = _ObjectCallMethod(hColumns, "get_Item", 1);
END
```

# IObjectViewColumns.ItemByName [Property][Get]

Returns a reference to the column object with the given name from this column's collection.

### Defined As

- [VBA] ItemByName(columnName As String) as Object
- [Cicode] OBJECT ItemByName(STRING columnName)
- [C++] ItemByName(STRING columnName, IObjectViewColumn\* Item)

**Parameters****columnName**

[in] Indicates the unique name of the column item to return from this collection.

**Execution Results**

If the method succeeds, the return value will be Success. If the column cannot be found, the return value will be InvalidArgument.

**Calling Syntax**

This example assumes there is a valid Columns collection object to be passed into the example methods.

**[VBA]**

```
Sub Example(columns As Object)
Dim column As Object
`Getting Property value
Set column = columns.ItemByName("Duration")
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hColumns)
// Getting property value
OBJECT hColumn = _ObjectCallMethod(hColumns, "get_ItemByName", "Duration");
END
```

## IObjectViewColumns.Remove [Method]

Removes the specified custom column from the Object View columns.

**Defined As**

- [VBA] Remove(columnName As String)
- [Cicode] OBJECT Remove(STRING columnName)
- [C++] Remove(STRING columnName)

**Parameters****columnName**

[in] Indicates the unique name of the column to remove from this collection.

**Execution Results**

If the method succeeds, the return value will be Success. If the column cannot be found, the return value will be InvalidArgument.

**Remarks**

Only user created custom columns can be removed.

**Calling Syntax**

This example assumes there is a valid Columns collection object to be passed into the example methods.

**[VBA]**

```
Sub Example(columns As Object)
Dim column As Object
`Getting Property value
Set column = columns.Remove("MyCustomColumn"
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hColumns)
// Getting property value
OBJECT hColumn = _ObjectCallMethod(hColumns, "get_ItemByName",
"MyCustomColumn");
END
```

## IOBJECTVIEWCOLUMNS.SHOW [Method]

Makes the specified column visible within the Object View.

**Defined As**

- [VBA] Show(columnName As String)
- [Cicode] Show(STRING columnName)
- [C++] HRESULT Show(BSTR columnName)

**Parameters**

columnName

[in] The string ID uniquely identifying the column you want to make visible

**Execution Result**

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the field cannot be set, then GeneralFailure is returned.

**Calling Syntax**

This example assumes there is a columns Collection from an ObjectView. (for example, VBA: objectView.Columns).

**[VBA]**

```
Sub Example(columns As Object)
columns.Show "Error"
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hColumns)
_ObjectCallMethod(hColumns, "Show", "Error");
END
```

## See Also

[IObjectViewColumns.Hide \[Method\]](#)

### IObjectViewItem Interface

#### Defined As

- [VBA] Object
- [Cicode] OBJECT
- [C++] IObjectViewItem

#### Methods (2)

- [IObjectViewItem.GetField \[Method\]](#)
- [IObjectViewItem.PutField \[Method\]](#)

#### Properties (3)

- [IObjectViewItem.Expanded \[Property\]\[Get/Set\]](#)
- [IObjectViewItem.Tag \[Property\]\[Get/Set\]](#)
- [IObjectViewItem.Items \[Property\]\[Get\]](#)

## IObjectViewItem.Expanded [Property][Get/Set]

Gets or Sets the expanded state of an item in the ObjectView. This change is reflected immediately in the visualization of the ObjectView.

#### Defined As

- [VBA] Boolean Expanded
- [Cicode] INT Expanded
- [C++] VARIANT\_BOOL Expanded

#### Execution Result

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

#### Limits

- True (-1): Expanded
- False (0): Collapsed

#### Calling Syntax

This example assumes there is a valid Item as retrieved from an Items Collection from an ObjectView. (for example, VBA: objectView.Items.Item(1)).

**[VBA]**

```
Sub Example(objectViewItem As Object)
Dim expanded As Boolean
`Getting Property value
expanded = objectViewItem.Expanded
`Setting Property value
objectViewItem.Expanded = False
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hObjectViewItem)
// Getting property value
INT nExpanded = _ObjectGetProperty(hObjectViewItem, "Expanded");
// Setting Property
_ObjectSetProperty(hObjectViewItem, "Expanded", 0);
END
```

## IOBJECTVIEWITEM.GETFIELD [Method]

Returns the string value of a displayed field for a specified column on this item.

The IObjectViewItem interface is hierarchical to two levels - pane and then pen. The result of the GetField method will depend on what type of item it is called on. To access the fields for a pen, for example, you have to first get the items collection for the pane item, then get the pen item.

**Defined As**

- [VBA] GetField(ColumnName As String) as String
- [Cicode] STRING GetField (STRING ColumnName)
- [C++] HRESULT GetField (BSTR ColumnName, BSTR \*Val)

**Parameters**

ColumnName

[in] The string ID uniquely identifying the column whose field value is being queried for.

**Execution Result**

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the ColumnName does not exist, InvalidArgument will be returned.

**Calling Syntax**

This example gets the Scale property of the first pen in the first pane. It assumes there is a valid Item as retrieved from an Items Collection from an ObjectView. (for example, VBA: objectView.Items.Item(1))

**[VBA]**

```
Sub Example()
Dim paneItem As Object
Dim penItem As Object
```

```
Dim fieldValue As String
Set paneItem = Test_CPA.ObjectView.Items.Item(1) ' Get the first pane of the ObjectView
Set penItem = paneItem.Items.Item(1) ' Get the first pen from the first pane
penItem.GetField "Scale", fieldValue ' Get the value of the scale field
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hObjectView)
OBJECT hPaneItems = _ObjectGetProperty(hObjectView, "Items");
OBJECT hPaneItem = _ObjectCallMethod(hPaneItems, "get_Item", 1);
// Get the first pane of the ObjectView
OBJECT hPenItems = _ObjectGetProperty(hPaneItem, "Items");
// Get the collection of pens from the first pane
OBJECT hPenItem = _ObjectCallMethod(hPenItems, "get_Item", 1);
// Get the first Pen item
STRING sValue;
_ObjectCallMethod(hPenItem, "GetField", "Scale", sValue);
// Get the value of the scale field
END
```

# IOBJECTVIEWITEM.ITEMS [Property][Get]

Gets the automation object representing the collection of child items under this item.

### Defined As

- [VBA] Object Items
- [Cicode] OBJECT Items
- [C++] IObjectViewItem\* Items

### Execution Result

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

### Remarks

Pane nodes are currently the only nodes that can have children.

### Calling Syntax

This example assumes there is a valid item as retrieved from an ObjectView. (for example, VBA: objectView.Items.Item(1). This will be a pane).

## [VBA]

```
Sub Example(objectViewItem As Object)
Dim items As Object
`Getting Property value
Set items = objectViewItem.Items
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hObjectViewItem)
// Getting property value
OBJECT hItems = _ObjectGetProperty(hObjectViewItem, "Items");
END
```

## IObjectViewItem.PutField [Method]

Sets the display string in a field's cell for a specified column on this item.

The IObjectViewItem interface is hierarchical to two levels - pane and then pen. The scope of the PutField method will depend on what type of item it is called on. To set fields for a pen, for example, you have to first get the items collection for the pane item, then get the pen item.

**Defined As**

- [VBA] PutField(columnName As String, fieldValue as String)
- [Cicode] PutField (STRING columnName, STRING fieldValue)
- [C++] HRESULT PutField (BSTR columnName, BSTR fieldValue)

**Parameters**

columnName

[in] The string ID uniquely identifying the column whose field value is being set.

fieldValue

[in] The string you would like to be displayed in the field for this column/pen intersection.

**Execution Result**

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the field cannot be set, then GeneralFailure is returned.

**Calling Syntax**

This example writes the value "someValue" to the CustomColumn field of the first pen in the first pane. It assumes there is a valid Item as retrieved from an Items Collection from an ObjectView. (for example, VBA: objectView.Items.Item(1)).

**[VBA]**

```
Sub Example()
Dim paneItem As Object
Dim penItem As Object
Set paneItem = Test_CPA.ObjectView.Items.Item(1)
' Get the first pane of the ObjectView
Set penItem = paneItem.Items.Item(1)
' Get the first pen from the first pane
penItem.PutField "CustomColumn", "someValue"
'set the value of the CustomColumn field
End Sub
```

**[Cicode]**

```

FUNCTION Example(OBJECT hObjectView)
OBJECT hPaneItems = _ObjectGetProperty(hObjectView, "Items");
OBJECT hPaneItem = _ObjectCallMethod(hPaneItems, "get_Item", 1);
// Get the first pane of the ObjectView
OBJECT hPenItems = _ObjectGetProperty(hPaneItem, "Items");
// Get the collection of pens from the first pane
OBJECT hPenItem = _ObjectCallMethod(hPenItems, "get_Item", 1);
// Get the first Pen item
_ObjectCallMethod(hPenItem, "PutField", "CustomColumn", "someValue");
// Set the value of the CustomColumn field
END

```

## IOBJECTVIEWITEM.TAG [Property][Get/Set]

Gets or Sets a user specified piece of data to associate with this Item.

**Defined As**

- [VBA] <Any Type> Tag
- [Cicode] <Any Type> Tag
- [C++] VARIANT Tag

**Remarks**

The user can associate any variant of data with a pen. This is handy for associating some custom data with a pen item, and then having direct access to it whenever any events with a pen item target occur.

**Calling Syntax**

This example assumes there is a valid Item as retrieved from an Items Collection from an ObjectView. (for example, VBA: objectView.Items.Item(1)).

**[VBA]**

```

Sub Example(objectViewItem As Object)
Dim tag As Variant
`Getting Property value
tag = objectViewItem.Tag
`Setting Property value to red
objectViewItem.Tag = tag
End Sub

```

**[Cicode]**

```

FUNCTION Example(OBJECT hObjectViewItem)
// Getting property value
INT nTag = _ObjectGetProperty(hObjectViewItem, "Tag");
// Setting Property to red
_ObjectSetProperty(hObjectView, "Tag", nTag);
END

```

## IObjectViewItems Interface

### Defined As

- [VBA] Object
- [Cicode] OBJECT
- [C++] IObjectViewItems

### Methods (0)

### Properties (3)

- IObjectViewItems.Count [Property][Get]
- IObjectViewItems.Item [Property][Get]
- IObjectViewItems.\_NewEnum [Property][Get]

## IObjectViewItems.\_NewEnum [Property][Get]

This allows For.. Each.. Next integration in VB.

### Calling Syntax

This example assumes there is a valid Items collection as retrieved from an ObjectView. (for example, VBA: objectView.Items). This property is not applicable to Cicode.

### [VBA]

```
Sub Example(Items As Object)
Dim item As Object
Dim count Object
`Using Property
For Each item In Items
count = count + 1
Next Item
End Sub
```

## IObjectViewItems.Count [Property][Get]

Gets the number of child items under this item.

### Defined As

- [VBA] Long Count
- [Cicode] INT Count
- [C++] int Count

### Execution Result

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will

be InvalidArgument.

#### Calling Syntax

This example assumes there is a valid Items collection as retrieved from an ObjectView. (for example, VBA: objectView.Items)

#### [VBA]

```
Sub Example(Items As Object)
Dim count As Long
`Getting Property value
count = Items.Count
End Sub
```

#### [Cicode]

```
FUNCTION Example(OBJECT hItems)
// Getting property value
INT nCount = _ObjectGetProperty(hItems, "Count");
END
```

## IObjectViewItem[Property][Get]

Gets the ObjectViewItem at a supplied index location in this collection.

#### Defined As

- [VBA] Item(index As Long) as Object
- [Cicode] OBJECT Item(INT index)
- [C++] Item(int index, IObjectViewItem\* Item)

#### Parameters

index

[in] Indicates the index location of the child item to return from this collection. (One based)

#### Calling Syntax

This example assumes there is a valid Items collection as retrieved from an ObjectView. (for example, VBA: objectView.Items).

#### [VBA]

```
Sub Example(Items As Object)
Dim item As Object
`Getting Property value
Set item = Items.Item(1)
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hItems)
// Getting property value
OBJECT hItem = _ObjectCallMethod(hItems, "get_Item", 1);
END
```

**IObjectViewPenItem Interface****Defined As**

- [VBA] Object
- [Cicode] OBJECT
- [C++] IObjectViewPenItem

**Methods (0)****Properties (3)**

- IObjectViewPenItem.BlockColor [Property][Get]
- IObjectViewPenItem.Checked [Property][Get/Set]
- IObjectViewPenItem.Selected [Property][Get]

## IObjectViewPenItem.BlockColor [Property][Get]

Gets the color representing this item in the ObjectView.

**Defined As**

- [VBA] Long BlockColor
- [Cicode] INT BlockColor
- [C++] OLE\_COLOR BlockColor

**Execution Result**

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

**Remarks**

The color value can be calculated using the following formula:  $\text{color} = (65536 * \text{Blue}) + (256 * \text{Green}) + (\text{Red})$  where red, green, and blue are 0-255.

**Calling Syntax**

This example assumes there is a valid pen item as retrieved from an ObjectView. (for example, VBA: ObjectView.Items.Item(1).Items.Item(1) This will be a pen).

**[VBA]**

```
Sub Example(objectViewPenItem As Object)
```

```
Dim blockColor As Long
`Getting Property value
blockColor = objectViewPenItem.BlockColor
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hObjectViewPenItem)
// Getting property value
INT blockColor = _ObjectGetProperty(hObjectViewItem, "BlockColor");
END
```

## IOBJECTVIEWPENITEM.CHECKED [Property][Get/Set]

Gets or Sets whether or not this pen item is checked.

#### Defined As

- [VBA] Boolean Checked
- [Cicode] INT Checked
- [C++] VARIANT\_BOOL Checked

#### Execution Result

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

#### Limits

- True (-1): Checked
- False (0): Unchecked

#### Remarks

This reflects the pens visibility property directly, and any sets to this property will reflect immediately in the update of the Process Analyst display.

#### Calling Syntax

This example assumes there is a valid pen item as retrieved from an ObjectView. (for example, VBA: objectView.Items.Item(1).Items.Item(1) This will be a pen).

### [VBA]

```
Sub Example(objectViewPenItem As Object)
Dim checked As Boolean
`Getting Property value
checked = objectViewPenItem.Checked
`Setting Property value
objectViewPenItem.Checked = False
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hObjectViewItem)
// Getting property value
INT checked = _ObjectGetProperty(hObjectViewItem, "Checked");
// Setting property value
_ObjectSetProperty(hObjectViewItem, "Checked", 0);
END
```

**See Also**

[OVItemChecked \[Event\]](#)

## IObjectViewPenItem.Selected [Property][Get]

Gets whether or not this pen is the selected pen in its pane.

**Defined As**

- [VBA] Boolean Selected
- [Cicode] INT Selected
- [C++] VARIANT\_BOOL Selected

**Execution Result**

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

**Limits**

- True (-1): Selected
- False (0): Unselected

**Remarks**

Each Pane has one selected pen. It is visually emphasized by a vertical gradient fill.

**Calling Syntax**

This example assumes there is a valid pen item as retrieved from an ObjectView. (for example, VBA: objectView.Items.Item(1).Items.Item(1) This will be a pen).

**[VBA]**

```
Sub Example(objectViewPenItem As Object)
Dim selected As Boolean
`Getting Property value
selected = objectViewPenItem.Selected
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hObjectViewItem)
// Getting property value
INT selected = _ObjectGetProperty(hObjectViewItem, "Selected");
END
```

## IPane Interface

### Defined As

- [VBA] Object
- [Cicode] OBJECT
- [C++] IPanes

### Methods (1)

- IPane.Delete [Method]

### Properties (6)

- IPane.Height [Property][Get/Set]
- IPane.Collection [Property][Get]
- IPane.Name [Property][Get/Set]
- IPane.BackgroundColor [Property][Get/Set]
- IPane.FixedHeight [Property][Get/Set]
- IPane.Pens [Property][Get]

## IPane.BackgroundColor [Property][Get/Set]

Gets or Sets the color of this Pane.

### Defined As

- [VBA] Long BackgroundColor
- [Cicode] INT BackgroundColor
- [C++] OLE\_COLOR BackgroundColor

### Execution Result

If the property get/set succeeds, the return value will be Success. If the pane is deleted, the return value will be GeneralFailure.

### Remarks

The color value can be calculated using the following formula: color = (65536 \* Blue) + (256 \* Green) + (Red). Where red, green and blue are 0-255.

### Calling Syntax

This example assumes there is a valid Pane object to be passed into the example methods.

### [VBA]

```
Sub Example(Pane As Object)
Dim backgroundColor As Long
`Getting Property value
backgroundColor = Pane.BackgroundColor
`Setting Property value to red
Pane.BackgroundColor = 255
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPane)
// Getting property value
INT nColor = _ObjectGetProperty(hPane, "BackgroundColor");
// Setting property value
_ObjectSetProperty(hPane, "BackgroundColor", 255);
END
```

## IPane.Collection [Property][Get]

Returns a reference to the Panes collection that this Pane belongs to.

### Defined As

- [VBA] Object Collection
- [Cicode] OBJECT Collection
- [C++] IPanes\* Collection

### Execution Result

If the property get succeeds the return value will be Success. If the pane is deleted the return value will be GeneralFailure.

### Calling Syntax

This example assumes there is a valid Pane object to be passed into the example methods.

### [VBA]

```
Sub Example(pane As Object)
Dim panes As Object
`Getting Property value
Set panes = pane.Collection
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPane)
```

```
// Getting property value
OBJECT hPanes = _ObjectGetProperty(hPane, "Collection");
END
```

## See Also

[IPanes Interface](#)

# IPane.Delete [Method]

Removes this Pane from the collection and the display.

### Defined As

- [VBA] Delete()
- [Cicode] Delete()
- [C++] HRESULT Delete()

### Execution Result

If the method succeeds, the return value will be Success. If the pane is already deleted, the return value will be GeneralFailure.

### Remarks

Any pen associated with the pane will also be deleted.

### Calling Syntax

This example assumes there is a valid Pane object to be passed into the example methods.

## [VBA]

```
Sub Panes(Pane As Object)
Pane.Delete()
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPane)
_ObjectCallMethod(hPane, "Delete");
END
```

# IPane.FixedHeight [Property][Get/Set]

Gets or Sets whether this pane has a fixed height.

### Defined As

- [VBA] Boolean FixedHeight

- [Cicode] INT FixedHeight
- [C++] VARIANT\_BOOL FixedHeight

### Execution Result

If the property get/set succeeds, the return value will be Success. If the pane is deleted, the return value will be GeneralFailure.

### Limits

- True (-1): Height is fixed
- False (0): Height is variable

### Remarks

When this property is true, the pane's Height reflects the pixel value size as gotten from the Pane's Height property. If the FixedHeight property is false, the Height property value is used as a ratio of the available 'Variable' real estate (all the left over room in the Process Analyst after Fixed Height panes have been added) which is shared out between the Variable Height panes.

### Calling Syntax

This example assumes there is a valid Pane object to be passed into the example methods.

## [VBA]

```
Sub Example(pane As Object)
Dim fixedHeight As Boolean
`Getting Property value
fixedHeight = pane.FixedHeight
`Setting Property value
pane.FixedHeight = True
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPane)
// Getting property value
INT bFixedHeight = _ObjectGetProperty(hPane, "FixedHeight");
// Setting property value
_ObjectSetProperty(hPane, "FixedHeight", -1);
END
```

## See Also

[IPane.Height \[Property\]\[Get/Set\]](#)

# IPane.Height [Property][Get/Set]

Gets or Sets the height of this pane.

### Defined As

- [VBA] Long Height
- [Cicode] INT Height
- [C++] int Height

### Execution Result

If the property get/set succeeds, the return value will be Success. If the height is out of range (16-1000), the return value will be InvalidArgument. If the pane is deleted, the return value will be GeneralFailure.

### Remarks

This property affects the visible height of the Pane in two different ways based on the Boolean value of the FixedHeight property. If the FixedHeight property is True, the Pane takes on a pixel height equivalent to the Height property value. Every pen inside the Pane is adjusted to fit. If the FixedHeight property is False, the Height property value is used as a ratio of the available 'Variable' real estate (all the left over room in the Process Analyst after Fixed Height panes have been added) which is shared out between the Variable Height panes.

### Calling Syntax

This example assumes there is a valid Pane object to be passed into the example methods.

## [VBA]

```
Sub Example(Pane As Object)
Dim height As Long
`Getting Property value
height = Pane.Height
`Setting Property value
Pane.Height = 250
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPane)
// Getting property value
INT nHeight = _ObjectGetProperty(hPane, "Height");
// Setting property value
_ObjectSetProperty(hPane, "Height", 250);
END
```

## See Also

[IPane.FixedHeight \[Property\]\[Get/Set\]](#)

# IPane.Name [Property][Get/Set]

Gets or Sets the name of this pane.

### Defined As

- [VBA] String Name
- [Cicode] STRING Name

- [C++] BSTR Name

#### Execution Result

If the property get/set succeeds, the return value will be Success. If a pane of the same name exists, the return value will be InvalidArgument. If the panes collection is deleted, the return value will be GeneralFailure.

#### Limits

Name needs to be between 1-250 characters.

#### Remarks

Pane names needs to be unique.

#### Calling Syntax

This example assumes there is a valid Pane object to be passed into the example methods.

### [VBA]

```
Sub Example(pane As Object)
Dim name As String
`Getting Property value
name = pane.Name
`Setting Property value
pane.Name = "Alarms"
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPane)
// Getting property value
STRING sName = _ObjectGetProperty(hPane, "Name");
// Setting property value
_ObjectSetProperty(hPane, "Name", "Alarms");
END
```

## IPane.Pens [Property][Get]

Gets a reference to the pens collection object containing the pens for this pane.

#### Defined As

- [VBA] Object Pens
- [Cicode] OBJECT Pens
- [C++] IPens\* Pens

#### Execution Result

If the property get succeeds the return value will be Success. If the pane is deleted the return value will be GeneralFailure.

#### Calling Syntax

This example assumes there is a valid Pane object to be passed into the example methods.

**[VBA]**

```
Sub Example(pane As Object)
Dim pens As Object
`Getting Property value
Set pens = pane.Pens
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPane)
// Getting property value
OBJECT hPens = _ObjectGetProperty(hPane, "Pens");
END
```

**See Also**

[IPens Interface](#)

**IPanes Interface****Defined As**

- [VBA] Object
- [Cicode] OBJECT
- [C++] IPanes

**Methods (2)**

- [IPanes.Create \[Method\]](#)
- [IPanes.RemoveAll \[Method\]](#)

**Properties (4)**

- [IPanes.Count \[Property\]\[Get\]](#)
- [IPanes.Item \[Property\]\[Get\]](#)
- [IPanes.\\_NewEnum \[Property\]\[Get\]](#)
- [IPanes.ItemByName \[Property\]\[Get\]](#)

## IPanes.\_NewEnum [Property][Get]

This allows For.. Each.. Next integration in VB.

**Calling Syntax**

This example assumes there is a valid Panes collection object to be passed into the example methods. This property is not applicable to Cicode.

**[VBA]**

```
Sub Example(Panes As Object)
Dim pane As Object
Dim count As Long
`Using Property
For Each pane In Panes
count = count + 1
Next pane
End Sub
```

## IPanes.Count [Property][Get]

Gets the number of Panes in this collection.

**Defined As**

- [VBA] Long Count
- [Cicode] INT Count
- [C++] int Count

**Execution Result**

If the property get succeeds, the return value will be Success. If the panes collection is deleted, the return value will be GeneralFailure.

**Calling Syntax**

This example assumes there is a valid Panes collection object to be passed into the example methods.

**[VBA]**

```
Sub Example(Panes As Object)
Dim count As Long
`Getting Property value
count = Panes.Count
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPanes)
// Getting property value
INT nCount = _ObjectGetProperty(hPanes, "Count");
END
```

## IPanes.Create [Method]

Adds a pane to this collection and returns a reference to it.

**Defined As**

- [VBA] Create(name as String) as Object
- [Cicode] OBJECT Create (STRING name)
- [C++] HRESULT Create(BSTR name, IPane\*\* pane)

#### Parameters

name

[in] The name to give to the pane (0-250 characters).

#### Execution Result

If the method succeeds, the return value will be Success. If a pane of the same name exists, the return value will be InvalidArgument. If the panes collection is deleted, the return value will be GeneralFailure.

#### Remarks

When this method succeeds it will return a reference to the new IPane object.

#### Calling Syntax

This example assumes there is a valid Panes collection object to be passed into the example methods.

### [VBA]

```
Sub Example(Panes As Object)
Dim pane As Object
Set pane = Panes.Create("Alarm Pane")
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPanes)
OBJECT hPane = _ObjectCallMethod(hPanes, "Create", "Alarm Pane");
END
```

### See Also

[IPanes Interface](#)

## IPanes.Item [Property][Get]

Gets the Pane at the given index in this Pane collection.

#### Defined As

- [VBA] Item(index As Long) as Object
- [Cicode] OBJECT Item(INT index)
- [C++] Item(int index, IPane\* Item)

#### Parameters

index

[in] Indicates the location of the Pane item to return from this collection. (One based)

#### Execution Result

If the property get succeeds the return value will be Success. If the index is out of range then the return value will be InvalidArgument. If the panes collection is deleted the return value will be GeneralFailure.

#### Calling Syntax

This example assumes there is a valid Panes collection object to be passed into the example methods.

### [VBA]

```
Sub Example(Panes As Object)
Dim pane As Object
`Getting Property value
Set pane = Panes.Item(1)
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPanes)
// Getting property value
OBJECT hPane = _ObjectCallMethod(hPanes, "get_Item", 1);
END
```

### See Also

[IPane Interface](#)

## IPanes.ItemByName [Property][Get]

Returns a reference to the pane object with the given name from this Panes collection.

#### Defined As

- [VBA] ByName(name As String) as Object
- [Cicode] OBJECT ByName(STRING name)
- [C++] ByName(STRING name, IPane\* Item)

#### Parameters

name

[in] Indicates the name of the Pane item to return from this collection.

#### Execution Result

If the property get succeeds, the return value will be Success. If the pane cannot be found, the return value will be InvalidArgument. If the panes collection is deleted, the return value will be GeneralFailure.

#### Calling Syntax

This example assumes there is a valid Panes collection object to be passed into the example methods.

**[VBA]**

```
Sub Example(Panes As Object)
Dim pane As Object
`Getting Property value
Set pane = Panes.ItemByName("Alarm Pane")
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPanes)
// Getting property value
OBJECT hPane = _ObjectCallMethod(hPanes, "get_ItemByName", "Alarm Pane");
END
```

## IPanes.RemoveAll [Method]

Removes every Pane from this Pane collection.

**Defined As**

- [VBA] RemoveAll()
- [Cicode] RemoveAll()
- [C++] HRESULT RemoveAll()

**Execution Result**

If the method succeeds, the return value will be Success. If the panes collection is deleted, the return value will be GeneralFailure.

**Calling Syntax**

This example assumes there is a valid Panes collection object to be passed into the example methods.

**[VBA]**

```
Sub Panes(Buttons As Object)
Panes.RemoveAll()
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPanes)
_ObjectCallMethod(hPanes, "RemoveAll");
END
```

## IPen Interface

**Methods**

- IPen.AddSample
- IPen.Clear [Method]
- IPen.Delete [Method]
- IPen.GetDefaultSpan [Method]
- IPen.GetHorizontalAxisTimeSpan [Method]
- IPen.GetInformation [Method]
- IPen.GetStatistic [Method]
- IPen.GetVerticalAxisSpan [Method]
- IPen.GoToNow [Method]
- IPen.HorizontalScrollBy [Method]
- IPen.HorizontalZoom [Method]
- IPen.PutHorizontalAxisTimeSpan [Method]
- IPen.PutVerticalAxisSpan [Method]
- IPen.RefreshData [Method]
- IPen.ResetToDefaultSpan [Method]
- IPen.Select [Method]
- IPen.SetDefaultSpan [Method]
- IPen.SetQualityCompactionPointType [Method]
- IPen.SetQualityLineStyle [Method]
- IPen.SetVerticalAxisLabelValue [Method]
- IPen.VerticalScrollBy [Method]
- IPen.VerticalZoom [Method]

## Properties

- IPen.AxisBackgroundColor [Property][Get/Set]
- IPen.BlockRepaint [Property][Get/Set]
- IPen.ClearOnResolutionChange [Property][Get/Set]
- IPen.Collection [Property][Get]
- IPen.DataPoint [Property][Get/Set]
- IPen.DataServer [Property][Get/Set]
- IPen.Height [Property][Get/Set]
- IPen.HorizontalAxisColor [Property][Get/Set]
- IPen.HorizontalAxisResize [Property][Get/Set]
- IPen.HorizontalAxisScroll [Property][Get/Set]
- IPen.HorizontalAxisWidth [Property][Get/Set]
- IPen.HorizontalGridlinesColor [Property][Get/Set]
- IPen.HorizontalGridlinesStyle [Property][Get/Set]
- IPen.HorizontalGridlinesWidth [Property][Get/Set]

- IPen.HorizontalMinorGridlinesColor [Property][Get/Set]
- IPen.HorizontalMinorGridlinesStyle [Property][Get/Set]
- IPen.InstantTrend [Property][Get/Set]
- IPen.IsDeleted [Property][Get]
- IPen.isSelected [Property][Get]
- IPen.LocalTime [Property][Get/Set]
- IPen.Name [Property][Get/Set]
- IPen.PointsVisible [Property][Get/Set]
- IPen.RequestMode [Property][Get/Set]
- IPen.SamplePeriod [Property][Get/Set]
- IPen.Stacked [Property][Get/Set]
- IPen.TrendCursorLabelFillColor [Property][Get/Set]
- IPen.TrendCursorLabelLineColor [Property][Get/Set]
- IPen.TrendCursorLabelTextColor [Property][Get/Set]
- IPen.VerticalAxisAutoscale [Property][Get/Set]
- IPen.VerticalAxisColor [Property][Get/Set]
- IPen.VerticalAxisLabelType [Property][Get/Set]
- IPen.VerticalAxisResize [Property][Get/Set]
- IPen.VerticalAxisScroll [Property][Get/Set]
- IPen.VerticalAxisWidth [Property][Get/Set]
- IPen.VerticalGridlinesColor [Property][Get/Set]
- IPen.VerticalGridlinesStyle [Property][Get/Set]
- IPen.VerticalGridlinesWidth [Property][Get/Set]
- IPen.VerticalMinorGridlinesColor [Property][Get/Set]
- IPen.VerticalMinorGridlinesStyle [Property][Get/Set]
- IPen.Visible [Property][Get/Set]

## IPen.AddSample

Adds a temporary sample to a pen.

**Note:** The pen you are adding samples to needs to have the **Server** field in the **Data Connection** properties set to <Unbound>. See [Configure the Pen Data Connection](#).

### Defined As

- [VBA] AddSample(value As Double, timeStamp as Date, milli as Integer, qualityType as Integer, compactionType as Integer)
- [Cicode] AddSample(REAL value, DATE timeStamp, INT milli, INT qualityType, INT compactionType)
- [C++] HRESULT AddSample(double value, DATE timeStamp, short milli, QualityType qualityType, QualityCompactionType compactionType)

## Parameters

value

[in] Indicates the value of the sample that will be added.

timeStamp

[in] Indicates at what time the sample will occur in UTC time.

milli

[in] Indicates the millisecond component of the time stamp (0 to 999).

qualityType

[in] Indicates the quality of the sample that will be added.

compactionType

[in] Indicates what display type the sample will be represented as.

## Execution Result

If the function succeeds, the return value will be Success. If an argument is bad, the return value will be InvalidArgument. If an argument is out of range, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure. If any other unexpected error occurs, the return value will be GeneralFailure.

## Remarks

You can only add samples to analog or digital pens.

This function has limited use as the samples added are stored in a temporary cache; they can be cleared anytime by time span changes, data refresh calls, or automation. You can change this behavior using [IPen.ClearOnResolutionChange \[Property\]\[Get/Set\]](#).

## Calling Syntax

Assumes you have passed a valid pen object into the function.

## [VBA]

```
Sub Example(pen As Object)
Dim timeStamp As Date
timestamp = Now
pen.AddSample 75.0, timeStamp, 100, 0, 0
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT iCitectTime;
REAL rOleTime;
iCitectTime = TimeCurrent(); // Returns seconds since 1970
rOleTime = TimeToOleDate(iCitectTime, 1); // Convert to OLE UTC time
_ObjectCallMethod(hPen, "AddSample", 75.0, rOleTime, 100, 0, 0);
END
```

## See Also

[QualityType \[Enumeration\]](#)

QualityCompactionType [Enumeration]

## IPen.AxisBackgroundColor [Property][Get/Set]

Gets or sets the background color of the axis of this pen.

### Defined As

- [VBA] Long BackgroundColor
- [Cicode] INT BackgroundColor
- [C++] OLE\_COLOR BackgroundColor

### Execution Result

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument. If the pen is deleted then the return value will be GeneralFailure.

### Remarks

The background is the area underneath the axis lines and values.

To calculate the integer value necessary for a color apply the following formula  $(65536 * \text{Blue}) + (256 * \text{Green}) + (\text{Red})$ . Where Red, Green and Blue are 0-255

### Calling Syntax

Assumes you have passed a valid pen object into the function.

### [VBA]

```
Sub Example(pen As Object)
Dim backgroundColor As Long
`Getting Property value
backgroundColor = pen.AxisBackgroundColor
`Setting Property value to Red
pen.AxisBackgroundColor = 255
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT backgroundColor;
// Getting current property value
backgroundColor = _ObjectGetProperty(hPen, "AxisBackgroundColor");
// Setting Property to Red
_ObjectSetProperty(hPen, "AxisBackgroundColor", PackedRGB(255, 0, 0));
END
```

## IPen.BlockRepaint [Property][Get/Set]

Use this property to halt or continue any drawing updates to this pen.

**Defined As**

- [VBA] Boolean BlockRepaint
- [Cicode] INT BlockRepaint
- [C++] VARIANT\_BOOL BlockRepaint

**Execution Result**

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument. If the pen is deleted then the return value will be GeneralFailure.

**Remarks**

This property is useful if you are modifying several properties at once as it will help reduce flicker and the amount of processing necessary. Simply set the property to True (-1), change as many properties as you want, and then set the property to False (0).

**Calling Syntax**

Assumes you have passed a valid pen object into the function.

**[VBA]**

```
Sub Example(pen As Object)
Dim blockRepaint As Boolean
`Getting Property value
blockRepaint = pen.BlockRepaint
`Setting Property value
pen.BlockRepaint = True
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPen)
INT bBlockRepaint;
// Getting current property value
bBlockRepaint = _ObjectGetProperty(hPen, "BlockRepaint");
// Setting Property
_ObjectSetProperty(hPen, "BlockRepaint", -1);
END
```

## IPen.Clear [Method]

Clears every sample belonging to this pen from the internal cache. (Note: This does not remove logged samples from the server)

**Defined As**

- [VBA] Clear()
- [Cicode] Clear()
- [C++] HRESULT Clear()

**Execution Result**

If the function succeeds the return value will be Success. If the pen is deleted then the return value will be GeneralFailure.

**Calling Syntax**

Assumes you have passed a valid pen object into the function.

**[VBA]**

```
Sub Example(pen As Object)
pen.Clear
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPen)
    _ObjectCallMethod(hPen, "Clear");
END
```

## IPen.ClearOnResolutionChange [Property][Get/ Set]

Controls whether the Process Analyst Control will automatically clear the data cache in response to human interaction methods that result in a data resolution change, for example, when the user manipulates the horizontal axis scaling. In normal operation this property is set to TRUE by default. Turning the property to FALSE will result in the action not clearing the data cache. This property can only be changed when the DataPoint property is not bound (that is, Empty). Once the cache is full it will roll samples off the back as new samples arrive. This property is useful when the pen is unbound and the AddSample method is used.

**Defined As**

- [VBA] Boolean ClearOnResolutionChange
- [Cicode] INT ClearOnResolutionChange
- [C++] VARIANT\_BOOL ClearOnResolutionChange

**Execution Result**

If the property get/set succeeds the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be AccessDenied.

**Limits**

- True (-1): User action clears the data cache
- False (0): User action does not clear the data cache

**Calling Syntax**

Assumes you have passed a valid pen object into the function.

**[VBA]**

```
Sub Example(pen As Object)
Dim ResoChange As Boolean
`Getting Property value
ResoChange = pen.ClearOnResolutionChange
`Setting Property value
pen.ClearOnResolutionChange = False
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPen)
INT bResoChange;
// Getting current property value
bResoChange = _ObjectGetProperty(hPen, "ClearOnResolutionChange");
// Setting Property
_ObjectSetProperty(hPen, "ClearOnResolutionChange", 0);
END
```

## IPen.Collection [Property][Get]

Returns a reference to the Pens collection object that this pen belongs to.

**Defined As**

- [VBA] Object Collection
- [Cicode] OBJECT Collection
- [C++] IPen\* Collection

**Execution Result**

If the property get succeeds the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

**Calling Syntax**

Assumes you have passed a valid pen object into the function.

**[VBA]**

```
Sub Example(pen As Object)
Dim pens As Object
`Getting Property value
Set pens = pen.Collection
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPen)
OBJECT pens;
```

```
// Getting current property value
pens = _ObjectGetProperty(hPen, "Collection");
END
```

## IPen.DataPoint [Property][Get/Set]

Get or Set the trend/alarm tag which this pen is bound to.

### Defined As

- [VBA] String DataPoint
- [Cicode] STRING DataPoint
- [C++] BSTR DataPoint

### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the tag is greater than 79 characters, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

### Remarks

This property works in conjunction with the DataServer property. This property can be changed during the lifetime of the pen. Changing the DataPoint property will result in the data cache being cleared and a new data request issued. A request for the tag's information will also be issued.

### Calling Syntax

Assumes you have passed a valid pen object into the function.

### [VBA]

```
Sub Example(pen As Object)
Dim tag As String
`Getting Property value
tag = pen.DataPoint
`Setting Property value
pen.DataPoint = "LOOP_1_PV"
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPen)
STRING tag;
// Getting current property value
tag = _ObjectGetProperty(hPen, "DataPoint");
// Setting Property
_ObjectSetProperty(hPen, "DataPoint", "LOOP_1_PV");
END
```

## See Also

[IPen.DataServer \[Property\]\[Get/Set\]](#)

# IPen.DataServer [Property][Get/Set]

Get or Set the server that this pen is bound to.

### Defined As

- [VBA] String DataServer
- [Cicode] STRING DataServer
- [C++] BSTR DataServer

### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the server connection cannot be found, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

### Remarks

This property currently only supports two options, "localhost" and "" (empty string), which indicates an unbound connection. Local host means the pen will use the local Plant SCADA client to source data from the Plant SCADA Trends/Alarm Servers.

This property works in conjunction with the DataPoint property. This property can be changed during the lifetime of the pen.

### Calling Syntax

Assumes you have passed a valid pen object into the function.

## [VBA]

```
Sub Example(pen As Object)
Dim server As String
`Getting Property value
server = pen.DataServer
`Setting Property value
pen.DataPoint = "localhost"
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPen)
STRING server;
// Getting current property value
server = _ObjectGetProperty(hPen, "DataServer");
// Setting Property
_ObjectSetProperty(hPen, "DataServer", "localhost");
END
```

## See Also

[IPen.DataPoint \[Property\]\[Get/Set\]](#)

# IPen.Delete [Method]

Deletes the pen from the Process Analyst.

### Defined As

- [VBA] Delete()
- [Cicode] Delete()
- [C++] HRESULT Delete()

### Execution Result

If the function succeeds, the return value will be Success. If the pen is already deleted, the return value will be GeneralFailure.

### Remarks

Calling this method will mark the pen for deletion, meaning any further calls to methods or properties on the pen will result in a GeneralFailure error. The pen will be removed from the display immediately after making this call.

### Calling Syntax

Assumes you have passed a valid pen object into the function.

## [VBA]

```
Sub Example(pen As Object)
pen.Delete
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPen)
_ObjectCallMethod(hPen, "Delete");
END
```

## See Also

[IPen.IsDeleted \[Property\]\[Get\]](#)

# IPen.GetDefaultSpan [Method]

Returns the default time span for this pen as a series of time components.

### Defined As

- [VBA] GetDefaultSpan(weeks As Integer, days As Integer, hours As Integer, minutes As Integer, seconds As Integer, milliseconds As Integer)
- [Cicode] GetDefaultSpan (INT weeks, INT days, DATE hours, INT minutes, INT seconds, INT milliseconds)
- [C++] HRESULT GetDefaultSpan (short\* weeks, short\* days, short\* hours, short\* minutes, short\* seconds, short\* milliseconds)

### Parameters

weeks

[out] Indicates the number of weeks in the span.

days

[out] Indicates the number of days in the span.

hours

[out] Indicates the number of hours in the span.

minutes

[out] Indicates the number of minutes in the span.

seconds

[out] Indicates the number of seconds in the span.

milliseconds

[out] Indicates the number of milliseconds in the span.

### Execution Result

If the function succeeds, the return value will be Success. If an argument is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure. If any other unexpected error occurs, the return value will be GeneralFailure.

### Calling Syntax

Assumes you have passed a valid pen object into the function.

## [VBA]

```
Sub Example(pen As Object)
Dim weeks As Integer
Dim days As Integer
Dim hours As Integer
Dim minutes As Integer
Dim seconds As Integer
Dim milliseconds As Integer
pen.GetDefaultSpan weeks, days, hours, minutes, seconds, milliseconds
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT weeks;
INT days;
INT hours;
INT minutes;
```

```
INT seconds;
INT milliseconds;
_ObjectCallMethod(hPen, "GetDefaultSpan", weeks, days, hours, minutes,
seconds, milliseconds);
END
```

## See Also

[IPen.SetDefaultSpan \[Method\]](#)  
[IPen.ResetToDefaultSpan \[Method\]](#)

# IPen.GetHorizontalAxisTimeSpan [Method]

Returns the start and end time of this pen in local or UTC time format.

### Defined As

- [VBA] GetHorizontalAxisTimeSpan(startTime As Date, startMs as Integer, endTime as Date, endMs as Integer, localTime as Boolean)
- [Cicode] GetHorizontalAxisTimeSpan (REAL startTime, INT startMs, REAL endTime, INT endMs, INT localTime)
- [C++] HRESULT GetHorizontalAxisTimeSpan (DATE\* startTime, short\* startMs, DATE\* endTime, short\* endMs, VARIANT\_BOOL localTime)

### Parameters

startTime

[out] This will contain the beginning date and time without milliseconds of the time span.

startMs

[out] This will contain the milliseconds component of the start time.

endTime

[out] This will contain the end date and time without milliseconds.of the time span.

endMs

[out] This will contain the milliseconds component of the end time.

localTime

[in] Indicates whether the times returned are in local time or UTC. True = -1, False (0) = UTC.

### Execution Result

If the function succeeds, the return value will be Success. If an argument is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure. If any other unexpected error occurs, the return value will be GeneralFailure.

### Calling Syntax

Assumes you have passed a valid pen object into the function.

## [VBA]

```
Sub Example(pen As Object)
```

```
Dim startDate As Date
Dim endDate As Date
Dim startMs As Integer
Dim endMs As Integer
pen.GetHorizontalAxisTimeSpan startDate, startMs, endDate, endMs, True
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPen)
REAL startDate;
REAL endDate;
INT startMs;
INT endMs;
_ObjectCallMethod(hPen, "GetHorizontalAxisTimeSpan", startDate, startMs,
endDate, endMs, -1);
END
```

## See Also

[IPen.PutHorizontalAxisTimeSpan \[Method\]](#)

# IPen.GetInformation [Method]

Returns information associated with this pen.

### Defined As

- [VBA] GetInformation(name As String) As String
- [Cicode] STRING GetInformation(STRING name)
- [C++] HRESULT GetDefaultSpan (BSTR name, BSTR\* value)

### Parameters

name

[in] Specify the pen information attribute you want to get the value for. See Remarks below for supported attributes.

value

[out] Indicates the value of the specified information attribute.

### Execution Result

If the function succeeds, the return value will be Success. If an argument is bad, the return value will be InvalidArgument. If the attribute does not exist, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure. If any other unexpected error occurs, the return value will be GeneralFailure.

### Information Attributes

Attribute	Returns	Applies to
Alarm Area	Alarm tag field	Alarm
Alarm Category	Alarm tag field	Alarm
Alarm Desc	Alarm tag field	Alarm
Alarm Name	Alarm tag field	Alarm
Alarm Type	Alarm tag field	Alarm
Comment	Alarm/Trend tag comment field	All
Duration	Process Analyst time span	All
End Time	Process Analyst axis end time	All
Engineering Full Scale	Trend tag field	Analog, Digital
Engineering Units	Trend tag field	Analog, Digital
Engineering Zero Scale	Trend tag field	Analog, Digital
Error	Process Analyst error status	All
Full Scale	Process Analyst vertical axis max scale	Analog,
Name	Process Analyst pen name	All
Raw Full Scale	Trend tag field	Analog, Digital
Raw Zero Scale	Trend tag field	Analog, Digital
Sample Period	Trend tag field	Analog, Digital
Start Time	Process Analyst axis start time	All
Tag	Process Analyst source binding field	All
Trend Type	Trend tag field	Analog, Digital
Zero Scale	Process Analyst vertical axis min scale	Analog
Scale	Process Analyst vertical axis scale range	Analog, Digital
Engineering Scale	Engineering scale range	Analog, Digital

**Calling Syntax**

Assumes you have passed a valid pen object into the function.

**[VBA]**

```
Sub Example(pen As Object)
Dim duration As String
duration = pen.GetInformation "Duration"
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPen)
STRING duration;
duration = _ObjectCallMethod(hPen, "GetInformation", "Duration");
END
```

## IPen.GetStatistic [Method]

Returns the result of a specified Process Analyst statistical operation.

**Defined As**

- [VBA] GetStatistic(name As String, value As String)
- [Cicode] GetStatistic(STRING name, STRING value)
- [C++] HRESULT GetStatistic(BSTR name, BSTR\* value)

**Parameters**

name

[in] Specify the statistic attribute you want to get the value for. See Remarks below for supported attributes.

value

[out] Indicates the value of the specified statistic attribute.

**Execution Result**

If the function succeeds, the return value will be Success. If an argument is bad, the return value will be InvalidArgument. If the attribute does not exist, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure. If any other unexpected error occurs, the return value will be GeneralFailure.

**Information Attributes**

Attribute	Returns	Applies to
Average	Process Analyst real-time average	Analog, Digital
Maximum	Process Analyst real-time maximum	Analog
Minimum	Process Analyst real-time minimum	Analog

**Calling Syntax**

Assumes you have passed a valid pen object into the function.

**[VBA]**

```
Sub Example(pen As Object)
Dim average As String
pen.GetStatistic "Average", average
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPen)
STRING average;
average = _ObjectCallMethod(hPen, "GetStatistic", "Average");
END
```

## IPen.GetVerticalAxisSpan [Method]

Returns the current span of the pen's vertical axis.

**Defined As**

- [VBA] GetVerticalAxisSpan(startValue As Double, endValue As Double)
- [Cicode] GetVerticalAxisSpan (REAL startValue, REAL endValue)
- [C++] HRESULT GetVerticalAxisSpan (double\* startValue, double\* endValue)

**Parameters**

startValue

[out] The current lower bound of the vertical axis.

endValue

[out] The current upper bound of the vertical axis.

**Execution Result**

If the function succeeds, the return value will be Success. If an argument is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure. If any other unexpected error occurs, the return value will be GeneralFailure.

**Calling Syntax**

Assumes you have passed a valid pen object into the function.

**[VBA]**

```
Sub Example(pen As Object)
Dim startValue As Double
Dim endValue As Double
pen.GetVerticalAxisSpan startValue, endValue
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPen)
REAL startValue;
REAL endValue;
_ObjectCallMethod(hPen, "GetVerticalAxisSpan", startValue, endValue);
END
```

**See Also**

[IPen.PutVerticalAxisSpan \[Method\]](#)

## IPen.GoToNow [Method]

Synchronizes the end time of the pen's span with your computer's current local time. The start time will also be moved to maintain the pen's current time span.

**Defined As**

- [VBA] GoToNow()
- [Cicode] GoToNow()
- [C++] HRESULT GoToNow()

**Execution Result**

If the function succeeds, the return value will be Success. If the pen is deleted, the return value will be GeneralFailure.

**Calling Syntax**

Assumes you have passed a valid pen object into the function.

**[VBA]**

```
Sub Example(pen As Object)
pen.GoToNow
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPen)
_ObjectCallMethod(hPen, "GoToNow");
END
```

## IPen.Height [Property][Get/Set]

Get or Set the physical height in pixels that the pen will allocate for itself when displayed in Stacked mode.

**Defined As**

- [VBA] Integer Height
- [Cicode] INT Height
- [C++] double Height

#### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If height set is out of range (16 - 1000), the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

#### Remarks

This property is ignored when the pen is not in Stacked mode.

#### Calling Syntax

Assumes you have passed a valid pen object into the function.

### [VBA]

```
Sub Example(pen As Object)
Dim height As Boolean
`Getting Property value
height = pen.Height
`Setting Property value
pen.Height = 75
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT height;
// Getting current property value
height = _ObjectGetProperty(hPen, "Height");
// Setting Property
_ObjectSetProperty(hPen, "Height", 75);
END
```

### See Also

[IPen.Stacked \[Property\]\[Get/Set\]](#)

## IPen.HorizontalAxisColor [Property][Get/Set]

Gets or sets the color used to draw the line, labels, and interval markers of the horizontal axis of this pen.

#### Defined As

- [VBA] Long HorizontalAxisColor
- [Cicode] INT HorizontalAxisColor
- [C++] OLE\_COLOR HorizontalAxisColor

### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

### Remarks

To calculate the integer value necessary for a color apply the following formula:

$$(65536 * \text{Blue}) + (256 * \text{Green}) + (\text{Red})$$

where Red, Green, and Blue are 0-255.

### Calling Syntax

Assumes you have passed a valid pen object into the function.

## [VBA]

```
Sub Example(pen As Object)
Dim color As Long
`Getting Property value
color = pen.HorizontalAxisColor
`Setting Property value to Red
pen.HorizontalAxisColor = 255
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT color;
// Getting current property value
color = _ObjectGetProperty(hPen, "HorizontalAxisColor");
// Setting Property to Red
_ObjectSetProperty(hPen, "HorizontalAxisColor", PackedRGB(255, 0, 0));
END
```

## See Also

[IPen.VerticalAxisColor \[Property\]\[Get/Set\]](#)

# IPen.HorizontalAxisResize [Property][Get/Set]

Gets or sets whether this pen allows the operator to interactively scale the horizontal axis using the mouse.

### Defined As

- [VBA] Boolean HorizontalAxisResize
- [Cicode] INT HorizontalAxisResize
- [C++] VARIANT\_BOOL HorizontalAxisResize

### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value

will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

#### Limits

- True (-1): Axis can be re-sized
- False (0): Axis cannot be re-sized

#### Calling Syntax

Assumes you have passed a valid pen object into the function.

### [VBA]

```
Sub Example(pen As Object)
Dim re-size As Boolean
`Getting Property value
re-size = pen.HorizontalAxisResize
`Setting Property value
pen.HorizontalAxisResize = False
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT bResize;
// Getting current property value
bResize = _ObjectGetProperty(hPen, "HorizontalAxisResize");
// Setting Property
_ObjectSetProperty(hPen, "HorizontalAxisResize",0);
END
```

### See Also

[IPen.VerticalAxisResize \[Property\]\[Get/Set\]](#)

## IPen.HorizontalAxisScroll [Property][Get/Set]

Gets or sets whether this pen allows the operator to interactively scroll the horizontal axis using the mouse.

#### Defined As

- [VBA] Boolean HorizontalAxisScroll
- [Cicode] INT HorizontalAxisScroll
- [C++] VARIANT\_BOOL HorizontalAxisScroll

#### Execution Result

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument. If the pen is deleted then the return value will be GeneralFailure.

#### Limits

- True (-1): Axis can be scrolled
- False (0): Axis cannot be scrolled

#### Calling Syntax

Assumes you have passed a valid pen object into the function.

#### [VBA]

```
Sub Example(pen As Object)
Dim scroll As Boolean
`Getting Property value
scroll = pen.HorizontalAxisScroll
`Setting Property value
pen.HorizontalAxisScroll = False
End Sub
```

#### [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT bScroll;
// Getting current property value
bScroll = _ObjectGetProperty(hPen, "HorizontalAxisScroll");
// Setting Property
_ObjectSetProperty(hPen, "HorizontalAxisScroll", 0);
END
```

#### See Also

[IPen.VerticalAxisScroll \[Property\]\[Get/Set\]](#)

## IPen.HorizontalAxisWidth [Property][Get/Set]

Gets or sets the width of the horizontal axis line and the associated interval markers.

#### Defined As

- [VBA] Integer HorizontalAxisWidth
- [Cicode] INT HorizontalAxisWidth
- [C++] short HorizontalAxisWidth

#### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

#### Limits

A valid width is 0-8 pixels.

#### Calling Syntax

Assumes you have passed a valid pen object into the function.

**[VBA]**

```
Sub Example(pen As Object)
Dim width As Integer
`Getting Property value
width = pen.HorizontalAxisWidth
`Setting Property value
pen.HorizontalAxisWidth = 3
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPen)
INT width;
// Getting current property value
width = _ObjectGetProperty(hPen, "HorizontalAxisWidth");
// Setting Property
_ObjectSetProperty(hPen, "HorizontalAxisWidth", 3);
END
```

**See Also**

[IPen.VerticalAxisWidth \[Property\]\[Get/Set\]](#)

## IPen.HorizontalGridlinesColor [Property][Get/Set]

Gets or sets the color used to draw the major horizontal gridlines.

**Defined As**

- [VBA] Long HorizontalGridlinesColor
- [Cicode] INT HorizontalGridlinesColor
- [C++] OLE\_COLOR HorizontalGridlinesColor

**Execution Result**

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

**Limits**

A valid width is 0-8 pixels.

**Remarks**

To calculate the integer value necessary for a color apply the following formula  $(65536 * \text{Blue}) + (256 * \text{Green}) + (\text{Red})$ . Where Red, Green and Blue are 0-255

**Calling Syntax**

Assumes you have passed a valid pen object into the function.

**[VBA]**

```
Sub Example(pen As Object)
Dim color As Long
`Getting Property value
color = pen.HorizontalGridlinesColor
`Setting Property value to Red
pen.HorizontalGridlinesColor = 255
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPen)
INT color;
// Getting current property value
color = _ObjectGetProperty(hPen, "HorizontalGridlinesColor");
// Setting Property to Red
_ObjectSetProperty(hPen, "HorizontalGridlinesColor", PackedRGB(255, 0, 0));
END
```

**See Also**

[IPen.HorizontalMinorGridlinesColor \[Property\]\[Get/Set\]](#)

## IPen.HorizontalGridlinesStyle [Property][Get/Set]

Gets or sets the line style used to draw the major horizontal gridlines.

**Defined As**

- [VBA] Long HorizontalGridlinesStyle
- [Cicode] INT HorizontalGridlinesStyle
- [C++] LineStyle HorizontalGridlinesStyle

**Execution Result**

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the style is out of range, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

**Calling Syntax**

Assumes you have passed a valid pen object into the function.

**[VBA]**

```
Sub Example(pen As Object)
Dim style As Long
`Getting Property value
style = pen.HorizontalGridlinesStyle
`Setting Property value to Dot
```

```
pen.HorizontalGridlinesColor = 2
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT style;
// Getting current property value
style = _ObjectGetProperty(hPen, "HorizontalGridlinesStyle");
// Setting Property to Dot
_ObjectSetProperty(hPen, "HorizontalGridlinesStyle", 2);
END
```

## See Also

[IPen.HorizontalMinorGridlinesColor \[Property\]\[Get/Set\]](#)

# IPen.HorizontalGridlinesWidth [Property][Get/ Set]

Gets or sets the line width used when drawing the major horizontal gridlines.

### Defined As

- [VBA] Integer HorizontalGridlinesWidth
- [Cicode] INT HorizontalGridlinesWidth
- [C++] short HorizontalGridlinesWidth

### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the width is out of range, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

### Calling Syntax

Assumes you have passed a valid pen object into the function.

## [VBA]

```
Sub Example(pen As Object)
Dim width As Integer
`Getting Property value
width = pen.HorizontalGridlinesWidth
`Setting Property value
pen.HorizontalGridlinesWidth = 3
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPen)
INT width;
// Getting current property value
width = _ObjectGetProperty(hPen, "HorizontalGridlinesWidth");
// Setting Property to
_ObjectSetProperty(hPen, "HorizontalGridlinesWidth", 3);
END
```

## IPen.HorizontalMinorGridlinesColor [Property][Get/Set]

Gets or sets the color used to draw the minor horizontal gridlines.

**Defined As**

- [VBA] Long HorizontalMinorGridlinesColor
- [Cicode] INT HorizontalMinorGridlinesColor
- [C++] OLE\_COLOR HorizontalMinorGridlinesColor

**Execution Result**

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

**Remarks**

To calculate the integer value necessary for a color apply the following formula:

$$(65536 * \text{Blue}) + (256 * \text{Green}) + (\text{Red})$$

where Red, Green, and Blue are 0-255.

**Calling Syntax**

Assumes you have passed a valid pen object into the function.

**[VBA]**

```
Sub Example(pen As Object)
Dim color As Long
`Getting Property value
color = pen.HorizontalMinorGridlinesColor
`Setting Property value to Red
pen.HorizontalMinorGridlinesColor = 255
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPen)
INT color;
// Getting current property value
```

```
color = _ObjectGetProperty(hPen, "HorizontalMinorGridlinesColor");
// Setting Property to Red
_ObjectSetProperty(hPen, "HorizontalMinorGridlinesColor",
PackedRGB(255, 0, 0));
END
```

## See Also

[IPen.HorizontalGridlinesColor \[Property\]\[Get/Set\]](#)

# IPen.HorizontalMinorGridlinesStyle [Property][Get/Set]

Gets or sets the line style used to draw the minor horizontal gridlines.

### Defined As

- [VBA] Long HorizontalMinorGridlinesStyle
- [Cicode] INT HorizontalMinorGridlinesStyle
- [C++] LineStyle HorizontalMinorGridlinesStyle

### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the style is out of range, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

### Calling Syntax

Assumes you have passed a valid pen object into the function.

## [VBA]

```
Sub Example(pen As Object)
Dim style As Long
`Getting Property value
style = pen.HorizontalMinorGridlinesStyle
`Setting Property value to Dot
pen.HorizontalMinorGridlinesColor = 2
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT style;
// Getting current property value
style = _ObjectGetProperty(hPen, "HorizontalMinorGridlinesStyle");
// Setting Property to Dot
_ObjectSetProperty(hPen, "HorizontalMinorGridlinesStyle", 2);
END
```

## See Also

[IPen.HorizontalGridlinesStyle \[Property\]\[Get/Set\]](#)  
[LineStyle \[Enumeration\]](#)

# IPen.HorizontalScrollBy [Method]

Scrolls the horizontal axis by the specified factor.

### Defined As

- [VBA] HorizontalScrollBy(factor As Double)
- [Cicode] HorizontalScrollBy(REAL factor)
- [C++] HRESULT HorizontalScrollBy(double factor)

### Parameters

factor

[in] Controls the direction and amount the axis will be scrolled. A negative value will move the axis back in time; a positive value will move the axis forward in time. The value is a percentage representing the current viewable span. So if the pen span is 1 hour, and you specify a factor of 0.5, you will move the time span 30 minutes into the future.

### Execution Result

If the function succeeds, the return value will be Success. If the argument is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

### Calling Syntax

Assumes you have passed a valid pen object into the function.

## [VBA]

```
Sub Example(pen As Object)
` Move the pen span back one complete span into history
pen.HorizontalScrollBy -1.0
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPen)
// Move the pen span back one complete span into history
_ObjectCallMethod(hPen, "HorizontalScrollby", -1.0);
END
```

## See Also

[IPen.VerticalScrollBy \[Method\]](#)

# IPen.HorizontalZoom [Method]

Zooms centrally into the time span by the given factor.

## Defined As

- [VBA] HorizontalZoom(factor As Double)
- [Cicode] HorizontalZoom(REAL factor)
- [C++] HRESULT HorizontalZoom(double factor)

## Parameters

factor

[in] Controls the direction and amount the axis will be zoomed. Acceptable zoom values are 0 to 1 (Zoom out) and > 1 (zoom in).

## Execution Result

If the function succeeds the return value will be Success. If the argument is bad then the return value will be InvalidArgument. If the argument is out of range then the return value will be InvalidArgument. If the pen is deleted then the return value will be GeneralFailure.

## Calling Syntax

Assumes you have passed a valid pen object into the function.

## [VBA]

```
Sub Example(pen As Object)
` Zoom out 50%
pen.HorizontalZoom 0.5
` Undo the Zoom
pen.HorizontalZoom 1.5
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPen)
// Zoom out 50%
_ObjectCallMethod(hPen, "HorizontalZoom", 0.5);
// Undo the Zoom
_ObjectCallMethod(hPen, "HorizontalZoom", 2.0);
END
```

## See Also

[IPen.VerticalZoom \[Method\]](#)

## IPen.InstantTrend [Property][Get/Set]

Get or set whether pen data is to be retrieved for an instant trend. If set to True, the pen's DataPoint property needs to specify the name of a Variable Tag or a Local Variable.

### Defined As

- [VBA] Boolean InstantTrend
- [Cicode] INT InstantTrend
- [C++] VARIANT\_BOOL InstantTrend

### Allowable values

True(-1): Data will be retrieved for an instant trend

False(0): Data will be retrieved as for a normal trend or alarm

### See Also

[IPen.SamplePeriod \[Property\]\[Get/Set\]](#)

## IPen.IsDeleted [Property][Get]

Returns whether this pen has been marked for deletion. That is, whether someone has called the Delete method on it or deleted it from the display.

### Defined As

- [VBA] Boolean IsDeleted
- [Cicode] INT IsDeleted
- [C++] VARIANT\_BOOL IsDeleted

### Execution Result

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

### Calling Syntax

Assumes you have passed a valid pen object into the function.

### [VBA]

```
Sub Example(pen As Object)
Dim deleted As Boolean
deleted = pen.IsDeleted
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPen)
INT bDeleted;
bDeleted = _ObjectGetProperty(hPen, "IsDeleted");
END
```

**See Also**

[IPen.Delete \[Method\]](#)

## IPen.isSelected [Property][Get]

Returns whether this pen has been selected in the Process Analyst.

**Defined As**

- [VBA] Boolean isSelected
- [Cicode] INT isSelected
- [C++] VARIANT\_BOOL isSelected

**Execution Result**

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

**Calling Syntax**

Assumes you have passed a valid pen object into the function.

**[VBA]**

```
Sub Example(pen As Object)
Dim selected As Boolean
selected = pen.isSelected
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPen)
INT bSelected;
bSelected = _ObjectGetProperty(hPen, "isSelected");
END
```

**See Also**

[IPen.Select \[Method\]](#)

# IPen.LocalTime [Property][Get/Set]

Get or Set whether the axis will display time in the computers current local format or in UTC (Universal Time Coordinate).

## Defined As

- [VBA] Boolean LocalTime
- [Cicode] INT LocalTime
- [C++] VARIANT\_BOOL LocalTime

## Execution Result

If the property get/set succeeds the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

## Limits

- True (-1): Local format
- False (0): UTC format

## Calling Syntax

Assumes you have passed a valid pen object into the function.

### [VBA]

```
Sub Example(pen As Object)
Dim localTime As Boolean
`Getting Property value
localTime = pen.LocalTime
`Display time in UTC
pen.LocalTime = False
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT bLocalTime;
// Getting current property value
bLocalTime = _ObjectGetProperty(hPen, "LocalTime");
// Display time in UTC
_ObjectSetProperty(hPen, "LocalTime", 0);
END
```

# IPen.Name [Property][Get/Set]

Get or Set the name of this pen.

## Defined As

- [VBA] String Name
- [Cicode] STRING Name
- [C++] BSTR Name

#### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the length of the name is wrong then the return value will be InvalidArgument. If the pen is deleted then the return value will be GeneralFailure.

#### Remarks

The Process Analyst will use this name extensively throughout the user interface to reference this pen.

The name of the pen does not have to be unique, but it needs to be between 1 and 250 character long.

#### Calling Syntax

Assumes you have passed a valid pen object into the function.

### [VBA]

```
Sub Example(pen As Object)
Dim name As String
`Getting Property value
name = pen.Name
`Setting property value
pen.Name = "NicePen"
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPen)
STRING name;
// Getting current property value
name = _ObjectGetProperty(hPen, "Name");
// Setting property value
_ObjectSetProperty(hPen, "Name", "NicePen");
END
```

## IPen.PointsVisible [Property][Get/Set]

Gets or Sets whether the sample points are displayed or hidden on the pen.

#### Defined As

- [VBA] Boolean PointsVisible
- [Cicode] INT PointsVisible
- [C++] VARIANT\_BOOL PointsVisible

#### Execution Results

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value

will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

#### Limits

- True(-1): Points are visible
- False(0): Points are hidden

#### Remarks

By default this property is False, meaning that any point type you have set using the SetQualityCompactionPointType function will be hidden.

#### Calling Syntax

Assumes you have passed a valid pen object into the function.

### [VBA]

```
Sub Example(pen As Object)
Dim visible As Boolean
`Getting Property value
visible = pen.PointsVisible
`Setting Property value
pen.PointsVisible = True
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT visible;
// Getting current property value
visible = _ObjectGetProperty(hPen, "PointsVisible");
// Setting Property value
_ObjectSetProperty(hPen, "PointsVisible", -1);
END
```

### See Also

[IPen.SetQualityCompactionPointType \[Method\]](#)

## IPen.PutHorizontalAxisTimeSpan [Method]

Sets the start and end time of this pen.

#### Defined As

- [VBA] PutHorizontalAxisTimeSpan(startTime As Date, startMs as Integer, endTime as Date, endMs as Integer)
- [Cicode] PutHorizontalAxisTimeSpan (REAL startTime, INT startMs, REAL endTime, INT endMs)
- [C++] HRESULT PutHorizontalAxisTimeSpan (DATE\* startTime, short\* startMs, DATE\* endTime, short\* endMs)

## Parameters

### startTime

[in] Indicates the beginning date and time without milliseconds of the time span in UTC format.

### startMs

[in] Indicates the milliseconds component of the start time.

### endTime

[in] Indicates the end date and time without milliseconds of the time span in UTC format.

### endMs

[in] This will contain the milliseconds component of the end time.

## Execution Result

If the function succeeds the return value will be Success. If an argument is bad then the return value will be InvalidArgument. If an argument is out of range then the return value will be InvalidArgument. If the pen is deleted then the return value will be GeneralFailure. If any other unexpected error occurs the return value will be GeneralFailure.

## Remarks

The Process Analyst only supports setting its axis in UTC (Universal Co-ordinated Time) format. This means you need to convert from local to UTC format yourself to make the axis display correctly in local time. Cicode provides several functions to do these conversions.

## Limits

The horizontal axis has an upper limit of 1/1/2100 12:00:00.000 and a lower limit of 1/1/1900 12:00:00.000. The minimum span is 100 milliseconds. The maximum span is 200 years.

## Calling Syntax

Assumes you have passed a valid pen object into the function.

## [VBA]

```
Sub Example(pen As Object)
Dim startDate As Date
Dim endDate As Date
Dim startMs As Integer
Dim endMs As Integer
startDate = CDate("16/6/2004 11:30:00")
endDate = CDate("16/6/2004 12:29:00")
startMs = 0
endMs = 0
pen.PutHorizontalAxisTimeSpan startDate, startMs, endDate, endMs
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPen)
REAL startDate;
REAL endDate;
startDate = StrToDate("16/6/04") + StrToTime("9:30:00");
endDate = StrToDate("16/6/04") + StrToTime("10:29:00");
startDate = TimeToOLEDate(startDate, 0); // Convert to UTC
```

```
endDate = TimeToOLEDate(endDate, 0); // Convert to UTC
_ObjectcallMethod(hPen, "PutHorizontalAxisTimeSpan", startDate, 0, endDate, 0);
END
```

## See Also

[IPen.GetHorizontalAxisSpan \[Method\]](#)

# IPen.PutVerticalAxisSpan [Method]

Sets the current position and span of the pens' vertical axis.

### Defined As

- [VBA] GetVerticalAxisSpan(startValue As Double, endValue As Double)
- [Cicode] GetVerticalAxisSpan (REAL startValue, REAL endValue)
- [C++] HRESULT GetVerticalAxisSpan (double\* startValue, double\* endValue)

### Parameters

startValue

[in] Indicates the new lower bound of the vertical axis.

endValue

[in] Indicates the new upper bound of the vertical axis.

### Execution Result

If the function succeeds, the return value will be Success. If an argument is bad, the return value will be InvalidArgument. If an argument is out of range, or the span is out of range, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

### Limits

The vertical axis has a upper limit of 1+e10 and a lower limit of 1-e10. However, the maximum span supported is 1+e10. The minimum span is 0.00001.

### Calling Syntax

Assumes you have passed a valid pen object into the function.

## [VBA]

```
Sub Example(pen As Object)
pen.PutVerticalAxisSpan 200.5, 300.34
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPen)
_ObjectCallMethod(hPen, "PutVerticalAxisSpan", 200.5, 300.34);
END
```

## See Also

[IPen.GetVerticalAxisSpan \[Method\]](#)

# IPen.RefreshData [Method]

Clears every sample belonging to this pen from the internal cache and then issues a new request for data.

### Defined As

- [VBA] RefreshData()
- [Cicode] RefreshData ()
- [C++] HRESULT RefreshData ()

### Execution Result

If the function succeeds the return value will be Success. If the pen is deleted then the return value will be GeneralFailure.

### Calling Syntax

Assumes you have passed a valid pen object into the function.

## [VBA]

```
Sub Example(pen As Object)
pen.RefreshData
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPen)
    _ObjectCallMethod(hPen, "RefreshData");
END
```

# IPen.RequestMode [Property][Get/Set]

Get or Set how multiple samples will be calculated on the server.

### Defined As

- [VBA] Integer RequestMode
- [Cicode] INT RequestMode
- [C++] RequestMode RequestMode

### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the mode is out of range, the return value will be InvalidArgument. If the pen is

deleted, the return value will be GeneralFailure.

#### Remarks

When the pen makes a request for data and samples need to be compacted, it will use this mode to determine how the compaction will occur.

Changing this mode will clear the data cache and issue a new request for data.

#### Calling Syntax

Assumes you have passed a valid pen object into the function.

### [VBA]

```
Sub Example(pen As Object)
Dim requestMode As Integer
`Getting Property value
requestMode = pen.RequestMode
`Setting mode to minimum
pen.RequestMode = 1
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT requestMode;
// Getting current property value
requestMode = _ObjectGetProperty(hPen, "RequestMode");
// Setting mode to minimum
_ObjectSetProperty(hPen, "RequestMode", 1);
END
```

### See Also

[RequestMethod \[Enumeration\]](#)

## IPen.ResetToDefaultSpan [Method]

Resets the span of this pen to its default span.

#### Defined As

- [VBA] ResetToDefaultSpan()
- [Cicode] ResetToDefaultSpan()
- [C++] HRESULT ResetToDefaultSpan()

#### Execution Result

If the function succeeds, the return value will be Success. If the pen is deleted, the return value will be GeneralFailure.

#### Remarks

The default span of pens is 10 minutes. This can be modified by using IPen.SetDefaultSpan.

#### Calling Syntax

Assumes you have passed a valid pen object into the function.

#### [VBA]

```
Sub Example(pen As Object)
pen.ResetToDefaultSpan
End Sub
```

#### [Cicode]

```
FUNCTION Example(OBJECT hPen)
_ObjectCallMethod(hPen, "ResetToDefaultSpan");
END
```

#### See Also

[IPen.GetDefaultSpan \[Method\]](#)  
[IPen.SetDefaultSpan \[Method\]](#)

## IPen.SamplePeriod [Property][Get/Set]

Determine how frequently data is collected for an instant trend. If the sample period for a pen is changed, existing samples for the pen may be discarded.

#### Defined As

- [VBA] Long SamplePeriod
- [Cicode] INT SamplePeriod
- [C++] int SamplePeriod

#### Limits

- Minimum = 50 milliseconds
- Maximum = 60000 milliseconds (1 minute)
- Default = the scan rate of the page which hosts the Process Analyst control.

#### See Also

[IPen.InstantTrend \[Property\]\[Get/Set\]](#)

# IPen.Select [Method]

Makes this pen the primary selected pen.

## Defined As

- [VBA] Select()
- [Cicode] Select()
- [C++] HRESULT Select()

## Execution Result

If the function succeeds, the return value will be Success. If the pen is deleted, the return value will be GeneralFailure.

## Remarks

Calling this method will also trigger [PenSelectionChanged \[Event\]](#).

## Calling Syntax

Assumes you have passed a valid pen object into the function.

### [VBA]

```
Sub Example(pen As Object)
pen.Select
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPen)
_ObjectCallMethod(hPen, "Select");
END
```

# IPen.SetDefaultSpan [Method]

Sets the default time span for this pen.

## Defined As

- [VBA] SetDefaultSpan(weeks As Integer, days As Integer, hours As Integer, minutes As Integer, seconds As Integer, milliseconds As Integer)
- [Cicode] SetDefaultSpan (INT weeks, INT days, DATE hours, INT minutes, INT seconds, INT milliseconds)
- [C++] HRESULT SetDefaultSpan (short weeks, short days, short hours, short minutes, short seconds, short milliseconds)

## Parameters

weeks

[in] Indicates the number of weeks in the span.

days

[in] Indicates the number of days in the span.

hours

[in] Indicates the number of hours in the span.

minutes

[in] Indicates the number of minutes in the span.

seconds

[in] Indicates the number of seconds in the span.

milliseconds

[in] Indicates the number of milliseconds in the span.

#### Execution Result

If the function succeeds the return value will be Success. If an argument is bad then the return value will be InvalidArgument. If the pen is deleted then the return value will be GeneralFailure.

#### Calling Syntax

Assumes you have passed a valid pen object into the function.

### [VBA]

```
Sub Example(pen As Object)
    ' Set span to 2 hours and 30 minutes
    pen.GetDefaultSpan 0, 0, 2, 30, 0, 0
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPen)
// Set span to 2 hours and 30 minutes
_ObjectCallMethod(hPen, "SetDefaultSpan", 0, 0, 2, 30, 0, 0);
END
```

### See Also

[IPen.GetDefaultSpan \[Method\]](#)

[IPen.ResetToDefaultSpan \[Method\]](#)

## IPen.SetQualityCompactionPointType [Method]

Use this function to indicate what visual cue to display for single and multiple samples.

#### Defined As

- [VBA] SetQualityCompactionPointType(compactionType As Integer, pointType As Integer)
- [Cicode] SetQualityCompactionPointType(INT compactionType, INT pointType)
- [C++] HRESULT SetQualityCompactionPointType(QualityCompactionType compactionType, PointType

pointType)

#### Parameters

compactionType

[in] Indicates which sample compaction type you want to set the visual cue for.

pointType

[in] Indicates which visual cue to use for the selected compaction type.

#### Execution Result

If the function succeeds the return value will be Success. If an argument is bad, the return value will be InvalidArgument. If an argument is out of range, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

#### Calling Syntax

Assumes you have passed a valid pen object into the function.

### [VBA]

```
Sub Example(pen As Object)
    ' Set single samples to look like triangles
    pen.SetQualityCompactionPointType 0, 5
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPen)
// Set single samples to look like triangles
_ObjectCallMethod(hPen, "SetQualityCompactionPointType", 0, 5);
END
```

### See Also

[QualityCompactionType \[Enumeration\]](#)

[PointType \[Enumeration\]](#)

## IPen.SetQualityLineStyle [Method]

This function can be used to change the type of line drawn for each of the quality states defined by the Process Analyst for this Pen only.

#### Defined As

- [VBA] SetQualityLineStyle(qualityType As Integer, lineStyle As Integer)
- [Cicode] SetQualityLineStyle(INT qualityType, INT lineStyle)
- [C++] HRESULT SetQualityLineStyle(QualityType qualityType, LineStyle lineStyle)

#### Parameters

**qualityType**

[in] Indicates which quality type you want to set the visual cue for.

**lineStyle**

[in] Indicates which line style visual cue to use for the selected quality type.

**Execution Result**

If the function succeeds the return value will be Success. If an argument is bad then the return value will be InvalidArgument. If an argument is out of range then the return value will be InvalidArgument. If the pen is deleted then the return value will be GeneralFailure.

**Remarks**

When a sample is added to the display, its quality value indicates how the line drawn from that sample to the next one will be displayed.

**Calling Syntax**

Assumes you have passed a valid pen object into the function.

**[VBA]**

```
Sub Example(pen As Object)
    ' Set lines drawn after NA samples to be drawn as dash_dot
    pen.SetQualityLineStyle 1, 3
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPen)
// Set lines drawn after NA samples to be drawn as dash_dot
_ObjectCallMethod(hPen, "SetQualityLineStyle", 1, 3);
END
```

**See Also**

[QualityType \[Enumeration\]](#)

[LineStyle \[Enumeration\]](#)

## IPen.SetVerticalAxisLabelValue [Method]

This function can be used to display custom text for a particular value on the Vertical Axis.

**Defined As**

- [VBA] SetVerticalAxisLabelValue(value As Double, label As String)
- [Cicode] SetVerticalAxisLabelValue(REAL value, STRING label)
- [C++] HRESULT SetVerticalAxisLabelValue(double value, BSTR label)

**Parameters**

value

[in] Indicates which value you want to replace with a custom label.

label

[in] Indicates the text that will be displayed instead of the specified value.

#### Execution Result

If the function succeeds the return value will be Success. If an argument is bad then the return value will be InvalidArgument. If the pen is deleted then the return value will be GeneralFailure.

#### Calling Syntax

Assumes you have passed a valid pen object into the function.

### [VBA]

```
Sub Example(pen As Object)
    ' Change the vertical axis to display High High instead of 95
    pen.SetVerticalAxisLabelValue 95, "High High"
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPen)
// Change the vertical axis to display High High instead of 95
_ObjectCallMethod(hPen, "SetVerticalAxisLabelValue", 95, "High High");
END
```

## IPen.Stacked [Property][Get/Set]

Get or Set whether the pen is visually displayed stacked or overlaid.

#### Defined As

- [VBA] Boolean Stacked
- [Cicode] INT Stacked
- [C++] VARIANT\_BOOL Stacked

#### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

#### Limits

- True (-1): Stacked
- False (0): Overlaid

#### Remarks

When stacked, pens will be drawn under each other; when overlaid, the pens will be drawn over the top of each other.

#### Calling Syntax

Assumes you have passed a valid pen object into the function.

#### [VBA]

```
Sub Example(pen As Object)
Dim stacked As Boolean
`Getting Property value
stacked = pen.Stacked
`Setting Property value
pen.Stacked = True
End Sub
```

#### [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT bStacked;
// Getting current property value
bStacked = _ObjectGetProperty(hPen, "Stacked");
// Setting property value
_ObjectSetProperty(hPen, "Stacked", -1);
END
```

## IPen.TrendCursorLabelFillColor [Property][Get/ Set]

Gets or sets the fill color used for any cursor label associated with this pen.

#### Defined As

- [VBA] Long TrendCursorLabelFillColor
- [Cicode] INT TrendCursorLabelFillColor
- [C++] OLE\_COLOR TrendCursorLabelFillColor

#### Execution Result

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument. If the pen is deleted then the return value will be GeneralFailure.

#### Remarks

To calculate the integer value necessary for a color apply the following formula  $(65536 * \text{Blue}) + (256 * \text{Green}) + (\text{Red})$ . Where Red, Green and Blue are 0-255.

#### Calling Syntax

Assumes you have passed a valid pen object into the function.

#### [VBA]

```
Sub Example(pen As Object)
Dim color As Long
```

```
`Getting Property value
color = pen.TrendCursorLabelFillColor
`Setting Property value to Red
pen.TrendCursorLabelFillColor = 255
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT color;
// Getting current property value
color = _ObjectGetProperty(hPen, "TrendCursorLabelFillColor");
// Setting Property to Red
_ObjectSetProperty(hPen, "TrendCursorLabelFillColor", PackedRGB(255, 0, 0));
END
```

# IPen.TrendCursorLabelLineColor [Property][Get/ Set]

Gets or sets the border color used for any cursor label associated with this pen.

### Defined As

- [VBA] Long TrendCursorLabelLineColor
- [Cicode] INT TrendCursorLabelLineColor
- [C++] OLE\_COLOR TrendCursorLabelLineColor

### Execution Result

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument. If the pen is deleted then the return value will be GeneralFailure.

### Remarks

To calculate the integer value necessary for a color apply the following formula  $(65536 * \text{Blue}) + (256 * \text{Green}) + (\text{Red})$ . Where Red, Green and Blue are 0-255.

### Calling Syntax

Assumes you have passed a valid pen object into the function.

## [VBA]

```
Sub Example(pen As Object)
Dim color As Long
`Getting Property value
color = pen.TrendCursorLabelLineColor
`Setting Property value to Red
pen.TrendCursorLabelLineColor = 255
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPen)
INT color;
// Getting current property value
color = _ObjectGetProperty(hPen, "TrendCursorLabelTextColor");
// Setting Property to Red
_ObjectSetProperty(hPen, "TrendCursorLabelTextColor", PackedRGB(255, 0, 0));
END
```

## IPen.TrendCursorLabelTextColor [Property][Get/ Set]

Gets or sets the text color used for any cursor label associated with this pen.

**Defined As**

- [VBA] Long TrendCursorLabelTextColor
- [Cicode] INT TrendCursorLabelTextColor
- [C++] OLE\_COLOR TrendCursorLabelTextColor

**Execution Result**

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument. If the pen is deleted then the return value will be GeneralFailure.

**Remarks**

To calculate the integer value necessary for a color apply the following formula  $(65536 * \text{Blue}) + (256 * \text{Green}) + (\text{Red})$ . Where Red, Green and Blue are 0-255.

**Calling Syntax**

Assumes you have passed a valid pen object into the function.

**[VBA]**

```
Sub Example(pen As Object)
Dim color As Long
`Getting Property value
color = pen.TrendCursorLabelTextColor
`Setting Property value to Red
pen.TrendCursorLabelTextColor = 255
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPen)
INT color;
// Getting current property value
color = _ObjectGetProperty(hPen, "TrendCursorLabelTextColor");
// Setting Property to Red
```

```
_ObjectSetProperty(hPen, "TrendCursorLabelTextColor", PackedRGB(255, 0, 0));
END
```

## IPen.VerticalAxisAutoscale [Property][Get/Set]

Gets or sets whether the vertical axis will automatically calculate its physical limits based on the sample values within its internal cache.

### Defined As

- [VBA] Boolean VerticalAxisAutoscale
- [Cicode] INT VerticalAxisAutoscale
- [C++] VARIANT\_BOOL VerticalAxisAutoscale

### Execution Result

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument. If the pen is deleted then the return value will be GeneralFailure.

### Remarks

Setting this property will turn off interactive Scrolling (IPen.HorizontalAxisScroll) and Scaling (IPen.HorizontalAxisResize).

### Limits

- True (-1): Autoscale enabled
- False (0): Autoscale disabled

### Calling Syntax

Assumes you have passed a valid pen object into the function.

### [VBA]

```
Sub Example(pen As Object)
Dim autoScale As Long
`Getting Property value
autoScale = pen.VerticalAxisAutoscale
`Setting Property value
pen.VerticalAxisAutoscale = True
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT autoScale;
// Getting current property value
autoScale = _ObjectGetProperty(hPen, "VerticalAxisAutoscale");
// Setting Property
_ObjectSetProperty(hPen, "VerticalAxisAutoscale", -1);
END
```

# IPen.VerticalAxisColor [Property][Get/Set]

Gets or sets the color used to draw the line, labels and interval markers of the vertical axis of this pen.

## Defined As

- [VBA] Long VerticalAxisColor
- [Cicode] INT VerticalAxisColor
- [C++] OLE\_COLOR VerticalAxisColor

## Execution Result

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument. If the pen is deleted then the return value will be GeneralFailure.

## Remarks

To calculate the integer value necessary for a color apply the following formula  $(65536 * \text{Blue}) + (256 * \text{Green}) + (\text{Red})$ . Where Red, Green and Blue are 0-255.

## Calling Syntax

Assumes you have passed a valid pen object into the function.

## [VBA]

```
Sub Example(pen As Object)
Dim color As Long
`Getting Property value
color = pen.VerticalAxisColor
`Setting Property value to Red
pen.VerticalAxisColor = 255
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT color;
// Getting current property value
color = _ObjectGetProperty(hPen, "VerticalAxisColor");
// Setting Property to Red
_ObjectSetProperty(hPen, "VerticalAxisColor", PackedRGB(255, 0, 0));
END
```

## See Also

[IPen.HorizontalAxisColor \[Property\]\[Get/Set\]](#)

# IPen.VerticalAxisLabelType [Property][Get/Set]

Gets or sets a unit type which can be applied to the axis labels. This allows numbers on the axis to display with

their unit. For example, setting the unit to "kg" will display "10 Kg" on the axis.

#### Defined As

- [VBA] Integer VerticalAxisLabelType
- [Cicode] INT VerticalAxisLabelType
- [C++] AxisLabelType VerticalAxisLabelType

#### Execution Result

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument. If the pen is deleted then the return value will be GeneralFailure.

#### Remarks

Label Types are fixed and cannot be added to.

#### Calling Syntax

Assumes you have passed a valid pen object into the function.

### [VBA]

```
Sub Example(pen As Object)
Dim labelType As Integer
`Getting Property value
labelType = pen.VerticalAxisLabelType
`Setting Property value to Percent
pen.VerticalAxisLabelType= 3
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT labelType;
// Getting current property value
labelType = _ObjectGetProperty(hPen, "VerticalAxisLabelType");
// Setting Property to Percent
_ObjectSetProperty(hPen, "VerticalAxisLabelType", 3);
END
```

### See Also

[AxisLabelType \[Enumeration\]](#)

## IPen.VerticalAxisResize [Property][Get/Set]

Gets or sets whether this pen allows the operator to interactively scale the vertical axis by using the mouse.

#### Defined As

- [VBA] Boolean VerticalAxisResize
- [Cicode] INT VerticalAxisResize

- [C++] VARIANT\_BOOL VerticalAxisResize

#### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

#### Limits

- True (-1): Enable re-size
- False (0): Disable re-size

#### Remarks

This only applies to analog pens.

#### Calling Syntax

Assumes you have passed a valid pen object into the function.

### [VBA]

```
Sub Example(pen As Object)
Dim re-size As Boolean
`Getting Property value
re-size = pen.VerticalAxisResize
`Setting Property value
pen.VerticalAxisResize = False
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT bResize;
// Getting current property value
bResize = _ObjectGetProperty(hPen, "VerticalAxisResize");
// Setting Property
_ObjectSetProperty(hPen, "VerticalAxisResize",0);
END
```

### See Also

[IPen.HorizontalAxisScroll \[Property\]\[Get/Set\]](#)

## IPen.VerticalAxisScroll [Property][Get/Set]

Gets or sets whether this pen allows the operator to interactively scroll the vertical axis by using the mouse.

#### Defined As

- [VBA] Boolean VerticalAxisScroll
- [Cicode] INT VerticalAxisScroll

- [C++] VARIANT\_BOOL VerticalAxisScroll

#### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

#### Limits

- True (-1): Enable scrolling
- False (0): Disable scrolling

#### Remarks

This only applies to analog pens.

#### Calling Syntax

Assumes you have passed a valid pen object into the function.

### [VBA]

```
Sub Example(pen As Object)
Dim scroll As Boolean
`Getting Property value
scroll = pen.VerticalAxisScroll
`Setting Property value
pen.VerticalAxisScroll = False
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT bScroll;
// Getting current property value
bScroll = _ObjectGetProperty(hPen, "VerticalAxisScroll");
// Setting Property
_ObjectSetProperty(hPen, "VerticalAxisScroll", 0);
END
```

### See Also

[IPen.HorizontalAxisScroll \[Property\]\[Get/Set\]](#)

## IPen.VerticalAxisWidth [Property][Get/Set]

Gets or sets the width of the vertical axis line and the associated interval markers.

#### Defined As

- [VBA] Integer VerticalAxisWidth
- [Cicode] INT VerticalAxisWidth

- [C++] short VerticalAxisWidth

#### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

#### Limits

A valid width is 0-8 pixels.

#### Remarks

This only applies to analog pens.

#### Calling Syntax

Assumes you have passed a valid pen object into the function.

### [VBA]

```
Sub Example(pen As Object)
Dim width As Integer
`Getting Property value
width = pen.VerticalAxisWidth
`Setting Property value
pen.VerticalAxisWidth = 3
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT width;
// Getting current property value
width = _ObjectGetProperty(hPen, "VerticalAxisWidth");
// Setting Property
_ObjectSetProperty(hPen, "VerticalAxisWidth", 3);
END
```

### See Also

[IPen.HorizontalAxisWidth \[Property\]\[Get/Set\]](#)

## IPen.VerticalGridlinesColor [Property][Get/Set]

Gets or sets the color used to draw the major vertical gridlines.

#### Defined As

- [VBA] Long VerticalGridlinesColor
- [Cicode] INT VerticalGridlinesColor
- [C++] OLE\_COLOR VerticalGridlinesColor

#### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

#### Remarks

To calculate the integer value necessary for a color apply the following formula  $(65536 * \text{Blue}) + (256 * \text{Green}) + (\text{Red})$ . Where Red, Green and Blue are 0-255.

#### Calling Syntax

Assumes you have passed a valid pen object into the function.

### [VBA]

```
Sub Example(pen As Object)
Dim color As Long
`Getting Property value
color = pen.VerticalGridlinesColor
`Setting Property value to Red
pen.VerticalGridlinesColor = 255
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT color;
// Getting current property value
color = _ObjectGetProperty(hPen, "VerticalGridlinesColor");
// Setting Property to Red
_ObjectSetProperty(hPen, "VerticalGridlinesColor", PackedRGB(255, 0, 0));
END
```

### See Also

[IPen.VerticalMinorGridlinesColor \[Property\]\[Get/Set\]](#)

## IPen.VerticalGridlinesStyle [Property][Get/Set]

Gets or sets the line style used to draw the major vertical gridlines.

#### Defined As

- [VBA] Long VerticalGridlinesColor
- [Cicode] INT VerticalGridlinesColor
- [C++] LineStyle VerticalGridlinesColor

#### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the style is out of range, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

#### Calling Syntax

Assumes you have passed a valid pen object into the function.

### [VBA]

```
Sub Example(pen As Object)
Dim style As Long
`Getting Property value
style = pen.VerticalGridlinesColor
`Setting Property value to Dot
pen.VerticalGridlinesColor = 2
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT style;
// Getting current property value
style = _ObjectGetProperty(hPen, "VerticalGridlinesStyle");
// Setting Property to Dot
_ObjectSetProperty(hPen, "VerticalGridlinesStyle", 2);
END
```

### See Also

[IPen.VerticalMinorGridlinesStyle \[Property\]\[Get/Set\]](#)

[LineStyle \[Enumeration\]](#)

## IPen.VerticalGridlinesWidth [Property][Get/Set]

Gets or sets the line width used when drawing the major vertical gridlines.

### Defined As

- [VBA] Integer VerticalGridlinesWidth
- [Cicode] INT VerticalGridlinesWidth
- [C++] short VerticalGridlinesWidth

### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the width is out of range, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

### Limits

A valid width is 0-8 pixels.

### Calling Syntax

Assumes you have passed a valid pen object into the function.

**[VBA]**

```
Sub Example(pen As Object)
Dim width As Integer
`Getting Property value
width = pen.VerticalGridlinesWidth
`Setting Property value
pen.VerticalGridlinesWidth = 3
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPen)
INT width;
// Getting current property value
width = _ObjectGetProperty(hPen, "VerticalGridlinesWidth");
// Setting Property t
_ObjectSetProperty(hPen, "VerticalGridlinesWidth", 3);
END
```

# IPen.VerticalMinorGridlinesColor [Property][Get/ Set]

Gets or sets the color used to draw the minor vertical gridlines.

**Defined As**

- [VBA] Long VerticalMinorGridlinesColor
- [Cicode] INT VerticalMinorGridlinesColor
- [C++] OLE\_COLOR VerticalMinorGridlinesColor

**Execution Result**

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

**Remarks**

To calculate the integer value necessary for a color apply the following formula  $(65536 * \text{Blue}) + (256 * \text{Green}) + (\text{Red})$ . Where Red, Green and Blue are 0-255.

**Calling Syntax**

Assumes you have passed a valid pen object into the function.

**[VBA]**

```
Sub Example(pen As Object)
Dim color As Long
`Getting Property value
color = pen.VerticalMinorGridlinesColor
`Setting Property value to Red
```

```
pen.VerticalMinorGridlinesColor = 255
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT color;
// Getting current property value
color = _ObjectGetProperty(hPen, "VerticalMinorGridlinesColor");
// Setting Property to Red
_ObjectSetProperty(hPen, "VerticalMinorGridlinesColor", PackedRGB(255, 0, 0));
END
```

## See Also

[IPen.VerticalGridlinesColor \[Property\]\[Get/Set\]](#)

# IPen.VerticalMinorGridlinesStyle [Property][Get/ Set]

Gets or sets the line style used to draw the minor vertical gridlines.

### Defined As

- [VBA] Long VerticalMinorGridlinesStyle
- [Cicode] INT VerticalMinorGridlinesStyle
- [C++] LineStyle VerticalMinorGridlinesStyle

### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the style is out of range, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

### Calling Syntax

Assumes you have passed a valid pen object into the function.

## [VBA]

```
Sub Example(pen As Object)
Dim style As Long
`Getting Property value
style = pen.VerticalMinorGridlinesStyle
`Setting Property value to Dot
pen.VerticalMinorGridlinesColor = 2
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPen)
INT style;
// Getting current property value
style = _ObjectGetProperty(hPen, "VerticalMinorGridlinesStyle");
// Setting Property to Dot
_ObjectSetProperty(hPen, "VerticalMinorGridlinesStyle", 2);
END
```

**See Also**

[IPen.VerticalGridlinesStyle \[Property\]\[Get/Set\]](#)

[LineStyle \[Enumeration\]](#)

## IPen.VerticalScrollBy [Method]

Scrolls the vertical axis by the specified factor.

**Defined As**

- [VBA] VerticalScrollBy(factor As Double)
- [Cicode] VerticalScrollBy(REAL factor)
- [C++] HRESULT VerticalScrollBy(double factor)

**Parameters**

factor

[in] Controls the direction and amount the axis will be scrolled. A negative value will move the axis in the negative direction. A positive value will move the axis forward in the positive direction. The value is a percentage representing the current viewable span. So if the pen span is 100 units (10 to 110), and you specify a factor of 0.5 then you will move the span 50 units (60 to 160).

**Execution Result**

If the function succeeds the return value will be Success. If the argument is bad then the return value will be InvalidArgument. If the pen is deleted then the return value will be GeneralFailure.

**Calling Syntax**

Assumes you have passed a valid pen object into the function.

**[VBA]**

```
Sub Example(pen As Object)
` Move the pen span forward one complete span
pen.VerticalScrollBy 1.0
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPen)
// Move the pen span forward one complete span
_ObjectCallMethod(hPen, "VerticalScrollby", 1.0);
END
```

## See Also

[IPen.HorizontalScrollBy \[Method\]](#)

# IPen.VerticalZoom [Method]

Zooms centrally into the time span by the given factor on the vertical axis

### Defined As

- [VBA] VerticalZoom(factor As Double)
- [Cicode] VerticalZoom (REAL factor)
- [C++] HRESULT VerticalZoom (double factor)

### Parameters

#### factor

[in] Controls the direction and amount the axis will be zoomed. Acceptable zoom values are 0 to 1 (Zoom out) and > 1 (zoom in).

### Execution Result

If the function succeeds the return value will be Success. If the argument is bad then the return value will be InvalidArgument. If the argument is out of range then the return value will be InvalidArgument. If the pen is deleted then the return value will be GeneralFailure.

### Calling Syntax

Assumes you have passed a valid pen object into the function.

## [VBA]

```
Sub Example(pen As Object)
` Zoom out 50%
pen.VerticalZoom 0.5
` Undo the Zoom
pen.VerticalZoom 1.5
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPen)
// Zoom out 50%
_ObjectCallMethod(hPen, "VerticalZoom", 0.5);
```

```
// Undo the Zoom
_ObjectCallMethod(hPen, "VerticalZoom", 2.0);
END
```

## See Also

[IPen.HorizontalZoom \[Method\]](#)

# IPen.Visible [Property][Get/Set]

Get or set whether this pen will be visually shown (True) or hidden (False) to the operator.

### Defined As

- [VBA] Boolean Visible
- [Cicode] INT Visible
- [C++] VARIANT\_BOOL Visible

### Execution Result

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the pen is deleted, the return value will be GeneralFailure.

### Limits

- True (-1): Visible
- False (0): Hidden

### Calling Syntax

Assumes you have passed a valid pen object into the function.

## [VBA]

```
Sub Example(pen As Object)
Dim visible As Boolean
`Get the property value
visible = pen.Visible
` Set the property value
pen.Visible = False
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPen)
INT bVisible;
// Get the property value
bVisible = _ObjectGetProperty(hPen, "Visible", 0.5);
// Set the property value
_ObjectSetProperty(hPen, "Visible", 0);
END
```

## IPens Interface

### Defined As

- [VBA] Object
- [Cicode] OBJECT
- [C++] IPens

### Methods (2)

- [IPens.Create \[Method\]](#)
- [IPens.RemoveAll \[Method\]](#)

### Properties (5)

- [IPens.Count \[Property\]\[Get\]](#)
- [IPens.Item \[Property\]\[Get\]](#)
- [IPens.\\_NewEnum \[Property\]\[Get\]](#)
- [IPens.ItemByName \[Property\]\[Get\]](#)
- [IPens.Pane\[Property\]\[Get\]](#)

## IPens.\_NewEnum [Property][Get]

This allows For.. Each.. Next integration in VB.

### Calling Syntax

This example assumes there is a valid Pens collection object to be passed into the example methods. This property is not applicable to Cicode.

### [VBA]

```
Sub Example(Pens As Object)
Dim pen As Object
Dim count As Long
`Using Property
For Each pen In Pens
count = count + 1
Next pen
End Sub
```

## IPens.Count [Property][Get]

Gets the number of Pens in this collection.

### Defined As

- [VBA] Long Count
- [Cicode] INT Count
- [C++] int Count

#### Execution Result

If the property get succeeds, the return value will be Success. If the pens collection is deleted, the return value will be GeneralFailure.

#### Calling Syntax

This example assumes there is a valid Pens collection object to be passed into the example methods.

### [VBA]

```
Sub Example(Pens As Object)
Dim count As Long
`Getting Property value
count = Pens.Count
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPens)
// Getting property value
INT nCount = _ObjectGetProperty(hPens, "Count");
END
```

## IPens.Create [Method]

Creates a new pen and adds it to this collection.

#### Defined As

- [VBA] Create(penType As Integer, nameMode As Integer) as Object
- [Cicode] OBJECT Create (INT penType, INT nameMode)
- [C++] HRESULT Create(PenType penType, PenNameMode penNameMode, IPen\*\* pen)

#### Parameters

##### penType

[in] Indicates the type of pen that will be created. See PenType Enumeration for the types of pen that can be created.

##### penNameMode

[in] Indicates how the name will be obtained for this pen. The Process Analyst provides options of PenNameMode\_Comment, PenNameMode\_Tag, and PenNameMode\_Custom. Specifying PenNameMode\_Comment will mean that the Process Analyst names the pen from the Comment field of the trend/alarm tag associated with the IPen.DataPoint property. Specifying PenNameMode\_Tag will mean that the Process Analyst will name the pen as the value of the IPen.DataPoint property. Specifying PenNameMode\_Custom causes the Process Analyst to provide a default name and leave setting the name to you

via the IPen.Name property.

#### Execution Results

If the method succeeds the return value will be Success. If the pens collection is deleted, the return value will be GeneralFailure.

#### Remarks

If this method succeeds, a new Pen of the specified type is created and appended to the pens collection.

#### Calling Syntax

This example assumes there is a valid Pens collection object to be passed in to the example methods.

### [VBA]

```
Sub Example(pens As Object)
Dim pen As Object
Set pen = pens.Create(4097, 1)
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hPens)
OBJECT hPen = _ObjectCallMethod(hPens, "Create" , 4097, 1);
END
```

### See Also

[IPen Interface](#)  
[PenType \[Enumeration\]](#)  
[PenNameMode \[Enumeration\]](#)

## IPens.Item [Property][Get]

Gets the Pen at the given index from this pen's collection.

#### Defined As

- [VBA] Item(index As Long) as Object
- [Cicode] OBJECT Item(INT index)
- [C++] Item(int index, IPen\* Item)

#### Parameters

index

[in] Indicates the index of the pen item to return from this collection. (One based)

#### Execution Result

If the property get succeeds, the return value will be Success. If the index is out of range, the return value will be InvalidArgument. If the pen's collection is deleted, the return value will be GeneralFailure.

**Calling Syntax**

This example assumes there is a valid Pens collection object to be passed into the example methods.

**[VBA]**

```
Sub Example(pens As Object)
Dim pen As Object
`Getting Property value
Set pen = pens.Item(1)
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPens)
// Getting property value
OBJECT hPen = _ObjectCallMethod(hPens, "get_Item", 1);
END
```

## IPens.ItemByName [Property][Get]

Gets the Pen of the given name from this Pens collection.

**Defined As**

- [VBA] ItemByName(name As String) as Object
- [Cicode] OBJECT ItemByName(STRING name)
- [C++] ItemByName(STRING name, IPen\* Item)

**Parameters**

name

[in] Indicates the name of the Pen item to return from this collection.

**Execution Result**

If the property get succeeds, the return value will be Success. If the pen does not exist, the return value will be InvalidArgument. If the pens collection is deleted, the return value will be GeneralFailure.

**Calling Syntax**

This example assumes there is a valid Pens collection object to be passed into the example methods.

**[VBA]**

```
Sub Example(Pens As Object)
Dim pen As Object
`Getting Property value
Set pen = Pens.ItemByName("CPU Usage")
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPens)
// Getting property value
OBJECT hPen = _ObjectCallMethod(hPens, "get_ItemByName", "CPU Usage");
END
```

## IPens.Pane[Property][Get]

Gets the Pane that this Pens collection belongs to.

**Defined As**

- [VBA] Object Pane
- [Cicode] OBJECT Pane
- [C++] HRESULT Pane(IPane\*\* Pane)

**Execution Result**

If the property get succeeds the return value will be Success. If the pens collection is deleted the return value will be GeneralFailure.

**Remarks**

Each Pens collection belongs to a Pane.

**Calling Syntax**

This example assumes there is a valid Pens collection object to be passed into the example methods.

**[VBA]**

```
Sub Example(pens As Object)
Dim pane As Object
`Getting Property value
Set pane = pens.Pane
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hPens)
// Getting property value
OBJECT hPane = _ObjectGetProperty(hPens, "Pane");
END
```

## IPens.RemoveAll [Method]

Removes every Pen from the Pens collection.

**Defined As**

- [VBA] RemoveAll()
- [Cicode] RemoveAll()
- [C++] HRESULT RemoveAll()

### Execution Result

If the property get/set succeeds the return value will be Success. If the pens collection is deleted the return value will be GeneralFailure.

### Calling Syntax

This example assumes there is a valid Pens collection object to be passed into the example methods.

## [VBA]

```
Sub Panes(pens As Object)
pens.RemoveAll()
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hPens)
_ObjectCallMethod(hPens, "RemoveAll");
END
```

## IProcessAnalyst Interface

### Defined As

- [VBA] Object
- [Cicode] OBJECT
- [C++] IProcessAnalyst

### Methods

- [IProcessAnalyst.BlockUpdates \[Method\]](#)
- [IProcessAnalyst.UnBlockUpdates \[Method\]](#)
- [IProcessAnalyst.CopyToClipboard \[Method\]](#)
- [IProcessAnalyst.CopyToFile \[Method\]](#)
- [IProcessAnalyst.FreezeEvent \[Method\]](#)
- [IProcessAnalyst.LoadFromFile \[Method\]](#)
- [IProcessAnalyst.PrintAll \[Method\]](#)
- [IProcessAnalyst.SaveToFile \[Method\]](#)
- [IProcessAnalyst.ShowProperties \[Method\]](#)
- [IProcessAnalyst.SubscribeForPropertyChange \[Method\]](#)
- [IProcessAnalyst.SynchroniseToNow \[Method\]](#)
- [IProcessAnalyst.UnsubscribePropertyChange \[Method\]](#)

## Properties

- IProcessAnalyst.AdminPrivilegeLevel [Property] [Get]
- IProcessAnalyst.AutoScroll [Property][Get/Set]
- IProcessAnalyst.BackgroundColor [Property][Get/Set]
- IProcessAnalyst.CommandSystem [Property][Get]
- IProcessAnalyst.ContextMenu [Property][Get/Set]
- IProcessAnalyst.Cursors [Property][Get]
- IProcessAnalyst.DataRequestRate [Property][Get/Set]
- IProcessAnalyst.DisplayRefreshRate [Property][Get/Set]
- IProcessAnalyst.Language [Property] [Get/Set]
- IProcessAnalyst.LastSelectedPen [Property][Get]
- IProcessAnalyst.LockedPens [Property][Get/Set]
- IProcessAnalyst.NumberofSamples[Property][Get/Set]
- IProcessAnalyst.ObjectView [Property][Get]
- IProcessAnalyst.Panes [Property][Get]
- IProcessAnalyst.PrimaryPath [Property][Get/Set]
- IProcessAnalyst.SecondaryPath [Property][Get/Set]
- IProcessAnalyst.Toolbars [Property][Get]
- IProcessAnalyst.ViewLocation [Property][Get]
- IProcessAnalyst.WritePrivilegeLevel [Property][Get]
- IProcessAnalyst.ZoomMode [Property][Get/Set]

## IProcessAnalyst.BlockUpdates [Method]

Blocks certain aspects of the Process Analyst's redrawing and data updating.

### Defined As

- [VBA] BlockUpdates()
- [Cicode] BlockUpdates()
- [C++] HRESULT BlockUpdates()

### Remarks

- This method blocks three redraw systems: redraw for the chart, the Object View, and the toolbars.
- Data updates are also blocked.
- The current data requests are cancelled.
- The Process Analyst has a built-in counter to store how many times the block and unblock have been called, so that only the final UnBlockUpdates call actually unblocks the above mentioned data updates and redraw systems.

### Execution Result

If the function succeeds, the return value will be Success. If the function does not succeed, the return value will be GeneralFailure.

### Calling Syntax

This example assumes there is a valid Process Analyst object to be passed into the example methods.

## [VBA]

```
Sub Example(ProcessAnalyst As Object)
ProcessAnalyst.BlockUpdates()
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hProcessAnalyst)
    _ObjectCallMethod(hProcessAnalyst, "BlockUpdates");
END
```

## See Also

[IProcessAnalyst.UnBlockUpdates \[Method\]](#)

# IProcessAnalyst.UnBlockUpdates [Method]

Unblocks certain aspects of the Process Analyst's redrawing and data updating.

### Defined As

- [VBA] UnblockUpdates()
- [Cicode] UnblockUpdates()
- [C++] HRESULT UnblockUpdates()

### Remarks

- This method unblocks three redraw systems: redraw for the chart, the Object View, and the toolbars.
- Data updates are also unblocked.
- The Process Analyst has a built-in counter to store how many times the block and unblock have been called, so that only the final UnBlockUpdates call actually unblocks the above mentioned data updates and redraw systems.

### Execution Result

If the function succeeds, the return value will be Success. If the function does not succeed, the return value will be GeneralFailure. If other BlockUpdates are in effect, a Success will be returned also (for those C++ users, S\_FALSE will be returned in this case).

### Calling Syntax

This example assumes there is a valid Process Analyst object to be passed into the example methods.

### [VBA]

```
Sub Example(ProcessAnalyst As Object)
ProcessAnalyst.UnblockUpdates()
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hProcessAnalyst)
    _ObjectCallMethod(hProcessAnalyst, "UnblockUpdates");
END
```

### See Also

[IProcessAnalyst.BlockUpdates \[Method\]](#)

## IProcessAnalyst.CopyToClipboard [Method]

Copies the data in the current viewable range for visible pens to the clipboard.

### Defined As

- [VBA] CopyToClipboard()
- [Cicode] CopyToClipboard()
- [C++] HRESULT CopyToClipboard()

### Remarks

The timestamp of each sample will be in local time defined by your computer. The start and end sample maybe generated for each pen to indicate the exported range of the data.

### Execution Result

If the function succeeds the return value will be Success. If the function does not succeed the return value will be GeneralFailure.

### Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

### [VBA]

```
Sub Example()
myPage_AN35.CopyToClipboard
End Sub
```

### [Cicode]

```
Sub Example()
```

```
OBJECT hProcessAnalyst = ObjectByName("AN35");
_ObjectCallMethod(hProcessAnalyst, "CopyToClipboard");
End Sub
```

## See Also

[IProcessAnalyst.CopyToFile \[Method\]](#)

# IProcessAnalyst.CopyToFile [Method]

Saves the data in the current viewable range for visible pens to the specified file.

### Defined As

- [VBA] CopyToFile(filename As String)
- [Cicode] CopyToFile(STRING filename)
- [C++] HRESULT CopyToClipboard(BSTR filename)

### Parameters

filename

[in] Indicates the name and path of the file that the data will be exported to.

### Execution Result

If the function succeeds the return value will be Success. If the function does not succeed the return value will be GeneralFailure.

### Remarks

The timestamp of each sample will be in local time defined by your computer. The start and end sample maybe generated for each pen to indicate the exported range of the data.

### Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

## [VBA]

```
Sub Example()
myPage_AN35.CopyToFile "test.xls"
End Sub
```

## [Cicode]

```
Sub Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
_ObjectCallMethod(hProcessAnalyst, "CopyToFile", "test.xls");
End Sub
```

## See Also

[IProcessAnalyst.CopyToClipboard \[Method\]](#)

# IProcessAnalyst.FreezeEvent [Method]

Enables or disables a specified event from triggering.

### Defined As

- [VBA] FreezeEvent(eventName As String, freeze As Boolean)
- [Cicode] FreezeEvent(STRING eventName, INT freeze)
- [C++] HRESULT FreezeEvent(BSTR columnName, VARIANT\_BOOL freeze)

### Parameters

eventName

[in] Specifies the event that you want to cease receiving notifications for.

freeze

[in] Indicates whether to enable or disable the event. True(-1) disable the event. False(0) enable the event.

### Execution Result

If the method succeeds, the return value will be Success. If the method does not succeed, the return value will be GeneralFailure. If eventName is bad or does not exist, the return value will be InvalidArgument.

### Remarks

All events exposed by the Process Analyst can be enabled or disabled. This method is particularly useful to minimize the likelihood of recursive behavior of functions that generate the same event that you are trying to handle.

### Calling Syntax

This example assumes there is a valid Process Analyst object to be passed into the example methods.

### [VBA]

```
Sub Example(analyst As Object)
analyst.FreezeEvent "HorizontalAxisChanged" True
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hAnalyst)
_ObjectCallMethod(hAnalyst, "FreezeEvent","HorizontalAxisChanged", -1);
END
```

# IProcessAnalyst.LoadFromFile [Method]

Loads a specified view into the Process Analyst.

## Defined As

- [VBA] LoadFromFile(filename As String, fileLocation As Integer)
- [Cicode] LoadFromFile(STRING filename, INT fileLocation)
- [C++] HRESULT LoadFromFile(BSTR filename, FileLocation fileLocation)

## Parameters

### filename

[in] Indicates a relative path and filename of the view to load into the Process Analyst. See Remarks, below.

### fileLocation

[in] Indicates which known location to load the file from.

## Execution Result

If the function succeeds the return value will be Success. If the filename is invalid the return value will be InvalidArgument. If the path indicated by fileLocation is invalid or offline then the return value will be PathNotFound. If any other problem occurs then the return value will be GeneralFailure.

## Remarks

This method will replace the current view with the one in the specified file.

Absolute paths are not necessary for the filename as the method has been designed to load the specified file from [Run]:\Analyst Views (FileLocation\_Local), my documents folder (FileLocation\_User) or from the primary/secondary paths (FileLocation\_Server).

When a file is loaded it will be synchronized with the other locations to verify each location has the file which is the latest. If the file you are loading is older than one which exists in another location it will be replaced.

## Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

## [VBA]

```
Sub Example()
` Load the view from the server
myPage_AN35.LoadFromFile "Test1.pav", 0
End Sub
```

## [Cicode]

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
_ObjectCallMethod(hProcessAnalyst, "LoadFromFile", "Test1.pav", 0);
END
```

## See Also

[FileLocation \[Enumeration\]](#)  
[IProcessAnalyst.PrimaryPath \[Property\]\[Get/Set\]](#)  
[IProcessAnalyst.SecondaryPath \[Property\]\[Get/Set\]](#)

# IProcessAnalyst.PrintAll [Method]

Displays the Print configuration dialog.

### Defined As

- [VBA] PrintAll
- [Cicode] PrintAll()
- [C++] HRESULT PrintAll()

### Execution Result

If the function succeeds the return value will be Success. If an unexpected error occurs then return value will be GeneralFailure.

### Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

## [VBA]

```
Sub Example()
myPage_AN35.PrintAll
End Sub
```

## [Cicode]

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
_ObjectCallMethod(hProcessAnalyst, "PrintAll");
END
```

# IProcessAnalyst.SaveToFile [Method]

Saves the current view using the specified name to the specified location.

### Defined As

- [VBA] SaveToFile(filename As String, fileLocation As Integer)
- [Cicode] SaveToFile(STRING filename, INT fileLocation)
- [C++] HRESULT SaveToFile(BSTR filename, FileLocation fileLocation)

**Parameters****filename**

[in] Indicates a relative path and filename which will be used during the saving of the view. See Remarks.

**fileLocation**

[in] Indicates which known location to save the file to.

**Execution Result**

If the function succeeds the return value will be Success. If the filename is invalid the return value will be InvalidArgument. If the path indicated by fileLocation is invalid or offline then the return value will be PathNotFound. If any other problem occurs, the return value will be GeneralFailure.

**Remarks**

On a client where the current user matches the WritePrivilegeLevel only the FileLocation\_Server and FileLocation\_User options will succeed. Saving using the FileLocation\_Server option will save to the locations indicated by PrimaryPath and SecondaryPath properties and into the Project directory.

On a client where the current user does not match the WritePrivilegeLevel only the FileLocation\_User will succeed.

**Calling Syntax**

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

**[VBA]**

```
Sub Example()
` Save the view to the server and project
myPage_AN35.SaveToFile "Analyst Views\Test1.pav", 1
End Sub
```

**[Cicode]**

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
` Save the view to the server and project
_ObjectCallMethod(hProcessAnalyst, "SaveToFile", "Analyst Views\Test1.pav", 1);
END
```

**See Also**

[FileLocation \[Enumeration\]](#)

[IProcessAnalyst.PrimaryPath \[Property\]\[Get/Set\]](#)

[IProcessAnalyst.SecondaryPath \[Property\]\[Get/Set\]](#)

[IProcessAnalyst.WritePrivilegeLevel \[Property\]\[Get\]](#)

## IProcessAnalyst.ShowProperties [Method]

Displays the Process Analyst's property configuration dialog.

**Define As**

- [VBA] ShowProperties
- [Cicode] ShowProperties()
- [C++] HRESULT ShowProperties ()

**Execution Result**

If the function succeeds the return value will be Success. If an unexpected error occurs then the return value will be GeneralFailure.

**Calling Syntax**

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

**[VBA]**

```
Sub Example()
myPage_AN35.ShowProperties
End Sub
```

**[Cicode]**

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
_ObjectCallMethod(hProcessAnalyst, "ShowProperties");
END
```

## IProcessAnalyst.SubscribeForPropertyChange [Method]

Use this method to receive notifications of when a particular property changes. Notifications will be sent via the PropertyChanged event.

**Defined As**

- [VBA] SubscribeForPropertyChange(interfaceName As String, propertyName As String)
- [Cicode] SubscribeForPropertyChange(STRING interfaceName, STRING propertyName)
- [C++] HRESULT SubscribeForPropertyChange(BSTR interfaceName, BSTR propertyName)

**Parameters**

interfaceName

[in] Specify the name of the interface that the property you want notifications for is defined on.

propertyName

[in] This is the name of the property you want to receive notifications for.

**Execution Result**

If the function succeeds, the return value will be Success. If the interfaceName or propertyName is a bad string,

the return value will be InvalidArgument. If any other problem occurs, the return value will be GeneralFailure.

#### Remarks

The following set of properties are supported:

Interface name	Property Name
IProcessAnalyst	AutoScroll
IProcessAnalyst	BackgroundColor
IProcessAnalyst	ContextMenu
IProcessAnalyst	LockedPens
IProcessAnalyst	DisplayRefreshRate
IProcessAnalyst	DataRequestRate
IProcessAnalyst	ZoomMode

#### Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

#### [VBA]

```
Sub Example()
    myPage_AN35.SubscribeForPropertyChange "IProcessAnalyst", "ZoomMode"
End Sub
```

#### [Cicode]

```
Sub Example()
    OBJECT hProcessAnalyst = ObjectByName("AN35");
    _ObjectCallMethod(hProcessAnalyst, "SubscribeForPropertyChange",
    "IProcessAnalyst", "ZoomMode");
End Sub
```

#### See Also

[IProcessAnalyst.UnsubscribePropertyChange \[Method\]](#)

[PropertyChanged \[Event\]](#)

## IProcessAnalyst.SynchroniseToNow [Method]

Synchronizes pens such that the date/time reflects "Now."

#### Defined As

- [VBA] SynchroniseToNow

- [Cicode] SynchroniseToNow()
- [C++] HRESULT SynchroniseToNow()

#### Execution Result

If the function succeeds, the return value will be Success. If any other problem occurs, the return value will be GeneralFailure.

#### Remarks

The current span for each pen will be maintained. 'Now' is defined as the current time on the client machine.

#### Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

### [VBA]

```
Sub Example()
    myPage_AN35.SynchroniseToNow
End Sub
```

### [Cicode]

```
Sub Example()
    OBJECT hProcessAnalyst = ObjectByName("AN35");
    _ObjectCallMethod(hProcessAnalyst, "SynchroniseToNow");
End Sub
```

## IProcessAnalyst.UnsubscribePropertyChanged [Method]

Use this method to cancel notifications of when the specified property changes. Notifications will cease to be sent via the PropertyChanged event.

#### Defined As

- [VBA] UnsubscribePropertyChanged(interfaceName As String, propertyName As String)
- [Cicode] UnsubscribePropertyChanged(STRING interfaceName, STRING propertyName)
- [C++] HRESULT UnsubscribePropertyChanged(BSTR interfaceName, BSTR propertyName)

#### Parameters

interfaceName

[in] Name of the interface that the property you want to remove notifications for is defined on.

propertyName

[in] Name of the property you want to remove notifications for.

#### Execution Result

If the function succeeds, the return value will be Success. If the interfaceName or propertyName is a bad string, the return value will be InvalidArgument. If any other problem occurs, the return value will be GeneralFailure.

#### Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

#### [VBA]

```
Sub Example()
    myPage_AN35.UnsubscribePropertyChanged "IProcessAnalyst", "ZoomMode"
End Sub
```

#### [Cicode]

```
Sub Example()
    OBJECT hProcessAnalyst = ObjectByName("AN35");
    _ObjectCallMethod(hProcessAnalyst, "UnsubscribePropertyChanged",
        "IProcessAnalyst", "ZoomMode");
End Sub
```

#### See Also

[IProcessAnalyst.SubscribeForPropertyChange \[Method\]](#)

[PropertyChanged \[Event\]](#)

## IProcessAnalyst.AdminPrivilegeLevel [Property] [Get]

Retrieves the privilege level currently set for controlling administration features of the Process Analyst at Run-time.

#### Defined As

- [VBA] Integer AdminPrivilegeLevel
- [Cicode] INT AdminPrivilegeLevel
- [C++] short AdminPrivilegeLevel

#### Execution Result

If the property get succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument.

#### Remarks

By default the level is 0 (Zero), which allows access to every feature at run time. Setting this to any other level will require the operator viewing the Process Analyst to have a privilege equal to that level.

This property can only be set at design time (in the Graphics Builder property pages) and is recommended to prevent Operators from changing performance properties such as DataRequestRate and DisplayRefreshRate.

**Limits**

Privilege level defined in Plant SCADA 1 - 8. 0 = no security.

**Calling Syntax**

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

**[VBA]**

```
Sub Example()
Dim privilege As Boolean
`Getting Property value
privilege = myPage_AN35.AdminPrivilegeLevel
End Sub
```

**[Cicode]**

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
INT privilege;
// Getting current property value
privilege = _ObjectGetProperty(hProcessAnalyst,"AdminPrivilegeLevel");
END
```

## IProcessAnalyst.AutoScroll [Property][Get/Set]

Gets or Sets the automatic scrolling of pens as time passes.

**Defined As**

- [VBA] Boolean AutoScroll
- [Cicode] INT AutoScroll
- [C++] VARIANT\_BOOL AutoScroll

**Execution Result**

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

**Remarks**

This function does not synchronize the pens to now. The display will be updated according to the value of the DisplayRefreshRate property.

**Limits**

- True (-1): Autoscroll is On
- False (0): Autoscroll is Off

**Calling Syntax**

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

**[VBA]**

```
Sub Example()
Dim autoScroll As Boolean
`Getting Property value
autoScroll = myPage_AN35.AutoScroll
`Setting Property value
myPage_AN35.AutoScroll = True
End Sub
```

**[Cicode]**

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
INT autoScroll;
// Getting current property value
autoScroll = _ObjectGetProperty(hProcessAnalyst, "AutoScroll");
// Setting Property to true
_ObjectSetProperty(hProcessAnalyst, "AutoScroll", -1);
END
```

## **IProcessAnalyst.BackgroundColor [Property][Get/ Set]**

Gets or sets the background color for the Process Analyst.

**Defined As**

- [VBA] Long BackgroundColor
- [Cicode] INT BackgroundColor
- [C++] OLECOLOR BackgroundColor

**Execution Result**

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

**Remarks**

The background is the area under the panes. To calculate the integer value necessary for a color, apply the following formula:

$$(65536 * Blue) + (256 * Green) + (Red)$$

where Red, Green, and Blue are 0-255.

**Limits**

- True (-1): Context menu is enabled
- False (0): Context menu is disabled

**Calling Syntax**

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

### [VBA]

```
Sub Example()
Dim backgroundColor As Long
`Getting Property value
backgroundColor = myPage_AN35.BackgroundColor
`Setting Property value to Red
myPage_AN35.BackgroundColor = 255
End Sub
```

### [Cicode]

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
INT backgroundColor;
// Getting current property value
backgroundColor = _ObjectGetProperty(hProcessAnalyst, "BackgroundColor");
// Setting Property to Red
_ObjectSetProperty(hProcessAnalyst, "BackgroundColor", 255);
END
```

## IProcessAnalyst.CommandSystem [Property][Get]

Gets a reference to the Process Analyst's Command System object. With this object you can execute commands, query command information, and create your own custom commands.

#### Defined As

- [VBA] Object CommandSystem
- [Cicode] OBJECT CommandSystem
- [C++] ICommandSystem\* CommandSystem

#### Execution Result

If the property get succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument.

#### Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

### [VBA]

```
Sub Example()
Dim commandSystem As Object
`Retrieve command system
Set commandSystem = myPage_AN35.CommandSystem
End Sub
```

**[Cicode]**

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
OBJECT hCommandSystem;
// Retrieve command system
hCommandSystem = _ObjectGetProperty(hProcessAnalyst, "CommandSystem");
END
```

## IProcessAnalyst.ContextMenu [Property][Get/Set]

Enables or disables the context menu which is displayed at Run-time when an operator clicks the right mouse button on the graphical display.

**Defines As**

- [VBA] Boolean ContextMenu
- [Cicode] INT ContextMenu
- [C++] VARIANT\_BOOL ContextMenu

**Execution Result**

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument.

**Calling Syntax**

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

**[VBA]**

```
Sub Example()
Dim contextMenu As Boolean
`Getting Property value
contextMenu = myPage_AN35.ContextMenu
`Disable the context menu
myPage_AN35.ContextMenu = False
End Sub
```

**[Cicode]**

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
INT contextMenu;
// Getting current property value
contextMenu = _ObjectGetProperty(hProcessAnalyst, "ContextMenu");
// Disable the context menu
_ObjectSetProperty(hProcessAnalyst, "ContextMenu", 0);
END
```

## IProcessAnalyst.Cursors [Property][Get]

Gets a reference to the Process Analyst's cursors collection. With this object you can create new, and browse existing cursors.

### Defined As

- [VBA] Object Cursors
- [Cicode] OBJECT Cursors
- [C++] ICursors\* Cursors

### Execution Result

If the property get succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument.

### Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

### [VBA]

```
Sub Example()
Dim cursors As Object
`Retrieve cursors collection
Set cursors = myPage_AN35.Cursors
End Sub
```

### [Cicode]

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
OBJECT hCursors;
// Retrieve cursor collection
hCursors = _ObjectGetProperty(hProcessAnalyst, "Cursors");
END
```

## IProcessAnalyst.DataRequestRate [Property][Get/ Set]

Indicates how often (in milliseconds) the Process Analyst Control will request data from the Trends Server(s). Internally the Process Analyst will choose the most optimum request rate for data, but this property can be used to slow the request down further.

### Defined As

- [VBA] Integer DataRequestRate
- [Cicode] INT DataRequestRate

- [C++] short DataRequestRate

#### Execution Result

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument.

#### Remarks

This property is useful for controlling the load on a Trends Server. The higher the figure the less load will be put on the Trends Server(s).

#### Limits

- Minimum = 10 milliseconds
- Maximum = 60000 milliseconds (1 minute)
- Default = 1000 milliseconds (1 second)

#### Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

### [VBA]

```
Sub Example()
Dim requestRate As Integer
`Retrieve request rate
requestRate = myPage_AN35.DataRequestRate
`Set request rate
myPage_AN35.DataRequestRate = 2000
End Sub
```

### [Cicode]

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
INT requestRate;
// Retrieve property value
requestRate = _ObjectGetProperty(hProcessAnalyst, "DataRequestRate");
// Set property value to 2 seconds
_Object SetProperty(hProcessAnalyst, "DataRequestRate", 2000);
END
```

## IProcessAnalyst.DisplayRefreshRate [Property][Get/Set]

Indicates how fast the Process Analyst Control display is updated in milliseconds. The default is an update of 1 second, which provides optimum client performance and visual feedback.

#### Defined As

- [VBA] Integer DisplayRefreshRate

- [Cicode] INT DisplayRefreshRate
- [C++] short DisplayRefreshRate

## ⚠ WARNING

### UNACCEPTABLY SLOW PROGRAM EXECUTION

- Do not specify a Display refresh rate less than 500 ms.
- Do not specify a Number of Samples greater than 500.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

### Remarks

This property is useful for controlling the performance of a client (CPU usage).

### Limits

- Minimum = 10 milliseconds (most machines will not be fast enough to keep up).
- Maximum = 60000 milliseconds (1 minute).
- Default = 1000 milliseconds (1 second).

### Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

## [VBA]

```
Sub Example()
Dim requestRate As Integer
`Retrieve request rate
requestRate = myPage_AN35.DisplayRefreshRate
`Set request rate
myPage_AN35.DisplayRefreshRate = 2000
End Sub
```

## [Cicode]

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
INT requestRate;
// Retrieve request rate
requestRate = _ObjectGetProperty(hProcessAnalyst, "DisplayRefreshRate");
// Set request rate
_Object SetProperty(hProcessAnalyst, "DisplayRefreshRate", 2000);
END
```

# IProcessAnalyst.DisplaySize[Property][Get/Set]

Gets or sets the size of graphical elements in the control's design and runtime surfaces.

## Defined As

- [VBA] Long DisplaySize
- [Cicode] INT DisplaySize
- [C++] DisplaySize DisplaySize

## Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

## Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

## [VBA]

```
Sub Example()
Dim displaySize As Long

'Getting Property value
displaySize = myPage_AN35.DisplaySize
'Setting Property value
myPage_AN35.DisplaySize = 1
End Sub
```

## [Cicode]

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");

// Getting property value
INT nDisplayMode = _ObjectGetProperty(hProcessAnalyst, "DisplaySize");
// Setting property value
_Object SetProperty(hProcessAnalyst, "DisplaySize", 1);
END
```

## See Also

[DisplaySize\[Enumeration\]](#)

# IProcessAnalyst.Language [Property] [Get/Set]

This function allows dynamic changing of the user interface to the language specified.

## Defined As

- [VBA] String Language
- [Cicode] STRING Language
- [C++] BSTR Language

### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

### Remarks

To change languages you need to have additional localized resource .dll files alongside the main resources.dll file. Additional language .dll files are named (and have to be named) using the format "Resources\_<languagecode>.dll". The Process Analyst expects this format or the language will not be loaded.

For example, if you have a Chinese resource dll named "Resources\_zh-CN.dll", set the Language property to "zh-CN". The .dll files are named according to the RFC 1766 standard for specifying culture names.

Specifying "." resets the language back to the default.

---

**Note:** This method is not necessary to be called if you are using Plant SCADA's multilanguage feature to make the Process Analyst switch languages. For details, see Multi-language Support.

### Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

## [VBA]

```
Sub Example()
Dim language As String
`Retrieve current language
language = myPage_AN35.Language
`Set language to Japanese
myPage_AN35.Language = "ja-JP"
End Sub
```

## [Cicode]

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
STRING language;
// Retrieve current language
language = _ObjectGetProperty(hProcessAnalyst, "Language");
// Set language to Japanese
_Object SetProperty(hProcessAnalyst, "Language", "ja-JP");
END
```

# IProcessAnalyst.LastSelectedPen [Property][Get]

Retrieves the last selected pen on the Process Analyst.

### Defined As

- [VBA] Object LastSelectedPen

- [Cicode] OBJECT LastSelectedPen
- [C++] IPen\* LastSelectedPen

#### Execution Result

If the property get succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument.

#### Remarks

The last selected pen is also referred to as the "primary" selection. If there are no pens in the view, an invalid object will be returned.

#### Limits

- A reference to the primary selected pen.
- Invalid object when there are no pens on the display.

#### Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

### [VBA]

```
Sub Example()
Dim selectedPen As Object
`Retrieve primary selection
Set selectedPen = myPage_AN35.LastSelectedPen
End Sub
```

### [Cicode]

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
OBJECT selectedPen;
// Retrieve primary selection
selectedPen = _ObjectGetProperty(hProcessAnalyst, "LastSelectedPen");
END
```

## IProcessAnalyst.LockedPens [Property][Get/Set]

Determines whether every the pen across every pane in the Process Analyst control are locked together.

#### Defined As

- [VBA] Boolean LockedPens
- [Cicode] INT LockedPens
- [C++] VARIANT\_BOOL LockedPens

#### Execution Result

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return

value will be InvalidArgument.

#### Remarks

While this property is enabled, any operation applied to the selected pen is applied to every pen. When the property is disabled, the pens will lose the lock logic, and any interaction technique will apply to the individual pen with selection focus.

If this property is disabled and then enabled, every pen assumes the same scale, timespan, and end time position as the selected pen.

#### Limits

- True (-1): Pens are locked.
- False (0): Pens are unlocked.

#### Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

### [VBA]

```
Sub Example()
Dim locked As Boolean
`Get current locked status
locked = myPage_AN35.LockedPens
`Turn off locked Pens
myPage_AN35.LockedPens = False
End Sub
```

### [Cicode]

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
OBJECT lockedPens;
// Get current locked status
lockedPens = _ObjectGetProperty(hProcessAnalyst, "LockedPens");
// Turn off locked Pens
_ObjectSetProperty(hProcessAnalyst, "LockedPens", 0);
END
```

## IProcessAnalyst.NumberofSamples[Property][Get/ Set]

Specifies the date/time axis span of each pen in number of samples. More or less detail for each pen can be displayed by increasing or decreasing the value of this property respectively.



UNACCEPTABLY SLOW PROGRAM EXECUTION

- Do not specify a Display refresh rate less than 500 ms.
- Do not specify a Number of Samples greater than 500.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** The value entered into the **Number of Samples** box in the Process Analyst Properties dialog box is closely tied to the display resolution. The default setting is ideal for screen resolutions from 1024x768 to 1280x1024. The association between Number of Samples and the display resolution occurs because for each sample shown on screen the Process Analyst attempts to leave a small gap to allow for sample markers. Because the Process Analyst shows samples when they occur, it requires less data than a traditional trend client. Retrieving data is expensive and the more data you retrieve the more time the request takes. *It is recommended that this parameter not exceed 500.*

#### Defined As

- [VBA] Integer NumberofSamples
- [Cicode] INT NumberofSamples
- [C++] short NumberofSamples

#### Execution Result

If the property get/set succeeds the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

#### Remarks

This property is useful for controlling the performance of a client. (CPU usage).

By dividing a pen's time span by the value of this property, you can calculate the current display period of the pen. The Process Analyst will only display a maximum of one sample per display period. See Data Compaction for details.

#### Limits

- Minimum = 10
- Maximum = 5000
- Default = 300

#### Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

### [VBA]

```
Sub Example()
Dim numOfSamples As Integer
`Retrieve number of samples
numOfSamples = myPage_AN35.NumberOfSamples
`Set request rate
myPage_AN35.NumberOfSamples = numOfSamples
End Sub
```

**[Cicode]**

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
INT numOfSamples;
// Retrieve number of samples
numOfSamples = _ObjectGetProperty(hProcessAnalyst, "NumberOfSamples");
// Set request rate
_ObjectSetProperty(hProcessAnalyst, "NumberOfSamples", 500);
END
```

**See Also**

[Exporting Pen Data](#)

## IProcessAnalyst.ObjectView [Property][Get]

Gets a reference to the ObjectView object. With this object you can manipulate the look of the ObjectView.

**Defined As**

- [VBA] Object ObjectView
- [Cicode] OBJECT ObjectView
- [C++] IObjectView\* ObjectView

**Execution Result**

If the property get succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument.

**Calling Syntax**

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

**[VBA]**

```
Sub Example()
Dim objectView As Object
`Retrieve the objectview
Set objectView = myPage_AN35.ObjectView
End Sub
```

**[Cicode]**

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
OBJECT objectView;
// Retrieve the objectview
objectView = _ObjectGetProperty(hProcessAnalyst, "ObjectView");
END
```

# IProcessAnalyst.Panes [Property][Get]

Gets a reference to the Panes collection. With this object you can create new, and browse existing panes.

## Defined As

- [VBA] Object Panes
- [Cicode] OBJECT Panes
- [C++] IPanes\* Panes

## Execution Result

If the property get succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument.

## Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

## [VBA]

```
Sub Example()
Dim panes As Object
`Retrieve the panes collection
Set panes = myPage_AN35.Panes
End Sub
```

## [Cicode]

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
OBJECT panes;
// Retrieve the panes collection
panes = _ObjectGetProperty(hProcessAnalyst, "Panes");
END
```

# IProcessAnalyst.PrimaryPath [Property][Get/Set]

Specifies the primary location for saving and loading Process Analyst views.

## Defined As

- [VBA] String PrimaryPath
- [Cicode] STRING PrimaryPath
- [C++] BSTR PrimaryPath

## Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

### Remarks

The primary and secondary path properties together provide a file redundancy option for large systems that need to store Process Analyst Views in a shared location. Whenever a load operation occurs from either of these locations, the loaded file will be synchronized with each location, such that the latest version of the file appears in both locations.

### Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

## [VBA]

```
Sub Example()
Dim path As String
`Retrieve the path
path = myPage_AN35.PrimaryPath
`Set the path
myPage_AN35.PrimaryPath = "\\\computer1\PA Views"
End Sub
```

## [Cicode]

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
STRING path;
// Retrieve the path
path = _ObjectGetProperty(hProcessAnalyst, "PrimaryPath");
// Set the path
_Object SetProperty(hProcessAnalyst, "PrimaryPath", "\\\computer1\PA Views");
END
```

## See Also

[IProcessAnalyst.SecondaryPath \[Property\]\[Get/Set\]](#)

[IProcessAnalyst.LoadFromFile \[Method\]](#)

[IProcessAnalyst.SaveToFile \[Method\]](#)

# IProcessAnalyst.SecondaryPath [Property][Get/ Set]

Specifies the secondary location for saving and loading Process Analyst views.

### Defined As

- [VBA] String SecondaryPath
- [Cicode] STRING SecondaryPath
- [C++] BSTR SecondaryPath

### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

### Remarks

The secondary and primary path properties together provide a file redundancy option for large systems that need to store Process Analyst Views in a shared location. Whenever a load operation occurs from either of these locations, the loaded file will be synchronized with each location, such that the latest version of the file appears in both locations.

### Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

## [VBA]

```
Sub Example()
Dim path As String
`Retrieve the path
path = myPage_AN35.PrimaryPath
`Set the path
myPage_AN35.SecondaryPath = "\computer1\PA Views"
End Sub
```

## [Cicode]

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
STRING path;
// Retrieve the path
path = _ObjectGetProperty(hProcessAnalyst, "SecondaryPath");
// Set the path
_Object SetProperty(hProcessAnalyst, "SecondaryPath", "\computer1\PA Views");
END
```

## See Also

- [IProcessAnalyst.LoadFromFile \[Method\]](#)
- [IProcessAnalyst.PrimaryPath \[Property\]\[Get/Set\]](#)
- [IProcessAnalyst.SaveToFile \[Method\]](#)

## IProcessAnalyst.Toolbars [Property][Get]

Gets a reference to the Toolbars collection. With this object you can browse and modify existing toolbars.

### Defined As

- [VBA] Object Toolbars
- [Cicode] OBJECT Toolbars

- [C++] IToolbars\* Toolbars

#### Execution Result

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

#### Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

### [VBA]

```
Sub Example()
Dim toolbars As Object
`Retrieve the toolbars collection
Set toolbars = myPage_AN35.Toolbars
End Sub
```

### [Cicode]

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
OBJECT toolbars;
// Retrieve the toolbars collection
toolbars = _ObjectGetProperty(hProcessAnalyst, "Toolbars");
END
```

## IProcessAnalyst.ViewLocation [Property][Get]

The location of last successfully loaded or saved PAV file. e.g. If you loaded a PAV file test.pav from the project folder, this property will be "[Local]test.pav".

#### Defined As

- [VBA] STRING ViewLocation
- [Cicode] STRING ViewLocation
- [C++] BSTR ViewLocation

#### Execution Result

If the property get succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument.

#### Calling Syntax

Assumes your Process Analyst's Object Name has been named "AN35".

### [VBA]

```
Sub Example()
Dim sViewLocation As String
`Retrieve view location
```

```
Set sViewLocation = myPage_AN35.ViewLocation
End Sub
```

### [Cicode]

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
STRING sViewLocation;
// Retrieve view location
sViewLocation = _ObjectGetProperty(hProcessAnalyst, "ViewLocation");
END
```

## IProcessAnalyst.WritePrivilegeLevel [Property][Get]

Returns the privilege level necessary to save Process Analyst views to the Primary and Secondary paths.

#### Defined As

- [VBA] Integer WritePrivilegeLevel
- [Cicode] INT WritePrivilegeLevel
- [C++] short WritePrivilegeLevel

#### Execution Result

If the property get succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument.

#### Remarks

The privilege cannot be set via automation. It needs to be set in the property pages at design time (for example, in Graphics Builder).

#### Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

### [VBA]

```
Sub Example()
Dim privilege As Integer
`Retrieve the privilege
privilege = myPage_AN35.WritePrivilegeLevel
End Sub
```

### [Cicode]

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
INTEGER privilege;
// Retrieve the privilege
```

```
privilege = _ObjectGetProperty(hProcessAnalyst, "WritePrivilegeLevel");
END
```

## See Also

[IProcessAnalyst.SaveToFile \[Method\]](#)  
[IProcessAnalyst.SecondaryPath \[Property\]\[Get/Set\]](#)

# IProcessAnalyst.ZoomMode [Property][Get/Set]

Enables or disables the box zooming mode for the Process Analyst.

### Defined As

- [VBA] Boolean ZoomMode
- [Cicode] INT ZoomMode
- [C++] VARIANT\_BOOL ZoomMode

### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

### Remarks

Setting this mode will verify only box zooming operations can occur; other operations such as interactive scrolling and scaling will cease.

### Limits

- True (-1): Enable zoom mode.
- False (0): Disable zoom mode.

### Calling Syntax

Assumes you have a page called "myPage" and the Process Analyst has been named "AN35".

## [VBA]

```
Sub Example()
Dim zoomMode As Boolean
`Retrieve the mode
zoomMode = myPage_AN35.ZoomMode
`Set the path
myPage_AN35.ZoomMode = True
End Sub
```

## [Cicode]

```
FUNCTION Example()
OBJECT hProcessAnalyst = ObjectByName("AN35");
```

```
INT zoomMode;
// Retrieve the zoomMode
zoomMode = _ObjectGetProperty(hProcessAnalyst, "ZoomMode");
// Set the zoomMode
_ObjectSetProperty(hProcessAnalyst, "ZoomMode", -1);
END
```

## IToolbar Interface

### Defined As

- [VBA] Object
- [Cicode] OBJECT
- [C++] IToolbar

### Methods (0)

### Properties (2)

[IToolbar.Visible \[Property\]\[Get/Set\]](#)

[IToolbar.Buttons \[Property\]\[Get\]](#)

## IToolbar.Buttons [Property][Get]

Gets this Toolbars collection of Buttons.

### Defined As

- [VBA] Object Buttons
- [Cicode] OBJECT Buttons
- [C++] IToolbarButtons\* Buttons

### Execution Result

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

### Calling Syntax

This example assumes there is a valid Toolbar object as retrieved from a Process Analyst's Toolbars collection.  
(for example, VBA: ProcessAnalyst.Toolbars.Item(1))

## [VBA]

```
Sub Example(Toolbar As Object)
Dim buttons As Object
`Getting Property value
Set buttons = Command.Buttons
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hToolbar)
// Getting property value
OBJECT hButtons = _ObjectGetProperty(hToolbar, "Buttons");
END
```

# IToolbar.Visible [Property][Get/Set]

Gets and Sets this Toolbars Visible state.

**Defined As**

- [VBA] Boolean Visible
- [Cicode] INT Visible
- [C++] VARIANT\_BOOL Visible

**Execution Result**

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

**Limits**

- True (-1): Visible
- False (0): Invisible

**Remarks**

Both the main and navigation toolbars can be hidden.

If a toolbar is not visible, the buttons tied to it will not appear.

---

**Note:** The navigation toolbar is automatically hidden when the chart contains no pens. Changing the value of this property while there are no pens will not have an effect until a pen is added.

**Calling Syntax**

This example assumes there is a valid Toolbar object as retrieved from a Process Analyst's Toolbars collection.  
(for example, VBA: ProcessAnalyst.Toolbars.Item(1))

**[VBA]**

```
Sub Example(Toolbar As Object)
Dim visible As Boolean
`Getting Property value
visible = Command.Visible
`Setting Property value
Command.Visible = False
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hToolbar)
// Getting property value
INT nVisible = _ObjectGetProperty(hToolbar, "Visible");
// Setting property value
_ObjectSetProperty(hToolbar, "Visible", 0);
END
```

**IToolbars Interface****Defined As**

- [VBA] Object
- [Cicode] OBJECT
- [C++] IToolbar

**Methods (0)****Properties (3)**

[IToolbars.Count \[Property\]\[Get\]](#)  
[IToolbars.Item \[Property\]\[Get\]](#)  
[IToolbars.\\_NewEnum \[Property\]\[Get\]](#)

## IToolbars.Item [Property][Get]

Gets the Toolbar at a supplied index location in this collection.

**Defined As**

- [VBA] Item(index As Long) as Object
- [Cicode] OBJECT Item(INT index)
- [C++] Item(int index, IToolbar\* Item)

**Parameters**

index

[in] Indicates the index location of the child item to return from this collection. (One based) 1 = Main toolbar; 2 = Navigation toolbar.

**Execution Result**

If the property get succeeds, the return value will be Success. If the index is out of range, the return value will be InvalidArgument.

**Calling Syntax**

This example assumes there is a valid Toolbars collection as retrieved from an ObjectView. (for example, VBA: objectView.Toolbars)

**[VBA]**

```
Sub Example(Toolbars As Object)
Dim toolbar As Object
`Getting Property value
Set toolbar = Toolbars.Item(1)
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hToolbars)
// Getting property value
OBJECT hToolbar = _ObjectCallMethod(hToolbars, "get_Item", 1);
END
```

## IToolbars.\_NewEnum [Property][Get]

This allows For.. Each.. Next integration in VB.

**Calling Syntax**

This example assumes there is a valid Toolbars collection as retrieved from an ObjectView (for example, VBA: objectView.Toolbars). This property is not applicable to Cicode.

**[VBA]**

```
Sub Example(Toolbars As Object)
Dim toolbar As Object
Dim count Object
`Using Property
For Each toolbar In Toolbars
count = count + 1
Next toolbar
End Sub
```

## IToolbars.Count [Property][Get]

Gets the number of Toolbars in this collection.

**Defined As**

- [VBA] Long Count
- [Cicode] INT Count
- [C++] int Count

**Execution Result**

If the property get succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument.

**Calling Syntax**

This example assumes there is a valid Toolbars collection as retrieved from an ObjectView. (for example, VBA: objectView.Toolbars).

**[VBA]**

```
Sub Example(Toolbars As Object)
Dim count As Long
`Getting Property value
count = Toolbars.Count
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hToolbars)
// Getting property value
INT nCount = _ObjectGetProperty(hToolbars, "Count");
END
```

**IToolBarButton Interface****Defined As**

- [VBA] Object
- [Cicode] OBJECT
- [C++] IToolbar

**Methods (0)****Properties (1)**

[IToolBarButton.CommandId \[Property\]\[Get\]](#)

## IToolBarButton.CommandId [Property][Get]

Gets the ID of the associated command for this button.

**Defined As**

- [VBA] String CommandId
- [Cicode] STRING CommandId
- [C++] BSTR CommandId

**Calling Syntax**

This example assumes there is a valid ToolbarButtons collection as retrieved from an ObjectView (for example, VBA: objectView.Toolbars.Item(1).Buttons(1).CommandID).

**[VBA]**

```
Sub Example(Button As Object)
Dim commandId As String
`Getting Property value
commandId = Buttons.CommandId
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hButton)
// Getting property value
STRING nCommandId = _ObjectGetProperty(hButton, "CommandId");
END
```

**IToolbarButtons Interface****Defined As**

- [VBA] Object
- [Cicode] OBJECT
- [C++] IToolbar

**Methods (3)**

- IToolbarButtons.Add [Method]
- IToolbarButtons.Remove [Method]
- IToolbarButtons.RemoveAll [Method]

**Properties (3)**

- IToolbarButtons.Count [Property][Get]
- IToolbarButtons.Item [Property][Get]
- IToolbarButtons.\_NewEnum [Property][Get]

## IToolbarButtons.\_NewEnum [Property][Get]

This allows For.. Each.. Next integration in VB.

**Calling Syntax**

This example assumes there is a valid ToolbarButtons collection as retrieved from an ObjectView (for example, VBA: objectView.Toolbars.Item(1).Buttons). This property is not applicable to Cicode.

**[VBA]**

```
Sub Example(Buttons As Object)
Dim button As Object
```

```
Dim count Object
`Using Property
For Each button In Buttons
count = count + 1
Next button
End Sub
```

## IToolbarButtons.Add [Method]

Adds a toolbar button linked to the command identified by the supplied Command Id to this Toolbar.

### Defined As

- [VBA] Add(CommandId as String)
- [Cicode] Add (STRING CommandId)
- [C++] HRESULT Add (BSTR CommandId)

### Parameters

CommandId

[in] The Command ID of a command to link to the new button that is being added.

### Execution Result

If this method succeeds, the return value will be Success. If the command ID is invalid, the return value will be InvalidArgument.

### Remarks

If this method succeeds, the ID supplied will be linked to the new button that is added. The Commands properties will be applied to that button. (its icon, tooltip, security) If this button is pressed, the CommandExecuted event will raise with this Command ID.

### Calling Syntax

This example assumes there is a valid ToolbarButtons collection as retrieved from an ObjectView (for example, VBA: objectView.Toolbars.Item(1).Buttons).

## [VBA]

```
Sub Example(Buttons As Object)
Buttons.Add("Citect_Command_Help")
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hButtons)
_ObjectCallMethod(hButtons, "Add", "Citect_Command_Help");
END
```

## See Also

[CommandExecuted \[Event\]](#)

## IToolbarButtons.Count [Property][Get]

Gets the number of Buttons in this collection.

### Defined As

- [VBA] Long Count
- [Cicode] INT Count
- [C++] int Count

### Calling Syntax

This example assumes there is a valid ToolbarButtons collection as retrieved from an ObjectView. (for example, VBA: objectView.Toolbars.Item(1).Buttons).

### [VBA]

```
Sub Example(Buttons As Object)
Dim count As Long
`Getting Property value
count = Buttons.Count
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hButtons)
// Getting property value
INT nCount = _ObjectGetProperty(hButtons, "Count");
END
```

## IToolbarButtons.Item [Property][Get]

Gets the button at a supplied index location in this collection.

### Defined As

- [VBA] Item(index As Long) as Object
- [Cicode] OBJECT Item(INT index)
- [C++] Item(int index, IToolBarButton\* Item)

### Parameters

index

[in] Indicates the index location of the button to return from this collection. (One based)

### Execution Results

If the property get succeeds, the return value will be success. If the index is out of range, the return value will be InvalidArgument. If the method does not succeed, the return value will be GeneralFailure.

**Calling Syntax**

This example assumes there is a valid ToolbarButtons collection as retrieved from an ObjectView (for example, VBA: objectView.Toolbars.Item(1).Buttons).

**[VBA]**

```
Sub Example(Buttons As Object)
Dim toolbar As Object
`Getting Property value
Set toolbar = Buttons.Item(1)
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hButtons)
// Getting property value
OBJECT hToolbar = _ObjectCallMethod(hButtons, "get_Item", 1);
END
```

## IToolbarButtons.Remove [Method]

Removes a button from this toolbar at the supplied index.

**Defined As**

- [VBA] Remove(Index as Integer)
- [Cicode] Remove (INT Index)
- [C++] HRESULT Remove (int Index)

**Parameters****Index**

[in] The index of the button to remove from this toolbar. (1 Based)

**Execution Results**

If the method succeeds, the return value will be Success. If the index is out of range, the return value will be InvalidRange. If the method does not succeed, the return value will be GeneralFailure.

**Calling Syntax**

This example assumes there is a valid ToolbarButtons collection as retrieved from an ObjectView. (for example, VBA: objectView.Toolbars.Item(1).Buttons).

**[VBA]**

```
Sub Example(Buttons As Object)
Buttons.Remove(1)
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hButtons)
    _ObjectCallMethod(hButtons, "Remove", 1);
END
```

## IToolbarButtons.RemoveAll [Method]

Removes every button from this Toolbar.

**Defined As**

- [VBA] RemoveAll()
- [Cicode] RemoveAll()
- [C++] HRESULT RemoveAll()

**Execution Results**

If this method succeeds, the return value will be Success. If this method does not succeed, the return value will be GeneralFailure.

**Calling Syntax**

This example assumes there is a valid ToolbarButtons collection as retrieved from an ObjectView. (for example, VBA: objectView.Toolbars.Item(1).Buttons)

**[VBA]**

```
Sub Example(Buttons As Object)
    Buttons.RemoveAll()
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hButtons)
    _ObjectCallMethod(hButtons, "RemoveAll");
END
```

**ITrendCursor Interface****Defined As**

- [VBA] Object
- [Cicode] OBJECT
- [C++] ITrendCursor

**Methods**

- [ITrendCursor.GetValue \[Method\]](#)

- ITrendCursor.Delete [Method]

#### Properties

- ITrendCursor.Color [Property][Get/Set]
- ITrendCursor.Width [Property][Get/Set]
- ITrendCursor.Position [Property][Get/Set]
- ITrendCursor.Visible [Property][Get/Set]
- ITrendCursor.Collection [Property][Get]
- ITrendCursor.Name [Property][Get/Set]
- ITrendCursor.PenLabelVisible [Property][Get/Set]
- ITrendCursor.PenLabelWidth [Property][Get/Set]
- ITrendCursor.PenLabelHeight [Property][Get/Set]
- ITrendCursor.PenLabelX [Property][Get/Set]
- ITrendCursor.PenLabelY [Property][Get/Set]
- ITrendCursor.LabelsLocked [Property][Get/Set]

## ITrendCursor.Collection [Property][Get]

Obtain a reference to the ICursors collection that contains the cursor.

#### Defined As

- [VBA] Object Collection
- [Cicode] OBJECT Collection
- [C++] HRESULT Collection(ICursors \*\*cursor)

#### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the cursor is deleted, the return value will be GeneralFailure.

#### Calling Syntax

This example assumes you have a valid reference to a cursor.

### [VBA]

```
Sub Example(cursor As Object)
Dim cursors As Object
`Getting Property collection
Set cursors = cursor.Collection
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hCursor)
```

```
OBJECT hCursors;
// Getting collection
hCursors = _ObjectGetProperty(hCursor, "Collection");
END
```

## ITrendCursor.Color [Property][Get/Set]

Gets or Sets the line color of the cursor.

### Defined As

- [VBA] Long Color
- [Cicode] INT Color
- [C++] OLE\_COLOR Color

### Execution Result

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument. If the cursor is deleted the return value will be GeneralFailure.

### Remarks

The Cicode function PackedRGB can be used to convert an RGB color specification to the OLE\_COLOR type used by the Process Analyst.

### Calling Syntax

This example assumes you have a valid reference to a cursor.

### [VBA]

```
Sub Example(cursor As Object)
Dim trendCursorColor As Long
`Getting Property value
trendCursorColor = cursor.Color
`Setting Property value (to red)
cursor.Color = 255
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hCursor)
INT trendCursorColor;
// Getting current property value
trendCursorColor = _ObjectGetProperty(hCursor, "Color");
// Setting Property to blue
_ObjectSetProperty(hCursor, "Color", PackedRGB(0, 0, 255));
END
```

# ITrendCursor.Delete [Method]

Deletes the cursor.

## Defined As

- [VBA] Delete()
- [Cicode] Delete()
- [C++] HRESULT Delete()

## Execution Result

If the function succeeds, the return value will be Success. If the cursor is deleted, the return value will be GeneralFailure.

## Remarks

This method will remove the cursor from the Process Analyst. Any current references to the cursor will continue to be valid; however, operations on them will result in errors.

## Calling Syntax

This example assumes you have a valid reference to a cursor.

## [VBA]

```
Sub Example(cursor As Object)
cursor.Delete
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hCursor)
_ObjectCallMethod(hCursor, "Delete");
END
```

# ITrendCursor.GetValue [Method]

Gets the value at the cursor for the given pen.

## Defined As

- [VBA] GetValue(pen As Object, asLocal As Boolean, time As Date, milli As Integer, value As String)
- [Cicode] GetValue(OBJECT pen, INT asLocal, REAL time, INT milli, STRING value)
- [C++] HRESULT Create GetValue(IPen\* pen, VARIANT\_BOOL asLocal, DATE \*time, short \*milli, BSTR \*value)

## Parameters

pen

[in] The pen for which the value is necessary.

asLocal

[in] Set to True (-1) if returned time is necessary in Local form (False (0) for UTC).

time

[out] The time represented by the cursor position. This is accurate to one second and needs to be combined with milli to give millisecond accuracy.

milli

[out] Added to time (see above) to give cursor time in millisecond accuracy.

value

[out] The value of the trend for the given pen at the returned time.

### Execution Result

If the function succeeds, the return value will be Success. If one of the return variables are bad, then the return value will be InvalidArgument. If the cursor is deleted, the return value will be GeneralFailure.

### Calling Syntax

This example assumes you have a valid reference to a cursor and a pen.

## [VBA]

```
Sub Example(cursor As Object, pen As Object)
Dim asLocal As Boolean
Dim cursorTime As Date
Dim milli As Integer
Dim cursorValue As String
asLocal = 0
cursor.GetValue pen, asLocal, cursorTime, milli, cursorValue
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hCursor, OBJECT hPen)
INT asLocal = 0;
REAL time;
INT milli;
STRING value;
_ObjectCallMethod(hCursor , "GetValue", hPen, asLocal, time, milli, value);
END
```

# ITrendCursor.LabelsLocked [Property][Get/Set]

Get or Set whether the cursor label positions are locked.

### Defined As

- [VBA] Boolean LabelsLocked
- [Cicode] INT LabelsLocked
- [C++] VARIANT\_BOOL LabelsLocked

### Limits

- True (-1): Labels are locked
- False (0): Labels are unlocked

#### Remarks

If labels are locked, they will not move when the cursor position is changed.

#### Calling Syntax

This example assumes you have a valid reference to a cursor.

### [VBA]

```
Sub Example(cursor As Object)
Dim labelsLocked As Boolean
`Getting Property value
labelsLocked = cursor.LabelsLocked
`Setting Property value (False)
cursor.LabelsLocked = False
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hCursor)
INT labelsLocked;
// Getting current property value
labelsLocked = _ObjectGetProperty(hCursor, "LabelsLocked");
// Setting Property to False (0)
_ObjectSetProperty(hCursor, "LabelsLocked", 0);
END
```

## ITrendCursor.Name [Property][Get/Set]

Get or Set the Name of the cursor.

#### Defined As

- [VBA] String Name
- [Cicode] STRING Name
- [C++] BSTR Name

#### Execution Result

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the name is 0 characters or greater than 250, the return value will be InvalidArgument.

If the cursor is deleted, the return value will be GeneralFailure.

#### Remarks

When setting the name property, remember that cursor names needs to be unique. Setting the Name property will not succeed if a cursor with that name already exists.

**Calling Syntax**

This example assumes you have a valid reference to a cursor.

**[VBA]**

```
Sub Example(cursor As Object)
Dim name As String
`Getting Property value
name = cursor.Name
`Setting Property value
cursor.Name = "NewCursor"
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hCursor)
STRING name;
// Getting current property value
name = _ObjectGetProperty(hCursor, "Name");
// Setting Property
_ObjectSetProperty(hCursor, "Name", "NewCursor");
END
```

## ITrendCursor.PenLabelHeight [Property][Get/Set]

Get or Set the label height of the specified pen on this cursor.

**Defined As**

- [VBA] Double PenLabelHeight(pen As Object)
- [Cicode] REAL PenLabelHeight (OBJECT pen)
- [C++] HRESULT PenLabelHeight (IPen\* pen, double labelHeight)

**Parameters**

pen

[in] The pen for which cursor label is to be referenced.

**Execution Result**

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument. If the cursor is deleted the return value will be GeneralFailure.

**Remarks**

The value of height is represented in pixels.

**Calling Syntax**

This example assumes you have a valid reference to a cursor and a pen.

**[VBA]**

```
Sub Example(cursor As Object, pen As Object)
Dim labelHeight As Double
`Getting Property value
labelHeight = cursor.PenLabelHeight(pen)
`Setting Property value (100)
cursor.PenLabelHeight (pen) = 100
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hCursor, OBJECT hPen)
REAL labelHeight;
// Getting current property value
labelHeight = _ObjectCallMethod(hCursor , "get_PenLabelHeight", hPen);
// Setting Property to 100
_ObjectCallMethod(hCursor , "put_PenLabelHeight", hPen, 100);
END
```

**See Also**

[ITrendCursor.PenLabelWidth \[Property\]\[Get/Set\]](#)

[ITrendCursor.PenLabelX \[Property\]\[Get/Set\]](#)

[ITrendCursor.PenLabelY \[Property\]\[Get/Set\]](#)

## ITrendCursor.PenLabelVisible [Property][Get/Set]

Get or Set the label visibility of the specified pen on this cursor.

**Defined As**

- [VBA] Boolean PenLabelVisible(pen As Object)
- [Cicode] INT PenLabelVisible(OBJECT pen)
- [C++] HRESULT PenLabelVisible(IPen\* pen, VARIANT\_BOOL labelVisible)

**Parameters**

pen

[in] The pen for which cursor label is to be referenced.

**Execution Result**

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the cursor is deleted, the return value will be GeneralFailure.

**Limits**

- True (-1): Label is visible.
- False (0): Label is hidden.

### Remarks

Setting the visibility of the cursor using the Visible property will override the pen label visibility. For example, if a particular label is hidden using PenLabelVisible, this label will be shown again if Visible is set to True (-1).

### Calling Syntax

This example assumes you have a valid reference to a cursor and a pen.

## [VBA]

```
Sub Example(cursor As Object, pen As Object)
Dim labelVisible As Boolean
`Getting Property value
labelVisible = cursor.PenLabelVisible(pen)
`Setting Property value (False)
cursor.PenLabelVisible(pen) = False
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hCursor, OBJECT hPen)
INT labelVisible;
// Getting current property value
labelVisible = _ObjectCallMethod(hCursor, "get_PenLabelVisible", hPen);
// Setting Property to FALSE
_ObjectCallMethod(hCursor , "put_PenLabelVisible", hPen, 0);
END
```

## See Also

[ITrendCursor.Visible \[Property\]\[Get/Set\]](#)

# ITrendCursor.PenLabelWidth [Property][Get/Set]

Get or Set the label width of the specified pen on this cursor.

### Defined As

- [VBA] Double PenLabelWidth(pen As Object)
- [Cicode] REAL PenLabelWidth (OBJECT pen)
- [C++] HRESULT PenLabelWidth (IPen\* pen, double labelWidth)

### Parameters

pen

[in] The pen for which cursor label is to be referenced.

### Execution Result

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument. If the cursor is deleted the return value will be GeneralFailure.

**Remarks**

The value of width is represented in pixels.

**Calling Syntax**

This example assumes you have a valid reference to a cursor and a pen.

**[VBA]**

```
Sub Example(cursor As Object, pen As Object)
Dim labelWidth As Double
`Getting Property value
labelWidth = cursor.PenLabelWidth(pen)
`Setting Property value (100)
cursor.PenLabelWidth(pen) = 100
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hCursor, OBJECT hPen)
REAL labelWidth;
// Getting current property value
labelWidth = _ObjectCallMethod(hCursor , "get_PenLabelWidth", hPen);
// Setting Property to 100
_ObjectCallMethod(hCursor , "put_PenLabelWidth", hPen, 100);
END
```

**See Also**

[ITrendCursor.PenLabelHeight \[Property\]\[Get/Set\]](#)

[ITrendCursor.PenLabelX \[Property\]\[Get/Set\]](#)

[ITrendCursor.PenLabelY \[Property\]\[Get/Set\]](#)

## ITrendCursor.PenLabelX [Property][Get/Set]

Get or Set the label's X-Axis position of the specified pen on this cursor.

**Defined As**

- [VBA] Double PenLabelX(pen As Object)
- [Cicode] REAL PenLabelX (OBJECT pen)
- [C++] HRESULT PenLabelX (IPen\* pen, double labelX)

**Parameters**

pen

[in] The pen for which cursor label is to be referenced.

**Execution Result**

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return

value will be InvalidArgument. If the cursor is deleted the return value will be GeneralFailure.

#### Remarks

The label position is represented in pixels.

#### Calling Syntax

This example assumes you have a valid reference to a cursor and a pen.

### [VBA]

```
Sub Example(cursor As Object, pen As Object)
Dim labelX As Double
`Getting Property value
labelX = cursor.PenLabelX(pen)
`Setting Property value (100)
cursor.PenLabelX(pen) = 100
End Sub
```

### [Cicode]

```
FUNCTION Example(OBJECT hCursor, OBJECT hPen)
REAL labelX;
// Getting current property value
labelX = _ObjectCallMethod(hCursor , "get_PenLabelX", hPen);
// Setting Property to 100
_ObjectCallMethod(hCursor , "put_PenLabelX", hPen, 100);
END
```

### See Also

[ITrendCursor.PenLabelWidth \[Property\]\[Get/Set\]](#)

[ITrendCursor.PenLabelHeight \[Property\]\[Get/Set\]](#)

[ITrendCursor.PenLabelY \[Property\]\[Get/Set\]](#)

## ITrendCursor.PenLabelY [Property][Get/Set]

Get or Set the label's Y-Axis position of the specified pen on this cursor.

#### Defined As

- [VBA] Double PenLabelY(pen As Object)
- [Cicode] REAL PenLabelY (OBJECT pen)
- [C++] HRESULT PenLabelY (IPen\* pIPen, double labelY)

#### Remarks

The label position is represented in pixels

#### Calling Syntax

This example assumes you have a valid reference to a cursor and a pen.

**[VBA]**

```
Sub Example(cursor As Object, pen As Object)
Dim labelY As Double
`Getting Property value
labelY = cursor.PenLabelY(pen)
`Setting Property value (100)
cursor.PenLabelY(pen) = 100
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hCursor, OBJECT hPen)
REAL labelY;
// Getting current property value
labelY = _ObjectCallMethod(hCursor , "get_PenLabelY", hPen);
// Setting Property to 100
_ObjectCallMethod(hCursor , "put_PenLabelY", hPen, 100);
END
```

**See Also**

[ITrendCursor.PenLabelWidth \[Property\]\[Get/Set\]](#)

[ITrendCursor.PenLabelHeight \[Property\]\[Get/Set\]](#)

[ITrendCursor.PenLabelX \[Property\]\[Get/Set\]](#)

## ITrendCursor.Position [Property][Get/Set]

Get or Set the cursor's physical position in the Process Analyst.

**Defined As**

- [VBA] Long Position
- [Cicode] INT Position
- [C++] int Position

**Execution Result**

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the cursor is deleted, the return value will be GeneralFailure.

**Remarks**

The cursor position is measured as the number of pixels from the left of the Process Analyst graph.

**Calling Syntax**

This example assumes you have a valid reference to a cursor.

**[VBA]**

```
Sub Example(cursor As Object)
Dim position As Integer
`Getting Property value
position = cursor.Position
`Setting Property value
cursor.Position = 300
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hCursor)
INT position;
// Getting current property value
position = _ObjectGetProperty(hCursor, "Position");
// Setting Property to (300)
_ObjectSetProperty(hCursor, "Position", 300);
END
```

## ITrendCursor.Visible [Property][Get/Set]

Get or Set whether the cursor is visible.

**Defined As**

- [VBA] Boolean Visible
- [Cicode] INT Visible
- [C++] VARIANT\_BOOL Visible

**Execution Result**

If the property get/set succeeds, the return value will be Success. If the return variable is bad, the return value will be InvalidArgument. If the cursor is deleted, the return value will be GeneralFailure.

**Remarks**

This property controls the visibility of the cursor. The visibility is also applied to labels associated with the cursor.

**Calling Syntax**

This example assumes you have a valid reference to a cursor.

**[VBA]**

```
Sub Example(cursor As Object)
Dim visibility As Boolean
`Getting Property value
visibility = cursor.Visible
`Setting Property value (False)
cursor.Visible = False
End Sub
```

**[Cicode]**

```
FUNCTION Example(OBJECT hCursor)
INT visibility;
// Getting current property value
visibility = _ObjectGetProperty(hCursor, "Visible");
// Setting Property to False (0)
_ObjectSetProperty(hCursor, "Visible", 0);
END
```

**See Also**

[ITrendCursor.PenLabelVisible \[Property\]\[Get/Set\]](#)

## ITrendCursor.Width [Property][Get/Set]

Gets or Sets the line width of the cursor.

**Defined As**

- [VBA] Long Width
- [Cicode] INT Width
- [C++] int Width

**Execution Result**

If the property get/set succeeds the return value will be Success. If the return variable is bad then the return value will be InvalidArgument. If the cursor is deleted the return value will be GeneralFailure.

**Limits**

- Minimum (0)
- Maximum (8)

**Calling Syntax**

This example assumes you have a valid reference to a cursor.

**[VBA]**

```
Sub Example(cursor As Object)
Dim lineWidth As Long
`Getting Property value
lineWidth = cursor.Width
`Setting Property value
cursor.Width = 5
End Sub
```

## [Cicode]

```
FUNCTION Example(OBJECT hCursor)
INT lineWidth;
// Getting current property value
lineWidth = _ObjectGetProperty(hCursor, "Width");
// Setting Property to 5
_ObjectSetProperty(hCursor, "Width", 5);
END
```

## Automation Examples

One way of understanding how to use the Process Analyst's automation model is to see some example code. The following examples cover some concepts of extensibility offered by the Process Analyst:

- [Handling an Event](#)
- [Enumerating Collections](#)
- [Implementing a Custom Command](#)
- [Implementing a Custom Column.](#)

The samples assume you are using the Example project and have pasted a new Process Analyst onto the test page provided by the project.

You will also need to configure the Process Analyst object's Name and Event class by doing the following:

1. In the Example project open the "test" page in Graphics Builder
2. Click the **Process Analyst** icon in the toolbox to insert the control.
3. Resize the control to fit the test page.
4. Double-click the **Process Analyst**.
5. Click the **Access/Identification** tab.
6. Change the **Object Name** to **CPA**.
7. Change the **Event class** to **CPA\_E**.
8. Click **Apply** and **OK**.

### Handling an Event

The Process Analyst contains many events that are triggered when certain actions occur. See Events.

To handle an event you need to provide a handler for the event by prepending your Process Analyst's event class name with the event name you want to handle and an underscore.

The example below shows how to define a handler for the "MouseClick" event with an event class defined as "CPA\_E".

## [VBA]

```
Sub CPA_E_MouseClick(pen As Object, button As Integer)
End Sub
```

**[Cicode]**

```
FUNCTION CPA_E_MouseClick(OBJECT hPA, OBJECT hPen, INT button)
END
```

The following example uses the MouseClick event to cancel the box zoom operation when the right mouse button is clicked.

**Note:** When referring to an ActiveX object in VBA, you need to prepend it with the page name and an underscore. In the example below, the page name is called "test". The object name is "CPA" and the event class name is "CPA\_E".

**[VBA]**

```
Sub CPA_MouseClick(pen As Object, button As Integer)
Dim bZoomMode As Boolean

If (button = 1) Then
bZoomMode = test_CPA.ZoomMode

If (bZoomMode = True) Then
test_CPA.ZoomMode = False
End If
End If
End Sub
```

**[Cicode]**

```
FUNCTION CPA_E_MouseClick(OBJECT hPA, OBJECT hPen, INT button)
INT bZoomMode = 0;
IF (button = 1) THEN
bZoomMode = _ObjectGetProperty(hPA, "ZoomMode")
IF (bZoomMode = -1) THEN
_ObjectSetProperty(hPA, "ZoomMode", 0)
ENDIF
ENDIF
ENDIF
```

**Enumerating Collections**

The Process Analyst contains many "collections" such as Panes, Pens, Cursors, Commands, and so on. This example shows you how to enumerate through the buttons on the navigation toolbar.

**[VBA]**

```
Sub EnumerateToolbarButtons()
Dim navBar As Object
Dim iButton As Integer
Dim button As Object
Dim nButtons As Integer

` The Navigation toolbar is the 2nd toolbar in the collection
```

```
Set navBar = test_CPA.Toolbars.Item(2)
If IsNull(navBar) = False Then
nButtons = navBar.Buttons.Count
For iButton = 1 To nButtons
Set button = navBar.Buttons(iButton)
Next iButton
End If
End Sub
```

## [Cicode]

```
FUNCTION EnumerateToolbarButtons()
OBJECT hPA = ObjectByName("CPA");
OBJECT hToolbars = _ObjectGetProperty(hPA, "Toolbars");
// The Navigation toolbar is the 2nd toolbar in the collection
OBJECT hNavBar = _ObjectCallMethod(hToolbars, "get_Item", 2);
OBJECT hButtons;
OBJECT hButton;
INT nButtons;
INT iButton;

IF (ObjectIsValid(hNavBar)) THEN
hButtons = _ObjectGetProperty(hNavBar, "Buttons");
nButtons = _ObjectGetProperty(hButtons, "Count");

FOR iButton = 1 TO nButtons DO
hButton = _ObjectCallMethod(hButtons, "get_Item", iButton);
END
END
END
```

**Note:** Many collections have an `ItemById` property that allows you to get the item you want without having to enumerate through the collection to find the item you want.

## Implementing a Custom Command

Custom commands are easy to implement and involve creating a new command, adding it to a toolbar, and responding to the `CommandExecuted` event.

### To add a new command and add it to the toolbar as a button:

- Display the properties for your Process Analyst in the Graphics Builder
- See Adding New Commands.
- Use the following settings:
  - ID = "SelectedPen"
  - Tooltip = "Show the name of the selected pen"
  - Button style = <Push>
  - Enabled = <Checked>

Once you've done this, you need to write an event handler for the `CommandExecuted` event. This event will be called when the command is executed, whether by Cicode or by clicking on the respective toolbar button. The

CommandExecuted event when triggered has a commandId parameter identifying the command executed by the operator.

This example implements a command that displays a message box showing the name of the primary selected pen.

## [VBA]

```
Sub CPA_E_CommandExecuted(commandId As String)
Select Case commandId
Case "SelectedPen"
Call OnSelectedPen()
End Select
End Sub

Sub OnSelectedPen()
Dim pen As Object
Dim sName As String
Dim sMessage As String

Set pen = test_CPA.LastSelectedPen

If IsNull(pen) = False Then
sName = pen.Name
End If
sMessage = "The name of the selected pen is:" + Chr(13) + sName
MsgBox sMessage, 48, "Citect"
End Sub
```

## [Cicode]

```
FUNCTION CPA_E_CommandExecuted(OBJECT hPA, STRING commandId)
SELECT CASE commandId
CASE "SelectedPen"
OnSelectedPen(hPA);
END SELECT
END

FUNCTION OnSelectedPen(OBJECT hPA)
OBJECT hPen;
STRING sName;
STRING sMessage;
IF ObjectIsValid(hPA) THEN
hPen = _ObjectGetProperty(hPA, "LastSelectedPen");

IF ObjectIsValid(hPen) THEN
sName = _ObjectGetProperty(hPen, "Name");
END
sMessage = "The name of the selected pen is:^n" + sName;
Message("Citect", sMessage, 48);
END
END
```

## See Also

[CommandExecuted \[Event\]](#)

### Enabling and Disabling a Command

You can also respond to the UpdateCommand event to control the enable/disable state of the command's toolbar button. The example below disables the button if there are no pens.

### [VBA]

```
Sub CPA_E_UpdateCommand(commandId As String)
Select Case commandId
Case "SelectedPen"
Call OnUpdatedSelectedPen()
End Select
End Sub

Sub OnUpdatedSelectedPen()
Dim pen As Object
Dim command As Object
Dim sName As String

On Error Goto errHandler

Set command = test_CPA.CommandSystem.ItemById("SelectedPen")
Set pen = test_CPA.LastSelectedPen

sName = pen.Name
command.Enabled = True
Exit Sub

errHandler:
command.Enabled = False
End Sub
```

### [Cicode]

```
FUNCTION CPA_E_UpdateCommand(OBJECT hPA, STRING commandId)
SELECT CASE commandId
CASE "PanelLock"
OnUpdatePanelLock(hPA);
CASE "SelectedPen"
OnUpdateSelectedPen(hPA);
END SELECT
END

FUNCTION OnUpdateSelectedPen(OBJECT hPA)
OBJECT hPen = _ObjectGetProperty(hPA, "LastSelectedPen");
OBJECT hCommandSystem = _ObjectGetProperty(hPA, "CommandSystem");
OBJECT hCommand = _ObjectCallMethod(hCommandSystem, "get_ItemById", "SelectedPen");
INT iError = 0;
```

```
ErrSet(1);
_ObjectGetProperty(hPen, "Name");
iError = IsError();

IF (iError <> 0) THEN
_ObjectSetProperty(hCommand, "Enabled", 0);
ELSE
_ObjectSetProperty(hCommand, "Enabled", -1);
END
ErrSet(0);
END
```

## See Also

[UpdateCommand \[Event\]](#)

### Implementing a Custom Column

Custom columns are added to the Object View allowing you to display your own information associated with a pen.

The sample below implements a column that calculates the "Display Period" for each pen on the Process Analyst. The sample consists of three functions: an update function and two event handlers to verify the column is updated when new pens are added and when the time span is changed.

#### The Update Function

The update function is complex since it needs to match an Object View pen item with a real pen object; however, this isn't too difficult because the Object View tree always reflects how many panes and pens are being displayed.

The code achieves this by iterating through each pane and pen object in the Process Analyst while simultaneously keeping a running index count of which pane/pen item it matches up to in the Object View tree.

By using these indexes the code knows which row to update. A row update is achieved using the `PutField` method.

---

**Note:** Implementing your own column is CPU-intensive. Try to keep the amount of code necessary to calculate a row value as efficient as possible and be aware how often the code will be executed. Note also that, for efficiency, the `BlockUpdates` and `UnblockUpdates` functions are used to limit the number of updates made to the Object View.

#### Event Handlers

Once you have a function that implements your custom column, you need to know when to update that column. The most common method of doing this is to implement event handlers for particular Process Analyst events. The example below uses the `PenCreated` event and the `HorizontalAxisChanged` event. These events will verify that the column values will be updated when pens are added to the display and when the time span changes.

## [VBA]

```
Sub UpdateMyColumn()
Dim iPaneItem As Integer
Dim iPane As Integer
Dim nPanes As Integer
Dim nSamples As Integer

Dim penItem As Object
Dim paneItem As Object

nPanes = test_CPA.NumberOfSamples
iPaneItem = 0
nPanes = test_CPA.Panes.Count
For iPane = 1 To nPanes
Dim pane As Object

Set pane = test_CPA.Panes.Item(iPane)
If IsNull(pane) = False Then
Dim pen As Object
Dim iPen As Integer
Dim iPenItem As Integer
Dim nPens As Integer

iPenItem = iPaneItem + 1

Set paneItem = test_CPA.ObjectView.Items.Item(iPenItem)

test_CPA.BlockUpdates
iPenItem = 0
nPens = pane.Pens.Count
For iPen = 1 To nPens
Set pen = pane.pens.Item(iPen)

If IsNull(pen) = False Then
Dim dDiff As Double
Dim sText As string

Dim dtStart As Date
Dim dtEnd As Date
Dim dtStartMs As Integer
Dim dtEndMs As Integer

iPenItem = iPenItem + 1

pen.GetHorizontalAxisTimeSpan dtStart, dtStartMs, dtEnd, dtEndMs, False

dDiff = ((CDbl(dtEnd) - CDbl(dtStart)) / nSamples) * 86400
If (dDiff >= 0.001) Then
sText = CStr(Format(dDiff, "#0.000")) + " seconds"
Else
sText = "0.001 seconds"
End If
Set penItem = paneItem.Items.Item(iPenItem)

If IsNull(penItem) = False Then
```

```
penItem.Putfield "DisplayPeriod", sText
End If
End If
Next
test_CPA.UnblockUpdates
End If
Next
End Sub

Sub CPA_E_HorizontalAxisChanged(hPen As Object)
UpdateMyColumn
End Sub

Sub CPA_E_PenCreated(hPen As Object)
UpdateMyColumn
End Sub
```

### [Cicode]

```
FUNCTION UpdateMyColumn()
OBJECT hPA = ObjectByName("CPA");
OBJECT hPanes = _ObjectGetProperty(hPA, "Panes");
OBJECT hPane;
OBJECT hPens;
OBJECT hPen;
OBJECT hObjectView = _ObjectGetProperty(hPA, "ObjectView");
OBJECT hPaneItems = _OBJECTGetProperty(hObjectView, "Items");
OBJECT hPaneItem;
OBJECT hPenItems;
OBJECT hPenItem;

INT nPanes = _ObjectGetProperty(hPanes, "Count");
INT nPens;
INT iPen;
INT iPane = 0;
INT nSamples = _ObjectGetProperty(hPA, "Numberofsamples");
INT iPenItem = 0;
INT iPaneItem = 0;

REAL dDiff;
REAL dtStart;
REAL dtEnd;
INT dtStartMs;
INT dtEndMs;

STRING sText;

FOR iPane = 1 TO nPanes DO
hPane = _ObjectCallMethod(hPanes, "get_Item", iPane);

IF ObjectIsValid(hPane) THEN
hPens = _ObjectGetProperty(hPane, "Pens");
nPens = _ObjectGetProperty(hPens, "Count");

iPaneItem = iPaneItem + 1;
```

```
_ObjectCallMethod(hPA, "BlockUpdates");
hPaneItem = _ObjectCallMethod(hPaneItems, "get_Item", iPaneItem);
iPenItem = 0;
FOR iPen = 1 TO nPens DO
hPen = _ObjectCallMethod(hPens, "get_Item", iPen);

IF ObjectIsValid(hPen) THEN

iPenItem = iPenItem + 1;
_ObjectCallMethod(hPen, "GetHorizontalAxisTimeSpan", dtStart, dtStartMs, dtEnd, dtEndMs,
0);

dDiff = ((dtEnd - dtStart) / nSamples) * 86400;

IF dDiff > 0.001 THEN
sText = StrFormat(dDiff, 10, 3, "seconds");
ELSE
sText = "0.001 seconds"
END
hPenItems = _ObjectGetProperty(hPaneItem, "Items");
hPenItem = _ObjectCallMethod(hPenItems, "get_Item", iPenItem);

_ObjectCallMethod(hPenItem, "PutField", "DisplayPeriod", sText);
END
END
_ObjectCallMethod(hPA, "UnblockUpdates");
END
END
END

FUNCTION CPA_E_HorizontalAxisChanged(OBJECT hPA, OBJECT hPen)
UpdateMyColumn();
END

FUNCTION CPA_E_PenCreated(OBJECT hPA, OBJECT hPen)
UpdateMyColumn();
END
```

## See Also

[IOBJECTVIEWITEM.PUTFIELD \[METHOD\]](#)  
[IPROCESSANALYST.BLOCKUPDATES \[METHOD\]](#)  
[IPROCESSANALYST.UNBLOCKUPDATES \[METHOD\]](#)  
[PENCREATED \[EVENT\]](#)  
[HORIZONTALAXISCHANGED \[EVENT\]](#)

## Schedules

You can use schedules to automate the operation of equipment in a Plant SCADA system. Schedules are configured in Plant SCADA using **Scheduler**, a tool that uses a simple calendar-based interface to execute state

changes for specific pieces of equipment at runtime.

## ⚠ WARNING

### UNINTENDED EQUIPMENT OPERATION

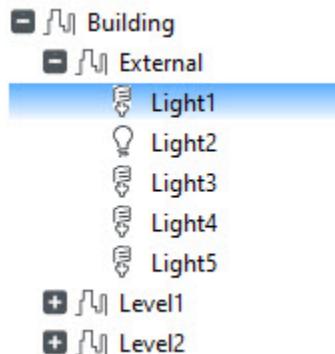
Do not use the Scheduler in situations where equipment requires a guaranteed sequence of successful actions. The Scheduler engine will attempt to run the actions in the correct order, but if entry actions take too long or a failure in communications occurs when an entry action is run, the engine does not guarantee that the equipment will receive every command.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

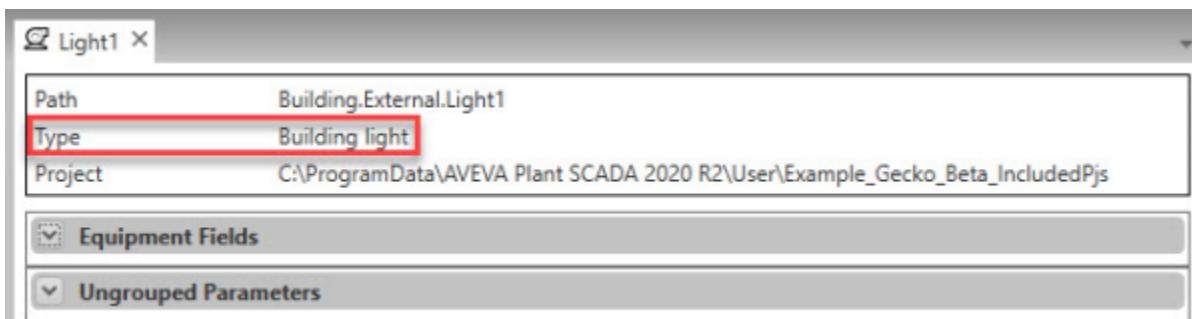
In Plant SCADA, a schedule specifies how a piece of equipment will operate at a particular time. This depends on the following:

- The piece of equipment needs to be defined within the system's equipment hierarchy (see [Equipment](#))
- The piece of equipment needs to include a defined set of "states" that can be used to manage its operation (see [Equipment States](#)).

For example, the equipment hierarchy in the Plant SCADA Example project includes a branch that represents a building.

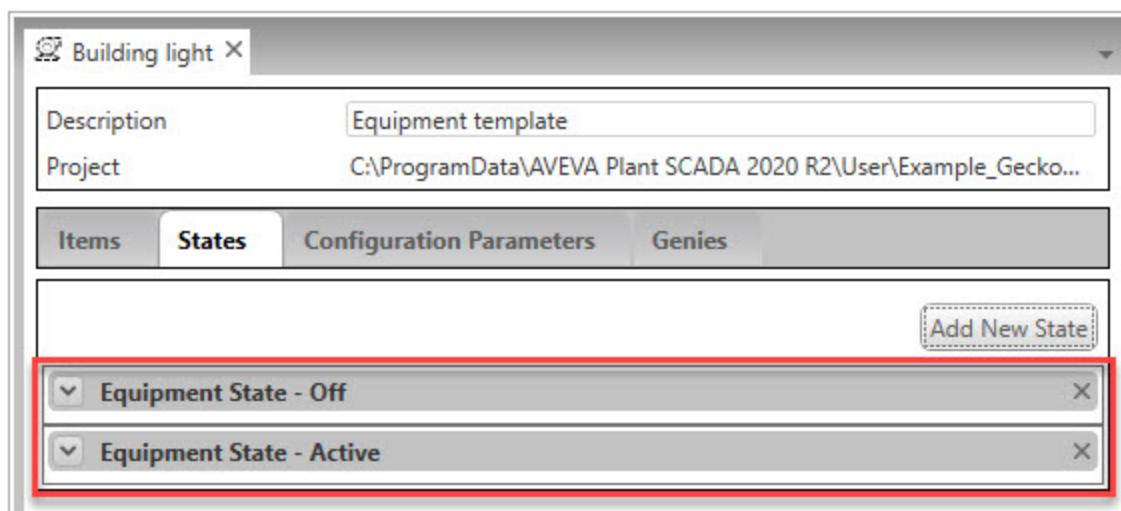


The piece of equipment selected above (Light1) represents an external light. When you view Light1 in Equipment Editor, you can see that it is an instance of an equipment type called "Building Light".



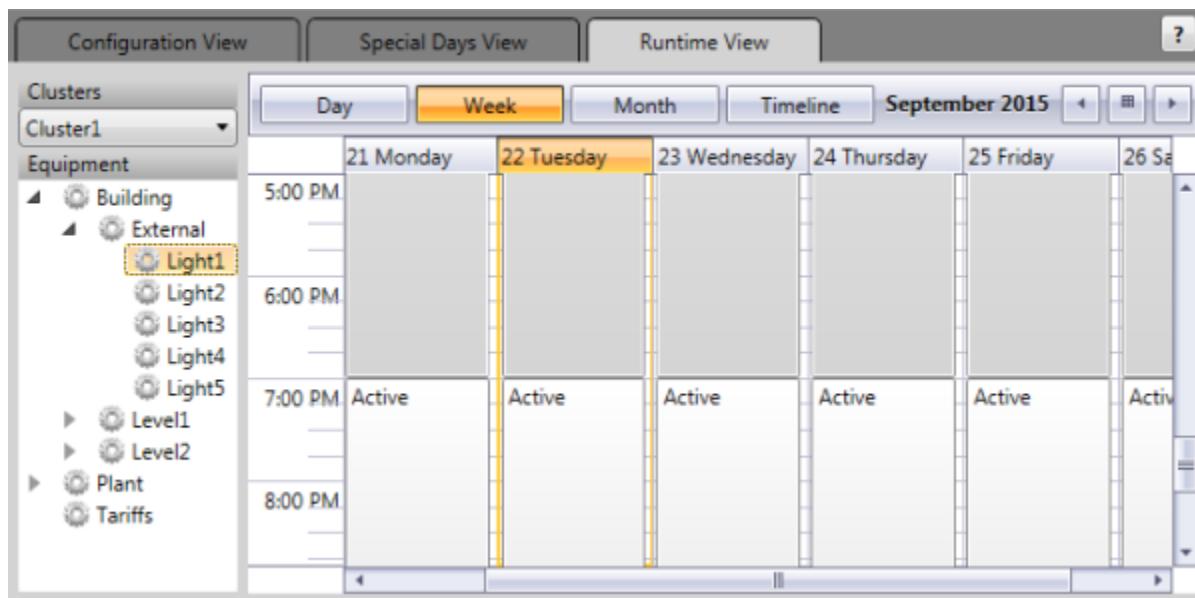
When you view the Building Light equipment type, you can see that it has two **States** configured:

- Off
- Active.



This allows a schedule to be applied to Light1 that uses the **Active** state to turn the external light on at 7pm every night, and the **Off** state to switch the light off again at 7am every morning.

Schedules are configured in Plant SCADA using [Scheduler](#), a tool that displays the equipment hierarchy for a project alongside a calendar. When a piece of equipment or branch is selected in the equipment hierarchy, the current schedule for it is displayed on the calendar.



A change of state is configured for a selected piece of equipment by adding a "schedule entry" to the calendar. A schedule entry specifies a **Start Time** and **End Time** for a particular **State**. You can also use a schedule entry to create a recurring schedule using patterns based on a day, week, month or year.

Scheduler also allows you to perform the following:

- Configure a schedule for all equipment within a particular branch in the equipment hierarchy (see [Schedule Inheritance](#))
- Set up "special days" that require a unique set of circumstances, such as public holidays or maintenance days (see [Special Days](#))
- Take manual control of a piece of equipment (see [Override Mode](#)).

- See [Prepare Your System for Scheduling](#) for information on how to prepare Plant SCADA to support schedules at runtime.
- See [Operate Scheduler at Runtime](#) for information about how to use Scheduler in the runtime environment.

**Note:** Scheduler supports the integration of schedules that are configured locally on a BACnet device. This means you can view and modify schedule-object and calendar-object properties on a BACnet device at runtime. For more information, see [Integrate BACnet Schedules into Scheduler](#).

## See Also

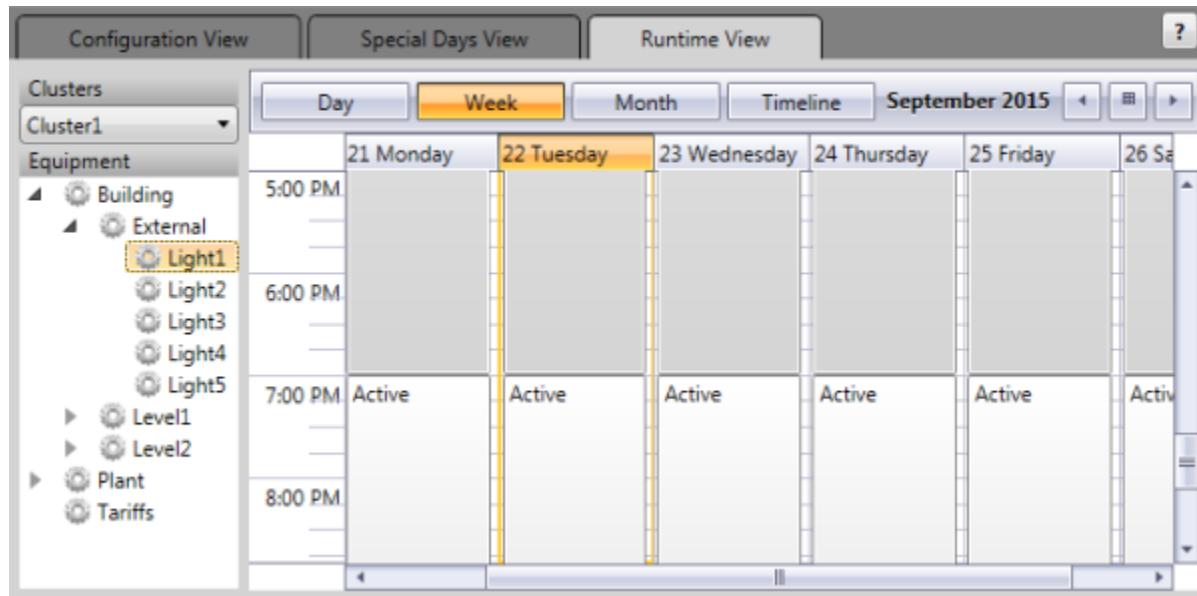
[Monitor Schedules](#)

## Scheduler

Scheduler is a runtime tool that allows you to view and manage schedule entries for a selected piece of equipment. You can also use it to specify when [Special Days](#) occur.

The Scheduler interface is made up two key components:

- [Equipment Tree](#) — displays the system's equipment hierarchy
- [Calendar](#) — displays the schedule entries for the equipment that is currently selected in the equipment tree.



You can use Scheduler to perform the following tasks:

- [Add a Schedule Entry](#)
- [Add a Recurring Schedule Entry](#)
- [Edit a Schedule Entry](#)
- [Enable Override Mode](#)
- [Manually Set a State Change](#)
- [Configure Special Days](#).

**Note:** Scheduler operates as a core component of the Plant SCADA reports server. This enables support for online changes. Any configuration changes to equipment or equipment states can be implemented during runtime via a report server restart using Runtime Manager or Cicode functions.

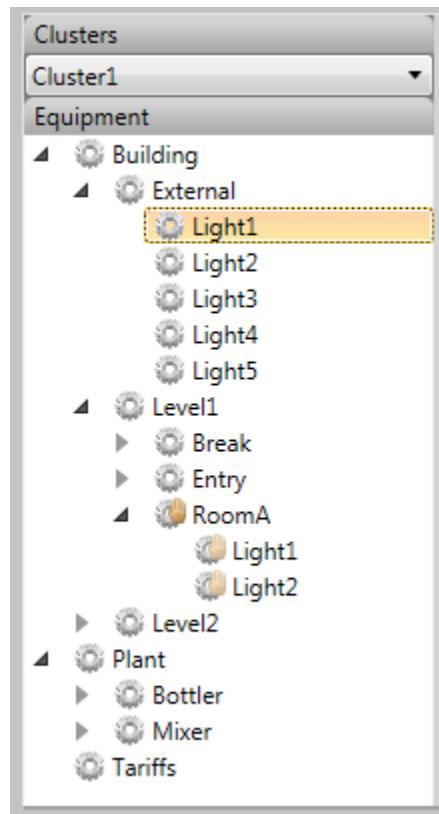
## See Also

[Manage Daylight Saving Transitions](#)

[View and Edit BACnet Schedules at Runtime](#)

## Equipment Tree

The equipment tree appears to the left of the Scheduler interface. It displays the equipment hierarchy for a selected cluster.



For instructions on how to prepare the equipment hierarchy that displays in the equipment tree, see the topic [Configure Equipment for Scheduling](#) in the Plant SCADA documentation.

When you select an item in the equipment tree, the schedule entries that are configured for the item appear in the calendar to the right.

The equipment tree uses a set of icons to indicate the status of the items included in the equipment hierarchy. For example, the icon for "Building.Level1.RoomA" in the equipment tree shown above indicates that RoomA is currently in override mode.

The current status of an item will determine the right-click menu options that are available for it. The following table describes the meaning of the icons that are used in the equipment tree, and the context menus that are supported in each case.

Icon	Equipment Mode	Context Menu
	This icon indicates that the equipment is in automatic mode. The current state is either the equipment's default state, or is set by any active schedules.	Override
	This icon indicates a piece of virtual equipment. Virtual equipment has no states defined, however it can have sub-directories.	No context menu
	This icon indicates the equipment is in <b>Override Mode</b> . The state is defined by the <b>Set to State</b> menu item.	Remove Override Set to State
	This icon indicates the equipment has inherited override mode from a parent directory. Its state is defined by the <b>Set to State</b> setting for the parent item. You can apply an override to counteract an inherited override setting.	Override

## See Also

[Calendar](#)

## Calendar

[Scheduler](#) includes a calendar that can be used to display a time-based representation of a schedule. When you make a selection in the [Equipment Tree](#), the associated schedule entries are presented on the calendar.

You can also use the calendar to determine when [Special Days](#) will occur.

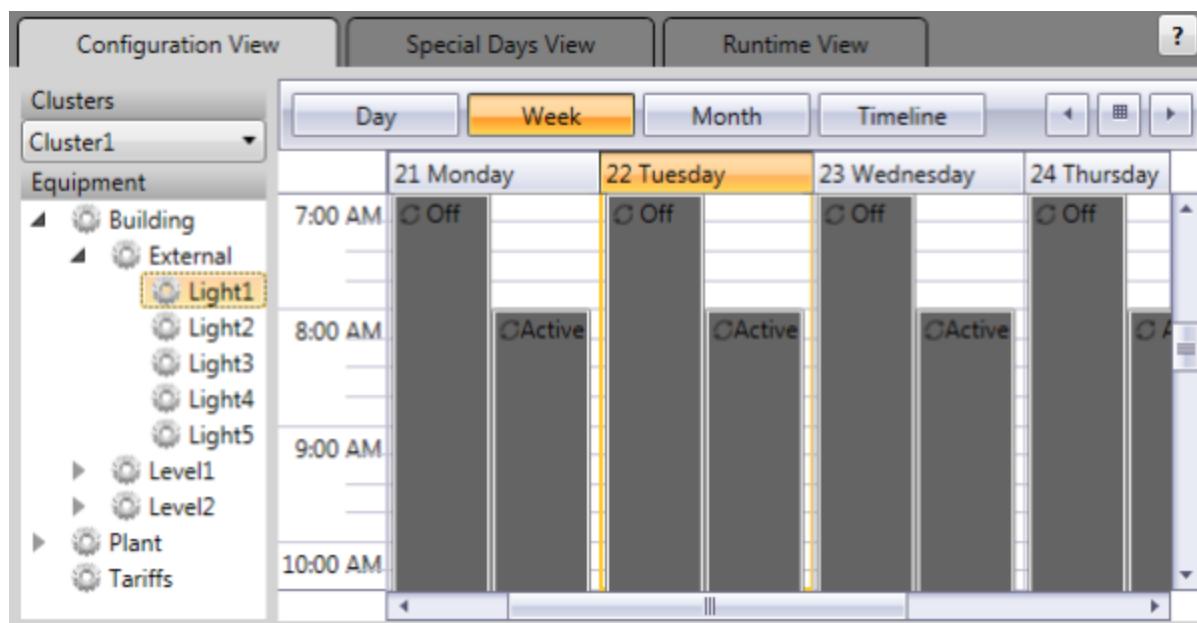
The calendar supports three different views:

- Configuration View
- Runtime View
- Special Days View.

Each view is accessible via the tabs that run across the top of the calendar.

- **Configuration View**

The Configuration View is used to add, modify or delete schedule entries for a selected piece of equipment.



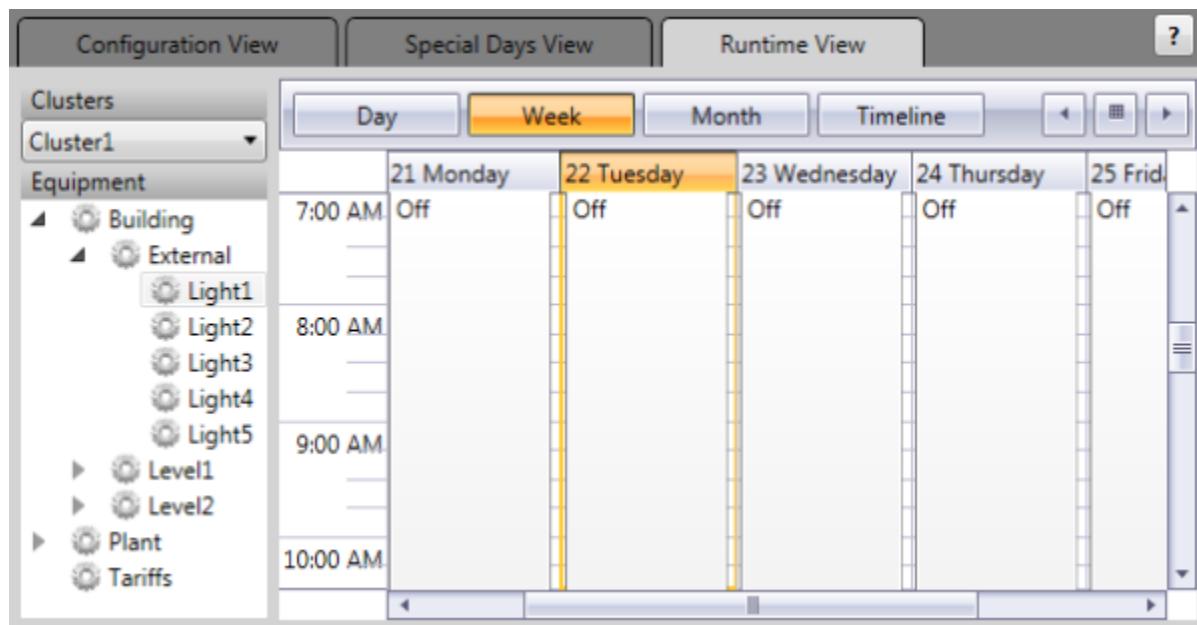
In this view, each schedule entry for a selected piece of equipment is shown as a block that spans the period between the **Start Time** and **End Time** specified for the associated equipment state. You can double-click on a block to view the properties for the schedule entry.

You can adjust the calendar to show a **Day**, **Week**, **Month**, or **Timeline**.

In some cases, a particular piece of equipment may have conflicting schedule entries. In the example above, "Light1" has two recurring schedule entries configured every day at 8:00 AM. When this situation occurs, precedence is given to the equipment state with the highest priority setting. For more information see [Schedule Priorities](#).

- **Runtime View**

The Runtime View is a read-only view that displays the activity that will occur for a piece of equipment (with any conflicts resolved).



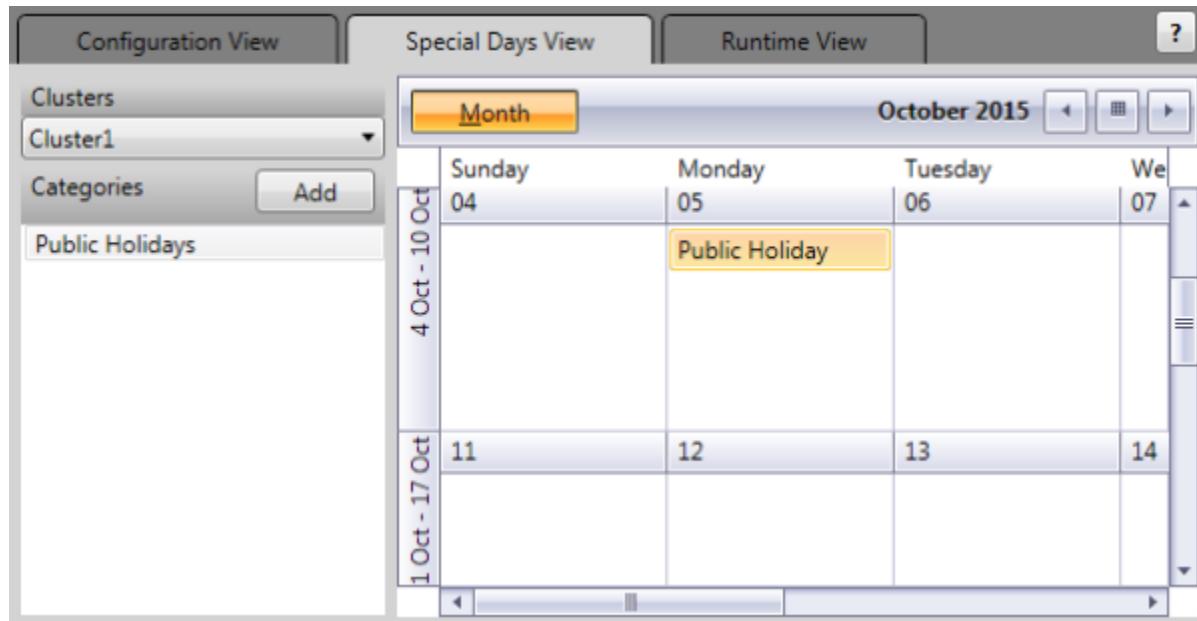
As with the Configuration View, the schedule is displayed as a series of blocks that indicate when each

equipment state will start and finish. You can double-click on a block to view the properties for the schedule entry, but you will be unable to make any changes.

You can adjust the calendar to show a **Day**, **Week**, **Month**, or **Timeline**.

- **Special Days View**

The Special Days View is used to view and configure [Special Days](#).



In this view, the calendar displays a month at a time. The equipment tree is replaced by a list of special day **Categories**. When you select a category, the associated special days are displayed on the calendar. Each is identified by a label that displays its **Name**.

## See Also

[Operate Scheduler at Runtime](#)

## Special Days

Special days are used to identify those days where regular scheduling does not apply to the equipment in a plant. For example, if your plant does not operate on certain public holidays, you could identify these days as special days in the Scheduler calendar.

Once you have set up special days, you can use them in the following ways:

- You can configure recurring schedule entries that exclude special days
- You can configure recurring schedule entries that only apply on special days.

Each special day needs to belong to a category. This allows you to manipulate the behavior of recurring schedules by including or excluding particular categories.

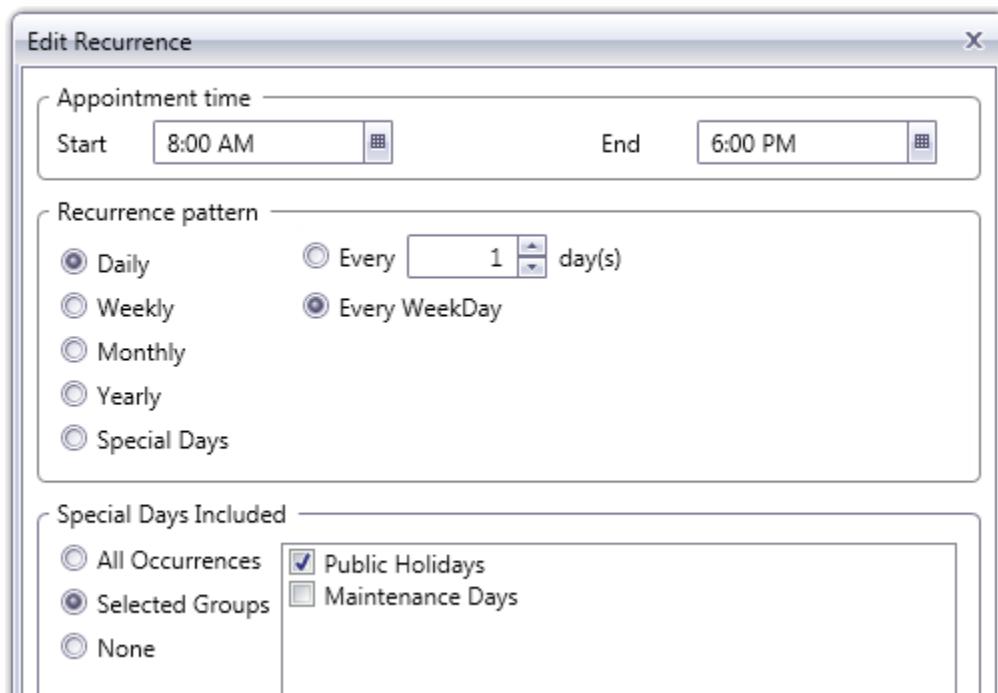
**Note:** If you are using Scheduler to display schedules imported from a BACnet device, Special Days will operate differently as they are used to display exception schedules. For more information, see [Integrate BACnet Schedules into Scheduler](#).

## Example

A production facility ceases operation on the last Friday of every month for maintenance. To manage this process, a special day category called "Maintenance Days" is created in the Scheduler, and the last Friday of every month is included.

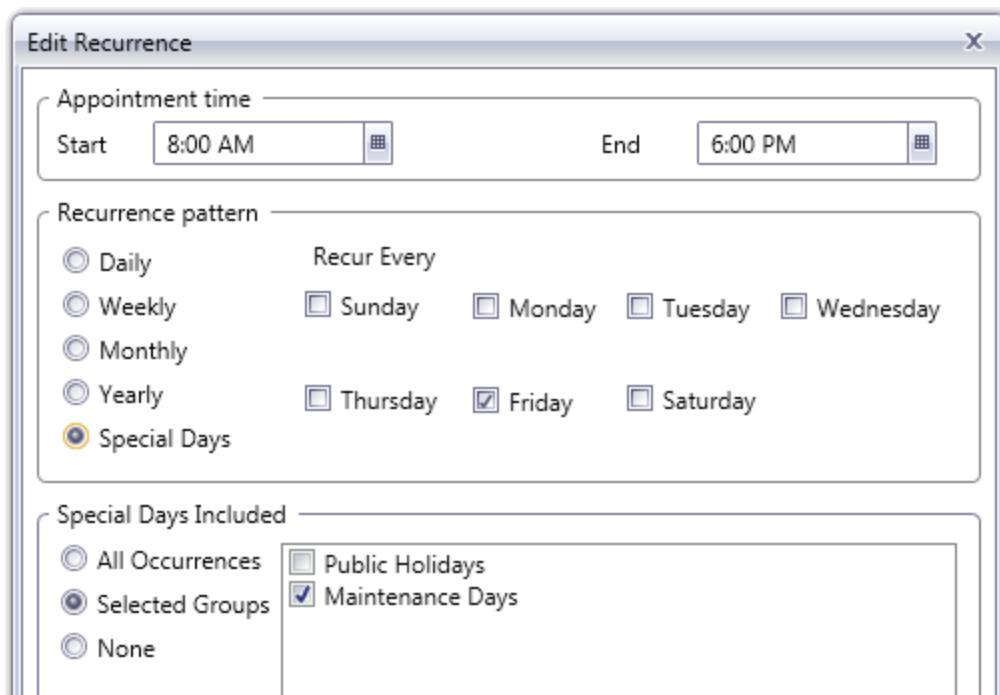
Schedules for both normal operation and maintenance days can then be configured in the following ways:

- Any recurring schedules that apply to normal operation will exclude the maintenance days. To achieve this, the following settings are used:



Observe that the "Maintenance Days" category is **not** selected in the **Special Days Included** section of the dialog.

- Any recurring schedules that are meant to run on maintenance days are configured to include the special days. To achieve this, the following settings are used:



In this case, the **Recurrence Pattern** is based on **Special Days**, and the "Maintenance Days" category is selected in the **Special Days Included** section of the dialog.

---

**Note:** In Plant SCADA 7.40, a different tool was used to configure special days. If you run a version 7.40 project that includes Scheduler, it will still display the old special days calendar control to the right of the Scheduler interface. This control will still work, however it is recommended that you use the new Special Days View to configure special days.

---

## See Also

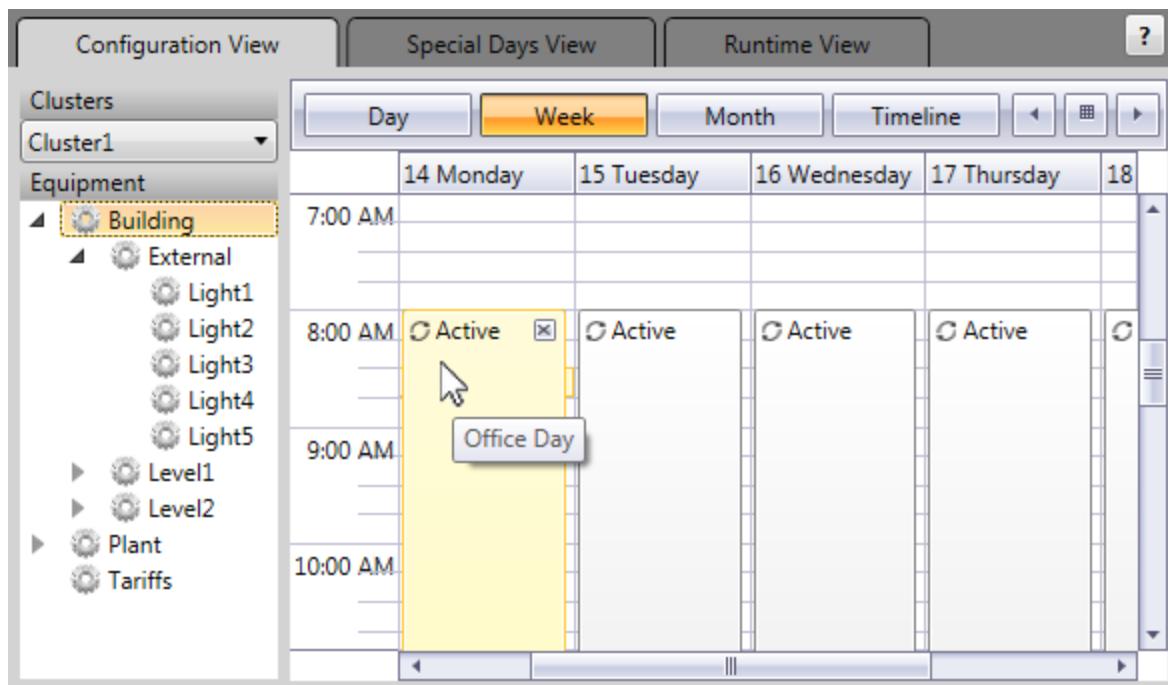
[Configure Special Days](#)

[Add a Recurring Schedule Entry](#)

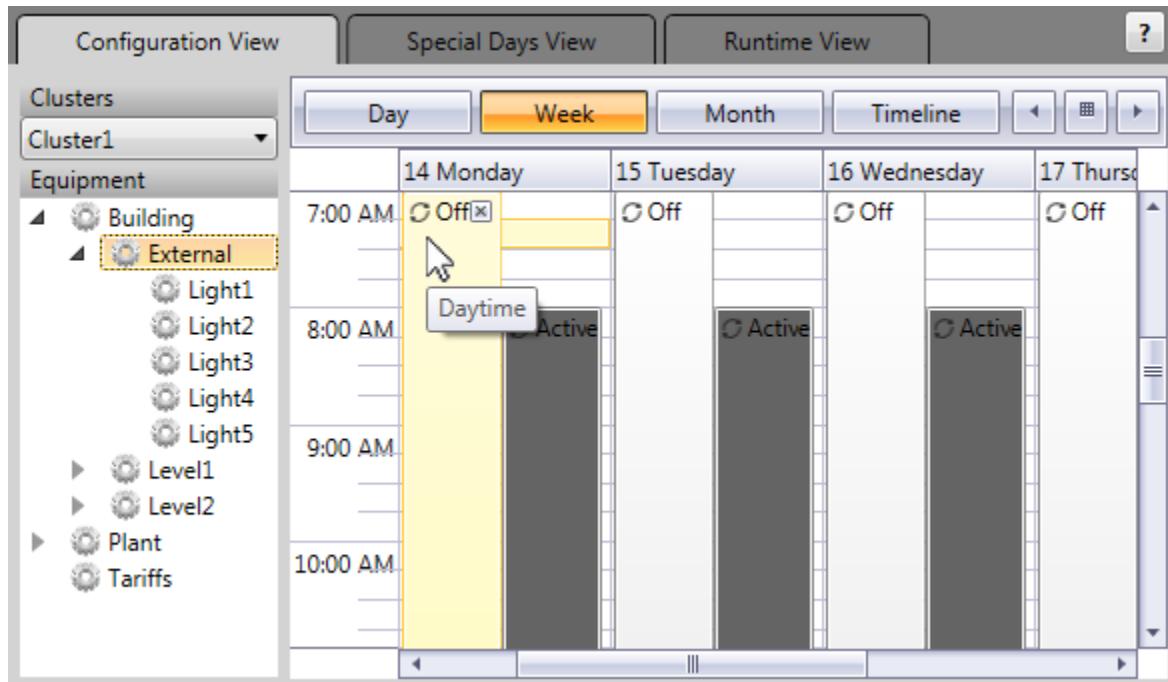
## Schedule Inheritance

If a schedule is applied to a location within an equipment hierarchy that has levels beneath it, the schedule is automatically inherited by the equipment within the lower levels.

For example, the equipment hierarchy in the Plant SCADA Example project includes a top level entry named "Building". A recurring schedule entry named "Office Day" is configured at this level. It specifies an active state commencing at 8:00 AM every weekday.



If you move down one level in the equipment hierarchy to "External", you will see that the "Office Day" schedule still applies (an inherited schedule is highlighted using dark gray). The inherited schedule entries display alongside a recurring schedule configured at the External level named "Daytime".



This means that all of the lights within the "External" branch of the equipment hierarchy will inherit both the "Daytime" and "Office Day" schedules. This creates a conflict at 8:00 AM every weekday, as the two inherited schedules have conflicting equipment states defined.

To determine how this type of conflict is resolved, see [Schedule Priorities](#).

---

**Note:** To view inherited schedules in the Scheduler calendar (including any conflicts), select the **Configuration View** tab (see [Calendar](#)).

## See Also

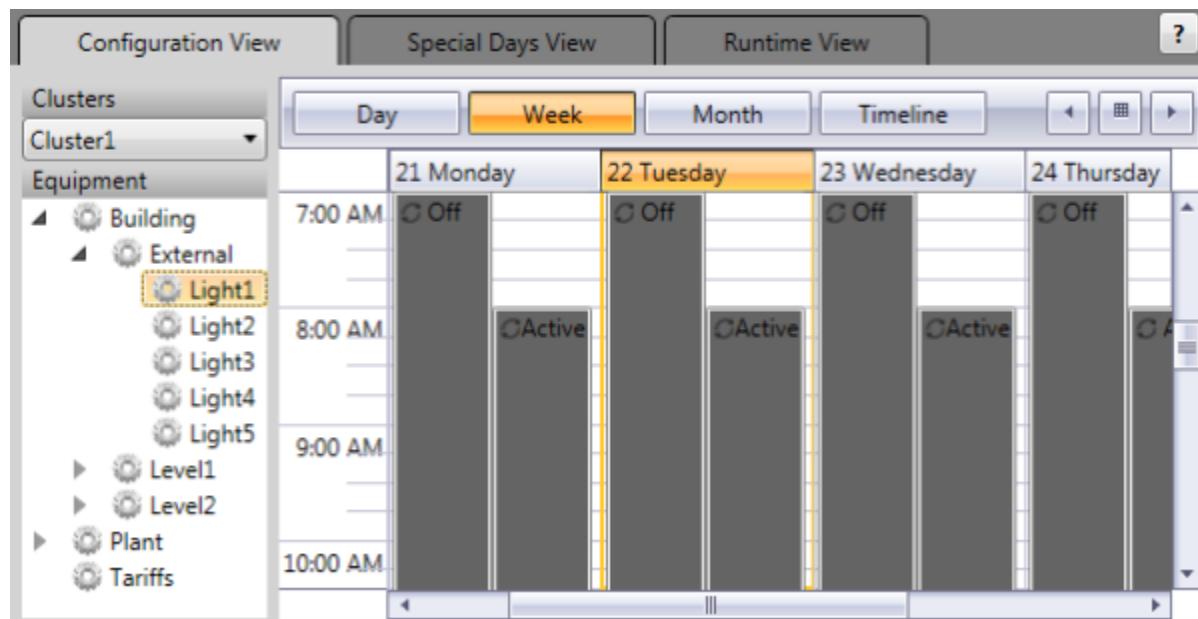
[Add a Schedule Entry](#)

[Add a Recurring Schedule Entry](#)

## Schedule Priorities

It is possible for a piece of equipment to have schedule entries that specify conflicting equipment states at the same time. This can be caused by [Schedule Inheritance](#), or by recurring schedules that overlap.

If two schedule entries appear alongside each other in the calendar's configuration view (see example below), it means that a scheduling conflict has occurred.



The situation is resolved according to the following priorities:

- If priorities are applied to the equipment states specified in the conflicting schedule entries, the schedule with the highest priority will take precedence (regardless of recurrence).
- If two (or more) schedule entries have the same priority level applied to their associated equipment state, the entry with the latest start time will take precedence.

If priorities are not specified for the conflicting equipment states:

- Inherited schedule entries will have the lowest priority.
- For schedule entries on the same branch, a recurring schedule entry will have a lower priority than a schedule entry set to run just once.
- For equipment with two or more schedule entries with the same start time, the schedule entry that was accessed last takes precedence.

**Note:** You can view the outcome of any schedule conflicts by displaying the Scheduler calendar's **Runtime View** (see [Calendar](#)). The Runtime View displays the activity that is scheduled for a piece of equipment with all conflicts resolved.

## Override Mode

Override mode allows you to take control of equipment scheduling and manually execute state changes. You can set equipment to override mode via the Scheduler [Equipment Tree](#).

When a piece of equipment is set to override mode, its associated schedule entries will not be executed. The following also occurs:

- Override mode is automatically propagated down the equipment hierarchy, which means it is inherited by the equipment on lower levels.
- Equipment state changes are also propagated down the equipment hierarchy to equipment that has inherited override mode.

If required, you can counteract a state specified by an inherited override by implementing another override at the level where you would like a different state to be set.

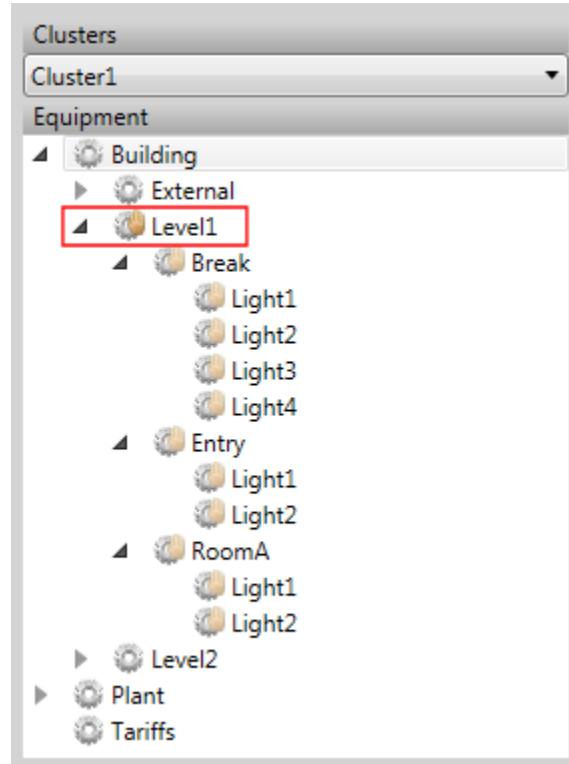
---

**Note:** If you hold the mouse over a piece of equipment in the Scheduler's equipment tree, its current state will display as a tool tip.

---

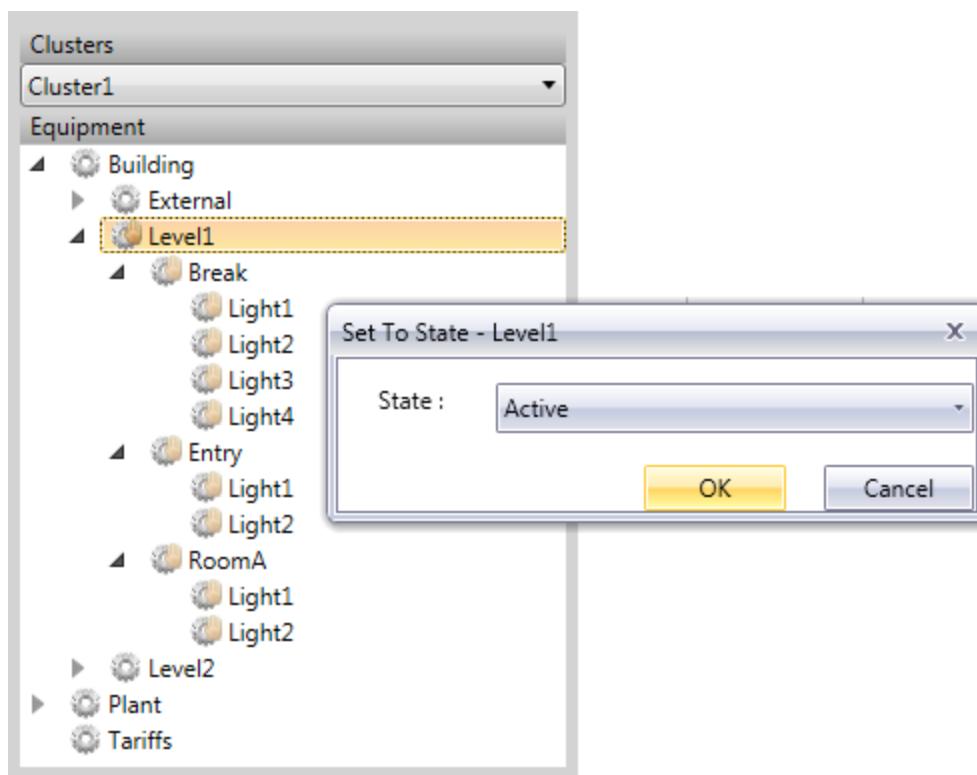
## Example

In the example below, the location "Building.Level1" is set to override mode. This is indicated by the small hand icon next to its location in the equipment hierarchy.

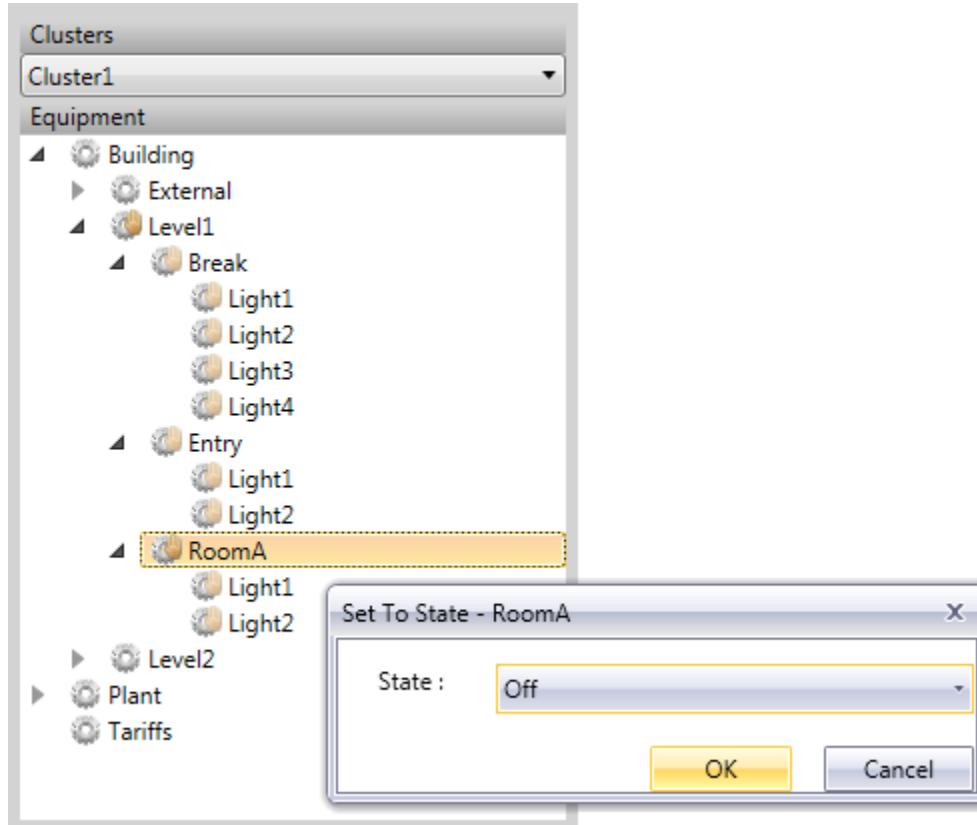


The override setting has propagated down the equipment hierarchy. This is indicated by the transparent icon located next to the equipment that branches off from Level1.

In the example below, the equipment state for Level1 is set to "Active" in the Scheduler equipment tree, which means every light on Level1 would turn on.



In the example below, a secondary override has been applied to "Building.Level1.RoomA" to counteract the Active state inherited override from Level1. This means the lights in RoomA can now be controlled independently and turned off.



## See Also

- [Enable Override Mode](#)
- [Manually Set a State Change](#)

## Prepare Your System for Scheduling

This section of the help describes the configuration changes that may be required in a Plant SCADA project to enable operation of Scheduler during runtime.

- [Configure Equipment for Scheduling](#)
- [Configure a Demand and Response Solution](#)
- [Add the Scheduler ActiveX Object to a Page](#)
- [Use ActiveX Properties to Control Scheduler's Current Selection](#)
- [Back Up Schedules](#)
- [Manage Security and Permissions](#)
- [Configure Scheduler Redundancy.](#)

---

**Note:** Scheduler supports the integration of schedules that are configured locally on a BACnet device. This means you can view and modify schedule-object and calendar-object properties on a BACnet device at runtime. For more information, see [Integrate BACnet Schedules into Scheduler](#).

---

## Configure Equipment for Scheduling

To use schedules in a Plant SCADA system, your project needs to include an [Equipment Hierarchy](#) (that incorporates the equipment you would like to automate) and [Equipment States](#) that control the operation of your equipment.

There are two ways you can configure equipment and equipment states in a Plant SCADA system:

- **Equipment Editor** — If you are starting a new project (or adding a new area or process to an existing project), you can use Equipment Editor to generate tags (see [Equipment Editor](#)).
- **Plant SCADA Studio** — If you have a project with existing tags, you can use Plant SCADA Studio to associate equipment with your tags (see [Define an Equipment Association for a Tag](#)).

In both cases, you need to consider the following equipment state properties when creating a schedule for a piece of equipment:

- **Entry Action** — defines the activity that occurs when an equipment state is initiated. It is configured using a Cicode expression, for example, you could use an expression that writes a value to a variable tag.

---

**Note:** Entry actions are not supported for equipment states that are associated with a BACnet device (see [Integrate BACnet Schedules into Scheduler](#)).

---

You can use the INI parameter [Scheduling]AlwaysExecuteEntryAction to specify that the entry actions for all equipment resource states are executed, regardless of whether or not their internal states are the same. By default, this parameter is set to false (0), which means only changes that have been made to the selected equipment will take effect.

- **Delay** — allows you to postpone an entry action for a specified period of time. A delay can be used to turn on equipment sequentially, thereby avoiding a power spike.
- **Repeat Action** — defines a repeat action that occurs when the entry action has completed. The repeat action will reoccur according to the specified **Period**, though it will not occur until the previous repeat action is complete.

If you are using Equipment Editor to configure equipment, these properties are defined when you add an equipment state to an equipment type (see [Add a State to an Equipment Type](#)).

If you are manually defining equipment for an existing set of tags, you can define these properties in the Equipment State Properties (see [Define Equipment States in Plant SCADA Studio](#)).

## See Also

[Configure a Demand and Response Solution](#)

## Configure a Demand and Response Solution

Demand and response (DR) refers to the mechanisms used for the management of electricity consumption in response to supply conditions. The Open ADR is an example of a standard that can be used to help automate demand response in commercial buildings. To provide demand and response, a system requires pre-defined levels that are used to adjust the expected consumption of energy.

You can use Plant SCADA to implement a demand and response solution.

In Scheduler, states can be defined multiple times to support a set of DR modes. This means the state that is scheduled to occur is determined by the DR mode for the equipment. In the example below "line2.lights" has multiple 'On' states to support multiple DR modes. This allows the room to be scheduled for 'On', with the amount of energy consumed adjusted according to the assigned DR mode.

Equipment	Equipment State	DR Mode	Entry Action	Description
Factory.Floor.Line2.Lights	On	0	TagI2_lights=1	When scheduled light will turn on at 100% energy output
	On	1	TagI2_lights=80	Energy saving 20%
	On	2	TagI2_lights=60	Energy savings is 40%
	On	3	TagI2_lights=40	Energy savings is 60%
	On	4	TagI2_lights=20	Energy Savings is 80%
	Off	0	TagI2_lights=0	

The default DR mode value is '0'. When the DR mode field is empty, the default value is used. The Scheduler engine selects the state with the nearest lower DR mode value in the case where an exact match to the current specified DR mode is not found.

Set the DR mode on equipment using the **EquipSetProperty** Cicode function, for example:

```
EquipSetProperty (Equipment, "DRMODE", "1", cluster)
```

## Add the Scheduler ActiveX Object to a Page

Scheduler is an ActiveX control that you can add to a Plant SCADA graphics page as an object. You can then use Scheduler when the page displays at runtime.

---

**Note:** If you are planning to add the Scheduler to a Situational Awareness Project, refer to the topic Use a Scheduler Control in a Situational Awareness Project.

---

### To add the Scheduler ActiveX control to a graphics page:

1. In Graphics Builder, open the page to which you would like to add the Scheduler ActiveX object.
2. On the **Edit** menu, select **Insert ActiveX Control**. The Insert ActiveX Control dialog box appears.
3. Double-click on the **SE Energy Scheduler Control** in the **ActiveX Controls** list.

The **Scheduler Control** properties dialog will display.

Or:

- On the **Drawing Toolbox**, select the **Scheduler** icon.



The **Scheduler Control** properties dialog will open.

Or:

- On the **Objects** menu, select **Scheduler**.

The **Scheduler Control** properties dialog will open.

For information on how to configure the properties for the control, see the following topics:

- ActiveX Object Properties - Appearance (Tag Association)
- Defining Common Object Properties.

Once you have placed the Scheduler control, you can position and re-size it. It is recommended that you size the control so the whole calendar will display in the runtime window.

## See Also

[Scheduler](#)

## Use ActiveX Properties to Control Scheduler's Current Selection

You can use the Scheduler ActiveX Control properties to customize the Scheduler Page in Runtime as per the requirement. Using these properties, you can:

- Select a cluster
- Select an equipment

- Select one of the Scheduler tabs
- Select a special day category
- Show only the **Runtime View** tab
- Show or hide the **Special Days View** tab
- Highlight special days
- Show or hide the **Add** button in the **Categories** panel on the **Special Days View** tab.

You can use the ActiveX properties in the following ways:

- Binding one of the Scheduler's ActiveX properties to a tag.
- Using the ActiveX properties as part of a page event command.

#### **Binding one of Scheduler's ActiveX properties to a tag:**

This functionality allows you to control the Scheduler's current selection by changing the associated tag value. For example, you can configure a button on a graphics page that displays the schedule for a particular piece of equipment.

To bind the scheduler's ActiveX properties to a tag perform the following steps:

1. In Graphics Builder, open the page that hosts the Scheduler ActiveX control (see [Open a Graphics Page](#)).
2. Select the Scheduler ActiveX control, then right-click on it and select Properties.
3. The SE Energy Scheduler Control Properties dialog box appears. Select the Appearance tab, and then the Tag Association tab on the right side.
4. Select one of the following options available in the Properties panel:
  - SelectedCluster
  - SelectedEquipment
  - SelectedSpecialDayCategory
  - SelectedTab
  - IsRuntimeViewOnly
  - IsSpecialDaysHidden
  - HighlightSpecialDays
  - IsAddSpecialDaysButtonVisible
5. In the panel labeled Associate property <property name> with tag..., enter the tag that you need to associate with the selected property.  
Refer to [ActiveX Object Properties - Appearance \(Tag Association\)](#) and associate a tag, or type a tag name.
6. Click OK.

Any value change to the associated variable tag reflects in the ActiveX object property, if the value meets the following requirements:

- SelectedCluster — The value is a string that specifies a valid cluster name (case sensitive).
- SelectedEquipment — The value is a string that represents a valid equipment path with periods (.) used to specify hierarchy levels (case sensitive).
- SelectedSpecialDayCategory — The value is a string that specifies a valid category name.

- SelectedTab — The value is an integer between 0 and 2:
  - 0 — Shows the **Configuration View** tab.
  - 1 — Shows **Special Days View** tab.
  - 2 — Shows the **Runtime View** tab.
- IsRuntimeViewOnly — The value is a boolean:
  - 0 (default) — Shows three tabs (**Configuration View**, **Special Days View** and **Runtime View**).
  - 1 — Hides the **Configuration View** and the **Special Days View** tab.
- IsSpecialDaysHidden — The value is a boolean:
  - 0 (default) — Shows the **Special Days View** tab.
  - 1 — Hides the **Special Days View** tab.
- HighlightSpecialDays — The value is a boolean:
  - 0 — Does not highlight special days.
  - 1 (default) — Highlights the special days in orange.
- IsAddSpecialDaysButtonVisible — The value is a boolean:
  - 0 — Hides the **Add** button in the **Categories** panel on the **Special Days View** tab.
  - 1 (default) — Shows the **Add** button in the **Categories** panel on the **Special Days View** tab.

### Using Scheduler's ActiveX properties as part of a page event command:

This functionality uses Cicode to specify the Scheduler selection when a page event occurs, typically the **On page shown** event.

To use Scheduler's ActiveX properties as part of a page event command perform the following steps:

1. In the Graphics Builder, open the page that hosts the Scheduler ActiveX control (see Open a Graphics Page).
2. Go to the **File** menu and select **Properties** to display the page properties.
3. Select the **Events** tab.
4. In the **Events** panel, select the event you need to associate with an ActiveX property. This is likely to be the **On page shown** event, as the other page events may produce illogical outcomes.
5. In the **<Event> command** panel, enter the required Cicode.

For example, the following Cicode (applied to the On page shown event) uses the function `_Object SetProperty` to display the schedule for "Building.External.Light1" in the Example project.

```
_Object SetProperty(ObjectByName("AN4"), "SelectedCluster", "Cluster1")
_Object SetProperty(ObjectByName("AN4"), "SelectedEquipment", "Building.External.Light1")
```

6. Click **OK**.

### See Also

[Tag Association](#)

## Back Up Schedules

### To back up the schedule entries created using Scheduler:

- Make a copy of the "[ClusterName].[ReportServerName].scheduling" XML file located in the Plant SCADA 'Data' directory on the local machine.

The file name be similar to the following:

Cluster1.ReportServer1.scheduling.xml

To restore the schedule entries in Scheduler, copy the xml file back into the Data directory.

---

**Note:** When you paste a backup of the schedules XML file into the Data directory, it will overwrite the schedule entries that are currently configured in Scheduler.

---

## Manage Security and Permissions

Scheduler integrates into the Plant SCADA security model by allowing access to certain features based on areas and user permissions.

When adding equipment to Scheduler, you can define the area to which the equipment belongs. This helps to add security around the use of equipment, as the Scheduler interface will only list equipment that is available to the current user. They will only be able to create, modify, delete and browse schedules for equipment that belongs to the area or areas that they can access. If the user attempts to access equipment in other through Cicode, an error code is returned.

When a new user logs on, the Scheduler interface will refresh and list the equipment which is available to that user, based on the areas the user has permission to access.

For more information regarding security refer to [Security Roles](#).

## Configure Scheduler Redundancy

Scheduler operates as a core component of the Plant SCADA reports server. This means you can use report server redundancy to help maintain the operation of schedules during periods of system instability.

Redundancy support for Scheduler is provided as follows:

- When a connection is established, Scheduler's files are synchronized (merged) between the peer report servers.
- Only the active server will execute the schedules. If both servers are running, the active server will be the one that started first.
- If a network outage occurs, the primary will become active after communication is restored.
- When both servers are online, changes to the Scheduler file are replicated across both servers.
- If one of the report servers becomes inoperative, the schedules will run on the remaining reports server. When the inoperative report server returns online, the client will remain with the current reports server.

For more information, see [Reports Server Redundancy](#).

---

**Note:** At startup, Scheduler may begin executing actions before other server processes in the system are available. This may result in the action not completing successfully. If this occurs, use the Citect.ini parameter

---

---

[Scheduling]StartDelay to delay the startup of the Scheduler engine. This allows time for the other server processes of the system to come online before any scheduled actions are initiated.

---

## Integrate BACnet Schedules into Scheduler

A BACnet device can be configured to incorporate its own schedule as a set of schedule-object and calendar-object properties. If you install the BACNET driver, Plant SCADA can read and write to these object properties at runtime. This allows you to integrate BACnet schedules into [Scheduler](#).

This functionality is enabled via the following process:

- A BACnet I/O device is configured in Plant SCADA and connected via the BACNET driver.
  - Variable tags and equipment definitions that represent the device's schedule and calendar objects are configured on the Plant SCADA I/O server. You can achieve this using the [Import Variable Tags](#) tool.
  - The Plant SCADA reports server subscribes to the tags on the I/O server and generates schedule entries in Scheduler.
  - Any changes to the schedule entries are written back to the BACnet device via the I/O server.
- 

**Note:**

1. The present value for a BACnet schedule is evaluated in the BACnet device, not by the Plant SCADA system. This means Plant SCADA entry actions are not supported for equipment states that are associated with a BACnet schedule.
  2. If a BACnet schedule object has exception schedules configured, they are presented in Scheduler using Special Days.
  3. If the date range specified for an exception schedule or calendar object is excessively long, it can affect system performance. For this reason, only the first six months of a date range will display in Scheduler.
  4. If the date range specified for an exception schedule or calendar object has a start date that is beyond three years in the future, it will not display in Scheduler.
  5. Wildcards (for example, "\*") are not supported by the Scheduler.
  6. Exception schedule or calendar objects that use a WeekNDay (WD) configuration will not display in Scheduler.
- 

### To integrate BACnet schedules into Scheduler:

1. Install the latest version of the BACNET driver.  
The latest version of the BACNET driver is available from the Plant SCADA Driver Web.
2. Add the BACnet device to your project as an I/O device (see [Using the Device Communications Wizard](#)).
3. Create the variable tags and equipment required to support the BACnet calendar and schedule object types.

You can automatically generate the required tags and equipment using the **Import Variable Tags** tool (see [Import Variable Tags from an External Data Source](#)). If you use this approach, you need to observe the following:

- The source **Database Type** specified on the Import Variable Tag dialog needs to be set to "BACnet".
- The BACnet Device Configuration dialog is used to configure the tag import. This dialog is accessible via the **Browse** button next to the **External Database** field. Information on how to use this dialog is accessible via the **Help** button.
- Select the **Import BACnet Schedule Instances as Equipment** option to create the equipment required to support the schedule entries in Scheduler.

4. Specify a data type for each BACnet schedule object so that they match the data type configured for the corresponding schedule entry on the BACnet device.

This can be done via Equipment Editor by modifying each Schedule Instance in the Equipment View. The parameter "DataType" can be set to the following values: "Int", "Dig", "Dbl", "Enum", "UInt", "Real". For more information, see [Specify a Data Type for a BACnet Schedule](#).

---

**Note:**

- Scheduler integration will not operate correctly if a data type configuration is not correctly matched with the device.
  - If you do not specify a data type for each BACnet schedule object, a "Bad IO device variable" error notification will be generated when you attempt to compile your project.
- 

5. Run an equipment update to generate the required tags (see [Update Equipment in Plant SCADA Studio](#)).

Alternatively, you can manually create instances of the required equipment using the BACnet equipment type templates provided in Plant SCADA's Include project.

The BACNET schedule entries will appear in Scheduler when you run your project (see [View and Edit BACnet Schedules at Runtime](#)).

## Specify a Data Type for a BACnet Schedule

Schedule entries can be configured locally on a BACnet device using any of the following data types:

- Integer
- Digital
- Double
- Enumerate
- Unsigned integer
- Real.

Plant SCADA accommodates these different data types with a corresponding set of tag addresses (stored as a 'string' data type):

- WeeklyScheduleExInt
- WeeklyScheduleExDig
- WeeklyScheduleExDbl
- WeeklyScheduleExEnum
- WeeklyScheduleExUInt
- WeeklyScheduleExReal.

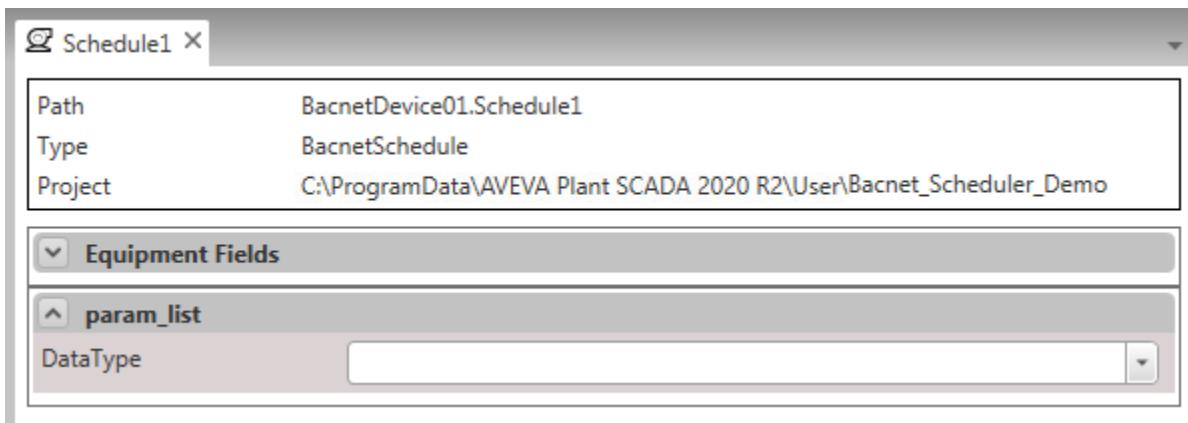
If you use the Tag Import tool to generate tags for BACnet scheduling, Plant SCADA is unable to detect the data type that has been used to configure the schedules on a device. This means you need to indicate the data type that should be used when the associated tag addresses are generated.

To simplify this task, the equipment instances generated by the Tag Import process to support BACnet schedule objects include a custom parameter called "DataType". Once you have set this parameter to the correct data type in the required equipment instances, you just need to run an equipment update to apply the correct

addresses to your tags.

#### To specify a data type for a BACnet schedule:

1. Open Equipment Editor and select the **Equipment** tab.
2. In the **Equipment** list to the right, locate the required BACnet device.  
BACnet devices are labeled using the **Parent Equipment** name specified during the tag import process.
3. In the equipment instances that branch off the required device, locate the BACnet Schedule instances.  
By default, equipment instances for BACnet schedule objects are named using "Schedule\_<n>", where *n* is the **Schedule ID** used to identify the associated schedule object in the BACnet device.
4. Open an equipment instance that requires updating (see [Open an Equipment Instance](#)).
5. Expand the **param\_list** fields to display the **DataType** parameter.



6. Select the required data type from the drop-down list to the right of the field. The options are:
  - **Dig** (digital)
  - **Dbl** (double)
  - **Int** (integer)
  - **Enum** (enumerate)
  - **UInt** (unsigned integer)
  - **Real** (real).

**Note:**

- Scheduler integration will not operate correctly if a data type configuration is not correctly matched with the device.
- If you do not specify a data type for each BACnet schedule object, a "Bad IO device variable" error notification will be generated when you attempt to compile your project.

7. Save your changes to the equipment instance.
8. Run an equipment update (see [Update Equipment](#)).

The variable tags associated with the equipment instance will now be configured to use the appropriate tag address.

#### See Also

[Integrate BACnet Schedules into Scheduler](#)

## Operate Scheduler at Runtime

You can use [Scheduler](#) to perform the following tasks at runtime:

- Add a Schedule Entry
- Add a Recurring Schedule Entry
- Edit a Schedule Entry
- Enable Override Mode
- Manually Set a State Change
- Configure Special Days
- View and Edit BACnet Schedules at Runtime.

**Note:** Special circumstances apply to schedule entries that occur during a daylight savings transition. For more information, see [Manage Daylight Saving Transitions](#).

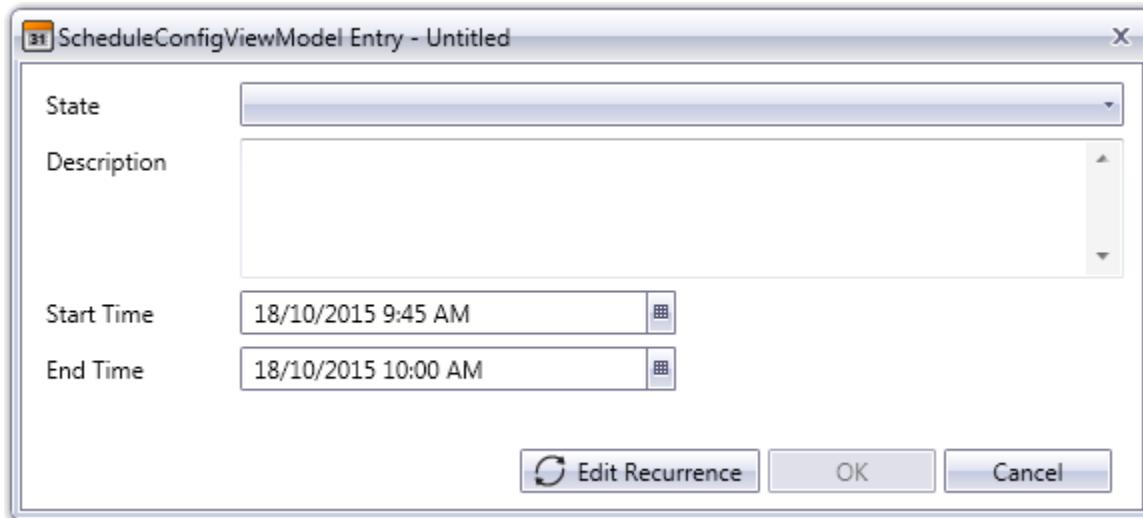
### Add a Schedule Entry

Schedule entries are used to specify a change of state for a selected piece of equipment. A schedule entry specifies a **Start Time** and **End Time** for a particular **State**.

**Note:** You can also create recurring schedule entries that follow patterns based on a day, week, month or year. See [Add a Recurring Schedule Entry](#).

#### To add a new schedule entry:

1. Open the Scheduler in the runtime environment and select the **Configuration View** tab.
2. In the [Equipment Tree](#), select the piece of equipment (or equipment hierarchy location) that requires a schedule entry.
3. In the [Calendar](#), double-click on the location that represents the date and time at which the schedule entry should occur. The Schedule Entry dialog will display.



**Note:** If you would like to add a schedule entry at a time where one already exists, you will need to select a time just outside the range of the existing entry and manually adjust the **Start Time** and **End Time**

---

accordingly. Be aware that this will create a schedule conflict (see [Schedule Priorities](#)).

4. In the **State** field, select the equipment state you would like the schedule entry to execute. The drop-down menu includes the states that are configured for the selected equipment.
5. Add a **Description** of the schedule entry (if required).
6. Use the **Start Time** and **End Time** fields to specify when the state change should commence and its duration. The button to the right of each field displays a date and time selection tool.
7. Click **OK**.

---

**Note:** Special circumstances apply to schedule entries that occur during a daylight savings transition. For example, the transition period is read-only which means schedule entries cannot be added. For more information, see [Manage Daylight Saving Transitions](#).

---

## See Also

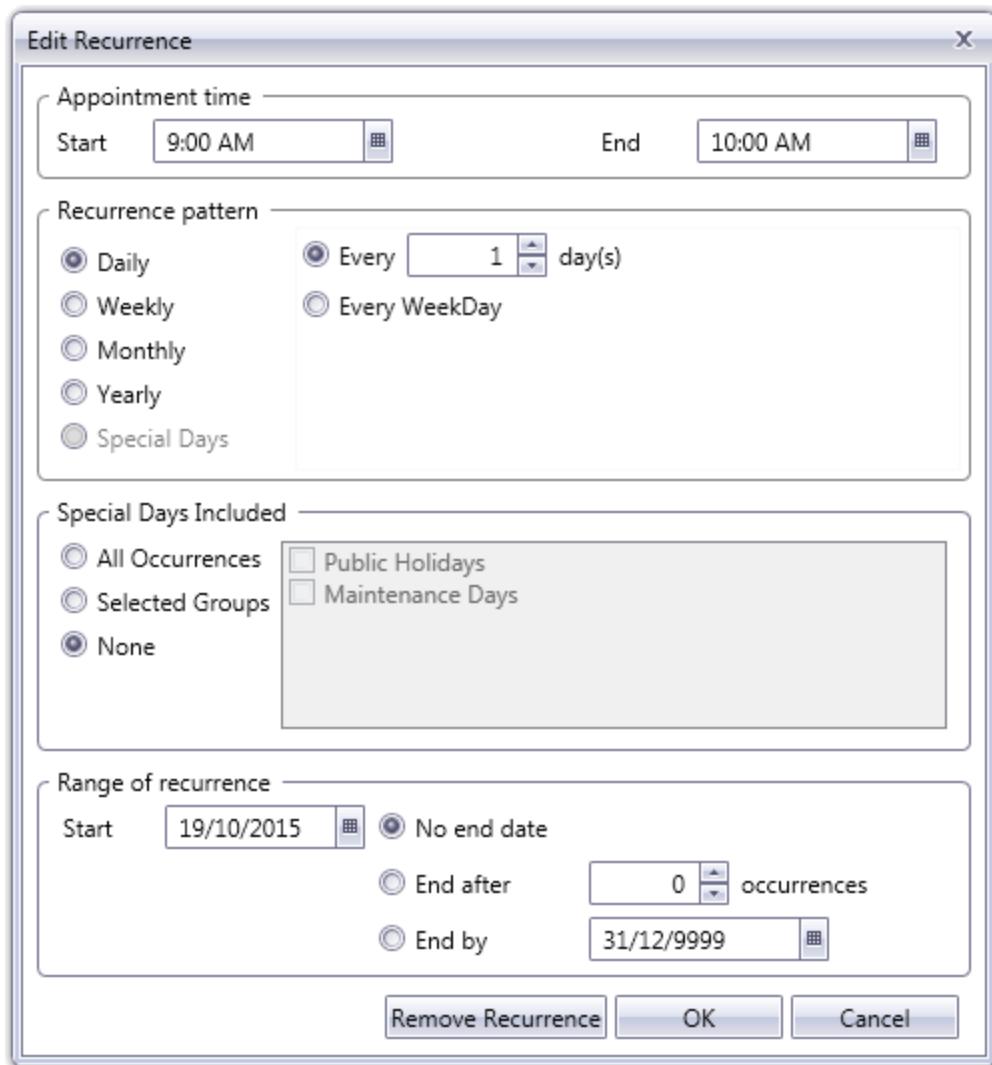
[Add a Recurring Schedule Entry](#)

## Add a Recurring Schedule Entry

A recurring schedule entry allows you to specify ongoing state changes for a piece of equipment based on a day, week, month or year.

### To add a recurring schedule entry:

1. Create a schedule entry following the steps described in [Add a Schedule Entry](#).
2. If required, double-click on the required schedule entry in the Scheduler calendar to open the Schedule Entry dialog.
3. Confirm that the period of time specified by the **Start Time** and **End Time** fields is not longer than 24 hours, as this is the maximum period of time supported by a recurring schedule entry.
4. Click on the **Edit Recurrence** button. The Edit Recurrence dialog will display.



5. In the **Recurrence pattern** section, select one of the following:

- **Daily** — Specifies how regularly the schedule will occur based on the **Every** field value. For example, "2" would mean the state change occurs every second day. Alternatively, you can specify that the state change will occur **Every Week Day** (Monday to Friday).
- **Weekly** — Specifies how regularly the schedule will occur based on the **Every** field value. For example, "2" would mean the state change occurs every fortnight. You can also use the check boxes to indicate that a state change should only happen on certain days of the week.
- **Monthly** — Specifies that the state change occurs on the same day across a monthly cycle (for example, "Day 1 of every 3 month(s)"), or on a particular day within a specific week (for example, "The Last Friday of every 1 month(s)").
- **Yearly** — Specifies that the state change occurs on the same date every year (for example, "Every January 1"), or on a particular day within a specific month (for example, "The First Tuesday of November")
- **Special Days** — Use this option if you want the state change to only occur on **Special Days** that fall on the selected week days. To enable the Special Days radio button, the **Special Days Included** section needs to be set to something other than **None**.

6. In the **Special Days Included** section, specify on which special days the state change will occur:

- **All Occurrences** — indicates the schedule entry will occur on every special day (regardless of the category).
  - **Selected Categories** — indicates the schedule entry will occur on the special days within the selected categories. You can select more than one category.
  - **None** — indicates the state change will not occur on any special days.
7. In the **Range of recurrence** section, specify the recurring schedule's **Start** date and an end date (if required). You can specify an end date using one of the following options:
- **No end date** — indicates that the state change will occur indefinitely
  - **End After** — specifies a selected number of occurrences (for example "9" indicates the state change will occur nine times).
  - **End by** — specifies a particular end date.
8. Click **OK** to close the **Edit Recurrence** dialog.
9. Click **OK** to save the schedule entry.

---

**Note:** To remove a recurrent pattern from a schedule entry, open the Edit Recurrence dialog and click on the **Remove Recurrence** button.

---

## See Also

[Configure Special Days](#)

## Edit a Schedule Entry

You can edit a schedule entry in the following ways:

- Update the schedule entry properties
- Move a schedule entry to a different time
- Delete a schedule entry.

---

**Note:** You are not able to edit an inherited schedule entry (see [Schedule Inheritance](#)). To edit a schedule entry, you need to determine the branch in the equipment tree on which it was created and make you required changes at that level.

---

### To update the properties for a schedule entry:

1. Open Scheduler in the runtime environment.
2. Display the **Configuration View** tab.
3. In the equipment tree, select the equipment you would like to update.
4. In the Scheduler calendar, double-click on the schedule entry you would like to change. The Schedule Entry dialog will appear.
5. Make the required changes to the schedule entry. Use the **Edit Recurrence** button if you would like to add or edit a [Add a Recurring Schedule Entry](#).
6. Click **OK** to save your changes.

**To move a schedule entry:**

1. Open Scheduler and display the Configuration View tab.
2. In the equipment tree, select the equipment you would like to update.
3. In the Scheduler calendar, locate the schedule entry you would like to move.
4. Click on the schedule entry and drag it to the required location on the calendar. The **Start Time** and **End Time** properties will update according to the entry's new location.

You can move a schedule entry to a location where one already exists, however this will create a schedule conflict (see [Schedule Priorities](#)).

---

**Note:** The drag-and-drop feature described here is not supported for recurring schedule entries.

---

**To delete a schedule entry:**

1. Open Scheduler and display the **Configuration View** tab.
2. In the equipment tree, select the equipment you would like to update.
3. In the Scheduler calendar, locate the schedule entry you would like to remove.
4. Hold the mouse over the schedule entry to display a small close icon. Click on this icon to delete the schedule entry.  
Or:  
Select the schedule entry and press the **Delete** key.
5. Click **OK** on the Delete Item dialog to confirm your selection.

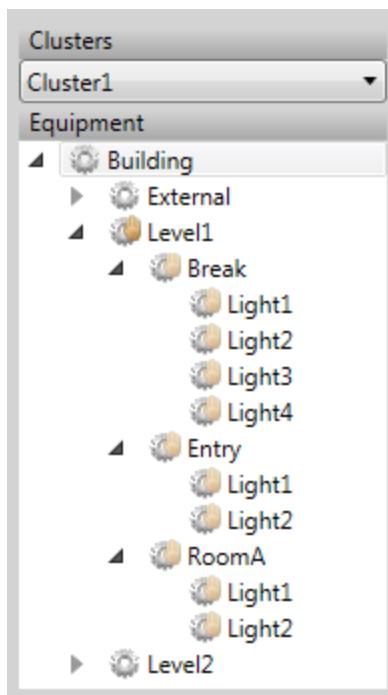
## Enable Override Mode

[Override Mode](#) allows you to take control of equipment scheduling and manually execute state changes.

**To enable override mode:**

1. Open [Scheduler](#) in the runtime environment.
2. In the equipment tree, locate the equipment you would like to set to override mode.
3. Right-click and select **Override**.

A small hand will appear on the icon next to the selected item indicating it is now in override mode. In the example below, "Level1" is in override mode.



**Note:** Override mode is automatically propagated down the equipment hierarchy, which means it is inherited by the equipment on lower levels (see [Override Mode](#)).

When override mode is enabled for equipment, you can manually set equipment state changes (see [Manually Set a State Change](#)).

#### To disable override mode:

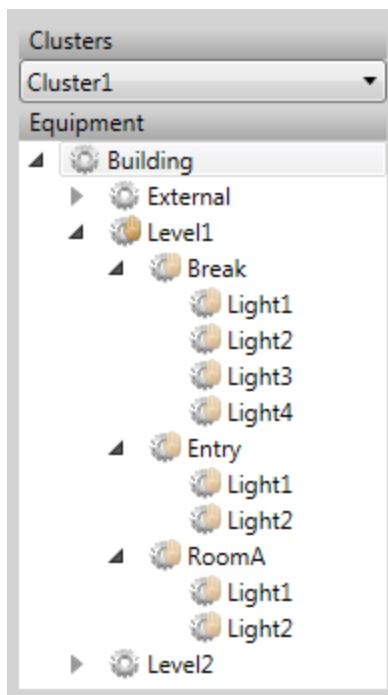
- In the equipment tree, right-click on the equipment and select **Remove Override**.

### Manually Set a State Change

When [Override Mode](#) is enabled for a piece of equipment, you can manually apply state changes to the equipment (and the equipment levels beneath it).

#### To manually set state changes to equipment in override mode:

1. Open [Scheduler](#) in the runtime environment.
  2. In the equipment tree, locate the equipment to which you would like to manually apply a state change.
- If the equipment is in override mode a small hand will appear on the icon to the left. In the example below, "Level1" is in override mode.



If this is not the case, follow the procedure described in [Enable Override Mode](#) to place the equipment into override mode.

3. Right click on the branch and select **Set to State**. The Set To State dialog will appear.
4. In the **State** field, select the equipment state you would like to implement. The drop-down menu to the right of the field includes a list of the equipment states that are configured for the selected piece of equipment.
5. Click **OK**.

**Note:** Equipment state changes are propagated down the equipment hierarchy to the equipment that has inherited override mode. If required, you can counteract a state specified by an inherited override by implementing another override at the level where you would like a different state to be set.

## Configure Special Days

[Special Days](#) are used to identify those days where regular scheduling does not apply to the equipment in a plant.

Each special day belongs to a category. To configure a special day, you should do the following:

### Creating a special day category

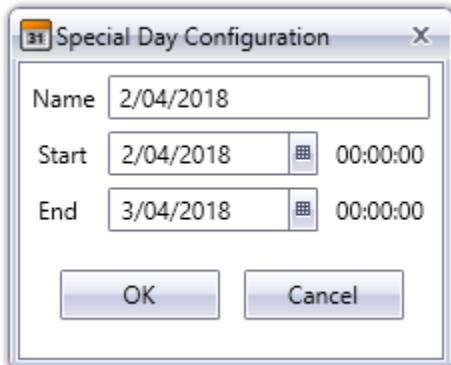
1. Open [Scheduler](#) in the runtime environment.
2. Click the **Special Days View** tab.
3. In the **Categories** panel, click the **Add** button.
4. The **Add Category** dialog box appears. Enter a unique name for the category and click **OK**. The new entry appears in the list of categories.

If the **Add** button is not available, go to the Scheduler page in Graphics Builder and set the "IsAddSpecialDaysButtonVisible" property value to "1." Refer to [Use ActiveX Properties to Control](#)

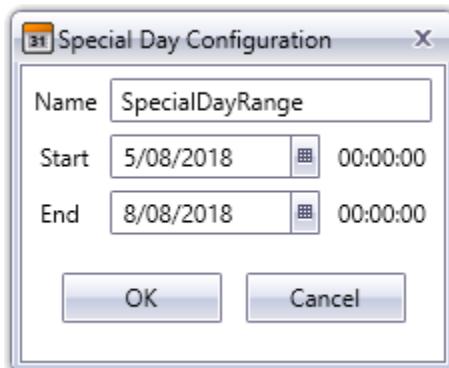
[Scheduler's Current Selection](#) for more details.

## Adding special days to the Scheduler calendar

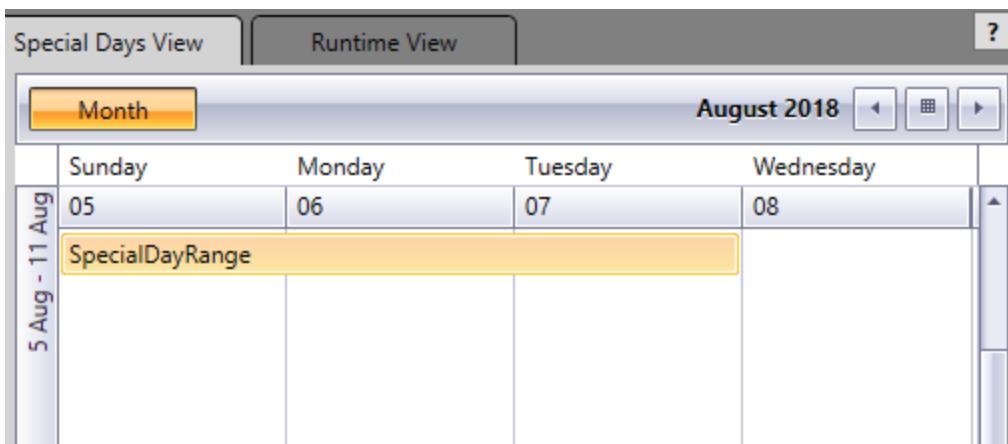
1. Open [Scheduler](#) in runtime environment and click the **Special Days View** tab.
2. In the **Categories** panel, select the category to which you need to add special days. (If required, follow the procedure described above to create a new category).
3. In the Scheduler calendar, navigate to the month with the day (or range of days) that you need to specify as a special day.
4. Double-click on a day that you want to specify as a special day.



5. The **Special Day Configuration** dialog box appears. Enter a name, start date and end date. A timestamp of "00:00:00" (the start of a day) applies to the **Start** and **End** fields. The configuration does not include the end date in the special days range, because the associated timestamp occurs as the day begins. For example, if you specify the dates from 5/08/2018 to 8/08/2018:



The span of the special days range highlights from 5/08/2018 to 7/08/2018.



6. Click **OK**.

If the special days are adjacent to each other and share the same name, then it will be grouped together in a single block.

## See Also

[Add a Recurring Schedule Entry](#)

## Manage Daylight Saving Transitions

In some time zones, the time is put forward or back one hour in response to the start and end of daylight saving time (DST).

In the [Scheduler](#) calendar, the period when a DST transition takes place is grayed out. You are unable to add a schedule entry that starts or ends in the transition period. When a schedule entry includes the transition period, the schedule will run according to the rules below:

### For the start of daylight saving:

For single and recurring schedule entries that include the transition to DST, the entries will run an hour less due to time shifting forward one hour. For example, if the schedule entry is from 1:30 to 3:30 A.M., and the time is set to switch at 2:00 A.M. (forward to 3:00 A.M.), the schedule will run from 1:30 to 2:00 A.M., and then from 3:00 A.M. to 3:30 A.M..

For recurring schedule entries set to end during the transition to DST, the entry will run from start time up to 1:59 A.M. (before the transition time). For example, if the schedule entry is set from 1:30 A.M. to 2:30 A.M., and the time is set to switch at 2:00 A.M. (forward to 3:00 A.M.), the schedule will run from 1:30 A.M. to 1:59 A.M., and on switching to DST will skip to 3:00 A.M. thus passing the end time of 2:30 A.M..

For recurring schedule entries set to start during the transition to DST, the entry will run from 3:00 A.M. to the end time. For example if the schedule entry is set from 2:30 A.M. to 4:00 A.M., the schedule will skip to 3:00 A.M. and run from 3:00 A.M. to 4:00 A.M..

## For the end of daylight saving:

For single schedule entries that include the transition back to normal time from DST, the entries will run for an additional hour due to the time shifting back one hour. For example, if the schedule entry is from 1:30 to 3:30 A.M., and the time is set to switch at 3:00 A.M. (back to 2:00 A.M.), the schedule will run from 1:30 to 3:00 A.M., and then from 2:00 A.M. to 3:30 A.M..

For single schedule entries that are set to end during the transition back from DST, the entry will run for an additional period from end time to 3:00 A.M. (2:00 A.M.). For example, if the schedule entry is set from 1:30 to 2:30, the schedule will run from 1:30 A.M. to 3:00 A.M. (2:00 A.M.), then from 2:00 A.M. to 2:30 A.M. for a total of 2 hours.

For recurring schedule entries that are set to end during the transition back from DST, the entry will run before the switch back from DST. For example, if the schedule entry is set from 1:30 A.M. to 2:30 A.M., the schedule will run from 1:30 A.M. to 2:30 A.M. for one hour only.

For single and recurring schedule entries that are set to start during the transition back from DST, the entry will run after the switch from DST. For example, if the schedule entry is set from 2:30 to 3:30, the schedule will run from 2:30 A.M. to 3:30 A.M. for 1 hour only.

---

**Note:** To see resolved schedule entries and how the Scheduler is set to behave during the transition to and from DST, display the Runtime View in the [Calendar](#).

---

## View and Edit BACnet Schedules at Runtime

Scheduler is able to display BACnet schedules that have been configured on a BACnet device and imported into Plant SCADA. This means you can read and write to schedule and calendar objects on a BACnet device at runtime using the Scheduler interface.

---

**Note:** Runtime View is not supported for BACnet schedule objects. It is recommended that you use the Configuration View when displaying BACnet schedules in Scheduler.

---

Schedules that were configured on a BACnet device will demonstrate the following when displayed in Scheduler:

- In the equipment tree, each schedule object associated with a BACnet device will appear as a branch.
- When a schedule object is selected in the equipment tree, the schedule entries that display will reflect the schedule configured on the BACnet device.
- Instead of specifying an equipment state change, BACnet schedule entries specify a value that can be interpreted by the BACnet device.

Exception schedules that were configured on a BACnet device will demonstrate the following when viewed in Scheduler:

- Exception schedules will appear on the Configuration View on a highlighted special day.
- Each exception schedule on the BACnet device is represented by a category in the Special Days View. The date or date range of the exception schedule will be shown as special days.
- Each calendar object will be available as a single category in the Special Days View.
- The **Type** field on the schedule entry properties will indicate if a schedule entry is a "weekly" or "exception" schedule.

---

**Note:**

- If the date range specified for an exception schedule or calendar object is excessively long, it can affect

---

system performance. For this reason, only the first six months of a date range will display in Scheduler.

- If the date range specified for an exception schedule or calendar object has a start date that is beyond three years in the future, it will not display in Scheduler.
- 

### You can make the following changes to BACnet schedule entries:

1. Edit the **Start Time** and **End Time** within the boundaries of a day (from midnight to midnight).  
A schedule entry will still end at midnight even if the specified **End Time** is later.
2. Edit the **Value**.
3. Delete a schedule entry.

### See Also

[Edit a Schedule Entry](#)

## Monitor Schedules

You can monitor the execution of schedule entries using the following trace messages:

- [Kernel Trace Messages](#)
- [Tracelog Trace Messages](#)

### Kernel Trace Messages

Scheduler outputs status information to the Kernel regarding the Runtime Transition Manager and the Redundancy engine.



### WARNING

#### UNINTENDED EQUIPMENT OPERATION

- Do not use the kernel for normal Plant SCADA operation. The kernel is only for diagnostics and debugging purposes.
- Configure your security so that only approved personnel can view or use the kernel.
- Do not view or use the kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Runtime Transition Manager

These messages are prefixed with "Runtime Transition Manager:"

- **Start delayed** - Start operation was requested but delayed due to the start delay period not being elapsed.

The message indicates that the start will occur later and specifies in how many milliseconds. A timestamp is supplied indicating the local time that the start was called.

- **Started** - Runtime Transitions Manager has been started, and a local timestamp supplied indicating when.
- **Delayed Start Canceled** - Prior start call which was queued due to startdelay has been dequeued and not started.
- **Stopped** - Runtime Transition Manager has halted its operation, a local timestamp indicating when.

## Redundancy Engine

These messages are prefixed with "Scheduler Redundancy:"

- **Synchronization stopped due to connect failed** - Synchronization has been aborted due to a lack of connection between the local and peer.
- **Synchronization stopped due to disconnect** - Synchronization has been aborted due to a connection disconnection between the local and peer.
- **Synchronization initiated from (peer or local)** – Synchronization has been started. Peer or local will be specified in this method, and indicates which server is initiating the synchronization start.
- **Synchronization Complete** - This indicates that a synchronization has successfully completed.

## Tracelog Trace Messages

The Scheduler outputs diagnostics information to the platform trace log.

For more information regarding log files and the rules belonging to platform logging systems refer to the section **Monitoring and Debugging Runtime** in the Plant SCADA main help.

The tracelog.Report.<Report server name>.dat file contains report server related trace information and is similar to the syslog.dat file.

Refer to the [Debug Parameters](#) help for information pertaining to logging. If you require additional information refer to the help for the [Computer Setup Editor](#).

---

**Note:** By default the logging levels are set to every category, but only Critical and Error level traces will be logged. As the traces from the Scheduler are currently Informational only, change this level to include information. (If that is the only change made, then informational traces from every component will be directed to the tracelog) With manipulation of the debug ini parameters, you can fine tune a fairly granular trace set.

Example for Scheduler critical, error, information ONLY:

```
[Debug]SeverityFilter= Critical,Error,Information  
CategoryFilterMode=0  
CategoryFilter=Scheduler
```

---

## Runtime Transition Manager

These messages are prefixed with "Runtime Transition Manager:"

- **Start delayed** - Start operation was requested but delayed due to the start delay period not having elapsed. The message indicates that the start will occur later and specifies in how many milliseconds. A timestamp is supplied indicating the local time that the start was called.
- **Started** - Runtime transitions manager has been started, and a local timestamp supplied indicating when.

- **Delayed Start Cancelled** - Prior start call which was queued due to startdelay has been dequeued and not started.
- **Stopped** - Runtime transition manager has halted its operation and a local timestamp indicating when.

## Equipment

These messages are prefixed with "Equipment"

- **Mode changed** - Gives an equipment name, the previous mode (Automatic or Manual) and the new mode, along with the standard timestamp information of when the transition occurred.
- **Demand Response Mode changed** - Gives an equipment name, the previous demand response mode (numeric) and the new mode, along with the standard timestamp information when the transition occurred.
- **Manual state changed** - Gives an equipment name, the previous state (string) and the new state, along with the standard timestamp information when the transition occurred. This message is only triggered on manual override state changes.
- **State changed** - Gives an equipment name, the previous state (string) and the new state, along with the standard timestamp information when the transition occurred. This message will be triggered for both automatic equipment state changes and manual ones.

---

**Note:** Currently traces for equipment property changes are logged on the server that they occurred on. No updates done via synchronization are logged.

# Graphics

This section of the documentation describes the tools and templates that are available to help you create graphical content for a Plant SCADA project.

## See Also

- [Graphics Builder](#)
- [AVEVA Industrial Graphics](#)
- [Situational Awareness Projects](#)

## Graphics Builder

Graphics Builder is the tool you use to create and configure graphics pages. You can add content to a page that makes it reflect the area or processes it will be used to monitor.

### To open Graphics Builder:

- On the Activity Bar, select **Graphics Builder** from the menu.
- Or:
- Click the Graphics Builder icon.



A graphics page is typically based on a template that conforms to the style applied to every page included in a project (see [Page Templates](#)). For example, if your current project is based on the SxW starter project, you would create a new graphics page from the "Normal" template based on the SxW style. The page that opens in Graphics Builder will include navigational tools and functionality that are common to every page in the project (for example, the alarm banner, navigation panel and equipment hierarchy).

A page template also allows you to build graphics pages that are an appropriate [Screen Resolution](#) for your client computers.

To create content for a graphics page, you use graphics objects that allow you to create a meaningful representation of the system that is being monitored. These objects can be configured to display system data and respond to tag value changes. A wide range of graphics objects are supported in Graphics Builder, from simple shapes to complex [Symbols](#), [Genies](#) and [Composite Genies](#).

When you add an object to a page, it is automatically allocated an animation point number (AN). This provides a way to reference animations at runtime (see [Animation Points](#)).

You can also add commands and controls to a graphics page, providing a way for an operator to interact with a system at runtime (see [Interact with Graphics Pages at Runtime](#)).

## See Also

- [Graphics Builder Settings](#)
- [Configure Graphics Pages](#)
- [Configure Graphics Objects](#)
- [Keyboard Commands](#)
- [Dynamic Associations](#)

## Graphics Builder Settings

The following topics explain how to configure and use the tools and features supported by Graphics Builder.

- [Set Graphics Builder Options](#)
- [Display the Drawing Toolbox](#)
- [Use a Grid to Align Objects](#)
- [Use Guidelines to Align Objects](#)
- [Display the Zoom Dialog Box](#)
- [Use a Cross Hair Cursor](#)
- [Customize Graphics Builder.](#)

## See Also

- [Configure Graphics Pages](#)

## Set Graphics Builder Options

You can define general options for the drawing environment.

### To define general options for Graphics Builder:

1. Open Graphics Builder.
2. Choose **Tools | Options**. The Options dialog box appears.
3. Select the required options (a description of the options is given below).
4. Click **OK**.

## Graphics Builder - General Options

Option	Description
<b>Display Properties on New</b>	Enables the automatic display of the relevant properties dialog when you add an object to the page.
<b>Display Properties on Copy</b>	Enables the automatic display of the relevant

Option	Description
	properties dialog when you copy an existing object on the page.
<b>Save Template Warning</b>	Enables the display of an alert message when you modify and then save an existing template. When you modify an existing template, any graphics pages that are associated with the template are not updated until you perform an <a href="#">Update Equipment in Plant SCADA Studio</a> to update each page based on the template.
<b>Modify AN Field</b>	Allows you to modify the number of the animation point (AN) of any object. You cannot change an AN to the same number as an existing AN on the graphics page.
<b>Disable Genie Forms</b>	Disables the display of Genie forms when a new Genie is added to the page or an existing Genie is edited. With Genie forms disabled, a form for each native object in the Genie displays.
<b>Display Group Button</b>	Enables the display of a group button on a Genie dialog. The group button displays a form for each native object in the Genie.
<b>Compile Enquiry Message</b>	Enables the "Do you want to compile?" message window when the project has been modified and <b>Run</b> is selected. Normally the project is compiled automatically (if the project has been modified) when <b>Run</b> is selected.
<b>Fast Runtime Display</b>	Enables the fast display of graphics pages in the runtime system.
<b>List System Pages</b>	<p>Specifies that system pages will be included in:</p> <ul style="list-style-type: none"> <li>• The list of pages in the Graphics Builder Open and Save dialog boxes.</li> <li>• The Page Properties Previous and Next menus, used for defining a browse sequence for your pages.</li> </ul> <p>System pages are prefixed with an exclamation mark (!).</p>
<b>Show version 3.xx/4.xx tools</b>	Enables the old (version 3 and version 4) toolbox. This toolbox contains old tools (such as Slider and Bar Graph), which are no longer necessary, as they can be configured using the Object properties.

Option	Description
<b>Transparent Paste</b>	<p>Allows you to specify a color that becomes transparent when a bitmap is pasted on a graphics page. This applies to bitmaps that are pasted from the clipboard, or imported from another application.</p> <p><b>Note:</b> Transparent data bits are not natively supported by other applications. If pasting a bitmap into an external application, transparent bits will appear as the transparent paste color.</p>

## See Also

[Graphics Builder Settings](#)

## Customize Graphics Builder

You can customize the Graphics Builder workspace and adjust the cursor speed.

Procedure	Description
To hide the status bar:	Choose <b>View   Show Status Bar</b> . Choose the command again to redisplay the status bar.
To hide the toolbar:	Choose <b>View   Show Tool Bar</b> . Choose the command again to redisplay the tool bar.
To change the speed of the cursor when using the mouse:	Choose <b>View   Mouse Slow Speed</b> .
To change the speed of the cursor when using the cursor keys:	<p>Choose <b>View   Cursor Keys Slow Speed</b>.</p> <p><b>Hint:</b> Move the cursor on screen by using the left, right, up, and down cursor keys.</p>

## See Also

[Graphics Builder Settings](#)

## Display the Drawing Toolbox

Plant SCADA's Drawing Toolbox allows you to add graphics objects to a page, symbol or template in Graphics Builder (see [Graphics Object Types](#)).

**To display the drawing toolbox:**

1. Open Graphics Builder.
2. On the **View** menu, select **Show Tools**.

Or:

Press the **F3** key.

**To hide the drawing toolbox:**

- On the **View** menu, deselect **Show Tools**.

Or:

Press the **F3** key.

Or:

Click on the button to the left of the toolbox title bar.

You can also minimize the drawing toolbox without closing it. To do this, select the button to the right of the toolbox title bar.

---

**Note:** The Drawing Toolbox may include a button that lets you insert a Pelco™ Viewer Camera Control. This button will only appear if you have previously installed this control on the local computer. The Pelco ActiveX control has been deprecated and is no longer part of the Plant SCADA installation. If you have upgraded your computer from a previous version with this control installed, it will continue to function. However, it is recommended that you communicate with Pelco cameras using a different method.

---

**See Also**

[Graphics Objects](#)

## Use a Grid to Align Objects

You can use a grid to align and place objects with precision. When the grid is active, any objects or groups of objects that you create, move, or re-size snap to the nearest grid intersection. Options for the grid can be set up in the Grid Setup dialog.

### To open the Grid Setup Dialog:

1. Open Graphics Builder.
2. From the **View** menu, select **Grid Setup**.
3. Set the required options (see the table below).
4. Click **OK**.

Field	Description
<b>Pitch</b>	The horizontal and vertical spacing of the grid points (in pixels). The smallest grid size you can specify is 3 x 3 pixels.
<b>Origin</b>	Specifies the grid origin (base point).
<b>Display Grid</b>	Displays the grid on screen.
<b>Snap to Grid (F8)</b>	Activates the grid. When the grid is active, any object or group of objects that you create, move, or re-size snaps to the nearest grid intersection.

## Use Guidelines to Align Objects

You can use horizontal and vertical guidelines as a straight-edge to align and place objects. When an edge or the center of an object gets close to a guideline, that edge or center automatically snaps to the guideline. Options for guidelines can be set up in the Guidelines Setup dialog. This dialog box lets you define 32 horizontal and 32 vertical guidelines. Guidelines act as a straight-edge, allowing you to align and place objects precisely. When an edge or center of an object gets close to a guide, that edge or center automatically snaps to the guide.

### To open the Guidelines Setup dialog:

1. Open Graphics Builder.
2. From the **View** menu, select **Guidelines Setup**.
3. The dialog provides the following options.

Field	Description
<b>Direction</b>	The direction of the guideline (horizontal or vertical).
<b>Display Guidelines</b>	Displays the guidelines on screen.
<b>Snap to Guidelines (F7)</b>	Activates the guidelines. When the guidelines are

Field	Description
	active, any object or group of objects that you create, move, or re-size snaps to the nearest guideline.

**To run a guideline horizontally across a page:**

1. Select **Horizontal**.
2. Enter a position (distance from the top of your page) for the guideline, and click **Set**.

**To run a guideline vertically down a page:**

1. Select **Vertical**.
2. Enter a position (distance from the left edge of your page) for the guideline, and click **Set**.

**To display the guidelines:**

1. From the **View** menu, select **Guidelines Setup**.
2. On the **Guideline Setup** dialog box, select the **Display Guidelines** check box.

**To snap objects to the nearest guideline:**

- On the **Guidelines Setup** dialog box, select the **Snap to Guidelines** check box.  
Or:  
Press the **F7** key.

**To move a guideline:**

1. Move the cursor over the guideline.
2. Click and hold the left mouse button.
3. Move the guideline to a new location.
4. Release the mouse button.

**To create a new guideline with the mouse:**

1. Move the cursor over the guideline.
2. Press and hold the **Ctrl** key.
3. Click and hold the left mouse button.
4. Move the guideline to a new position.
5. Release the mouse button and **Ctrl** key.

**To delete a guideline with the mouse:**

- Drag the guideline to the edge of the page.

## Display the Zoom Dialog Box

Use the Zoom dialog box to display an enlarged view of your drawing. The Zoom dialog box magnifies the area around your cursor, enabling you to position or draw objects precisely.

You can alter the magnification level by selecting **Zoom In** or **Zoom Out** on dialog's control menu (accessible to the left of the dialog's title bar).

### To display the Zoom dialog box:

1. Open Graphics Builder.
2. On the **View** menu, select **Show Zoom**.  
Or:  
Press the **F10** key.

### To hide the Zoom dialog box:

1. On the **View** menu, deselect **Show Zoom**.  
Or:  
Press the **F10** key.  
Or:  
Double click on the dialog's control menu.

## See Also

[Customize Graphics Builder](#)

## Use a Cross Hair Cursor

You can change the cursor into cross hairs that extend the full width and length of the drawing area. When you move away from the drawing area, the normal pointer reappears, allowing you to select commands and tools.

### To display a cross hair cursor:

1. Open Graphics Builder.
2. From the **View** menu, select **Cross Hair Cursor**.

## See Also

[Customize Graphics Builder](#)

## Colors

Plant SCADA supports True Color (16.7 million colors) for static and animated objects, including page backgrounds, imported images, symbols, metafiles and bitmaps.

Wherever a particular page or object property has a color value, a current color button will appear.



To choose a different color to the one currently displayed on the button, click on the small black arrow to the right. This launches the [Color Picker](#).

The following tools are also provided to help manage the colors used on your graphics pages:

- [Edit Favorite Colors Dialog Box](#)
- [Swap Color Dialog Box](#)
- [Adjust Colors Dialog Box](#)

## Color Picker

The first 11 rows of the Color Picker show a set of standard colors, including transparent (marked with a black cross). The remaining rows display any user defined colors, referred to as **Color Favorites**. This includes flashing colors, represented by a two color block, divided diagonally.

To select one of the colors displayed on the color picker, click on it. If the necessary color does not appear, you have the option to create a custom color, or match an existing color from one of your graphics pages.

### To match an existing color on a graphics page:

1. Verify that the color you would like to match is present on the page currently displayed in Graphics Builder.
2. From the Color Picker, choose the Color Selector tool (looks like an eyedropper).
3. Use the Color Selector to click on the color you would like to match.
4. The color you have chosen will now appear in the Color Value Display button.

### To create a customized color:

1. From the Color Picker, click the **Edit** button. The [Edit Favorite Colors Dialog Box](#) is displayed. Use the to create the color you would like to use.
2. **Name** the color if necessary.
3. Use the **Add** button to include the color with the Color Favorites displayed on the Color Picker.
4. Click **OK**. The color you have just created will now be selected.

## See Also

[Edit Favorite Colors Dialog Box](#)

[Swap Color Dialog Box](#)

[Adjust Colors Dialog Box](#)

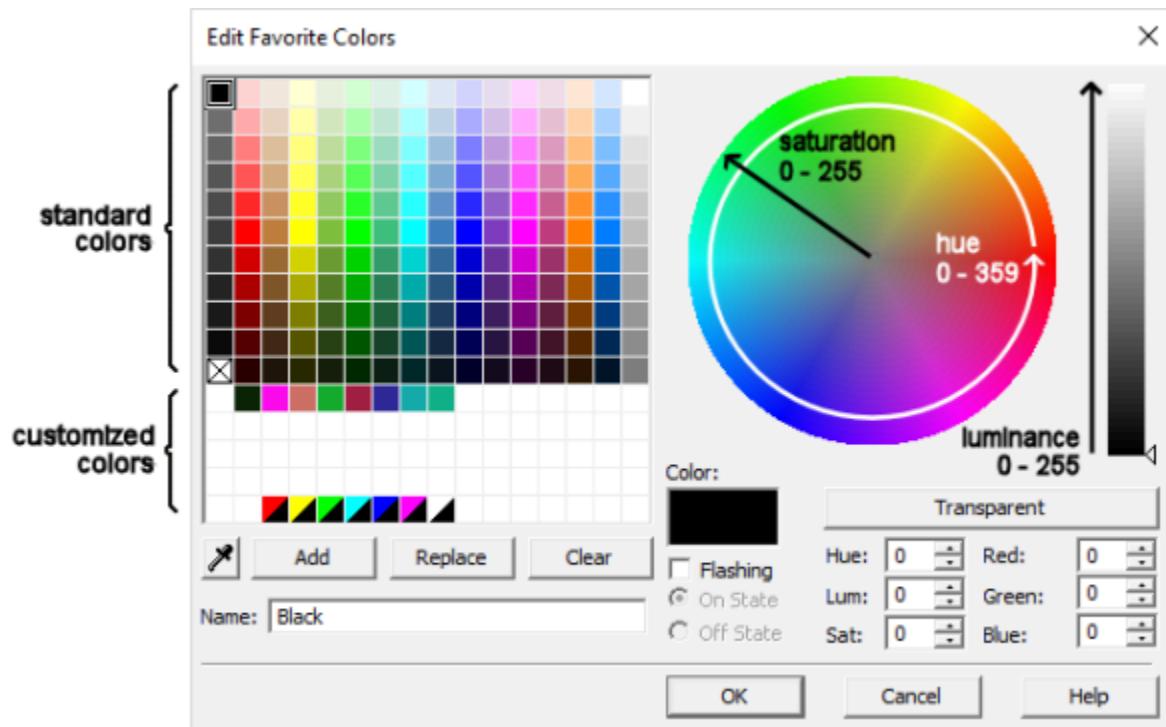
## Edit Favorite Colors Dialog Box

From the Color Picker, click the **Edit** button. The Edit Favorite Color dialog appears.

With the Edit Favorite Colors dialog box, you can:

- Make a visual selection of a color you would like to use by clicking on the color wheel.

- Decide the brightness level for a color by clicking on the brightness bar.
- Create a color by entering hue, saturation and luminance values.
- Create a color by entering red, green and blue color values.
- Modify the colors available on the Color Picker by adding, replacing and removing colors on the color grid.
- Name a color, allowing it to be identified on the color grid via a tool tip.



## Edit Favorite Colors Dialog Fields

Field	Description
<b>Colors Grid</b>	Displays a selection of predefined colors. The first 11 rows show a set of standard colors, including transparent (marked with a black cross). The remaining rows display the user-defined colors currently added as favorites. This includes flashing colors, represented by a two color block, divided diagonally. The colors that appear in this grid represent those that are available from the Color Picker.
<b>Add</b>	Adds the color currently displayed in the <b>Color</b> panel to the user-defined favorites in the color grid. If there are no places available on the grid, you will have to use the <b>Replace</b> or <b>Clear</b> button instead.
<b>Replace</b>	This button allows to you alter a color on the color

Field	Description
	grid. Select the color you would like to change, adjust its color value, and then hit the <b>Replace</b> button. The selected color will be updated to reflect the changes that were made.
<b>Clear</b>	Removes the selected color, leaving an "unused" position on the color grid.
<b>Name</b>	<p>Allows you to associate a name with a predefined color. The name can be viewed as a tool tip in the Color Picker, making it easy to distinguish a specific color among similar shades.</p> <p>You can associate a name with a newly created color by typing in a name before clicking the <b>Add</b> button. You can also apply a name to an existing color by selecting the color, keying in a name and clicking the <b>Replace</b> button.</p> <p><b>Note:</b> The predefined color labels are already defined as color names.</p>
<b>Color</b>	<p>The <b>Color</b> panel displays the color created by the current settings applied to the Edit Favorite Color dialog.</p> <p>The values displayed on the Edit Favorite Color dialog automatically adjust to correctly represent the color currently displayed in this panel. The values in each field are not independent.</p>
<b>Color wheel</b>	The color wheel allows you to visually select a color. It represents the full spectrum of colors in a cyclic layout, with color saturation increasing towards the outside edge. Simply click on the wheel to select a color.
<b>Brightness bar</b>	Allows you to visually select the brightness you would like applied to a color. Click on the bar in the appropriate location to apply a brightness level. The bar represents luminance, as the colors move away from pure black as they progress up the bar to pure white.
<b>Hue</b>	Specifies the hue value for the color currently displayed in the <b>Color</b> panel. Hue primarily distinguishes one color from another. The value can be between 0 (zero) and 359, representing degrees around the color wheel. For example, zero is a pure red to the right, 180 is pure cyan on the left.

Field	Description
<b>Sat</b>	Specifies the saturation level for the color currently displayed in the <b>Color</b> panel. Saturation level increases the further the selected color moves away from gray scale to a pure primary color. The value can be between 0 (zero) and 255.
<b>Lum</b>	<p>Specifies the luminance of the color currently displayed in the <b>Color</b> panel. Luminance represents the brightness of a color, the value increasing the further the color moves away from black towards pure white. The value can be between 0 (zero) and 255.</p> <p><b>Note:</b> When you create a color by using HLS values, you may find that the HLS values you specified for a color have changed when you reopen the dialog box. This happens because RGB values are less precise than HLS values, sometimes resulting in several HLS values being assigned the same RGB value. As a result, when the HLS values are generated from the RGB values, some values may change.</p>
<b>Red</b>	Indicates the amount of red used to create the color currently displayed in the <b>Color</b> panel. The value can be between 0 (zero) and 255. Adjust this setting if you want to create a color using RGB values.
<b>Green</b>	Indicates the amount of green used to create the color currently displayed in the <b>Color</b> panel. The value can be between 0 (zero) and 255. Adjust this setting if you want to create a color using RGB values.
<b>Blue</b>	Indicates the amount of blue used to create the color currently displayed in the <b>Color</b> panel. The value can be between 0 (zero) and 255. Adjust this setting if you want to create a color using RGB values.
<b>Transparent</b>	Click this button to select transparent. Wherever transparent is used as a color, the background color, or color behind the transparent object, will be displayed.
<b>Flashing</b>	Check this box if you want to create a flashing color. A flashing color appears as a diagonally divided panel in the color grid. To create a flashing color, you will have to select an <b>On State</b> and <b>Off State</b> color.

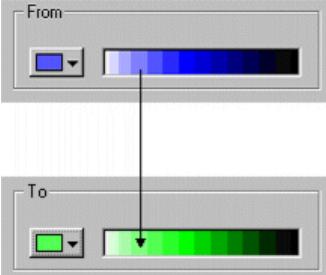
## Swap Color Dialog Box

You use the Swap Color dialog box to swap the colors of an object (or group of objects, or a bitmap) to new

colors. From the **Tools** menu in Graphics Builder, select **Swap Colors**. The Swap Colors dialog appears.

**Note:** You may find that you get unexpected results when swapping colors in your bitmaps if they contain areas that have either very high or very low luminance values (that is, white or black areas). This is due to the fact that these areas contain almost no color, only luminance. For example, areas that represent specular highlights (the shine on a brightly illuminated steel conveyor belt, for example) will remain colorless no matter how the color for the rest of the bitmap is adjusted. Consequently, before working with the Swap Color dialog box and other color tools, familiarize yourself with the concepts of hue, luminance, and saturation in order to avoid unexpected results with the images on your graphics pages.

## Swap Colors Dialog Fields

Field	Description
<b>From</b>	<p>The current color of the object. If you click the <b>Swap Range</b> check box, it presents a range of colors in varying degrees of brightness ranging from white to black. Any colors in this range that exist in the object are replaced by the corresponding colors from the <b>To</b> range.</p>  <p>If a color in the <b>From</b> range exists in the object, it will be replaced by the <b>To</b> color in the same relative position in the range.</p>
<b>To</b>	<p>The color to which the original object color will be changed. If you click the <b>Swap Range</b> check box, it presents a range of colors in varying degrees of brightness from white to black. This allows you to swap a whole range of colors at once.</p>
<b>From any color</b>	<p>Specifies to change every color in the object to the new color.</p>
<b>Swap range</b>	<p>Specifies to swap a range of colors. The <b>From</b> and <b>To</b> boxes indicate the starting colors in the ranges.</p> <p><b>Note:</b> You cannot invert colors with <b>Swap Range</b> selected. This means, for example, that you could not swap dark red for light green and light red for dark green in one go.</p>

## Adjust Colors Dialog Box

The Adjust Colors dialog box allows refined color re-mapping with Graphics Builder objects. From the **Tools** menu in Graphics Builder, select **Adjust Colors**. The Adjust Colors dialog appears.

## Adjust Colors Dialog Fields

Field	Description
<b>Hue Degrees</b>	<p>The Hue area allows the user to set the color hue range to map to and from. The bars displayed span values of 0 (zero) to 359 degrees, representing the cyclic nature of hue color values. A numeric value for each slider can be keyed in to the field to the right.</p> <ul style="list-style-type: none"><li>• The slider above the <b>From Hue Range</b> bar selects the start of the color range that will be mapped.</li><li>• The slider below the <b>From Hue Range</b> bar selects the end of the color range to be mapped.</li><li>• The slider above the <b>To Hue Range</b> bar selects the start point for the color range you'll be mapping to. Because the value range is cyclic, the selected area can wrap around to the left side of the bar.</li></ul> <p>The range of colors that is excluded by your selection is grayed out, allowing for a visual assessment of the selected range.</p>
<b>Lightness (%)</b>	The lightness slider allows you to boost the lightness of colors across a range of (negative) - 100% to 100%. If the slider is increased above zero, colors will tend towards white. If the slider is set below zero, colors will move towards black. A numeric value for the slider can be entered into the field to the right.
<b>Selected Hues Only</b>	Applies the lightness setting to only those colors that will be remapped. Leaving the box unchecked allows the lightness to be adjusted for every color.
<b>Saturation (%)</b>	The saturation slider allows you to boost the saturation of color across a range of (negative) -100% to 100%. If the slider is increased above 0, colors will tend towards primary colors. If the slider is set below zero, colors will move towards grayscale. A numeric value for the slider can be entered into the field to the right.
<b>Selected Hues Only</b>	Applies the saturation setting to only those colors that

Field	Description
	will be remapped. Leaving the box unchecked allows saturation to be adjusted for every color.

## Page Templates

Page templates are used in Plant SCADA to produce a consistent look and feel across all of the pages included in a project.

---

**Note:** If a project is based on a Situational Awareness Starter Project, it will not use page templates in the way described here. Situational Awareness projects use a workspace architecture that is based on a master page. For more information, see [Create a Master Page for a Customized Workspace](#).

---

The templates that are included with a Plant SCADA installation are grouped according to "styles". These styles include:

- **situational\_awareness** — see note above.
- **sxw\_style\_1** — a feature-rich template set designed for use with systems that conform to Schneider Electric's StruxureWare specification.
- **tab\_style\_1** — a feature-rich template set that includes a layout with a tabbed ribbon menu.
- **standard** — a legacy style with basic functionality, simple page navigation and alarm icons.
- **bottom** — a variation of the legacy standard style with the toolbars at the bottom of the screen.
- **top** — a variation of the legacy standard style with the toolbars at the top of the screen.

Each style has templates for the pages you are likely to use in a typical runtime project.

---

**Note:** If you need to generate multiple graphics pages that share a common look and feel and behavior, you can also create your own page template from any existing template (see [Create a Template](#)). Use your own template to apply changes in a single location and update all of your pages automatically (see [Link a Page to a Template](#)).

---

## Page Templates Common to All Styles

Template Name	Page Description
<b>alarm</b>	The Active Alarms page.
<b>blank</b>	A blank page.
<b>disabled</b>	The Disabled Alarms page.
<b>doublepa</b>	A page containing two instances of the Process Analyst.  Some older styles will contain a variation named "doubletrend" that displays two trends (without using Process Analyst).
<b>file</b>	Used to display a file.

Template Name	Page Description
	Versions of this template may exist that specifically support an RTF or HTML file (for example, "file_html" or "file_rtf").
<b>hardware</b>	The Hardware Alarms page.
<b>meanmeanchart</b>	A page that displays control charts for the mean of SPC tags.
<b>normal</b>	A page to which you can add content that reflects an area or process to be monitored.
<b>poppa</b>	A popup page that contains an instance of the Process Analyst.  Some legacy styles will contain a variation named "poptrend" that displays a trend (without using Process Analyst).
<b>singlepa</b>	A page containing a single instance of the Process Analyst.  Some legacy styles will contain a variation named "singletrend" that displays a single trend (without using Process Analyst).
<b>soe</b>	The Sequence Of Events page.
<b>standardchart</b>	A page with charts that display the standard deviation and mean for SPC tags.
<b>summary</b>	The Alarm Summary page.

## Example

A project that is created using the SxW starter project will automatically include a set of pages based on the style "sxw\_style\_1". This style supports visual components that are included on all pages, such as a menu panel, an equipment hierarchy and an alarm banner. The templates used to generate these pages are stored in the system project named "SxW\_Style\_Include".

To add a graphics page to this project, use the sxw\_style\_1 "normal" template. The page that is created will already include the visual components that are common to the other pages in the project. You can then add your own content to the page.

---

**Note:** To browse the page templates associated with the active project, use the **Libraries** view in the **Visualization** activity (see [Browse Libraries in Plant SCADA Studio](#)).

---

## See Also

[Screen Resolution](#)

## Screen Resolution

The page templates that are delivered with Plant SCADA are grouped according to "styles" (see [Page Templates](#)). Within each style, the templates are available in a variety of resolutions to support a range of possible screen sizes on a client computer.

### Screen Resolutions Supported by Each Template Style.

Style	Available Resolutions
<b>bottom</b>	VGA (640x480, 4:3) SVGA (800x600, 4:3) XGA (1024x768, 4:3)
<b>standard</b>	VGA (640x480, 4:3) SVGA (800x600, 4:3) XGA (1024x768, 4:3) SXGA (1280x1024; 5:4)
<b>sxw_style_1</b>	HD768 (1366x768, 16:9) HD1080 (1920x1080, 16:9)
<b>tab_style_1</b>	SXGA (1280x1024; 5:4) HD768 (1366x768, 16:9) HD1080 (1920x1080, 16:9) WUXGA (1920x1200, 16:10)
<b>top</b>	VGA (640x480, 4:3) SVGA (800x600, 4:3) XGA (1024x768, 4:3)

When you create a graphics page, you should consider the screen resolution of the client computer that will display the page at runtime. While you are able to adjust the size of the runtime window (see [Sizing the Runtime Window](#)), choosing a template in the correct resolution will make a page the appropriate size for full screen operation. It will also avoid the need for scroll bars (due to a page being too large).

If required, you may also need to consider the impact of running your project on multiple monitors (see [MultiMonitors Parameters](#)).

---

**Note:** The legacy standard templates styles (**standard**, **bottom** and **top**) come in "no title bar" variations. These are provided to support projects that will run in fullscreen mode with no title bar (where the citect.ini parameter [\[Animator\]FullScreen](#) is set to 1). The **sxw\_style\_1** and **tab\_style\_1** templates are designed to always include a title bar. If you use these templates, you can restrict usage of the standard title bar buttons with Cicode (see [Secure the Window Title Bar](#)).

If you need to adjust the screen resolution for a page (for example, if you are creating a popup page or your own template set from a blank template), you can change the screen resolution for a page at any time by adjusting the **Resolution** property (see [Edit the Properties for a Page](#)).

## Sizing the Runtime Window

By default, when the Plant SCADA runtime window initially opens, its size is determined by the dimensions of the project's startup page. You are able to re-size the runtime window by dragging its frame, and you can use the standard title bar buttons (Maximize, Minimize/Restore and Close).

However, there are a number of ways you can manipulate this behavior. For example, you can:

- operate in fullscreen mode
- open as a maximized window
- disable dynamic sizing
- lock the aspect ratio of the runtime window.

To implement these customizations, you use the following `citect.ini` parameters:

- [\[Animator\]FullScreen](#)
- [\[Animator\]MaximizedWindow](#)
- [\[Page\]DynamicSizing](#)
- [\[Page\]MaintainAspectRatio](#)
- [\[Page\]StartupMode](#)

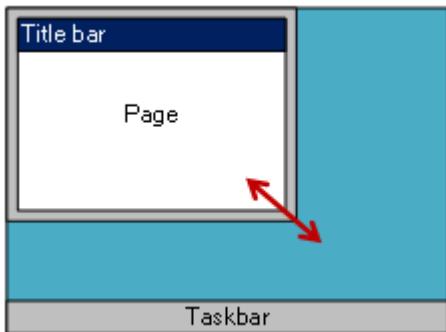
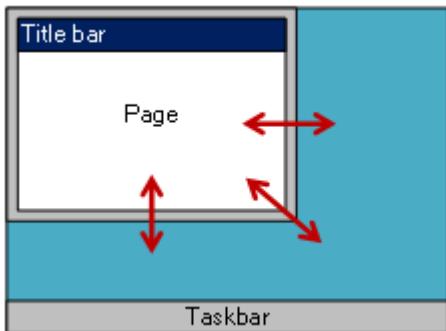
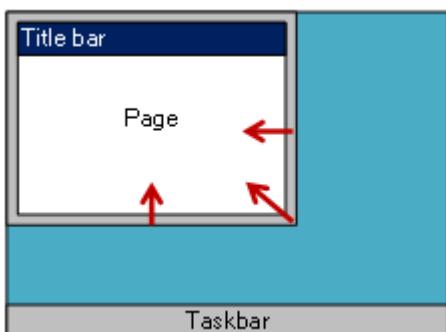
---

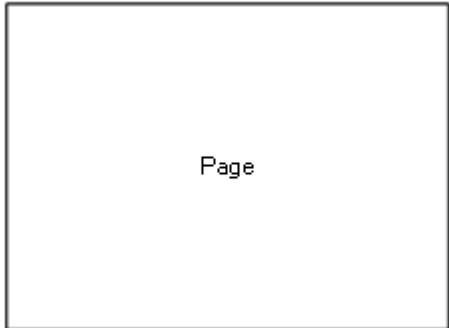
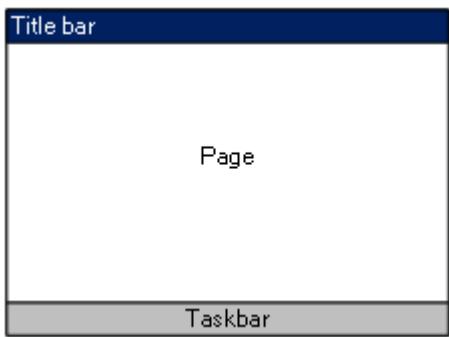
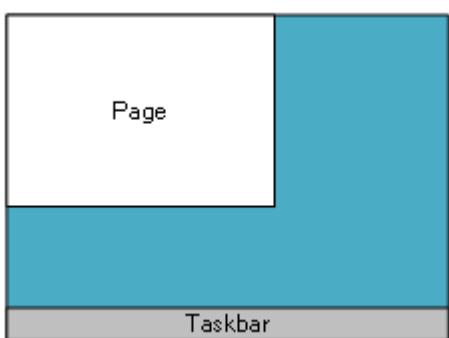
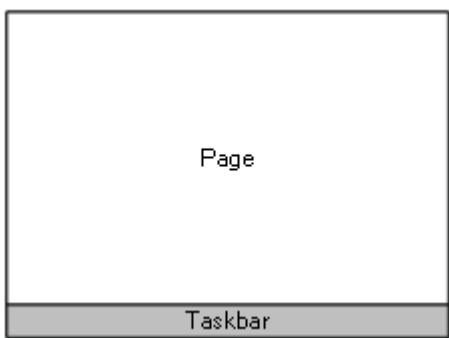
**Note:** Certain runtime window customizations may determine which template you need to use to create your project pages. For example, if your project uses the "standard" style and `[Animator]FullScreen` is set to 2 (no title bar), you will need to use the "no title bar" variation of a page template (see [Page Templates](#)).

---

## Examples

The following examples demonstrate the outcome of using different combinations of the `citect.ini` parameters listed above.

Example	Parameter Settings
	<p>The default behavior. Resizing the window maintains the aspect ratio.</p> <p>[Animator]FullScreen = 0  [Animator]MaximizedWindow = 0  [Page]DynamicSizing = 1  [Page]MaintainAspectRatio = 1</p>
	<p>The default behavior, with aspect ratio control turned off.</p> <p>[Animator]FullScreen = 0  [Animator]MaximizedWindow = 0  [Page]DynamicSizing = 1  [Page]MaintainAspectRatio = 0</p>
	<p>Dynamic sizing is turned off, which means you are not able to increase the window size and content is not scaled. Scroll bars are enabled if the window size is decreased and no longer large enough to show the entire page.</p> <p>[Animator]FullScreen = 0  [Animator]MaximizedWindow = 0  [Page]DynamicSizing = 0</p>
	<p>Fullscreen mode, with a title bar.</p> <p>[Animator]FullScreen = 2  [Page]DynamicSizing=1  [Page]MaintainAspectRatio = 0</p> <p><b>Note:</b> If dynamic sizing is turned off, the page content cannot be scaled and fullscreen mode does not work.</p>

Example	Parameter Settings
 <p>Page</p>	<p>Fullscreen mode, without a title bar.</p> <p>[Animator]FullScreen = <b>1</b>  [Page]DynamicSizing=1  [Page]MaintainAspectRatio = 0</p>
 <p>Title bar</p> <p>Page</p> <p>Taskbar</p>	<p>The window is maximized on startup. If you select the restore button on the title bar, you can adjust the size of the window.</p> <p>[Animator]FullScreen = 0  [Animator]MaximizedWindow = <b>1</b>  [Page]DynamicSizing = 1  [Page]MaintainAspectRatio = 0</p> <p><b>Note:</b> If [Page]MaintainAspectRatio is set to 1, the maximized window will maintain the aspect ratio of displayed page, not the monitor. This means the page may not cover the entire desktop.</p>
 <p>Page</p> <p>Taskbar</p>	<p>Startup mode is set to 16, which means the window is displayed with thin borders, no caption, and no maximize/minimize or system menu icons. The window cannot be re-sized.</p> <p>[Animator]FullScreen = 0  [Animator]MaximizedWindow = 0  [Page]StartupMode = <b>16</b></p>
 <p>Page</p> <p>Taskbar</p>	<p>Startup mode is set to 16, which means the window is displayed with thin borders, no caption, and no maximize/minimize or system menu icons. With dynamic sizing enabled, the window can be maximized.</p> <p>[Animator]FullScreen = 0  [Animator]MaximizedWindow = <b>1</b>  [Page]DynamicSizing = <b>1</b>  [Page]MaintainAspectRatio = 0  [Page]StartupMode = <b>16</b></p> <p><b>Note:</b> If [Page]MaintainAspectRatio is set to 1, the</p>

Example	Parameter Settings
	maximized window will maintain the aspect ratio of displayed page, not the monitor. This means the page may not cover the entire desktop.

## See Also

[Secure the Windows Title Bar](#)

## Configure Graphics Pages

This section of the help explains how to create and manage graphics pages. It includes the following topics:

- [Create a Graphics Page](#)
- [Open or Save a Graphics Page](#)
- [Locate a Graphics Page](#)
- [Edit the Properties for a Page](#)
- [Edit the Default Page Settings](#)
- [Use a Browse Sequence](#)
- [Configure a Startup Page and Splash Screen](#)
- [Secure the Window Title Bar](#)
- [Display Tags on a Page](#)
- [Create a Template](#)

## Create a Graphics Page

### To create a new page:

1. Open Graphics Builder.
2. On the **File** menu, select **New**.

Or:

Select the **New** button.



The New Dialog box is displayed. This dialog box is used to create a page, template, symbol, Genie, or Super Genie by selecting a button.

3. On the New dialog box, click **Page**. The Use Template dialog box is displayed.
4. Complete the fields in the Use Template dialog box (see below for a description).
5. Click **OK**.

If you create a new page using the Graphics Builder, you need to edit the page record to define a browse

sequence.

**Note:** You can initiate the process to create a page directly from the **Pages** view in Plant SCADA Studio's **Visualization** activity. See [Browse Pages in Plant SCADA Studio](#).

## Use Template (New Page/Template) Dialog Box

This dialog box lets you create a new page or template based on an existing template.

Element	Description
<b>Template</b>	A table of templates on which you can base the new page or template. Use the scroll bar to locate the thumbnail image of the template, then choose the template and click <b>OK</b> (or double-click the thumbnail image). <b>Note:</b> To edit the template, select it and click <b>Edit</b> .
<b>Style</b>	The style of the page. Plant SCADA templates are grouped into several styles and are available in various page resolutions (see <a href="#">Page Templates</a> ). When you create a new project, you can choose the style that best suits your requirements.
<b>Linked</b>	To maintain the link with the original template, select this check box. A page or template that is linked with its original template is automatically updated if the template is changed. <b>Note:</b> You can remove the link to the template at any time with the <b>Cut Link</b> command from the <b>Edit</b> menu, but you cannot re-link a page or template with its original template after this has occurred.
<b>Designed for showing title Bar</b>	Determines whether to display the Windows title bar (at the top of each graphics page). The title bar contains the title of the window, maximize, minimize and close buttons (at the right hand end of the title bar), and the control menu button (at the left hand end of the title bar). To display a page in fullscreen (without a title bar), the size of the page needs to be the same size as the display (or larger). If the page is smaller than the display, the title bar still displays, even if fullscreen mode is enabled. Standard templates styles are available for both page sizes. <b>Note:</b> The <b>swx_style_1</b> and <b>tab_style_1</b> templates are designed to always include a title bar. If you use these templates, you can restrict usage of the standard title bar buttons with Cicode (see <a href="#">Secure the Window Title Bar</a> ).

Element	Description
<b>Resolution</b>	<p>The screen resolution of the page or template (see <a href="#">Screen Resolution</a>):</p> <p><b>Default</b> - width x height of the screen on the computer currently in use</p> <p><b>VGA</b> - 640 x 480</p> <p><b>SVGA</b> - 800 x 600</p> <p><b>XGA</b> - 1024 x 768</p> <p><b>SXGA</b> - 1280 x 1024</p> <p><b>User</b> - User-defined width x height</p>

## Open or Save a Graphics Page

You can open and/or save a graphics page using the Open/Save As Dialog Box.

---

**Note:** This dialog box lets you open or save a page, template, symbol, Genie, or Super Genie. To select the type of entity, click the appropriate tab.

---

### To open an existing page:

1. Click **Open**, or choose **File | Open**. The Open/Save As dialog box is displayed. (See below for a description of the fields in this dialog box).
2. Choose **Type: Page**.
3. Select the **Project** where the page is stored.
4. Select the **Page**.
5. Click **OK**.

---

**Note:** To delete a page from the project, select the page name and click **Delete**.

---

### To save the current page:

1. Click **Save**, or choose **File | Save**.
2. Select the **Project** in which to store the page.
3. Click **OK**.

### To save the current page with a new name:

1. Choose **File | Save As**.
2. Select the project in which the page is stored.
3. Enter a name for the page in **Page** (64 characters maximum).
4. Click **OK**.

### To save all current pages that are open:

- Choose **File | Save All**.

**To delete a page:**

- Select the page and click **Delete**.

**Open/Save As Dialog Box**

Element	Description
<b>Page / Symbol / Template / Genie</b>	<p>The name of the graphics page, template, symbol, Genie, or Super Genie.</p> <p>If you are opening a page, template, symbol, Genie, or Super Genie, select its name from the large window.</p> <p>If you are saving a page, template, symbol, Genie, or Super Genie, type a name into the smaller input box (or select a name from the large window if you want to overwrite an existing page, template, symbol, Genie, or Super Genie).</p> <p><b>Note:</b> If you are using distributed servers, the name needs to be unique to the cluster (for example you cannot have the same page name in more than one cluster).</p>
<b>Library</b>	(For symbols, Genies, and Super Genies only.) The library in which to save the symbol, Genie, or Super Genie. To create a new library, click <b>New</b> .
<b>Style</b>	(For templates only.) The style of the template. To create a new style, click <b>New</b> .
<b>Preview Enable</b>	Displays a thumbnail image of the page, template, symbol, Genie, or Super Genie.
<b>Title bar</b>	(For templates only.) Specifies whether to include a space for the title bar. If you use a title bar, you will have slightly less display space on screen.
<b>Resolution</b>	<p>(For templates only.) The screen resolution of the template:</p> <p><b>Default</b> - width x height of the screen on the computer currently in use</p> <p><b>VGA</b> - 640 x 480</p> <p><b>SVGA</b> - 800 x 600</p> <p><b>XGA</b> - 1024 x 768</p> <p><b>SXGA</b> - 1280 x 1024</p> <p><b>User</b> - User-defined width x height</p>

## See Also

[Browse Pages in Plant SCADA Studio](#)

## Locate a Graphics Page

**To locate a graphics page:**

1. Open Graphics Builder.
2. Choose **File | Find**.
3. Select a project from the **Project Name** list.
4. Browse through the pages in the **Pages** list.
5. Click **OK** to view the page.

**Note:** You can use the **Pages** view in Plant SCADA Studio's **Visualization** activity to browse a project's pages.  
See [Browse Pages in Plant SCADA Studio](#).

---

## See Also

[Using Find and Replace](#)

## Edit the Properties for a Page

You can define properties for your graphics pages.

**To define the properties for a page:**

1. Open a page in the Graphics Builder.
2. On the **File** menu, select **Properties**.
3. Use the Page Properties dialog box to define the properties for the graphics page.

**Note:** You can also use the Property Grid to view and edit some of the properties for any selected pages in Plant SCADA Studio's Visualization activity (see [View and Edit Page Properties in Plant SCADA Studio](#)).

---

## See Also

[Page Properties - General](#)

[Page Properties - Appearance](#)

[Page Properties - Keyboard Commands](#)

[Page Properties - Events](#)

[Page Properties - Environment](#)

[Page Properties - Associations](#)

[Edit the Default Page Settings](#)

## Page Properties - General

Graphics pages have the following general properties:

Field	Description
<b>Window title</b>	<p>The title to be displayed on the page at runtime. A window title can have a maximum size of 64 characters.</p> <p>The title can be plain text, Super Genie syntax or both (for example, "Pump ?AREA? status").</p>
<b>Description</b>	<p>Enter a description of the page, and its various functions and so on up to a maximum of 250 characters. The description is designed for comments only and does not affect how your system performs, nor does the description appear during runtime.</p>
<b>Previous page</b>	<p>The page that will precede the current page in the runtime browse sequence (maximum 64 characters). Choose an existing page from the menu or type in a page name.</p> <p>This property is optional. If not specified, the Page Previous command is inoperative while this page is displayed.</p>
<b>Next page</b>	<p>The page that will follow the current page in the runtime browse sequence (maximum 64 characters). Choose an existing page from the menu or type in a page name.</p> <p>This property is optional. If not specified, the Page Next command is inoperative while this page is displayed.</p>
<b>[Security] All areas</b>	<p>Select the check box if you want the page to belong to every area (the page can be seen by any operator with view access to at least one area).</p>
<b>[Security] Area</b>	<p>Enter the area to which this page belongs. Only users with access to this area can view this page. Click the menu to select an area, or type in an area number directly. If an area is not specified, the page is accessible to every user.</p>
<b>[Page scan time] Default</b>	<p>The Page scan time defines how often this graphics page is updated at runtime. The Page scan time also determines the rate of execution of the While page shown events (i.e. the command(s) which are executed while the page is displayed at runtime).</p>

Field	Description
	<p>Select this check box to use the default page scan time (as set using the <a href="#">[Page]ScanTime</a> parameter); otherwise, leave it blank, and enter (or select) another value in the field below. For example, if you enter a page scan time of 200 milliseconds, there will be an attempt to update the page every 200 milliseconds, and any While page shown events are executed every 200 milliseconds.</p> <p><b>Note:</b> You can set the default page scan time using the Setup Wizard.</p>
<b>[Logging] Log device</b>	<p>This is the device to which messages are logged for the page's keyboard commands. Click the menu to the right of the field to select a device, or type a device name.</p> <p><b>Note:</b> You need to include the <i>MsgLog</i> field in the format of the log device for the message to be sent.</p>
<b>[Cluster context] Inherit from caller</b>	<p>Select this check box to make the current page inherit the default cluster context from the page or Cicode task that opened this page (for example using the <i>PageDisplay</i> or <i>WinNewAt</i> Cicode functions). If this is checked you cannot select a specific default cluster. See <i>Cluster Context</i> in the topic <a href="#">Clusters</a> for more information.</p>
<b>[Cluster context] Cluster</b>	<p>Specifies a default cluster context for this page.</p>

Click **Apply**, and then click **OK**. To define further properties for the page, select the relevant tabs.

## See Also

[Edit the Properties for a Page](#)

### Page Properties - Appearance

You define the appearance of your graphics pages by using the controls on the **Appearance** tab.

Graphics pages have the following appearance properties:

Field	Description
<b>[Template] Style</b>	<p>The style (appearance) of the graphics page in the runtime system. You can set a default style (to be applied to new pages) using the page defaults of existing pages and templates using the Page Properties. For a description of the available styles,</p>

Field	Description
	see the topic <a href="#">Page Templates</a> .
<b>[Template] Resolution</b>	<p>The default screen resolution of the pages:</p> <ul style="list-style-type: none"> <li>• VGA — 640 x 480 pixels (4:3)</li> <li>• SVGA — 800 x 600 pixels (4:3)</li> <li>• XGA — 1024 x 768 pixels (4:3)</li> <li>• SXGA — 1280 x 1024 pixels (5:4)</li> <li>• HD768 — 1366 x 768 pixels (16:9)</li> <li>• HD1080 — 1920 x 1080 pixels (16:9)</li> <li>• WUXGA — 1920 x 1200 pixels (16:10)</li> <li>• UHD4K — 3840 x 2160 pixels (16:9)</li> <li>• User Defined.</li> </ul>
<b>[Template] Name</b>	<p>The name of the template on which the page is based. Choose a template name from the menu.</p> <p><b>Note:</b> If you are looking for a template that you created yourself, check that you entered the correct <b>Style</b> and <b>Resolution</b> above.</p>
<b>[Template] Show title bar</b>	<p>Determines whether the Windows title bar displays (at the top of the page). The title bar contains the title of the window, maximize, minimize and close buttons (at the right hand end of the title bar), and the control menu button (at the left hand end of the title bar).</p> <p>To display a page in full screen (without a title bar), the size of the page needs to be the same size as the display (or larger). If the page is smaller than the display, the title bar still displays, even if full screen mode is enabled. Standard templates styles are available for both page sizes.</p>
<b>[View area] Width</b>	<p>The width (in pixels) of the area that the operator can view at runtime. Click the up and down arrows to increase and decrease the width, or type in another value directly.</p>
<b>[View area] Height</b>	<p>The height (in pixels) of the area that the operator can view at runtime. Click the up and down arrows to increase and decrease the height, or type in another value directly.</p>
<b>Page color</b>	<p>The color that will display in the background of the graphics page.</p>

Field	Description
Ignore Quality	<p>This field allows you to override the global [Page]IgnoreValueQuality parameter setting in the Citect.ini file for a particular page.</p> <p>The options are:</p> <ul style="list-style-type: none"><li>• Default Indication - use the current setting for the [Page]IgnorevalueQuality parameter setting in the Citect.ini file.</li><li>• #COM Indication - change the setting to the equivalent of 0 for the [Page]IgnorevalueQuality parameter setting in the Citect.ini file.</li><li>• No Indication - change the setting to the equivalent of 1 for the [Page]IgnorevalueQuality parameter setting in the Citect.ini file.</li><li>• Background Indication - change the setting to the equivalent of 2 for the [Page]IgnorevalueQuality parameter setting in the Citect.ini file.</li></ul>

The preview field to the right of this dialog displays a picture of the selected template. Click **Apply**, then click **OK**. To define further properties for the page, click the relevant tabs.

## See Also

[Edit the Properties for a Page](#)

### Page Properties - Keyboard Commands

The Keyboard Commands property lets you define keyboard commands for the page. A keyboard command is a particular key sequence which executes a command when it is typed in by the operator at runtime.

You can also define a message which will log every time the key sequence is entered.

If an operator does not satisfy the Access requirements specified under **Security** (see below), they will not be able to enter keyboard commands for this page at runtime.

Keyboard commands have the following properties:

Field	Description
<b>Key sequence</b>	<p>Enter the key sequences that the operator can enter to execute a command (32 characters or less). You can enter as many key sequences as you like. To add a key sequence, click <b>Add</b>, and type in the sequence or select one from the menu. To edit an existing sequence, click the relevant line, and click <b>Edit</b>. You can also remove key sequences by clicking <b>Delete</b>. See <a href="#">Define Key Sequences for Commands</a>.</p>
<b>Key sequence command</b>	<p>The commands (set of instructions) to be executed immediately when the selected key sequence is entered. The key sequence command can have a maximum length of 254 characters.</p> <p>To insert a tag or a function, click the <b>Wizard</b> button to the right of this field. This button displays two options: <b>Insert Tag</b>, and <b>Insert Function</b>.</p>
<b>[Security] Same area as page</b>	<p>Select this check box to assign the keyboard command to the same area as the page (see <a href="#">Page Properties - General</a>). Only users with access to this area (and any necessary privileges) can issue this command or log the message. If you want to assign this keyboard command to another area, leave this check box clear and enter another area below.</p>
<b>[Security] Command area</b>	<p>Enter the area to which this keyboard command belongs up to a maximum of 16 characters. Only users with access to this area (and any necessary privileges) can issue this command or log the message. For example, if you enter Area 1 here, operators need to have access to Area 1 to issue this command.</p> <p>Click an area from the menu or type in an area number.</p>
<b>[Security] No privilege restrictions</b>	<p>Tick this box to disable privilege restrictions; otherwise, leave it blank, and enter another privilege below.</p> <p>The result of not assigning a privilege restriction differ according to whether you have used areas in your security setup:</p> <ul style="list-style-type: none"> <li>• <b>No Areas:</b> Every operator has full control of the page.</li> <li>• <b>Areas:</b> An operator will only need view access to the area assigned to this page to have full control</li> </ul>

Field	Description
	over the page.
<b>[Security] Privilege level</b>	<p>Enter the privilege level that a user needs to possess to issue this command or log the message. For example, if you enter Privilege Level 1 here, operators need to possess Privilege Level 1 to issue this command. You can also combine this restriction with area restrictions (see above). For example, if you assign the keyboard command to Area 5, with Privilege Level 2, the user needs to be set up with Privilege 2 for Area 5.</p> <p>Choose a privilege from the menu or type in an area number.</p>
<b>[Logging] Log Message</b>	<p>A text message sent to the <i>MsgLog</i> field of the Log Device when the selected action is performed by the operator at runtime (32 characters maximum). The message can be plain text, Super Genie syntax, or a combination of the two.</p> <p>If you want to include field data as part of a logged message, you need to insert the field name as part of the device format when you configure the device. For instance, in the <b>Format</b> field of the <b>Devices</b> form, you could enter {MsgLog,20} {FullName,15}. This would accommodate the logging of messages such as <b>P2 started by John Smith</b>.</p> <p>The log device to which the message is sent is specified through the General page properties.</p>

Click **Apply**, and then click **OK**. Click **Clear Property** to clear property details, and disable the property. To define further properties for the page, click the relevant tabs.

## See Also

[Edit the Properties for a Page](#)

### Page Properties - Events

You use the **Events** tab to assign commands to your graphics pages. These commands can be executed when the page is opened, closed, or whenever the page is open. You can also define different messages to log at the same time.

Page events have the following properties:

Field	Description
<b>Event</b>	<p>There are three events to which commands can be attached. You can select more than one type of event. Unique commands can be attached to each (i.e., you can perform one task when the page is opened, and another when it is closed).</p>
<b>[Event] On page entry</b>	<p>Select this option if you want a command to be executed when the page is initializing. For example, you could execute a command to extract recipe data from a database into a Cicode variable, to be displayed on the page.</p> <p>The following functions should not be called: Login() or UserLogin().</p> <p><b>Note:</b> Use the "On Page Shown" event if your command uses ActiveX controls. The "On page entry" event may not properly initialize ActiveX controls. The resulting value from a tag write that is performed using the "On page entry" event may not be displayed and accessed correctly if the tag is used directly within the page. The "On page shown" event can be used for initializing tags that are to be accessed on the page.</p>
<b>[Event] On page exit</b>	<p>Select this option if you want a command to be executed when the operator exits the page. For example, this command could be used to close a database that was opened at page entry.</p> <p>The following functions should not be called: PageGoto(), PageNext(), PagePrev(), PageDisplay(), or PageLast().</p> <p><b>Note:</b> If you shut down the system, "On page exit" functions for the currently open pages will not execute. If a particular page exit code needs to run, call the code before calling the Shutdown() function.</p>
<b>[Event] While page is shown</b>	<p>Select this option if you want a command to execute continually for the entire time that the page is open. For example, the <b>While page is shown</b> command could be used to perform background calculations for this page.</p>

Field	Description
<b>[Event] On page shown</b>	<p>Select this option if you want a command to execute when a page is first displayed and objects on it (including ActiveX controls) have been initialized. If you want to reference ActiveX controls in your command then use this event instead of "On page entry".</p> <p><b>Note:</b></p> <ol style="list-style-type: none"> <li>1. Use the "On Page Shown" event if your command uses ActiveX controls. The "On page entry" event may occur before the ActiveX controls have been initialized.</li> <li>2. Use the "On Page Shown" event if your command requires access to the latest variable tag value. The "On page entry" event may occur before the tag has been resolved.</li> </ol> <p>See <a href="#">PageInfo</a> Cicode function, type 25.</p>
<b>[Event] On activate</b>	<p>Select this option if you want the command to execute when the window is activated at runtime. The automation interface allows you to set or get commands for this event using <a href="#">PageEventPutEx</a> and <a href="#">PageEventGetEx</a>.</p>
<b>[Event] On deactivate</b>	<p>Select this option if you want the command to execute when the window is deactivated. This command will close the window. The automation interface allows you to set or get commands for this event using <a href="#">PageEventPutEx</a> and <a href="#">PageEventGetEx</a>.</p>
<b>Command</b>	<p>The commands (set of instructions) to be executed immediately when the selected Event occurs.</p> <p>To insert a tag or a function, click the <b>Wizard</b> button to the right of this field. This button displays two options; <b>Insert Tag</b>, and <b>Insert Function</b>.</p>

Click **Apply**, and then click **OK**. Click **Clear Property** to clear property details, and disable the property. To define further properties for the page, click the relevant tabs.

## See Also

[Edit the Properties for a Page](#)

### Page Properties - Environment

You use the **Environment** tab to define environment information for your graphics pages. You can add, remove, or edit page environment variables for a graphics page, template, or Super Genie.

For example, you can design your loop pages to expand when clicked (a **Tune >>** button) to show tuning parameters. You can use the environment variable to define the size of the expanded window. The **DspGetEnv()** function would be used as part of the Cicode for the button that expands the window. This means that the Cicode used to expand the window can be generic; you can use the same Cicode each time.

Environment variables are propagated from Super Genie templates to Super Genie pages automatically.

### Variables

The environment variables to be added to the page. When the page is opened during runtime, these variables are created. Their values can then be read using the **DspGetEnv()** function. When the page is closed, the environment variable memory is freed (discarded). For the example above, you would add one variable to define the width of the page, and another to define the height.

**Note:** When using Cicode, be aware that page variables are case sensitive.

To add an environment variable, click **Add**. To edit an existing environment variable, select it, and click **Edit**. (If you click **Add** or **Edit**, a small dialog appears, containing two fields, one for the property and one for its value.) To remove an environment variable, select it, and click **Remove**.

## See Also

[Edit the Properties for a Page](#)

### Page Properties - Associations

The Page Property association Name Tab is used to define default Associations. The name you use when defining a Super Genie association is entered in the Name Field.

Use this tab to reference or edit existing associations. In using this tab you can add those associations that are currently in use in a Super Genie association and define the appropriate default value and value on error fields. You can also add a description.

Double-click in the row to make it editable, or Click **Add**.

Field	Description
Name	<p><b>Note:</b> On upgrading from an earlier version of Plant SCADA existing numbered substitutions are taken from defined Super Genie pages and automatically listed in the drop down box.</p> <p>For new associations enter the name of the association directly in the field [Name] provided: or select the name of the association from the drop-down box. Only those association names that are 'in-use' as part of a Super Genie association are listed. For example, if the association in the expression field of a button was '?Speed?' then the association name is Speed.</p> <p>Name Associations:</p> <ul style="list-style-type: none"><li>• To start with an alphanumeric or "_" character</li></ul>

Field	Description
	<ul style="list-style-type: none"> <li>• To contain only those characters that are allowed in variable tag names (excluding the period character '.'), and not contain any white space.</li> <li>• Be no more than 253 characters in length</li> </ul>
<b>Default (optional)</b>	Enter the default value to be used if the page association has not been performed using one of the Association Cicode functions at runtime. The default value needs to be either a literal string constant enclosed in single quotes (e.g. <'a literal value'> or <'1.23'>) or a valid full or partial tag name, equipment.item reference.
<b>Value on Error (optional):</b>	Enter the text to be used if no default value is defined and the association is not performed, or if the defined association did not resolve to a valid tag name. Since the value on error cannot be a variable tag, it is not necessary to enclose the text in quotation marks.  If the association on the page is specified in a typed format, the value will be converted to the specified type. If the value cannot be converted a hardware alarm will be raised. For example, if the association is specified as ?INT AlmDesc? and the value on error is set to <LowLevel>, the animation will display zero (0) and a hardware alarm will be raised. Typeless substitutions such as ?AlmDesc? will display the value on error as specified.
<b>Description (optional):</b>	Enter a description of the page association. The description is useful when maintaining associations.
<b>In Use:</b>	This column can not be edited and simply indicates if the association is in use on the current page in a Super Genie.

When finished click **Add**, the association is then added to the table.

To edit an association select an item and click **Edit** or double-click on the row to make it editable. You can not edit the name of an existing association that is in use. However you can select a new name in an existing row, and when this name is <ENTERED>, a new row of information will be displayed and the existing row of data will disappear.

To delete an association select the row in the table and click **Delete**. You can delete associations in use or not in use. If you delete the object from the page, the status of the association used in the Super Genie substitution will be updated to not in-use.

Click **Ok** or **Apply** to save the association table or click **Cancel** to discard any changes you may have made.

## See Also

- [Edit the Properties for a Page](#)
- [Passing Animation Point Metadata as Super Genie Associations](#)
- [Define Dynamic Associations](#)
- [Use Constants with Dynamic Associations](#)

## Edit the Default Page Settings

You can edit the default properties that all new graphics pages will use.

**To set the properties to be used for new graphics pages:**

1. Choose **File | Defaults**.
2. Complete the **Page Defaults** dialog box (see below for a description of the fields), then click **OK**.

## Page Defaults Dialog Box

Field	Description
<b>Template Resolution</b>	<p>Default screen resolution of the standard graphics pages (for example, alarms pages and standard trend pages):</p> <p><b>Default</b> - width x height of the screen on the computer currently in use</p> <p><b>VGA</b> - 640 x 480</p> <p><b>SVGA</b> - 800 x 600</p> <p><b>XGA</b> - 1024 x 768</p> <p><b>SXGA</b> - 1280 x 1024</p> <p><b>User</b> - User-defined width x height</p>
<b>Template Style</b>	The style (appearance) of the graphics pages in the runtime system. The style you select is the default style for new pages you add to the project. You can change the style of existing pages and templates using the Page Properties. For a description of the available styles, see the topic <a href="#">Page Templates</a> .
<b>[Template] Show title bar</b>	Determines whether the Windows title bar displays at the top of each graphics page. The title bar contains the title of the window, maximize, minimize and close buttons (at the right hand end of the title bar), and the control menu button (at the left hand end of the title bar).  To display a page in full screen (without a title bar),

Field	Description
	the size of the page needs to be the same size as the display (or larger). If the page is smaller than the display, the title bar still displays, even if full screen mode is enabled. Standard templates styles are available for both page sizes  You can override this default for your own pages at the time you create them, or later.
Background color	The color that will display in the background of new graphics pages.
Preview	This dialog also displays a preview of your page with the defaults applied.

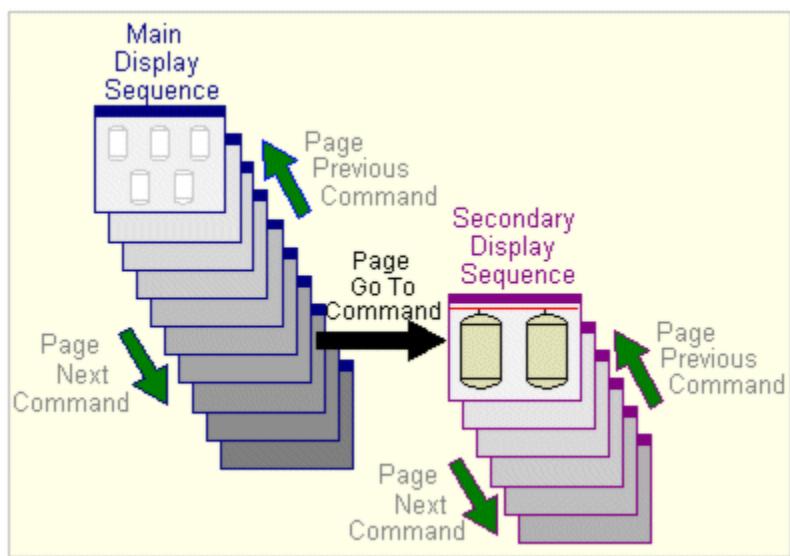
## Use a Browse Sequence

You can link related pages together with a browse sequence. A browse sequence creates a linear navigation sequence for the pages in your system.

When you define a graphics page, you can specify where in the browse sequence the page displays. Within a browse sequence, an operator can display a preceding or following page by choosing **Page Previous** and **Page Next** commands (or a similar set of buttons defined on each page using the PageNext and PagePrev Cicode functions).

When you save a page for the first time it is automatically added to the browse sequence.

You can also use multiple browse sequences by defining a Page GoTo command that displays a page in another (secondary) sequence. The Page Next and Page Previous commands then display the next and previous pages in the secondary sequence, as in the following diagram:



A display sequence is optional. You can define several Page GoTo commands that display specific pages in an hierarchical structure.

## See Also

[Configure a Startup Page and Splash Screen](#)

# Configure a Startup Page and Splash Screen

Every system needs to have a startup page, and some may use a splash screen.

- **Splash screen**

A splash screen is the very first window that is displayed when Plant SCADA starts. It is defined by the parameter [\[Page\]Splash](#).

The splash window has a thin border, is non-movable and is often set to "always on top". It can be used to display project information while the system is initializing, such as a company logo, software version, project version, and so on. By default, it is only displayed for a brief period of time and then automatically dismissed.

- **Startup page**

The startup page is the page that is displayed when Plant SCADA runtime has initialized and become operational. It is defined by the parameter [\[Page\]Startup](#).

The startup page can be displayed before or after the splash window is dismissed. The timing is controlled by the following set of parameters:

- [\[Page\]SplashTimeout](#) - determines how long the splash window will be displayed.
- [\[Page\]StartupDelay](#) - determines how long the startup page will wait before it is displayed.

If the StartupDelay is set to less than the SplashTimeout, the startup page will be displayed while the splash screen is still shown. On the other hand, setting the StartupDelay to be greater than the SplashTimeout will display the startup page after the splash screen is dismissed.

The Example project makes use of the splash screen parameters to specify a page that is displayed as splash screen, and Cicode functions to display product and project information on the splash screen.

The following parameters are used to set up the splash screen in the Example project:

```
[Page]
Splash = !Splash
SplashWinName = Splash
SplashTimeout = 5000
Startup = Menu
StartupWinName = Main
StartupDelay = 5100
HomePage = Menu
```

## See Also

[Secure the Window Title Bar](#)

## Secure the Window Title Bar

You can now specify confirmation actions for standard title bar buttons (minimize, maximize, restore, and close) to override the standard behavior for graphic page windows by writing a Cicode function that defines the alternative action, and specifying that Cicode function using the appropriate [OnEvent](#) function event code. The Cicode function needs to return a value.

If your Cicode function returns a zero value, then the Windows message will be passed on to the default message handler so that the window will continue to execute the default behavior.

If your Cicode function returns non-zero, then the Windows message will not be passed on to the default message handler.

In addition, when you call Cicode function [Shutdown](#), you can decide whether to bypass the confirmation function or not.

## See Also

[Event Functions](#)

[Window Functions](#)

## Display Tags on a Page

You can apply visual cues to a tag displayed on a graphics page, providing greater emphasis to the data presented. For example, you could highlight a bad quality tag value by presenting it in a different color or style.

This type of activity is enabled using the [\[Page\]IgnoreValueQuality](#) parameter, which can be used to display a graphical representation of data quality.

[\[Page\]IgnoreValueQuality](#) can be used in tandem with a number of [\[Page\]](#) parameters to provide different visual outcomes based on the state of a tag. For example, you could change the text background color for a tag according to its current state using the following parameters:

- [\[Page\]BadTextBackgroundColor](#)
- [\[Page\]ErrorTextBackgroundColor](#)
- [\[Page\]UncertainTextBackgroundColor](#)
- [\[Page\]OverrideTextBackgroundColor](#)
- [\[Page\]ControlInhibitTextBackgroundColor](#)

You can also use the parameter [\[Page\]EnableQualityToolTip](#) to enable tool tips that present quality and timestamp data when the cursor is held over a tag.

The Tag Extension Parameters page in the Plant SCADA Example Project provides an interactive example of how these parameters can be implemented.

## See Also

[Tag Extensions](#)

## Create a Template

If you need to generate multiple graphics pages that share a common set of objects, you can create your own page template (from any existing template). You can then use this template to create the required pages with the shared objects, and add individual objects to each.

If you want to delete or change the location of a common object, or to add a new common object, you will not have to change each page; instead, you can change the template. Pages based on that template are automatically updated.

**Note:** When you create a template, save it in the project directory. It is then backed up when you back up the project. Don't modify the supplied standard templates. When you edit a template, you need to use one of the [Update Pages](#) commands (from the **Tools** menu) to update each page based on the template. The properties of the template are not updated automatically.

### Create a Template:

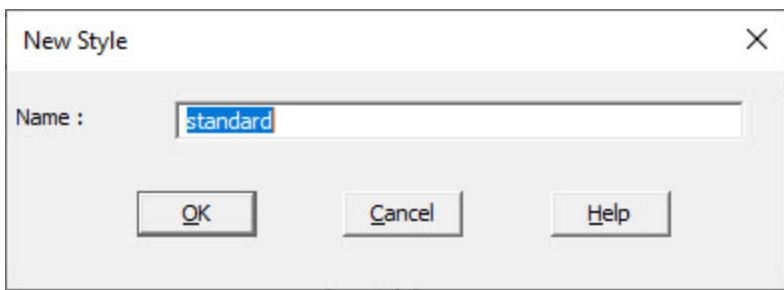
1. Open Graphics Builder.
2. On the **File** menu, select **New**.  
Or:  
Select the **New** button.  

3. On the New dialog box, click **Template**.
4. On the Use Template dialog box, choose a **Template** upon which to base your template (see [Page Templates](#)).
5. Choose a **Style** for the template.
6. If required, select or clear the **Linked** check box (see [Link a Page to a Template](#)).
7. If required, select or clear the **Designed for showing title bar** check box.
8. Choose the **Resolution** for the template (see [Screen Resolution](#)).
9. Click **OK**.

### Save the Current Template:

1. On the **File** menu, select **Save**.  
Or:  
Select the **Save** button.
2. On the Save As dialog box, select the **Project** in which to store the template, and its **Resolution**.
3. Click **OK**.

**Note:** To create a new style for the template, click **New**. You create a new template style using the New Style dialog box. Enter a name for your new style in the **Name** field.



## Link a Page to a Template

When using a template to create a new page, a link can be kept to the template. A page (or template) that is linked with its original template is automatically updated if the template is changed.

When a page is linked to a template, the objects that form the template cannot be accessed from the page by the usual double-click. To display the properties of these objects, hold the **Control** (CTRL) key down and double-click the object you want to view properties for. Alternatively, choose **Tools | Goto Object**, select the group, and click **OK**. However, most of these properties are read-only.

---

**Note:** You can remove the link to the template at any time using **Edit | Cut Link**, but you cannot re-link a page or template with its original template after this has occurred.

---

## See Also

[Create a Template](#)

## The Bitmap Editor

A bitmap image is a drawing object represented as an array of pixels (or dots), rather than as individual entities. A bitmap is treated as a single object that you can move, copy, and reshape. Because you can edit individual pixels in a bitmap, you can use bitmaps for vignettes and image blending. In a runtime system, bitmaps are displayed differently to other objects. Bitmaps are mapped directly to the screen (that is, each pixel in the image corresponds to a pixel on the screen). Objects are stored as a series of instructions, and are drawn on the screen in the same order as they were drawn on the graphics page.

You can create bitmaps with the Bitmap Editor, or import bitmaps from any other Windows-based bitmap editor. You can use bitmap images on your graphics pages, and as animated symbols. The background color in the Bitmap Editor is always transparent, indicated as a white pixel with a black dot at the center. To draw with the background (transparent) color, click (and hold) the right mouse button.

Flashing colors are represented by a diagonally split pixel, indicating the on-state and off-state colors used.

### To open the Bitmap Editor:

1. Select **Tools | Bitmap Editor**.

### Bitmap Editor Toolbar Buttons

Button	Description
	Exits the Bitmap Editor and saves editing changes.
	Exits the Bitmap Editor and discards changes.
	Zooms in on the image.
	Zooms out on the image.
	Selects a color from the image to set as the current color (keyboard shortcut is <b>Shift+P</b> ). You can also select the current color from the color swatch.
	Displays the <b>Bitmap Size</b> dialog box where you can view the image's current dimensions and edit the image's edge.

**To re-size a bitmap:**

1. In Graphics Builder, click the bitmap.
2. Choose **Tools | Bitmap Editor**, or press **F9**.
3. Click **Resize**. The Bitmap Size dialog box appears.
4. Select a mode. Click **Grow** to enlarge the image, or **Shrink** to reduce it.
5. For each side of the bitmap, specify how many pixels you want to add or remove, then click **OK**.

**To set a color from a bitmap as the current color:**

1. In Graphics Builder, click a bitmap.
2. Choose **Tools | Bitmap Editor**, or press **F9**.
3. Click **Eye Dropper**, and then click a color in the image. The selected color becomes the current color, and is used when you click elsewhere on the image.

**To convert an object (or objects) to a bitmap:**

1. Select the object(s).
2. Choose **Tools | Convert to Bitmap**.

---

**Note:** The Convert to Bitmap operation is only supported in 8-bit (256) color mode.

---

**To paste a bitmap (from another application):**

1. Create the image in an external application.
2. Use the external application's copy command to copy the image to your computer's clipboard.
3. Switch to the graphics builder.

4. Choose **Edit | Paste**.

You can edit pasted bitmaps by selecting the object and then choosing **Edit | Properties**.

**To import a graphic:**

1. Choose **File | Import**. The Import dialog box appears.
2. Select the file you want to import by using the Import dialog box.
3. Click **OK** (or click the file that you want to import and drag it onto a page in Graphics Builder).

You can edit imported bitmaps by selecting the object and then choosing **Edit | Properties**.

**To import a flashing graphic:**

1. Choose **File | Import as Flashing**. The Primary Import dialog box appears.
2. Select the first file you want to use for your flashing image.
3. Click **OK**. The Flashing Import dialog box appears.
4. Select the second file you want to use for your flashing image.

## Update Pages

If you have modified a template, library object, symbol, Genie or Super Genie, you can update all instances where the template, library object, symbol, Genie or Super Genie is in use in your project.

---

**Note:** Before updating your pages it is recommended you back up your project. If your project contains a page using an ActiveX control , and you update pages for that project on a machine that does not have the ActiveX control installed, the page may become corrupted after the update is completed.

---

To update pages in Graphics Builder, go to the **Tools** menu and select **Update Pages in Active and Included Projects**.

## See Also

[Pack Libraries](#)

## Pack Libraries

When you delete library objects (symbols, Genies, Super Genies and page templates), they are marked for deletion and can not be used. However, they will remain in the associated database file. Packing a library deletes all items marked for deletion from the various database files and re-indexes these files.

It is recommended to run the pack operation at regular intervals.

To pack libraries in the Graphics Builder go to the **Tools** menu. You can choose to:

- **Pack Libraries in Active Pages** - this option only deletes items from the active project.
- **Pack Libraries in Active and Include Projects** - this option deletes items from the selected project and its include projects.

## See Also

[Libraries](#)

# Graphics Objects

Graphics objects are the components you add to a graphics page to create a meaningful representation of the process that the page will monitor. Objects are added to a page via Graphics Builder's drawing toolbox (see [Display the Drawing Toolbox](#)). Each button on the drawing toolbox allows you to add a particular graphic object type to the page that is currently displayed (see [Graphics Object Types](#)). Objects can be moved, reshaped, or copied after they have been added to a page.

Objects are defined by a set of properties that control how the object behaves at runtime. These properties can be used to assign keyboard commands and access rights to an object, and can be configured in such a way that the object will dynamically change at runtime when an expression returns a certain value or a variable tag changes state.

Each object has:

- Properties that relate to the particular object type
- Properties that are common to all objects.

Common object properties can be used to manipulate the following aspects of an object:

- 3D effects
- Visibility
- Movement
- Scaling
- Fill
- Input commands
- Slider control
- Access
- Metadata.

For example, you can associate a tool tip with any object. This is achieved via the object's [General Access to Objects](#) properties. For more information, see [Edit Common Object Properties](#).

---

**Note:** Some objects may not support every common property set. For example, buttons will not support slider behavior.

---

The Object Properties dialog box appears when an object is initially drawn. You can also access it at any time by double-clicking an object.

---

**Note:** Object properties will override any conflicting Cicode display functions.

## See Also

[Object Groups](#)

## Graphics Object Types

You can add the following graphics object types to a graphics page using the [Display the Drawing Toolbox](#).

Drawing Tool Icon	Description
	<b>Free Hand Line</b> — a hand-drawn line (see <a href="#">Add a Free Hand Line</a> ).
	<b>Straight Line</b> — a straight line (see <a href="#">Add a Straight Line</a> ).
	<b>Rectangle</b> — a square or rectangle (see <a href="#">Add a Rectangle</a> ).
	<b>Ellipse</b> — a circle or ellipse (see <a href="#">Add an Ellipse</a> ).
	<b>Polygon</b> — a multi-sided shape (see <a href="#">Add a Polygon</a> ).
	<b>Pipe</b> — a multi-segment line that feature shading to create a tubular appearance (see <a href="#">Add a Pipe</a> ).
	<b>Text</b> — displays text within a specified area (see <a href="#">Add Text</a> ).
	<b>Number</b> — displays numerical values within a specified area (see <a href="#">Add Numbers</a> ).
	<b>Button</b> — a simple control switch with 'on' and 'off' states (see <a href="#">Add a Button</a> ).
	<b>Symbol Set</b> — a collection of symbols that can be used to create an animation (see <a href="#">Add a Symbol Set</a> ).
	<b>Trend</b> — displays up to eight trends within a specified area (see <a href="#">Add a Trend</a> ).
	<b>Cicode</b> — adds a Cicode object (see <a href="#">Add a Cicode Object</a> ).
	<b>Web Content</b> — adds an object on a graphics page that allows you to view web-based content at runtime

Drawing Tool Icon	Description
	(see <a href="#">Add a Web Content Object</a> ).
	<b>Pasted Symbol</b> — inserts a <a href="#">Symbols</a> on a page. Pasted symbols have different appearance properties to those of normal objects as you can only specify a visibility property (see <a href="#">Paste a Symbol</a> ).
	<b>Pasted Genie</b> — inserts a <a href="#">Genies</a> onto the graphics page (see <a href="#">Paste a Genie</a> ).
	<b>Composite Genie</b> — inserts a <a href="#">Composite Genies</a> onto the graphics page (see <a href="#">Insert a Composite Genie</a> ).
	<b>Process Analyst</b> — displays an instance of Process Analyst within a specified area (see <a href="#">Process Analyst</a> ).
	<b>Database Exchange Control</b> — allows you to connect to a data source (see <a href="#">Add a Database Exchange Control</a> ).
	<b>Scheduler</b> — inserts the <a href="#">Scheduler</a> control onto a graphics page (see <a href="#">Add the Scheduler ActiveX Object to a Page</a> ).
	<b>ActiveX</b> — an ActiveX object (see <a href="#">Add an ActiveX Object</a> ).

## See Also

[Graphics Objects](#)

[Object Groups](#)

## Object Groups

You can group multiple objects together to create an object group. An object group can even include a mix of objects and other object groups.

An object group has a set of properties (similar to an individual object) which determine the runtime behavior of the group as a whole (for example, when an expression returns a certain value or a variable tag changes state). The properties of the individual objects in the group remain unchanged.

To edit or view the properties of a group, double-click it. If there are several groups on your page, you can use the **Go to Object** tool (accessible via the **Tools** menu). This allows you to see which groups and objects are on your page, making it easier for you to select the object you want to edit. It also allows you to display the properties of the objects (or groups) that make up the group (see [Locate an Object](#)).

You can also edit the properties of an object in the group by holding down the **Control** (CTRL) key and double-clicking the object.

## See Also

[Group Objects](#)

## Configure Graphics Objects

Graphics objects are added to a page via Graphics Builder's drawing tool (see [Display the Drawing Toolbox](#)).

You can use this tool to perform the following tasks in Graphics Builder:

- [Add a Free Hand Line](#)
- [Add a Straight Line](#)
- [Add a Rectangle](#)
- [Add an Ellipse](#)
- [Add a Polygon](#)
- [Add a Cicode Object](#)
- [Paste a Symbol](#)
- [Paste a Genie](#)
- [Add a Web Content Object](#)
- [Add an ActiveX Object](#)
- [Add a Database Exchange Control](#)

Each object type also has a set of properties that are common to all objects. For more information, see [Edit Common Object Properties](#).

## See Also

[Graphics Object Types](#)

## Add a Free Hand Line

The Free Hand Line tool allows you to draw lines. Lines can be moved, re-sized, reshaped, brought to the front and so on, and their properties edited just like other types of object.

**To draw a free hand line:**

1. Click the Free Hand tool.  

2. Move the cursor to where you want the line to start.
3. Click and drag the cursor to draw the line.

When you release the mouse button, the object properties dialog for the line is displayed.

## Freehand Line Properties - Appearance (General)

Free Hand Line drawings have the following general appearance properties.

Property	Description
[Line] Width	The width of the line (in pixels). You can change the width by clicking the up and down arrows to the right of the field, or by entering another value in this field. If you make the line more than 1 pixel wide, it needs to be solid.
[Line] Style	The style of the line. You can choose one of the following line styles: <ul style="list-style-type: none"> <li> - Solid</li> <li> - Dash</li> <li> - Dot</li> <li> - Dash Dot</li> <li> - Dash Dot Dot</li> </ul> To change the style, choose a style from the menu to the right of this field.
[Line] Color	The color of the line.
[Fill] Filled	The Filled check box determines whether the object will be filled with a color. If you check this box, an invisible line is drawn from one end of your line to the other. Everything between the invisible line and your line will be filled.  - unfilled  - filled
[Fill] Color	The color with which the object will be filled. The color that you select as your fill color here is static. To specify a fill color that changes with runtime conditions, click the <b>Fill</b> tab. If you have enabled the Fill (Color) properties, be aware that the color you select here will override the OFF color for Fill Color (On/Off), the ABC color for Fill Color (Multi-state), Array Color 0 for Fill Color (Array), and the At minimum color for Fill Color (Gradient).

For help on the remaining properties tabs, see [Edit Common Object Properties](#).

## See Also

[Graphics Object Types](#)

### Add a Straight Line

The Straight Line tool allows you to draw straight lines. Straight lines can be moved, re-sized, reshaped, brought to the front and so on, and their properties edited just like any other type of object.

#### To draw a straight line:

1. Click the **Straight Line** tool.

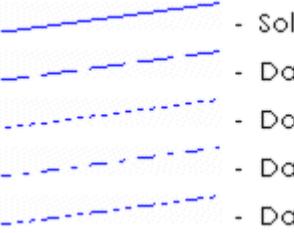


2. Move the cursor to where you want to start the line.
3. Click and drag to draw the line. (If you hold the **Ctrl** key while drawing the line it is constrained to the vertical or horizontal).

When you release the mouse button, the object properties dialog for the line is displayed.

### Straight Line Properties - Appearance (General)

Straight Lines have the following general appearance properties.

Property	Description
[Line] Width	The width of the line (in pixels). You can change the width by clicking the up and down arrows to the right of the field, or by entering another value in this field. If you make the line more than 1 pixel wide, it needs to be solid.
[Line] Style	The width of the line (in pixels). You can change the width by clicking the up and down arrows to the right of the field, or by entering another value in this field. If you make the line more than 1 pixel wide, it needs to be solid.  The style of the line. You can choose from the following line styles:   <ul style="list-style-type: none"><li>- Solid</li><li>- Dash</li><li>- Dot</li><li>- Dash Dot</li><li>- Dash Dot Dot</li></ul> <p>To change the style, choose a style from the menu to</p>

Property	Description
	the right of this field.
[Line] Color	The color of the line.

For help on the remaining properties tabs, see [Edit Common Object Properties](#).

## See Also

[Graphics Object Types](#)

### Add a Rectangle

Use the Rectangle tool to draw rectangles and squares. Rectangles and squares can be moved, re-sized, reshaped, brought to the front and so on, and their properties edited just like other types of object.

#### To draw a rectangle:

1. Click the **Rectangle** tool.



2. Move the cursor to where you want the rectangle to start.
3. Click and drag the mouse to the opposite corner of the rectangle and release the mouse button. If you hold the **Shift** key before you start drawing the rectangle, it is drawn from its center outwards.

When you release the mouse button, the object properties dialog for the rectangle is displayed.

#### To draw a square:

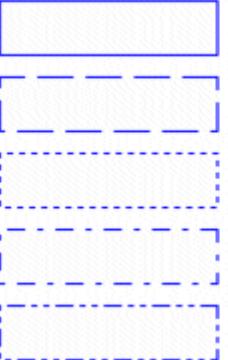
1. Click the **Rectangle** tool.
2. Click (and hold) the **Ctrl** key.
3. Move the cursor to where you want the square to start and click (and hold) the mouse button.
4. Drag the cursor to the opposite corner of the square and release the mouse button. If you hold the **Shift** key (and the **Ctrl** key) before you start drawing the square, it is drawn from its center outwards.

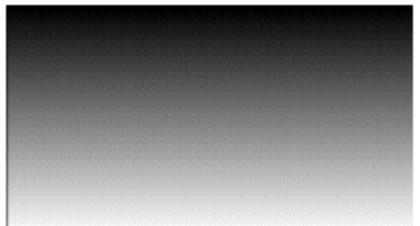
When you release the mouse button, the object properties dialog for the square is displayed.

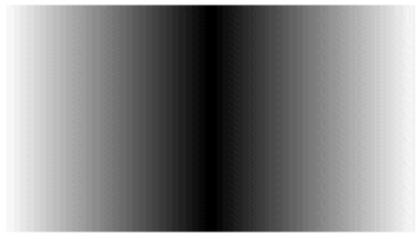
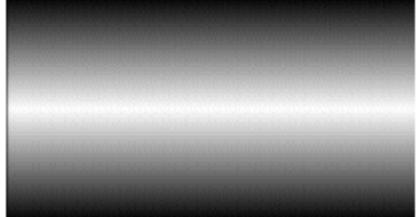
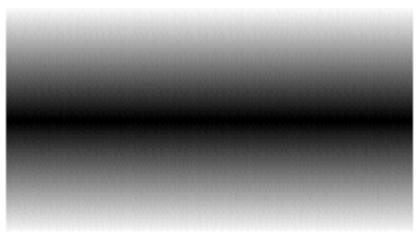
## Rectangle Properties - Appearance (General)

Rectangles have the following general appearance properties.

Property	Description
[Line] Width	The width of the line (in pixels). You can change the width by clicking the up and down arrows to the right of the field, or by entering another value in this field. If you make the line more than 1 pixel wide, it needs

Property	Description
	to be solid.
<b>[Line] Style</b>	<p>The outline style of the rectangle. You can choose one of the following line styles:</p>  <ul style="list-style-type: none"> <li>- Solid</li> <li>- Dash</li> <li>- Dot</li> <li>- Dash Dot</li> <li>- Dash Dot Dot</li> </ul> <p>To change the style, choose a style from the menu to the right of this field.</p>
<b>[Line] Color</b>	The outline color of the rectangle.
<b>[Fill] Filled</b>	The Filled check box determines whether the object will be filled with a color.
<b>[Fill] Color</b>	<p>The color with which the rectangle will be filled. The color that you select as your fill color here is static.</p> <p>To specify a fill color that changes with runtime conditions, click the <b>Fill</b> tab.</p> <p>If you have enabled the Fill (Color) properties, be aware that the color you select here will override the OFF color for Fill Color (On/Off), the ABC color for Fill Color (Multi-state), Array Color 0 for Fill Color (Array), and the At minimum color for Fill Color (Gradient).</p>
<b>[Object type] Extra line</b>	Adds an extra line (1 pixel width) of lowlight color to the rectangle, if the rectangle is defined as Raised or Lowered (click the 3D Effects tab).
<b>[Gradient] Color</b>	<p>Controls the color of the gradient fill between the fill color and the gradient color. This option is available only when the <b>Filled</b> and <b>Gradient Fill</b> check boxes are selected. The gradient is updated at runtime to reflect the gradient between the two colors selected.</p> <p>Gradient fills support flashing colors.</p> <p>Gradients will not rotate with an object; for example, if an object contains a left-to-right gradient fill and is rotated 90 degrees (either at runtime or in Graphics</p>

Property	Description
	Builder), the gradient is still left to right.
<b>[Gradient] Direction</b>	<p>The direction to be used for the gradient color. Use the table below as a guide to choose the gradient color direction you want.</p> 
	<p>Left to Right</p> 
	<p>Right to Left</p> 
	<p>Top to Bottom</p> 
	<p>Bottom to Top</p>

Property	Description
	
	
	
	
<b>[Object type] Border</b>	Adds an extra line (1 pixel width) of black to the perimeter of the rectangle.
<b>[Object type] Corner Radius</b>	Controls the radius of the corners of the rectangle. Enter a value between 0 and 32. The higher the value, the more rounded the corners of the rectangle.  When the radius is greater than 0, the <b>Extra line</b> and <b>Border</b> options are not available.

For help on the remaining properties tabs, see [Edit Common Object Properties](#).

## See Also

[Graphics Object Types](#)

### Add an Ellipse

You use the Ellipse tool to draw ellipses, circles, arcs, and pie-slices. Ellipse objects can be moved, re-sized, reshaped, brought to the front and so on, and their properties edited just like other types of object.

#### To draw an ellipse:

1. Click the **Ellipse** tool.



2. Move the cursor to a corner of the bounding rectangle (marquee) and click (and hold) the mouse button.
3. Drag the cursor to the opposite corner of the bounding rectangle and release the mouse button. If you hold the **Shift** key before you start drawing the ellipse, it is drawn from its center outwards.

When you release the mouse button, the object properties dialog for the ellipse is displayed.

#### To draw a circle:

1. Click the **Ellipse** tool.
2. Click (and hold) the **Ctrl** key.
3. Move the cursor to a corner of the bounding rectangle (marquee) and click (and hold) the mouse button.
4. Drag the cursor to the opposite corner of the bounding rectangle and release the mouse button. If you hold the **Shift** key and the **Ctrl** key before you start drawing the circle, it is drawn from its center outwards.

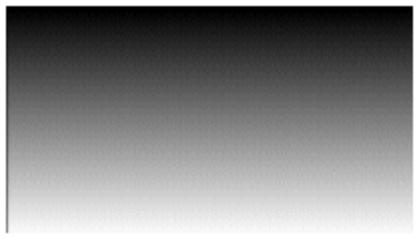
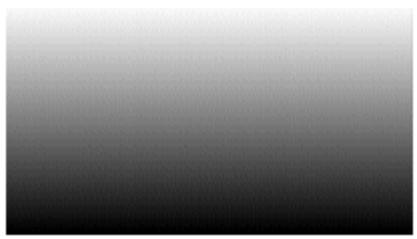
When you release the mouse button, the object properties dialog for the circle is displayed.

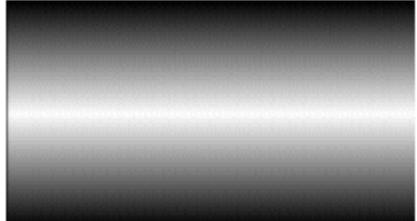
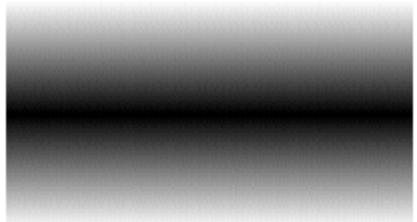
### Ellipse Properties - Appearance (General)

Ellipses have the following general appearance properties:

Property	Description
[Line] Width	The width of the outline of the ellipse (in pixels). You can change the width by clicking the up and down arrows to the right of the field, or by entering another value in this field. If you make the line more than 1 pixel wide, the line style will be solid.
[Line] Style	The outline style of the ellipse. You can choose one of the following line styles:

Property	Description
	 <p>- Solid  - Dash  - Dot  - Dash Dot  - Dash Dot Dot</p> <p>To change the style, choose a style from the menu to the right of this box.</p>
[Line] Color	The outline color of the ellipse.
[Fill] Filled	The Filled check box determines whether the ellipse will be filled with a color.
[Fill] Color	<p>The color with which the ellipse will be filled. The color that you select as your fill color here is static.</p> <p>To specify a fill color that changes with runtime conditions, click the Fill tab. If you have enabled the Fill (Color) properties, be aware that the color you select here will override the OFF color for Fill Color (On/Off), the ABC color for Fill Color (Multi-state), Array Color 0 for Fill Color (Array), and the At minimum color for Fill Color (Gradient).</p>
[Gradient] Color	<p>Controls the color of the gradient fill between the fill color and the gradient color. This option is available only when the <b>Filled</b> and <b>Gradient Fill</b> check boxes are selected. The gradient is updated at runtime to reflect the gradient between the two colors selected.</p> <p>Gradient fills support flashing colors.</p> <p>Gradients will not rotate with an object; for example, if an object contains a left-to-right gradient fill and is rotated 90 degrees (either at runtime or in Graphics Builder), the gradient is still left to right.</p>
[Gradient] Direction	The direction to be used for the gradient color. Use the table below as a guide to choose the gradient color direction you want.

Property	Description
	 A horizontal gradient bar transitioning from black on the left to white on the right.
	<p>Left to Right</p>  A horizontal gradient bar transitioning from white on the left to black on the right.
	<p>Right to Left</p>  A horizontal gradient bar transitioning from white at the top to black at the bottom.
	<p>Top to Bottom</p>  A horizontal gradient bar transitioning from black at the top to white at the bottom.
	<p>Bottom to Top</p>  A horizontal gradient bar transitioning from black on the left and right to white in the center.
	<p>Horizontal Gradient to Middle</p>

Property	Description
	 Horizontal Gradient from Middle   Vertical Gradient to Middle   Vertical Gradient from Middle
<b>[Object type] Ellipse</b>	<p>Select this radio button if you want to the object to be a full ellipse.</p>  For a full ellipse, you are not required to specify Start and End angles.
<b>[Object type] Pie-slice</b>	<p>Select this radio button if you want to remove a section from your ellipse (i.e., you want it to resemble a pie-slice).</p> <p>If you select this option, you can specify a Start angle, and an End angle:</p>
<b>Start angle</b>	<p>The angle (measured clockwise from 0 degrees) of the section to be removed from the ellipse. For example, if you enter a start angle of 50 degrees, your pie-slice would look something like this:</p>

Property	Description
	
<b>End angle</b>	<p>The angle (measuring clockwise from 0 degrees) of the section of the ellipse which is to remain. For example, if you enter an end angle of 150 degrees, your pie-slice would look something like this:</p>  <p>Start and End angles can be combined for various effects. For example, a Start angle of 270 degrees, and an End angle of 150 degrees would produce the following pie-slice:</p> 
<b>[Object type] Arc</b>	<p>Select this radio button if you want to draw an arc. If you select this option, you can specify a Start angle, and an End angle:</p>
<b>Start angle</b>	<p>The angle (measured clockwise from 0 degrees) defining the segment to be removed from the ellipse, leaving an arc. For example, if you enter a start angle of 50 degrees, your arc would look something like this:</p> 
<b>End angle</b>	<p>The angle (measuring clockwise from 0 degrees) defining the segment of the ellipse which is to remain. For example, if you enter an end angle of 150 degrees, your pie-slice would look something like this:</p>  <p>Start and End angles can be combined for various effects. For example, a Start angle of 270 degrees, and an End angle of 150 degrees would produce the following arc:</p> 

For help with the other properties, see [Edit Common Object Properties](#).

## See Also

[Graphics Object Types](#)

### Add a Polygon

Use the Polygon tool to draw polygons and polylines. Polygons can be moved, re-sized, reshaped, brought to the front and so on, and their properties edited just like other types of object.

---

**Note:** You can manipulate the shape of a polygon at runtime by associating an expression with the polygon vertices. See [Animate a Polygon at Runtime](#) for a description of a polygon's Vertices properties.

---

#### To draw a polygon:

1. Click the **Polygon** tool.



2. Move the cursor to where you want the polygon to start and click and hold the mouse button.
3. At the end of the first line segment, release the mouse button.
4. Move the cursor to each point on the polygon in turn, and click the mouse button (clicking and dragging is not necessary after the first segment).
5. To draw horizontally or vertically only, hold the **Ctrl** key down when you are drawing the polygon.
6. To complete the polygon, double-click the mouse button.

When you complete the polygon, the object properties dialog is displayed.

#### To draw a polyline:

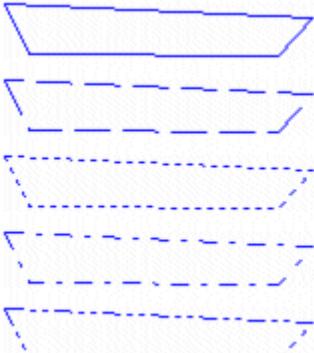
1. Click the **Polygon** tool.
2. Move the cursor to where you want the polyline to start and click and hold the mouse button.
3. At the end of the first line segment, release the mouse button.
4. Move the cursor to each point on the polyline in turn and click the mouse button (clicking and dragging is not necessary after the first segment).
5. To draw horizontally or vertically only, hold the **Ctrl** key down when you are drawing the polyline.
6. To complete the polyline, double-click the mouse button.

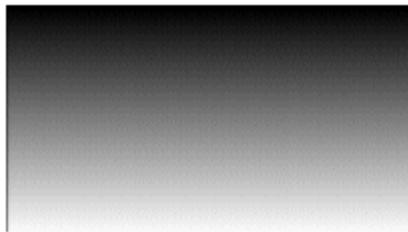
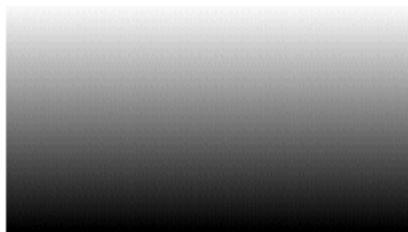
When you release the mouse button, the object properties dialog for the circle is displayed.

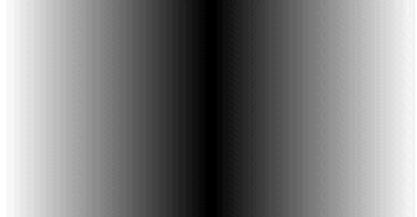
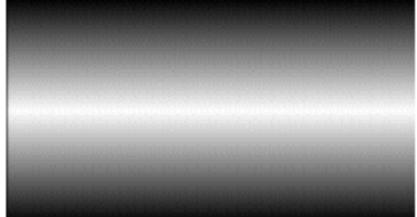
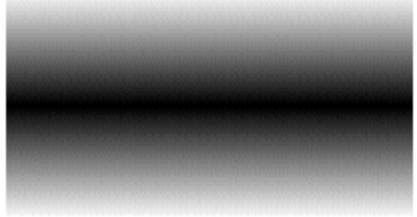
Initially, the object will actually be a polygon. To change it to a polyline, in the object's properties dialog define it as **[Object type] Open**.

### Polygon Properties - Appearance (General)

Polygons have the following general appearance properties:

Property	Description
[Line] Width	<p>The width of the outline of the polygon (in pixels). You can change the width by clicking the up and down arrows to the right of the field, or by entering another value in this field.</p> <p>If you make the line more than 1 pixel wide, the line style will be solid.</p>
[Line] Style	<p>The outline style of the polygon. You can choose one of the following line styles:</p>  <ul style="list-style-type: none"> <li>- Solid</li> <li>- Dash</li> <li>- Dot</li> <li>- Dash Dot</li> <li>- Dash Dot Dot</li> </ul> <p>To change the style, choose a style from the menu to the right of this box.</p>
[Line] Color	The outline color of the polygon.
[Fill] Filled	The Filled check box determines whether the polygon will be filled with a color.
[Fill] Color	<p>The color with which the polygon will be filled. The color that you select as your fill color here is static. To specify a fill color that changes with runtime conditions, click the Fill tab. If you have enabled the Fill (Color) properties, be aware that the color you select here will override the OFF color for Fill Color (On/Off), the ABC color for Fill Color (Multi-state), Array Color 0 for Fill Color (Array), and the At minimum color for Fill Color (Gradient).</p>
[Gradient] Color	<p>Controls the color of the gradient fill between the fill color and the gradient color. This option is available only when the <b>Filled</b> and <b>Gradient Fill</b> check boxes are selected. The gradient is updated at runtime to reflect the gradient between the two colors selected. Gradient fills support flashing colors.</p> <p>Gradients will not rotate with an object; for example, if an object contains a left-to-right gradient fill and is</p>

Property	Description
	rotated 90 degrees (either at runtime or in Graphics Builder), the gradient is still left to right.
<b>[Gradient] Direction</b>	<p>The direction to be used for the gradient color. Use the examples below as a guide to choose the gradient color direction you want.</p> 
	<p>Left to Right</p> 
	<p>Right to Left</p> 
	<p>Top to Bottom</p>  <p>Bottom to Top</p> 

Property	Description
	
	Horizontal Gradient to Middle
	
	Horizontal Gradient from Middle
	
	Vertical Gradient to Middle
	
	Vertical Gradient from Middle
<b>[Object type] Open</b>	Defines the object as a polyline (the first point and the last point are not joined)..
<b>[Object type] Closed</b>	Defines the object as a polygon (the first point and the last point are joined).

For a description of a polygon's Vertices properties, see [Animate a Polygon at Runtime](#).

For help with the other properties, see [Edit Common Object Properties](#).

## See Also

[Graphics Object Types](#)

### Animate a Polygon at Runtime

Plant SCADA allows you to associate a tag or Cicode expression with the vertices of a polygon. This means the shape of a polygon can be manipulated at runtime in response to values generated by a production system.

This is achieved by applying offset values to a vertex (in relation to its location) that define a path along which the vertex can move at runtime.

- **From Offset X** and **From Offset Y** define the starting point for the vertex path.
- **To Offset X** and **To Offset Y** define the end point for the vertex path.

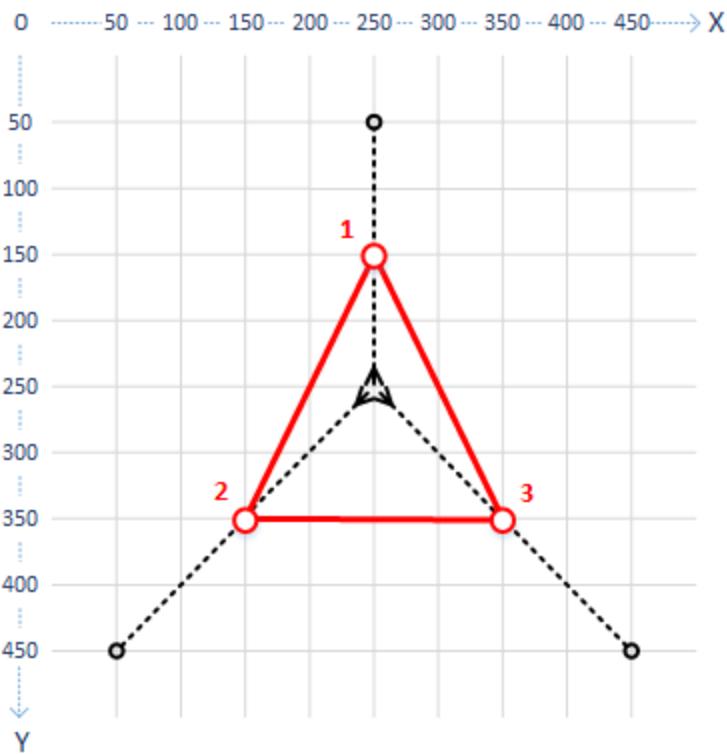
As the return value for the expression changes, the vertex will move along the path defined for it, altering the shape of the polygon.

## Example

A triangle is placed on a graphics page with the following settings configured for its three vertices.

Vertex	Location		From Offset		To Offset		Specified Range	
	X	Y	X	Y	X	Y	Min	Max
1	250	150	0	+100	0	-100	0	100
2	150	350	+100	-100	-100	+100	0	100
3	350	350	-100	-100	+100	+100	0	100

The following diagram shows the triangle with the path defined for each vertex by its offset values. In each case, the defined path starts at the center of the triangle and extends outwards.

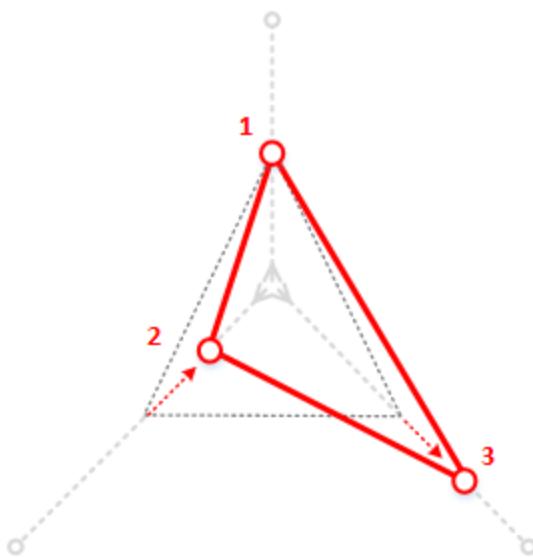


To manipulate the shape of the triangle at runtime, the following expressions could be added to the vertices.

Vertex	Expression
1	Tag1
2	Tag2
3	Tag3

Assuming the following list represents the current tag values, the triangle would appear as displayed below.

- Tag1 = 50
- Tag2 = 25
- Tag3 = 75



You can also apply a range to a vertex that specifies the start point and end point values for its path. By default, this range is set to 0—100.

If you need to accommodate a different range of return values for an expression, select the **Specify Range** option when configuring the vertex properties.

## Vertices Properties

Polygon vertices have the following properties:

Property	Description
[Vertices] Vertex	A number automatically applied to each vertex in a polygon, based on the order in which they were drawn. To set the properties for a particular vertex, select it within this list.
[Vertices] X	The location of a vertex along the horizontal axis of the page (in pixels). Zero (0) represents the left edge of the page.
[Vertices] Y	The location of a vertex along the vertical axis of the page (in pixels). Zero (0) represents the top edge of the page.
Expression for Vertex: <n>	The tag value or Cicode expression used to determine the location of the selected vertex at runtime. As the return value for the expression changes, the vertex will move along the path defined by its offset values (see below). To insert a tag or a function, click the Wizard button to

Property	Description
	the right of this field. This button displays two options: Insert Tag and Insert Function.
<b>Specify Range</b>	<p>Select this option if you would like to specify the start and end values for the vertex path.</p> <p>This option will be required if the return values from an expression span a different range to the default setting of 0—100.</p>
<b>[Specify Range] Minimum</b>	<p>Enter the minimum return value that you would like to use for the vertex range. When this value is returned, the vertex will shift to the location specified by the <b>From Offset</b> values.</p> <p>You can only enter a value here if you have selected the <b>Specify</b> range box.</p>
<b>[Specify Range] Maximum</b>	<p>Enter the maximum return value that you would like to use for the vertex range. When this value is returned, the vertex will shift to the location specified by the <b>To Offset</b> values.</p> <p>You can only enter a value here if you have selected the <b>Specify</b> range box.</p>
<b>[From Offset] X</b>	<p>The distance (in pixels) along a horizontal axis that the vertex will shift when the expression returns the minimum value in the specified range. You can shift the vertex to the left by entering a negative offset value.</p> <p>You can change the offset value by pressing the up and down arrows to the right of the field, or by directly entering a value.</p>
<b>[From Offset] Y</b>	<p>The distance (in pixels) along a vertical axis that the vertex will shift when the expression returns the minimum value in the specified range. You can shift the vertex towards the top of the page by entering a negative offset value.</p> <p>You can change the offset value by pressing the up and down arrows to the right of the field, or by directly entering a value.</p>
<b>[To Offset] X</b>	<p>The distance (in pixels) along a horizontal axis that the vertex will shift when the expression returns the maximum value in the specified range. You can shift the vertex to the left by entering a negative offset value.</p>

Property	Description
	You can change the offset value by pressing the up and down arrows to the right of the field, or by directly entering a value.
[To Offset] Y	The distance (in pixels) along a vertical axis that the vertex will shift when the expression returns the maximum value in the specified range. You can shift the vertex towards the top of the page by entering a negative offset value.  You can change the offset value by pressing the up and down arrows to the right of the field, or by directly entering a value.

## See Also

[Add a Polygon](#)

## Add a Pipe

Use the Pipe tool to draw pipes with automatic three-dimensional shading. Pipes can be moved, re-sized, reshaped, brought to the front and so on, and their properties edited just like other types of object.

### To draw a pipe:

1. Click the **Pipe** tool.



2. Move the cursor to where you want the pipe to start, and click and hold the mouse button.
3. At the end of the first line segment, release the mouse button.
4. Move the cursor to each point on the path in turn and click the mouse button (clicking and dragging is not necessary after the first segment).
5. To complete the pipe, double-click the mouse button.

When you complete the pipe, the object properties dialog is displayed.

To draw horizontally or vertically only, hold the **Ctrl** key down when drawing the pipe.

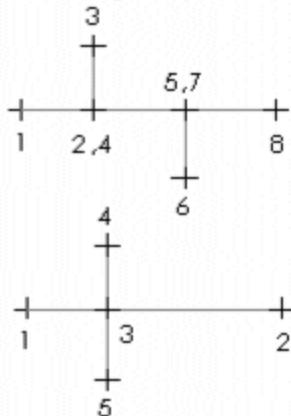
## Drawing Complex Pipe Arrangements

Use the Pipe tool to draw complex pipe arrangements (including 'T' pieces and junctions). The illustration below shows some pipes, and the sequence of mouse clicks needed to draw each of them:

To draw this pipe ...



... follow this mouse clicking sequence



**Hint:** Use the grid to assist in accurate positioning for each click.

## Pipe Properties - Appearance (General)

Pipes have the following general appearance properties.

Property	Description
[Line] Width	The width of the pipe (in pixels). You can change the width by clicking the up and down arrows to the right of the field, or by entering another value in this field. Every pipe needs to be at least 1 pixel wide.
[Line] Highlight Color	The color of the pipe where it is "in the light"; that is, the brightest color on the pipe.
[Line] Lowlight Color	The color of the pipe where it is "in shadow"; that is, the dullest color on the pipe.

For help on the remaining properties tabs, see [Edit Common Object Properties](#).

## See Also

[Graphics Object Types](#)

## Add Text

Use the Text tool to type text on the page. Text can be moved, re-sized, reshaped, brought to the front and so on, and their properties edited just like other types of object.

### To add text:

1. Click the **Text** tool.



2. Type your text on the keyboard. (Press **Enter** to start a new line.)
3. Move the cursor to where you want to position the text and click the left mouse button.

When you click the mouse button to place the text, the object properties dialog is displayed.

**Note:** You can adjust the appearance properties for a text object by selecting **Font** from Graphics Builder's **Text** menu.

## Text Properties - Appearance (General)

Text has the following general appearance properties

Property	Description
<b>Font</b>	The font used to display the text. Use the scroll bar to the right to view available fonts, or type the font name, or part of it, directly into this field.
<b>Style</b>	Select whether you would like the text to be Regular, <b>Bold</b> , <b>Bold Italic</b> , or <i>Italic</i> .
<b>Size</b>	Define the size of the text (point size). Available sizes might vary according to the selected printer and the selected font.
<b>[Alignment] Left</b>	Select this radio button to align the text to the left of the text box.
<b>[Alignment] Right</b>	Select this radio button to align the text to the right of the text box.
<b>[Alignment] Center</b>	Select this radio button to align the text in the center of the text box.
<b>[Effect] Strikeout</b>	Select this box to make the text will appear with a line through it.
<b>[Effect] Underline</b>	Select this box to underline the text.
<b>Text</b>	This field contains the text that will display on the page. You can enter any keyboard character(s). You can edit the text here, or directly from the graphics page. It is useful to edit text at this field, as you can apply text changes at the same time as you apply other font and color changes.  This text changes automatically depending on the Display Value properties that you define.

Property	Description
<b>Foreground</b>	The color of the text.
<b>Script</b>	<p>The script for the text.</p> <p><b>Note:</b> If the selected script is not supported by the selected font, a warning icon will appear next to the drop-down. The icon will display a tool tip that explains, the combination is unsupported and a different font will be used at runtime.</p>

**Note:** There are several radio buttons in Display Value (On/Off, Multi-state and so on). When selected, these radio buttons change the appearance of the right hand side of the dialog.

## Text Properties - Appearance (Display Value)

Text has the following display value appearance properties. Selecting one of the following five options changes the appearance of the right hand side of the dialog.

Property	Description
<b>[Type] On / Off</b>	<p>Changes the text which displays when a particular condition is met, and another when it is not. For example, you could display an alarm message when a particular variable tag is in alarm, and a normal message when it is not.</p> <p>See <a href="#">Text Properties - Appearance Display Value (On/Off)</a>.</p>
<b>[Type] Multi-state</b>	<p>This option is useful when you have several possible conditions, occurring together in different combinations, at different times. Select this option to display different text for each combination.</p> <p>For example, three digital variable tags (A,B, and C) can each be ON or OFF at any time. You can display a different message for each ON/OFF combination. In other words, you could display a different message for each of the following ON/OFF combinations ABC, ABC, ABC, ABC, ABC, ABC, ABC.</p> <p>.</p>
<b>[Type] Array</b>	<p>Allows you to enter an expression which returns an integer. For each integer (from 0-255), you can display different text. For example, you could display a different message for each state of an analog tag.</p> <p>See <a href="#">Text Properties - Appearance Display Value (Array)</a>.</p>

Property	Description
[Type] Numeric	Displays the value of a tag or expression in numeric format (you can specify the format). See <a href="#">Text Properties - Appearance Display Value (Numeric)</a> .
[Type] String	Displays the value of an expression as a string. See <a href="#">Text Properties - Appearance Display Value (String)</a>

## See Also

[Graphics Object Types](#)

### Text Properties - Appearance Display Value (On/Off)

Text has the following On/Off Display Value properties:

Property	Description
<b>ON text when</b>	The text entered in the <b>ON text</b> field (below) appears when the condition entered here is true. The text can be a maximum of 128 characters long. To insert a tag or a function, click the <b>Wizard</b> button to the right of this field. This button displays two options; <b>Insert Tag</b> , and <b>Insert Function</b> .
<b>OFF text</b>	The text that will display whenever the condition entered above is false. You can use any keyboard character(s) to a maximum length of 256 characters. For example, you could display the message <b>Conveyor 110 Normal</b> when <b>CV110_ERROR.On</b> is false (i.e., there is no alarm at conveyor 110).
<b>ON text</b>	The text that will display whenever the condition entered above is true. You can enter any keyboard character(s) to a maximum length of 256 characters. For example, you could display the message <b>Conveyor 110 Alarm</b> when <b>CV110_ERROR.On</b> is true (i.e. there is no alarm at conveyor 110).

Click **Clear Property** to clear property details and disable the property.

## See Also

[Text Properties - Appearance Display Value \(Multi-state\)](#)

[Text Properties - Appearance Display Value \(Array\)](#)

[Text Properties - Appearance Display Value \(Numeric\)](#)

[Text Properties - Appearance Display Value \(String\)](#)

### Text Properties - Appearance Display Value (Multi-state)

Text has the following multi-state display value properties:

#### Conditions

The conditions you enter here will occur together in different ways, at different times. You can use each different combination to determine the text that will display.

The default number of conditions is 3, but you can add more (to a maximum of 5 conditions, providing 32 combinations), using the **Add** button. You can also delete conditions using the **Delete** button, but you need to enter a condition in this field. To enter a condition, click the relevant line (A, B, C, etc.), and click **Edit**. To insert a tag or function, click the **Wizard** button. This button displays two options: Insert Tag and Insert Function.

**Note:** You can also insert Equipment.item references into expression fields using the **insert tag** option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with available tags.

#### State text

The text that is to display for each combination of the above conditions. You can enter any keyboard character(s).

For example:

To display different messages about the status of a valve, you could fill out the **Conditions** and **State text** fields as follows:

Conditions		State text	
A	Open Feedback	ABC	Valve Inoperable
B	Close Feedback	ABC	Valve Inoperable
C	Open Output	ABC	Valve Closed
		ABC	Valve Inoperable
		ABC	Valve Inoperable
		ABC	Valve Open
		ABC	Valve Inoperable
		ABC	Valve Inoperable

In this example, **Open\_Feedback** and **Close\_Feedback** are variable tags representing digital inputs on the valve; **Open\_Output** is a variable tag representing an output on the valve. So, ABC means **Open\_Feedback** is on, and **Close\_Feedback** and **Open\_Output** are both off. For this combination, the text "Valve Inoperable" will display, because the valve is open when it is meant to be closed. The same type of logic applies to the rest of the states.

Click **Clear Property** to clear property details and disable the property.

### See Also

[Text Properties - Appearance Display Value \(On/Off\)](#)

[Text Properties - Appearance Display Value \(Array\)](#)

[Text Properties - Appearance Display Value \(Numeric\)](#)

[Text Properties - Appearance Display Value \(String\)](#)

## Text Properties - Appearance Display Value (Array)

Text has the following Array Display Value properties:

### Array expression

Enter the expression which is to return one or more integers. For each returned integer, a different piece of text is displayed.

If the return value is:

- Less than 0 (zero), it will be set to 0 (zero), and a runtime hardware alarm will be triggered.
- Greater than 255, it will be set to 255, and a runtime hardware alarm will be triggered.
- A real (non-integer) number, it will be rounded off to the nearest integer.

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: Insert Tag and Insert Function.

**Note:** You can also insert Equipment.item references into expression fields using the **insert tag** option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with available tags.

### Array text

The text that is to display for each integer returned by the Array expression entered above. You can enter any keyboard character(s).

For example, to display different messages about the status of a motor, you could fill out the **Array expression** and **Array text** fields as follows:

Array expression	Array text	
MOTOR_STATUS	0	Running
	1	Starting
	2	Stopping
	3	Stopped - Normal
	4	Stopped - Error
	5	Isolated

In this example, MOTOR\_STATUS is an analog variable tag representing the status of a motor. When the motor changes state, an integer is returned (0 = Running, 1 = Starting etc.) and the appropriate text displays.

Click **Clear Property** to clear property details and disable the property.

## See Also

[Text Properties - Appearance Display Value \(On/Off\)](#)

[Text Properties - Appearance Display Value \(Multi-state\)](#)

[Text Properties - Appearance Display Value \(Numeric\)](#)

## [Text Properties - Appearance Display Value \(String\)](#)

### **Text Properties - Appearance Display Value (Numeric)**

Text has the following Numeric Display Value properties.

#### **Numeric expression**

The value of the expression entered here will be displayed on the graphics page. It will be formatted according to the format selected.

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: Insert tag and Insert Function.

---

**Note:** You can also insert Equipment.item references into expression fields using the **insert tag** option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with available tags.

#### **Format**

The value returned for the expression entered above displayed according to the format you enter here. For example, the analog variable tag **MOTOR\_STATUS** returns integers 0-5. If you enter this tag as the Numeric expression above, and enter **#.##** as your format, the display will alternate between **0.00**, **1.00**, **2.00**, **3.00**, **4.00**, and **5.00**. You can select a format from the drop-down list, or type in your own. If the numeric expression is a single variable, its format is overwritten by the format you enter here.

---

**Note:** To display the Variable Tag's unit as defined in the Eng unit ( System Model | Variables) field select **#!** from the drop down list.

Click **Clear Property** to clear property details and disable the property.

## **See Also**

[Text Properties - Appearance Display Value \(On/Off\)](#)

[Text Properties - Appearance Display Value \(Multi-state\)](#)

[Text Properties - Appearance Display Value \(Array\)](#)

[Text Properties - Appearance Display Value \(String\)](#)

### **Text Properties - Appearance Display Value (String)**

Text has the following String Display Value properties.

#### **String Expression**

The value of the expression entered here will be displayed as a string on the graphics page.

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: Insert tag and Insert Function.

---

**Note:** You can also insert Equipment.item references into expression fields using the **insert tag** option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with available tags.

---

Click **Clear Property** to clear property details, and disable the property. To define further properties for the object, click the relevant tabs.

For help on the remaining properties tabs, see [Edit Common Object Properties](#).

## See Also

- [Text Properties - Appearance Display Value \(On/Off\)](#)
- [Text Properties - Appearance Display Value \(Multi-state\)](#)
- [Text Properties - Appearance Display Value \(Array\)](#)
- [Text Properties - Appearance Display Value \(Numeric\)](#)

## Add a Button

Use the **Button** tool to draw buttons on graphics page. You can then assign security rights and attach commands to it.

Buttons can be moved, re-sized, reshaped, brought to the front, and so on, and their properties edited just like other types of object.

### To draw a button:

1. Click the **Button** tool.



2. Move the mouse to where you want the button to start and press (and hold) the mouse button.
3. Drag the mouse to where you want the button to finish and release the mouse button.

When you release the mouse button, the object properties dialog is displayed.

## Button Properties - Appearance (General)

Buttons have the following General Appearance properties.

Property	Description
<b>[Type] Text</b>	<p>Select this option to display text on the button. If you select this option, the <b>Text</b> and <b>Font</b> fields will display to the right of the dialog.</p> <p>If either the <b>Text</b> or <b>Symbol</b> type option is selected, you can use XP style buttons. To configure a button to use the Windows XP style, select the <b>XP Style</b> check box. During runtime an XP style button has a blue border when it has keyboard focus, and an orange border when the mouse is on the button. When you place a button on a page, the <b>XP Style</b> check box is selected by default.</p> <p>Pressing '^n' or 'Enter' wraps the text onto the next line. For example, Start^nMotor would display as:</p> 
<b>Font</b>	<p>The font used to display the text. Use the scroll bar to the right to view available fonts, or type part of a font name directly into this field.</p> <p><b>Note:</b> The Background Color property is no longer supported for button fonts. Any imported buttons from a previous version will have the Background Color set to the default color.</p>
<b>Style</b>	<p>Select whether you would like the text to be Regular, Bold, Bold Italic, or Italic.</p>
<b>Size</b>	<p>Define the size of the text (point size). Available sizes might vary according to the selected printer and the selected font.</p>
<b>Custom Fill Color</b>	<p>Select this check box to enable the Fill Color Up and Fill Color down boxes.</p>
<b>Fill Color Up</b>	<p>The button color when in the Up state.</p>
<b>Fill Color Down</b>	<p>The button color when in the Down state.</p> <p><b>Note:</b> Fill Color Up and Down options available for Text and Symbol Types only.</p>
<b>[Alignment] Left</b>	<p>Select this radio button to left-align multi-line text. The aligned text will remain in the center of the button. This has no effect on single lines of text.</p>
<b>[Alignment] Right</b>	<p>Select this radio button to right-align multi-line text. The aligned text will remain in the center of the</p>

Property	Description
	button. This has no effect on single lines of text.
[Alignment] Center	Select this radio button to center-align multi-line text. The aligned text will remain in the center of the button. This has no effect on single lines of text.
[Effect] Strikeout	Select this box to make the text appear with a line through it.
[Effect] Underline	Select this box to underline the text.
Text	<p>This field contains the text that will display on the page. You can enter any keyboard character(s). You can edit the text here, or directly from the graphics page. It is useful to edit text at this field, as you can apply text changes at the same time as you apply other font and color changes.</p> <p>This text changes automatically depending on the Display Value properties that you define.</p>
Foreground	The color of the text.
Script	<p>The script for the text.</p> <p><b>Note:</b> If the selected script is not supported by the selected font, a warning icon will appear next to the drop-down. The icon will display a tool tip that explains, the combination is unsupported and a different font will be used at runtime.</p>
[Type] Symbol	<p>Select this option to display a symbol on the button. If you select this option, the <b>Set</b> button will display to the right of the dialog.</p> <p>Click <b>Set</b> to choose the symbol which is to display on the button. A picture of the selected symbol will also display.</p>
[Type] Target	When this option is selected, the button will not have any text or symbols on it, and it will have a transparent face.
Mode	<p>There are three different modes of transparent buttons:</p> <ul style="list-style-type: none"> <li>• <b>BORDER_3D:</b> The button is drawn with only the 3-D border (transparent face).</li> <li>• <b>BORDER:</b> The button is drawn with only a thin line</li> </ul>

Property	Description
	<p>border.</p> <ul style="list-style-type: none"><li>• <b>TARGET:</b> The button is totally transparent. This constitutes a screen target.</li></ul>

For help on the remaining properties tabs, see [Edit Common Object Properties](#).

## See Also

[Graphics Object Types](#)

## Add Numbers

Use the Number tool to represent a tag or expression as a number. When you place a number on your page, simply enter the relevant variable tag or expression. Numbers can be moved, re-sized, brought to the front, and so on, and their properties edited just like other types of object.

(The same functionality is also available through the Text tool.)

**To add a number to your graphics page:**

1. Click the **Number** tool.



2. Move the cursor to where you want the number to display and click the mouse button. The Text Properties dialog box appears where you enter the relevant variable tag or expression.

## See Also

[Add Text](#)

[Graphics Object Types](#)

## Add a Symbol Set

The Symbol Set tool allows you to represent changing runtime conditions with changing symbols. By clicking on this tool, then clicking on the graphics page, you can define the symbols which are to display for each condition.

After a symbol set has been added to the page, it can be moved, re-sized, re-shaped, brought to the front etc., and its properties can be edited, just like any other type of object.

**To add a Symbol Set:**

1. Click the **Symbol** tool.



2. Move the mouse pointer to the desired position on the page, and click with the left mouse button.
3. When you click the mouse button, the object properties dialog is displayed.
4. Fill out the relevant properties for the symbol set (see below for a description), and click **OK**.

**Note:** There are several radio buttons in Symbol Sets Appearance (described below). When selected, these radio buttons change the appearance of the right hand side of the dialog.

## Symbol Set Properties

Property	Description
<b>[Type] On / Off</b>	<p>Select this radio button to display one symbol when a particular expression is TRUE, and another when it is FALSE. For example, you could display a red symbol when a particular variable tag is in alarm, and a green symbol when it is not.</p> <p>See <a href="#">Symbol Set Properties - Appearance General (On/Off)</a>.</p>
<b>[Type] Multi-state</b>	<p>This option is useful when you have several possible conditions, occurring together in different combinations, at different times. Select this option to display different symbols for each combination.</p> <p>For example, three digital variable tags (A,B, and C) can each be ON or OFF at any time. You can display a different symbol for each ON/OFF combination. In other words, you could display a different symbol for each of the following ON/OFF combinations ABC, ABC, ABC, ABC, ABC, ABC, ABC.</p> <p>See <a href="#">Symbol Set Properties - Appearance General (Multi-state)</a>.</p>
<b>[Type] Array</b>	<p>The Array option allows you to enter an expression which returns an integer. For each unique integer (from 0 to 255), you can display a unique symbol. For example, you could display a different symbol for each threshold of an analog alarm.</p> <p>See <a href="#">Symbol Set Properties - Appearance General (Array)</a>.</p>
<b>[Type] Animated</b>	<p>Select this radio button to display an actual animation (several different symbols in sequence).</p> <p>When selected, the radio buttons on the dialog box change the appearance of the right hand side of the dialog. These radio buttons are only documented once below.</p> <p>See <a href="#">Symbol Set Properties - Appearance General</a></p>

Property	Description
	(Animated).

## See Also

[Configure Graphics Objects](#)

### Symbol Set Properties - Appearance General (On/Off)

Symbol Sets have the following general appearance (On/Off) properties:

Property	Description
<b>ON symbol when</b> (128 Chars.)	The symbol entered in the <b>ON symbol</b> field (below) will display whenever the condition entered here is TRUE. The symbol entered in the <b>OFF symbol</b> field (below) will display whenever the condition entered here is FALSE.  To insert a tag or a function, click the Wizard button to the right of this field. This button displays two options; <b>Insert tag</b> , and <b>Insert Function</b> .
<b>OFF symbol</b>	The symbol that will display whenever the condition entered above is false. Click <b>Set</b> to select a symbol, or <b>Clear</b> to clear the current selection.  For example, you could display the OFF symbol when <b>MIX_RUNNING</b> is false.
<b>ON symbol</b>	The symbol that will display whenever the condition entered above is true. Click <b>Set</b> to select a symbol, or <b>Clear</b> to clear the current selection.  For example, you could display the ON symbol when <b>MIX_RUNNING</b> is true.

Click **Apply** or **OK** to bring your changes into effect, or **Cancel** to discard them and exit. Click **ClearProperty** to clear property details, and disable the property. To define further properties for the object, click the relevant tabs.

## See Also

[Add a Symbol Set](#)

[Symbol Set Properties - Appearance General \(Multi-state\)](#)

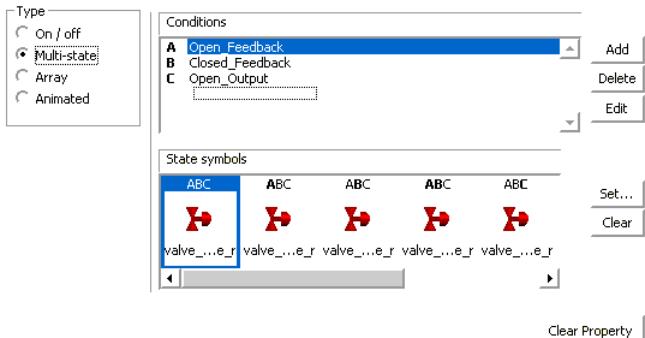
[Symbol Set Properties - Appearance General \(Array\)](#)

[Symbol Set Properties - Appearance General \(Animated\)](#)

[Configure Graphics Objects](#)

## Symbol Set Properties - Appearance General (Multi-state)

Symbol Sets have the following general appearance (Multi-state) properties:

Property	Description
<b>Conditions</b>	<p>The conditions you enter here will occur together in different ways, at different times. You can use each different combination to determine which symbol will display.</p> <p>To enter a condition, click the relevant line (A, B, C, etc.), and click <b>Edit</b>. You can add more conditions (to a maximum of 5, providing 32 combinations), using the <b>Add</b> button. To insert a tag or function, click the <b>Wizard</b> button. This button displays two options; <b>Insert Tag</b> and <b>Insert Function</b>. You can also delete conditions using the <b>Delete</b> button, but there needs to be a condition in this field. Conditions which are left black (instead of deleted) will be evaluated as false at runtime.</p>
<b>State symbols</b>	<p>The symbols that will display for each combination of the above conditions. Click the <b>Set</b> button to select a symbol, or <b>Clear</b> to clear the current selection.</p> <p>For example:</p> <p>To display different symbols each time the status of a valve changes, you could fill out the <b>Conditions</b> and <b>State symbols</b> fields as follows:</p>  <p>In this example, <b>Open_Feedback</b>, and <b>Close_Feedback</b> are variable tags representing digital inputs on the valve, and <b>Open_Output</b> is a variable tag representing an output on the valve. So, ABC means <b>Open_Feedback</b> is ON, and <b>Close_Feedback</b> and <b>Open_Output</b> are both OFF. For this combination, the inoperable symbol will display, because the valve is open when it is meant to be closed. The same type of logic applies to the rest of the states.</p>

Click **Clear Property** to clear property details, and disable the property. To define further properties for the object, click the relevant tabs.

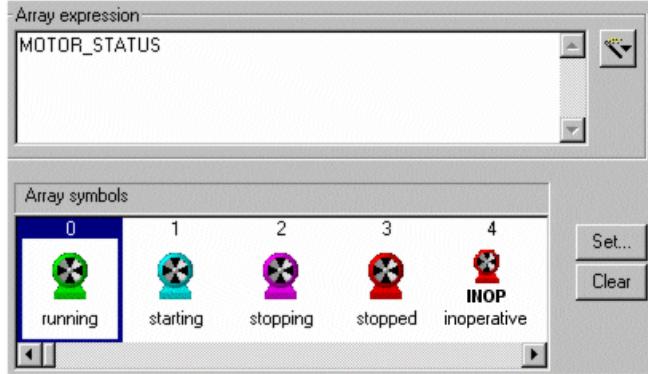
## See Also

- [Add a Symbol Set](#)
- [Symbol Set Properties - Appearance General \(On/Off\)](#)
- [Symbol Set Properties - Appearance General \(Array\)](#)
- [Symbol Set Properties - Appearance General \(Animated\)](#)
- [Configure Graphics Objects](#)

### Symbol Set Properties - Appearance General (Array)

Symbol Sets have the following general appearance (Array) properties:

Property	Description
<b>Array expression</b>	<p>Enter the expression which is to return one or more integers. For each returned integer, a different symbol will be displayed.</p> <p>If the return value is:</p> <ul style="list-style-type: none"><li>• Less than 0 (zero), it will be set to 0 (zero), and a runtime hardware alarm will be triggered.</li><li>• Greater than 255, it will be set to 255, and a runtime hardware alarm will be triggered.</li><li>• A real (non-integer) number, it will be rounded off to the nearest integer.</li></ul> <p>To insert a tag or a function, click the <b>Wizard</b> button to the right of this field. This button displays two options; <b>Insert Tag</b>, and <b>Insert Function</b>.</p>
<b>Array symbol</b>	<p>The symbol that is to display for each integer returned by the Array expression entered above (symbol 0 will be used when the expression returns integer 0, symbol 1 will be used when integer 1 is returned etc.).</p> <p>Click the <b>Set</b> button to select a symbol, or <b>Clear</b> to clear the current selection.</p> <p>For example, to display different symbols illustrating the various states of a motor, you could fill out the <b>Array expression</b> and <b>Array symbol</b> fields as follows:</p>

Property	Description
	 <p>In this example, <b>MOTOR_STATUS</b> is an analog variable tag representing the status of a motor. Each time the motor changes state, an integer is returned (0 = Running, 1 = Starting etc.), and the appropriate symbol displays.</p>

Click **Clear Property** to clear property details, and disable the property. To define further properties for the object, click tabs.

## See Also

[Add a Symbol Set](#)

[Symbol Set Properties - Appearance General \(On/Off\)](#)

[Symbol Set Properties - Appearance General \(Multi-state\)](#)

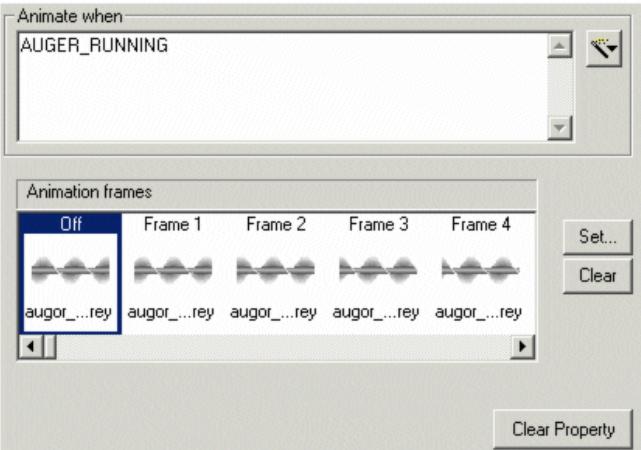
[Symbol Set Properties - Appearance General \(Animated\)](#)

[Configure Graphics Objects](#)

## Symbol Set Properties - Appearance General (Animated)

Symbol Sets have the following general appearance (Animated) properties:

Property	Description
<b>Animate when</b>	<p>Whenever this expression is true, the animation will run. Whenever the expression is false, the Off frame (below) will display.</p> <p>To insert a tag or a function, click the <b>Wizard</b> button to the right of this field. This button displays two options; <b>Insert Tag</b>, and <b>Insert Function</b>.</p>
<b>Animation frames</b>	<p>The symbols from Frame 1 onwards are those that will be used as the animation. They are displayed in sequence when the expression above is TRUE. The frequency at which the symbols are displayed is</p>

Property	Description
	<p>determined by the <a href="#">[Page]AnmDelay</a> parameter. The symbol in the Off frame will display when the expression above is FALSE.</p> <p>For example, to animate a running auger, you could fill out the <b>Animate when</b> and <b>Animation frames</b> fields as follows:</p>  <p>In this example, <b>AUGER_RUNNING</b>, is a variable tag which is TRUE when the auger is running. The symbols in the animation frames (Frame 1 onwards) have been designed so that when displayed in sequence, they animate a running auger. The symbol in the Off animation frame will display when <b>AUGER_RUNNING</b> is FALSE.</p>

Click **Apply** or **OK** to bring your changes into effect, or **Cancel** to discard them and exit. Click **Clear Property** to clear property details, and disable the property. To define further properties for the object, click the relevant tabs.

For help on the remaining properties tabs, see [Edit Common Object Properties](#).

## See Also

- [Add a Symbol Set](#)
- [Symbol Set Properties - Appearance General \(On/Off\)](#)
- [Symbol Set Properties - Appearance General \(Multi-state\)](#)
- [Symbol Set Properties - Appearance General \(Array\)](#)
- [Configure Graphics Objects](#)

## Add a Trend

The Trend tool allows you to add a trend to the graphics page with the mouse (click and drag).

After a trend object is drawn, it can be moved, re-sized, re-shaped, brought to the front etc., and its properties

can be edited, just like any other type of object.

### To add a trend to a page:

1. Click the **Trend** tool



or choose **Objects | Trend**.

2. Move the mouse to where you want the trend to start and click (and hold) the mouse button.
3. Drag the mouse to the opposite corner of the trend and release the mouse button.

The **Trend Properties** dialog appears. Assign the Trend Tags to the pens, choosing appropriate colors.

## Trend Properties

Trends have the following general appearance properties:

Property	Description
<b>Cluster Name</b>	<p>The name of the cluster that runs the trends being graphed. Each trend graph can only communicate with one cluster, so you cannot mix trends from multiple clusters on a single trend graph.</p> <p><b>Note:</b> To mix trends from different clusters on a single trend graph you will need to use Process Analyst.</p> <p>If there is only one cluster, or, the client is connected to only one cluster, this property can be left blank and the value of the single connected cluster is inferred.</p> <p>If the client is connected to more than one cluster, then this field needs to be specified.</p>
<b>Pens</b>	<p>The pens (including color) to be displayed on the graph (31 characters maximum). You can use up to eight pens.</p> <p>Double-clicking a selected pen or clicking <b>Edit</b> allows you to change the trend tag and pen color. To insert a trend tag, click <b>Wizard</b> to display the <a href="#">Insert Trend Dialog Box</a>.</p> <p>If more than one trend tag is displayed in a trend window and each has a different sample period, the trend with the smallest sample period is used as the general display period.</p> <p><b>Note:</b> If the trend object is part of a group, part of a pasted Genie or symbol, or part of the page's template, you can still access its properties: hold down the <b>Control</b> (CTRL) key and double-click the object.</p> <p>Alternatively, choose <b>Goto Object</b> from the Tools</p>

Property	Description
	<p>menu, click the object, then click <b>OK</b>. Be aware, however, that if it is part of a pasted Genie or symbol, or part of the template, you cannot edit existing pens, only new ones.</p> <p>If you are configuring an SPC control chart, you need to add a suffix to the trend tag to indicate the type of SPC. There are SPC templates that are easily configured through Genies. Use the Genies rather than defining these trend tags for yourself. The following table lists available SPC types:</p> <p>SPC Definition - SPC Type</p> <p>&lt;tag name&gt;.X - Mean of raw data in a subgroup (X - bar)</p> <p>&lt;tag name&gt;.XCL - Center line of X - bar</p> <p>&lt;tag name&gt;.XUCL - Upper control limit of X - bar</p> <p>&lt;tag name&gt;.XLCL - Lower control limit of X - bar</p> <p>&lt;tag name&gt;.R - Range of raw data in a subgroup (R - bar)</p> <p>&lt;tag name&gt;.RCL - Center line of R - bar</p> <p>&lt;tag name&gt;.RUCL - Upper control limit of R - bar</p> <p>&lt;tag name&gt;.RLCL - Lower control limit of R - bar</p> <p>&lt;tag name&gt;.S - Standard deviation of raw data in a subgroup (S - bar)</p> <p>&lt;tag name&gt;.SCL - Center line of S - bar</p> <p>&lt;tag name&gt;.SUCL - Upper control limit of S - bar</p> <p>&lt;tag name&gt;.SLCL - Lower control limit of S - bar</p> <p>Where &lt;tag name&gt; is any trend tag, for example:</p> <p>Pen 1 - PIC117_PV.XCL</p> <p>Pen 2 - PIC117_PV.XUCL</p> <p>Pen 3 - PIC117_PV.XLCL</p> <p>Pen 4 - PIC117_PV.X</p> <p>If you are using the PageTrend() function to display this trend page, leave these fields blank.</p>
<b>Display Trend Types as Periodic</b>	<p>When selected, enables trend pens (both periodic and event) to be displayed as periodic. Event and periodic trend data can then be displayed on the same graph. If this box is not selected, event and periodic pens have different styles and needs to be displayed on separate graphs.</p> <p><b>Note:</b> This option is set by default in the predefined</p>

Property	Description
	templates designed for use with periodic trends. It will only need to be enabled for customized templates.
<b>[Samples] Number of samples</b> (5 Chars.)	<p>The number of samples (1-32767) you can display in your trend window without scrolling (i.e., the width of your trend object). The default depends on the number of pixels per sample and your display resolution. The width of a trend object is equal to <b>Pixels per sample x Number of samples</b>.</p> <p><b>Note:</b> For a meaningful trend graph, make <b>Pixels per sample x Number of samples</b> less than the width of the display. For example, an XGA screen has a width of 1024 pixels. If you use 10 pixels per sample, 102 samples can be displayed on the screen without scrolling.</p>
<b>[Samples] Pixels per sample</b> (2 Chars.)	The display width of each sample. The width of a trend object is equal to <b>Pixels per sample x Number of samples</b> . The default is 1 pixel.

Click **Clear Property** to clear property details, and disable the property. To define other properties for the object, click the relevant tabs.

For help on the remaining properties tabs, see [Edit Common Object Properties](#).

## See Also

[Insert Trend Dialog Box](#)

[Configure Graphics Objects](#)

## Insert Trend Dialog Box

This dialog box lets you select a trend tag. To insert a trend tag, select the tag name, then click **OK**. The tag is inserted at the location of the cursor.

## See Also

[Add a Trend](#)

[Configure Graphics Objects](#)

## Add a Cicode Object

The Cicode Object tool allows you to add a Cicode Object to the graphics page with the mouse (click and drag).

A Cicode Object can be any command (such as a function and so on). When the graphics page is displayed at runtime, the command is run continually. Cicode objects can also be assigned a key sequence, allowing you to

enter keyboard commands when it is selected at runtime.

After a Cicode object is added, it can be moved and so on and its properties edited, just like other types of object.

### To add a Cicode Object to a page:

1. Click the **Cicode Object** tool,



or choose **Objects | Cicode Object**.

2. Move the mouse to where you want to add the object, and click the left mouse button.
3. Define the relevant properties for the object, and click **OK**.

### Cicode Object Properties - Cicode (General)

Cicode Objects have the following General properties:

Property	Description
<b>Command</b> (254 Chars.)	<p>A Cicode command that is continually executed. You can use any Cicode command, built-in Cicode function or user-written function. The command is executed continually (while the page is displayed), for example:</p> <p>Command :</p> <pre>DspSymAnm(25, "Pumps.Slurry1", "Pumps.Slurry2", "Pumps.Slurry3");</pre> <p>The command in this example uses the built-in function DspSymAnm(). The function displays three symbols ("Pumps.Slurry1", "Pumps.Slurry2", "Pumps.Slurry3") continually (at AN 25).</p> <p>You can also write generic functions by using the Cicode function DspGetAnCur() to get the AN number, for example:</p> <p>Command:</p> <pre>DspSymAnm(DspGetAnCur(), "Pumps.Slurry1", "Pumps.Slurry2", "Pumps.Slurry3");</pre> <p>The command in this example displays three symbols ("Pumps.Slurry1", "Pumps.Slurry2", "Pumps.Slurry3") continually (at the current AN).</p> <p>If you are using an actual animation, each symbol is displayed at a frequency that is set using the Setup Wizard (also determined by the <a href="#">[Page]AnmDelay</a> parameter). To add just an animation point to the page, add a Cicode Object, without a command.</p>

Click **Clear Property** to clear property details, and disable the property. To define other properties for the object, click the relevant tabs.

For help on the remaining properties tabs, see [Edit Common Object Properties](#).

## See Also

[Configure Graphics Objects](#)

### Paste a Symbol

The Paste Symbol tool allows you to insert a [Symbol](#) from a library onto the graphics page.



After a symbol is pasted using this tool, it can be moved, re-sized, reshaped, brought to the front and so on, and its properties edited just like any other type of object.

You can paste a symbol from the library on to a page in the following ways:

- As an **unlinked** symbol; the pasted symbol is not updated with changes to the symbol in the library.
- As a **linked** symbol; the symbol on the page is updated when the symbol in the library is changed (to alter the properties of a symbol in the library, open the library and edit it in the library). If you edit the symbol from the page and then change the source symbol in the library, the pasted symbol changes. For example, if you double the size of a pasted symbol, then double the size of the symbol in the library, the pasted symbol doubles again. You can remove the link to the library by using the **Edit | Cut Link** command.

#### To paste a symbol from the library to a page:

1. Click the **Paste Symbol** tool or choose **Edit | Paste Symbol**.
2. To paste a linked symbol, select the **Linked** check box. To paste an unlinked symbol, deselect the **Linked** check box.

#### To break the link:

1. Select the symbol.
2. Choose **Edit | Cut Link**.

To display the properties of the objects in the symbol (after pasting), hold the **Control** (CTRL) key down and double-click the specific object. Alternatively, choose **Tools | Goto Object**, click the object, then click **OK**.

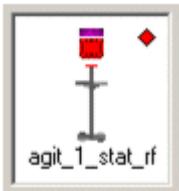
### Paste Symbol as Flashing

If you have selected **Paste Symbol as Flashing**, two dialog boxes appear in sequence, allowing you to choose two images that you want to implement as a flashing symbol. The **Primary Select Symbol** dialog allows you to select the initial image used, the **Flashing Select Symbol** dialog the second image. If the bitmaps are different in size, the flashing symbol is scaled to the size of the primary image. If one of the symbols is itself a flashing symbol, only the primary state will be displayed.

## Paste Symbol Dialog Box

This dialog box lets you paste a symbol from a library to the graphics page (or template).

## Paste Symbol Dialog Box Properties

Property	Description
<b>Symbol</b>	A table of symbols in the project. To add a symbol to a graphics page, use the scroll bar to locate the thumbnail image of the symbol, then select it and click <b>OK</b> (or double-click the thumbnail image). To edit the object in the library, select it and click <b>Edit</b> . To create a new symbol, click <b>New</b> . <b>Note:</b> If the symbol has a small diamond-shaped badge next to it, it indicates it is a flashing symbol (see example below.) 
<b>Library</b>	The library where the symbol is stored.
<b>Linked</b>	To paste a symbol that maintains the link with its library, check this box. A symbol that is linked will be automatically updated if the symbol in the library is changed. You can remove the link at any time with the <b>Cut Link</b> command from the Edit menu, but you cannot re-link a symbol with the library after this has occurred.

## See Also

[Symbol Properties - Appearance \(General\)](#)

### Symbol Properties - Appearance (General)

This dialog displays a picture of the selected symbol, name, and path. Click **Set** to change the symbol, or double-click the image. The Select Symbol dialog appears, letting you select a new symbol.

For help on the remaining properties tabs, see [Edit Common Object Properties](#).

#### To change the properties of an object in a pasted Symbol:

1. Click the **Select** tool.

2. Hold down the Control (CTRL) key and double-click the object.
3. Change the relevant properties in the dialog box. Alternatively, select **Tools | Goto Object**, select the object, and then click **OK**.

## See Also

[Paste a Symbol](#)

[Configure Graphics Objects](#)

## Add a Genie

The Paste Genie tool allows you to insert a Genie onto the graphics page.



See [Paste a Genie](#).

After a Genie is pasted using this tool, it can be re-sized, rotated, moved, copied, duplicated, pasted, brought to the front, and so on.

To display the properties of the objects in the Genie (after pasting), hold the Control (CTRL) key down and double-click the specific object.

## See Also

[Configure a Genie](#)

## Insert a Composite Genie

A Composite Genie can be inserted into any graphics page, template or Super Genie. It cannot be inserted into a Genie or symbol.

---

**Note:** The appearance of Composite Genies can be affected by different DPI settings. It is recommended that when running Graphics Builder, the **Scale and Layout** setting should be set to "100%" in the Windows™ **Display** settings. After changing the Scale and Layout setting for a computer, you should restart it.

---

### To insert a Composite Genie:

1. Open Graphics Builder.
2. Click the **Paste Composite Genie** button in the Objects toolbox.



The Open dialog box is displayed.

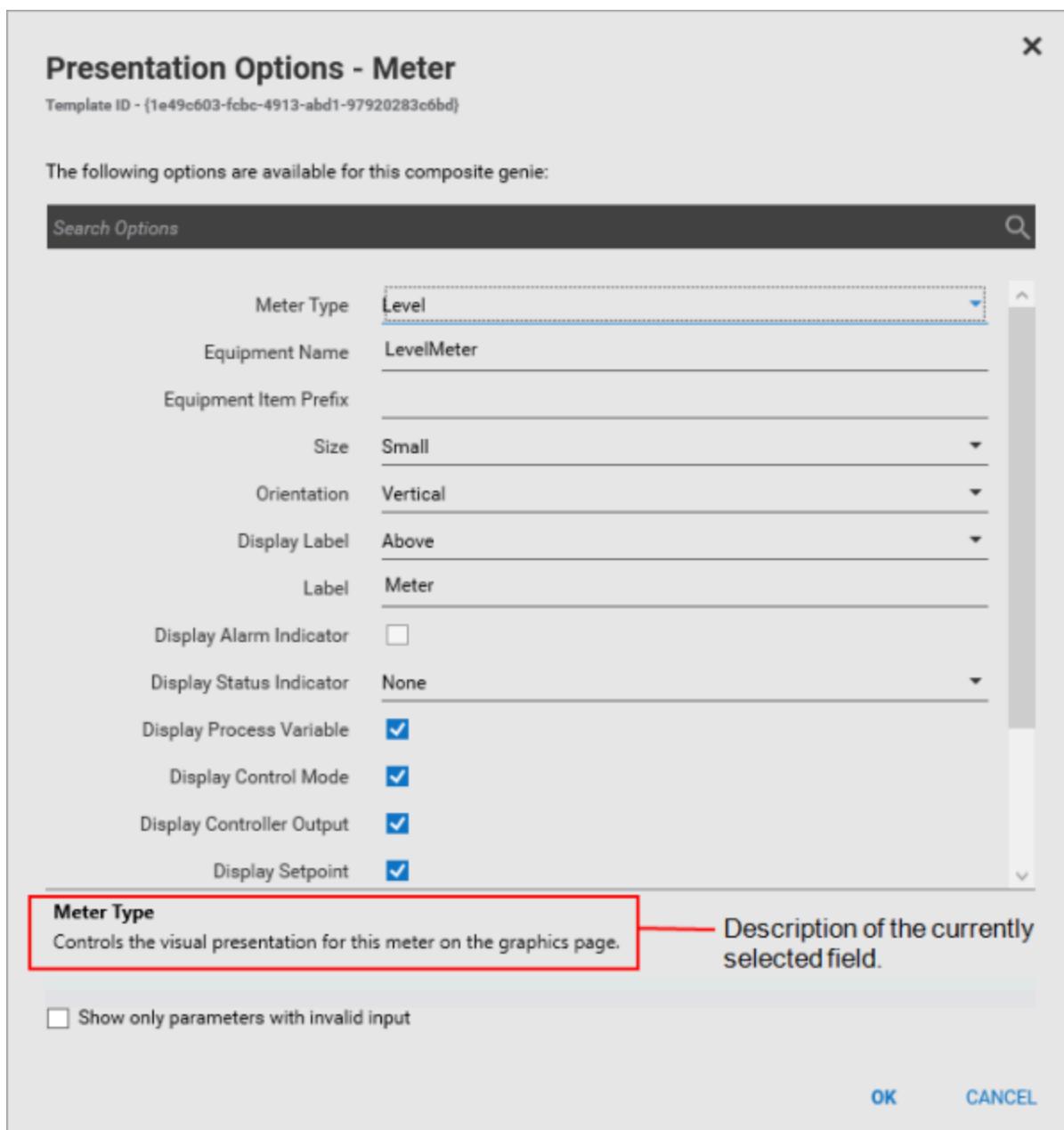
3. Select the XML template for the Composite Genie from the list of templates.

---

**Note:** The template file that you select needs to be located in the Composite Genies folder in the current project folder (or one of its included projects).

---

4. Click **OK**. The Presentation Options dialog box is displayed.



- Type or select the values for the parameters displayed.

**Note:** Depending upon the options you choose, some items may be hidden or displayed. For example, on the Meter Composite Genies, when you select the **Display Control Output** check box, the **Display Trend** and **Trend Type** options are displayed.

- Use the **Search** box to search for parameters in the dialog box. Type the parameter name or part of the parameter name to locate a parameter.

**Note:** Clear the Search box before clicking **OK** on the Presentation Options dialog box.

- Use the **Show only parameters with invalid input** option to display options for which incorrect or no values have been selected or specified.
- Click **OK** to insert the Composite Genie, or **Cancel** to close the Presentation Options dialog box without inserting the Composite Genie.

---

**Note:** You cannot rotate or mirror a Composite Genie that has been inserted on a page.

---

## See Also

[Composite Genies](#)

[Edit/Update a Composite Genie](#)

[Delete Unused Composite Genie Instances](#)

[Delete a Composite Genie](#)

## Add a Web Content Object

The Web Content tool allows you to add a browser window to a graphics page that displays web-based content at runtime.

Each instance of a Web Content object has a **URL** property that allows you to specify the content that is displayed. For more information about the supported functionality, see [Web Content](#).

After a browser window has been drawn on a page, it can be moved, re-sized, re-shaped, and so on. Its properties can be edited just like any other type of object.

---

**Note:** Web Content objects always appear on top other objects at runtime.

---

### To add a web content object to a page:

1. Click the **Web Content** tool in the Graphics Builder Toolbox.



Or:

From the **Objects** menu, select **Web Content**.

2. Move the mouse to where you want the browser to be located, then click (and hold) the mouse button.
  3. Drag the mouse to the opposite corner of the browser and release the mouse button.
- The **Web Content Properties** dialog appears.
4. Enter the **URL** you would like to display in the **Appearance (General)** properties (see below).

### **WARNING**

#### **LOSS OF CONTROL**

- Do not use the Plant SCADA Web Content control to display an untrusted or potentially unsafe web site, as this may leave your runtime system vulnerable to malicious activity.
- Do not enter a URL in the Web Content control that will allow users to navigate to an untrusted or potentially unsafe web site (for example, a search engine).

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

---

**Note:** Before you implement the Web Content control as part of your runtime system, you should carefully consider the security implications of the content that will be accessible. If using an internet-based website or

---

---

application, you will be externally exposing your SCADA system and network.

You can also adjust the following [Edit Common Object Properties](#):

- **Appearance (Visibility)** — see [Visibility](#).
  - **Movement (Horizontal)** — see [Object Properties - Movement \(Horizontal\)](#).
  - **Movement (Vertical)** — see [Object Properties - Movement \(Vertical\)](#).
  - **Scaling (Horizontal)** — see [Object Properties - Scaling \(Horizontal\)](#).
  - **Scaling (Vertical)** — see [Object Properties - Scaling \(Vertical\)](#).
  - **Access (General)** — see [General Access to Objects](#).
- The **Tooltip** and **Log Device** fields are disabled for Web Content objects.
- **Metadata (General)** — see [Metadata](#).

## Web Content Object Properties

A Web Content object will have the following General Appearance properties:

Property	Description
<b>URL</b>	Enter the URL for the web content you would like the object to display at runtime (maximum 2048 characters). You need to use the full URL, including the protocol and subdomain. For example: <code>https://www.aveva.com/</code> You also need to enter a static URL as expressions are not supported. If this property is set using the Cicode function <a href="#">DspWebContentSetURL</a> , then a 255 character limit will apply.
<b>Enable 'Print' in context menu</b>	Select this option if you like to enable printing from the browser's context menu.

## See Also

[Configure Graphics Objects](#)

## Add an ActiveX Object

You can incorporate ActiveX objects into your project. This means you can use components that have been developed independently of Plant SCADA.

The ActiveX tool can be used to insert ActiveX objects in graphics pages. After you have selected and positioned an ActiveX object, it can be moved, re-sized, re-shaped, brought to the front etc., and its properties can be edited, just like any other object.

## ActiveX Object Properties

The two properties tabs common to ActiveX objects are the **Tag Association** and **Visibility** tabs which appear vertically on the **Appearance** tab. The content and number of other tabs depends on the individual design of each ActiveX object. This is determined by the amount of flexibility and support its creator has included.

For details on configuring these additional tabs, refer to the documentation provided with the ActiveX object.

**Note:** The Insert an ActiveX Control windows may include the Pelco™ Viewer Camera Control. This ActiveX control has been deprecated and is no longer part of the Plant SCADA installation. If you have upgraded your computer from a previous version with this control installed, it will continue to function. However, it is recommended that you communicate with Pelco cameras using a different method.

## See Also

[Tag Association](#)

[Managing Associated Data Sources](#)

[Object Identification](#)

### Tag Association

You can create an association between a property of an ActiveX object and a variable tag.

**To create an association between a property of an ActiveX object and a variable tag:**

1. Double-click the ActiveX object. The Properties dialog box appears.
2. Click the **Tag Association** tab.
3. Select a property from the **Properties** list.
4. Click the **Wizard | Insert Tag** button.
5. Select a tag from the list and click **OK**.

### ActiveX Object Properties - Appearance (Tag Association)

ActiveX objects have the following appearance properties:

Property	Description
<b>Properties</b>	The "properties" of an ActiveX object relate to the elements that define the object's functionality and appearance. The available properties for a selected ActiveX object are listed here, as defined by the object's creator.  The check box to the left of each item on the list indicates whether or not a tag has been associated with the Property. If the box is checked, it means an association has already been defined for the property. If you want to clear this tag association, simply uncheck the box, or clear the tag from the "Associate

Property	Description
	property with tag . . ." field.
<b>Associate property with tag</b>	<p>You can create an association between an ActiveX object property and a variable tag so that changing values in one are reflected in the other. To create an association, you need to first choose a property from the property list. The label <b>Associate property with tag</b> will change to display the property name.</p> <p>Select the variable tag you would like to associate with a property by clicking on the Wizard button and choosing from the list of available tags. Alternatively, the tag name can be typed in to the <b>Associate property with tag</b> field.</p> <p><b>Note:</b> You can only use variable tag names in this field. Functions, expressions and constants are not supported when defining ActiveX property tag associations.</p> <p>You can only associate a property with a variable tag if the tag type is compatible with the property. To display a list of compatible tag types, select the property, and click <b>List Property Types</b>. A list of compatible data types will display, or a message will inform you that there are no compatible types.</p> <p>If there are no types compatible with the property, the <b>Associate Property with tag</b> and <b>Update association on</b> fields will be disabled.</p> <p>If you specify a tag which might be inappropriate for the selected ActiveX property, the <b>Type Evaluation dialog</b> will display an alert. This might happen if:</p> <ul style="list-style-type: none"> <li>• The types compatible with the property are different from the tag type.</li> <li>• The tag type is smaller than the types compatible with the property, meaning that data could be truncated or lost.</li> </ul>
<b>Bindable</b>	<p>If an object property is "bindable", it means it can automatically send notification of value changes to Plant SCADA, and acknowledge any value changes from an associated tag. This means both the property and an associated tag will automatically update whenever the value for either changes.</p> <p>If a property is not bindable, the property/tag association can only be synchronized according to the</p>

Property	Description
	<p>event selected in the <b>Update association on Event</b> field.</p> <p>Be aware that if an object property is bindable, the <b>Update association on Event</b> field will be automatically set to &lt;Property change notification&gt; by default. You can change this setting if you want the tag association to be updated by a more specific event.</p> <p>For properties that are not bindable, you can mimic the behavior of &lt;property change notification&gt; by selecting <b>After update</b> for the <b>Update association on Event</b> field.</p>
<b>Property status</b>	<p>This indicates the read/write status of the selected property. With ActiveX objects, some properties are read-only, whereas others accept value changes. If a property is marked "read only", it indicates that its value is fixed and can only be read by Plant SCADA. If its status is read/write, can modify the property during runtime via a tag association.</p>
<b>Update association on Event:</b>	<p>This defines when you want a value update to occur for the selected property and its associated tag. Use the menu to the right of the <b>Event</b> field to view events that can be used to trigger a tag association update.</p> <p>The available events that can be used with a particular property are predefined by an object's creator. They typically include user interaction events (for example, mouse clicks), time events (such as a new day or new month), or value changes (such as "after update").</p> <p><b>Note:</b> If you are associating multiple ActiveX object properties with the same variable tag, then only one property should be configured with an update event. The Runtime will create only one event association per variable tag in that ActiveX object.</p>
<b>Property documentation</b>	<p>Most ActiveX objects have documentation describing the object's controls and functionality. Some have a separate Help file included, others, simple text prompts to explain each property. This depends on what an object's creator has included.</p> <p>The Property documentation field displays Help information for a selected property, or give instructions to obtain the Help necessary for a selected property. Usually the following message will appear:</p>

Property	Description
	"Click the '?' button to the right to display the Help topic for this property" The <b>Help</b> button displays the ActiveX object Help file (if included), usually with the topic displayed that relates to the selected property. It will also provide information about settings provided on additional tabs the ActiveX object might call up on the properties dialog.

Click the **Clear Property** button to clear the tag associations for each of the ActiveX object properties. To clear a tag association for a particular object property, clear the check box to the left of the property.

If you accidentally click **Clear Property**, you can restore your tag associations by clicking **Cancel** and reopening the ActiveX properties dialog.

## See Also

[Configure Graphics Objects](#)

### Managing Associated Data Sources

If an ActiveX object has an association with a data source (for example, it stores data to a DBF file), you need to consider the impact of running a project that contains it on a different machine.

If the path to the data source is hardcoded to a location on the local machine, the data source will not be found if the project is moved or run remotely. For example, the Database Exchange ActiveX control connects to a recipe.DBF file in the project path. If you restore a project that uses it on a different computer with a different installation path, you will need to recreate the data source to retrieve any recipes.

### **WARNING**

#### UNINTENDED EQUIPMENT OPERATION

- Whenever possible, avoid programming the literal paths to data sources in your project.
- When a project or a data source is relocated to a different computer, examine the program and program objects and correct any literal paths before placing the project back into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

One way to avoid having to recreate data sources is to locate any associated data sources in a central location on a network. For example, if the data source is located on a SQL server, it will be accessible from every machine on the common network.

#### To insert an ActiveX Control:

1. Click the **ActiveX** tool,



or choose **Edit | Insert ActiveX Control**.

2. Select an ActiveX Control and click **Insert**.

## See Also

[Tag Association](#)

[Object Identification](#)

### Object Identification

You can identify your ActiveX object.

#### To identify your ActiveX object:

1. From Graphics Builder, double-click the ActiveX object. The Properties dialog appears.
2. Click the **Access** tab.
3. Click the **Identification** tab.
4. Assign your ActiveX object a name in the **Object Name** field.
5. Assign your ActiveX object an **Event Class**.
6. Click **OK**.

### Object Properties - Access (Identification)

Objects have the following Access Identification properties:

Property	Description
<b>[Identification] Object Name</b>	This field allows you to assign a name to your ActiveX object (254 characters maximum). It identifies the object when using the <a href="#">ObjectByName()</a> , <a href="#">AnByName()</a> , and <a href="#">CreateControlObject()</a> Cicode functions.  The name can be any combination of alpha or numeric characters.
<b>[Identification] Event Class</b>	Allocate a name for the event class of your ActiveX object (16 characters). You can then use this name to create a Cicode function to trap an event.  Don't change the default value if you want to access the ActiveX object using VBA. If you do, VBA can't access the object. If the Event Class is changed, you can reset it to the default value by clicking <b>Clear Property</b> .
<b>[Persistence] Persist ActiveX data between page</b>	Select this check box to allow changes made to the

Property	Description
<b>transitions</b>	control to be persisted between pages. For example, you can select this check box if you want an ActiveX edit control to keep the current text in the control, so that next time the page is entered, the same text is displayed.

**Note:**

- An ActiveX control may or may not implement data persistence in the manner you expect. Some controls will not persist certain data, and therefore you will not be able to save and restore that data.
- Even when an ActiveX control implements data persistence in an expected manner, be advised that data is only persisted for the current session. If Plant SCADA runtime is shut down and restarted the updated data is not available.

## See Also

[Add an ActiveX Object](#)

### Add a Database Exchange Control

The Database Exchange ActiveX control lets you connect to a data source (for example a database), and extract, display, and edit recipe values.

To use the Database Exchange control, you need to insert it on a graphics page. At runtime, values are displayed in a table. Each column of data can be associated with a Plant SCADA variable tag, enabling recipe values to be written to tags. Data can also be sorted, filtered, printed, edited, and written back to the data source.

Plant SCADA supports connection from the Database Exchange control to all Windows-supported databases, including DBF, MDB, SQL Server, and OLEDB data sources.

You can access more information about the Database Exchange control by pasting it on to a graphics page and clicking the **Help** button on the associated Object Properties dialog.

### Import Graphics

The Graphics Builder has several file format filters to allow you to import graphics from other applications, such as drafting programs, illustration programs, presentation packages, scanners, etc. After a graphic is imported, you can use the Graphics Builder to edit the image.

Graphics files can be dragged from a third-party application (such as Windows Explorer), and dropped onto a page in the Graphics Builder.

You use the Import dialog box to import a graphic produced with a different application.

If you have selected **Import as Flashing**, two Import dialog boxes appear in sequence, allowing you to choose two images that you want to implement as a flashing symbol. The **Import Primary** dialog allows you to select the initial image used; the **Import Flashing** dialog allows you to choose the second images used.

### Import Graphics Dialog

Field	Description
Look in	The drive and directory where the graphic is stored.
File Name	The name of the graphics file.
Files of Type	<p>The type of graphics file. The following file formats are supported:</p> <ul style="list-style-type: none"> <li>• Windows bitmaps (*.BMP, *.DIB, *.RLE)</li> <li>• PCX format bitmaps (*.PCX)</li> <li>• Text files (*.TXT)</li> <li>• AutoCAD DXF Files (*.DXF) 2D only (the binary format is also supported)</li> <li>• Windows metafiles (*.WMF)</li> <li>• Fax Image (*.FAX)</li> <li>• Ventura Image (*.IMG)</li> <li>• JPEG (*.JPG, *.JIF, *.JFF, *.JGE)</li> <li>• Photo CD (*.PCD)</li> <li>• Portable Network Graphic (*.PNG). (PNG files with alpha channels are not supported)</li> <li>• Targa (*.TGA)</li> <li>• Tagged Image Format (*.TIF)</li> <li>• WordPerfect (*.WPG)</li> <li>• Encapsulated Postscript (*.EPS)</li> </ul> <p><b>Note:</b> For EPS graphics, only the TIFF or WMF preview in the EPS file header is imported (if available).</p>

## Edit Common Object Properties

This section describes properties that are common to several object types. Some are also common to object groups.

- [Appearance](#)
- [3D Effects](#)
- [Visibility](#)
- [Movement](#)
- [Scaling](#)
- [Fill](#)
- [Fill Color](#)
- [Fill Level](#)

- [Object Touch Commands](#)
- [Object Keyboard Commands](#)
- [Sliders](#)
- [Access](#)
- [Metadata](#)
- [Alarm Indicator.](#)

## Appearance

Click the **Appearance** tab to define the appearance of the object, such as line style, and shadowing. You can also specify when the object will be hidden from the operator (for example when DIGITAL\_TAG is OFF).

The check mark to the left of the Appearance tab tells you when an appearance property has been configured. The check marks in the tabs down the right of the dialog indicate which property is configured.

## See Also

- [3D Effects](#)
- [Visibility](#)
- [Edit Common Object Properties](#)

## 3D Effects

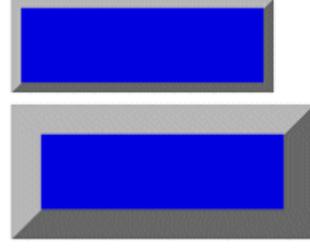
You can apply 3D effects to objects to make them more realistic.

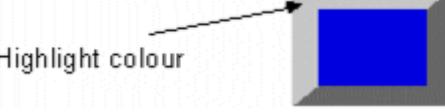
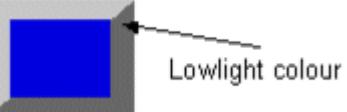
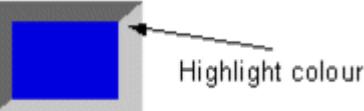
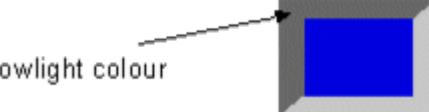
### To apply 3D effects to an object:

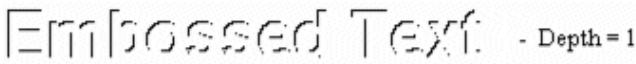
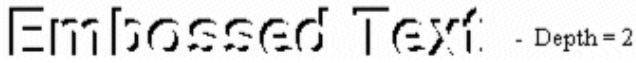
1. Draw the object/group (or paste a symbol). The properties tab dialog will automatically display, unless you have turned off the Display properties on new option in the Graphics Builder. (For a group, the properties dialog will not display automatically; you need to double-click the group).
2. Click the **Appearance** tab.
3. Click the **3D Effects** tab (to the right of the dialog).
4. The dialog will then display several options to enable you to manipulate your object. To activate any of these options click the option (or the radio bullet to the left of the option).
5. By selecting certain options additional fields will display to enable you to further manipulate your object. Enter further details as necessary, using the Help button for detailed information about each field.
6. Click **OK**.

### Object Properties - Appearance (3D Effects)

Objects have the following 3D Effects properties.

Property	Description
[Effects] None	Select this option to display the object without any special effects (such as shadowing, embossing and so on).
[Effects] Shadowed	Select this option to display the object with a shadow; for example: 
Depth	The distance (in pixels) that the shadow extends below and to the right of the object. This option alters the apparent distance between the object and its shadow, for example:  <ul style="list-style-type: none"><li>- Depth = 5</li><li>- Depth = 15</li></ul>
Shadow color	The color of the shadow. The shadow color will not change dynamically with runtime conditions.
[Effects] Raised	Select this option to display the object as a raised three dimensional solid, for example: 
Depth	The distance (in pixels) that the sides of the object extend out from the raised surface. This option alters the apparent distance from the raised surface down to your graphics page, for example:  <ul style="list-style-type: none"><li>- Depth = 5</li><li>- Depth = 15</li></ul>
Highlight color	The color of the directly illuminated "edges" of the object.

Property	Description
	 <p>Highlight colour</p>
<b>Lowlight color</b>	<p>The color of the "edges" of the object that are in shadow.</p>  <p>Lowlight colour</p>
<b>[Effects] Lowered</b>	<p>Select this option to display the object as if it is actually lower than your graphics page, for example:</p> 
<b>Depth</b>	<p>The distance (in pixels) that the sides of the object extend out from the lowered surface. This option alters the apparent distance from the lowered surface up to your graphics page, for example:</p>  <p>- Depth=5</p>  <p>- Depth=15</p>
<b>Highlight color</b>	<p>The color of the directly illuminated "edges" of the object.</p>  <p>Highlight colour</p>
<b>Lowlight color</b>	<p>The color of the "edges" of the object that are in shadow.</p>  <p>Lowlight colour</p>
<b>[Effects] Embossed</b>	<p>Select this option to display the object as if it has been embossed on your graphics page, for example:</p>

Property	Description
	<b>Embossed Text</b>
<b>Depth</b>	The distance (in pixels) that the embossed surface is lowered. This option alters the apparent distance from the embossed surface up to your graphics page, for example:  
<b>Highlight color</b>	The color of the right and lower edges of the object. 
<b>Lowlight color</b>	The color of the left and upper edges of the object. 

Click **Apply** or **OK** to save your changes, or **Cancel** to exit. To define further properties for the object, click the relevant tabs.

## See Also

[Edit Common Object Properties](#)

## Visibility

You can determine whether an object is visible or not.

### To hide/unhide an object:

1. Double click the object you would like to hide.
2. Select the **Appearance** tab.
3. Select the **Visibility** tab (to the right of the dialog).
4. Click the **Wizard** button to the right of the **Hidden when** field.
5. Select either **Insert Tag** or **Insert Function** depending on which you would like to relate to your object.
6. Enter an expression in the **Hidden when** field. When this expression is true your object will be hidden.
7. Click **OK**.

## Object Properties - Appearance (Visibility)

Objects and groups have the following visibility properties:

Property	Description
<b>Hidden when</b>	<p>The object/group will be hidden whenever the expression entered in this field is TRUE. Enter an expression of 254 characters or less. For example, if you want the object/group to be hidden for every operator except the superintendent, you could enter the following:</p> <p>NOT GetPriv( _Super, _SectionA ) where _Super and _SectionA are labels.</p> <p>To insert a tag or a function, click the <b>Wizard</b> button to the right of this field. This button displays two options: Insert Tag and Insert Function.</p> <p><b>Note:</b> You can also insert Equipment.item references into expression fields using the <b>insert tag</b> option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with available tags.</p> <p><b>Note:</b> If a group is hidden, every object (and other groups) in the group will also be hidden regardless of their individual properties. If the group is visible, its objects will behave according to their individual properties.</p>

Click **Clear Property** to clear property details and disable the property.

## See Also

[Edit Common Object Properties](#)

## Movement

You can move an object vertically or horizontally, or to rotate it, depending on the return of an expression or the state of a tag.

### To configure an object or group that moves:

1. Draw the object/group (or paste a symbol). The properties tab dialog automatically display, unless you have turned off the **Display properties on new** option in the Graphics Builder. (For a group, the properties dialog will not display automatically; you need to double-click the group.)
2. Click the **Movement** tab.
3. Click the **Horizontal**, **Vertical** or **Rotational** tab (to the right of the dialog).

4. Enter a Movement **expression** (the expression that will move the object/group at runtime).
5. Enter further details as necessary, using the **Help** button for detailed information about each field.
6. Click **OK**.

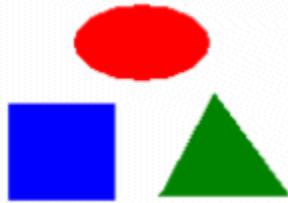
## Group and Object Movement - Examples

A group and its objects can be configured with any combination of movement (horizontal, vertical, and rotational). The following examples illustrate how some of these combinations work.

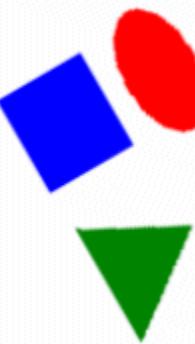
### Example 1: Rotating the group and moving the object left to right

If your group is configured to rotate from 0 degrees to 60 degrees, and one of its objects is configured to move left and right, the object will do both. It will move left and right as per its own properties, and, at the same time, it will rotate as part of the group. Remember, however, that 'left' and 'right' are relative to the original orientation of the group, not the page. As the group rotates, 'horizontal' also rotates. When the group has rotated 15 degrees, 'left' is actually 285 degrees; (not 270 degrees), and 'right' is actually 105 degrees (not 90 degrees). When the group has rotated 50 degrees, 'left' is 320 degrees, and 'right' is 140 degrees, and so on.

#### Original state of group



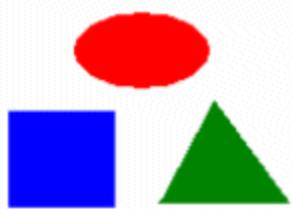
#### Group rotated right, and ellipse moved left



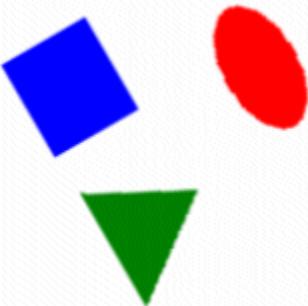
### Example 2: Rotating the group and moving the object up and down

If your group is configured to rotate from 0 degrees to 60 degrees, and one of its objects is configured to move up and down, the object will do both. It will move up and down as per its own properties, and, at the same time, it will rotate as part of the group. Remember, however, that 'up' and 'down' are relative to the original orientation of the group, not the page. As the group rotates, 'vertical' also rotates. When the group has rotated 15 degrees, 'up' is actually 15 degrees (not 0 degrees), and 'down' is actually 195 degrees (not 180 degrees). When the group has rotated 50 degrees, 'up' is 50 degrees, and 'down' is 230 degrees, and so on.

#### Original state of group



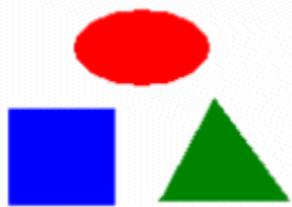
Group rotated right, and ellipse moved up



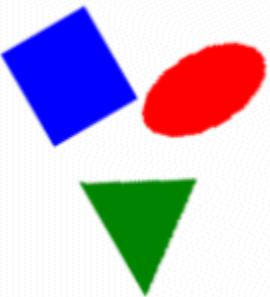
### Example 3: Rotating the group clockwise and rotating the object anticlockwise

If your group is configured to rotate from 0 degrees-60 degrees, and one of its objects is configured to rotate from 90 degrees-0 degrees, the object will do both. It will rotate as per its own properties, and, at the same time, it will rotate as part of the group. Remember, however, that the object's rotation is relative to the group, not the page. If the group rotates 60 degrees to the right, and the object rotates 90 degrees to the left, the object has only rotated 30 degrees to the left relative to the page.

Original state of group



Group rotated clockwise, and ellipse rotated anticlockwise



**Note:** By moving the ellipse as shown in the above examples, you are actually changing the overall size of the

group. You need to remember this as it might affect object fill levels.

## See Also

- [Object Properties - Movement \(Horizontal\)](#)
- [Object Properties - Movement \(Vertical\)](#)
- [Object Properties - Movement \(Rotational\)](#)
- [Edit Common Object Properties](#)

### Object Properties - Movement (Horizontal)

Objects and groups can be moved from side to side during runtime, changing dynamically whenever the value of a particular expression changes. By default, as the value of the expression increases, the object/group will move (in increments) to the right. As the value of the expression decreases, the object/group will move (in increments) to the left.

This property could, for example, be used to display the position of a coal stacker moving along a stockpile.

**Note:** Horizontal movement cannot be used if the horizontal slider is enabled. A group and its objects can be configured with any movement combination (i.e., a group can move vertically while one of its objects rotates, and so on).

## Horizontal Movement Properties

Objects and groups have the following horizontal movement properties:

Property	Description
<b>Movement expression</b>	<p>The value of the expression entered in this field (253 characters maximum) will determine the horizontal movement of the object/group. By default, when the expression returns its minimum value, the object/group will shift hard to the left. When the expression returns its maximum, the object/group will shift hard to the right. For intermediate values, the object/group will move to the appropriate position between the minimum and maximum offset.</p> <p>To insert a tag or a function, click the <b>Wizard</b> button to the right of this field. This button displays two options: Insert Tag and Insert Function.</p> <p><b>Note:</b> You can also insert Equipment.item references into expression fields using the <b>insert tag</b> option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with available tags.</p>
<b>[Movement expression] Specify range</b>	<p>Select this box to manually specify <b>Minimum</b> and <b>Maximum</b> values for the movement expression, rather than using the default values. For an expression containing an analog variable tag, the defaults are the Engineering Zero and Full Scale values from the last variable tag in the expression. If the analog variable tag does not have Engineering Zero and Full Scale values, the defaults are 0 (zero) and 32000. For expressions without tags, the defaults are 0 (zero) and 100.</p>
<b>[Movement expression] Minimum</b>	<p>Enter the minimum value for the expression. When this value is returned by the expression, the object/group will shift to the left, by the <b>At minimum</b> offset. You can only enter a value here if you have selected the <b>Specify</b> range box.</p>
<b>[Movement expression] Maximum</b>	<p>Enter the maximum value for the expression. When this value is returned by the expression, the object/group will shift to the right, by the <b>At maximum</b> offset. You can only enter a value here if you have selected the <b>Specify</b> range box.</p>
<b>[Offset] At minimum</b>	<p>The distance (number of pixels from the original object/group center) that the object/group will shift to the left when the <b>Movement expression</b> returns its minimum value.</p>

Property	Description
	You can change the offset value by pressing the up and down arrows to the right of the field, or by entering another value in this field.
<b>[Offset] At maximum</b>	<p>The distance (number of pixels from the original object/group center) that the object/group will shift to the right when the <b>Movement expression</b> returns its maximum value.</p> <p>You can change the offset value by pressing the up and down arrows to the right of the field, or by entering another value in this field.</p> <p><b>Note:</b> You can shift the object/group right at minimum and left at maximum, by entering negative distances in the Offset fields, or by swapping the expression limits (in the <b>Minimum</b> and <b>Maximum</b> fields).</p>

Click **Clear Property** to clear property details, and disable the property.

## See Also

[Object Properties - Movement \(Vertical\)](#)

[Object Properties - Movement \(Rotational\)](#)

### Object Properties - Movement (Vertical)

Objects and groups can be moved up and down during runtime, changing dynamically whenever the value of a particular expression changes. By default, as the value of the expression increases, the object/group will move up (in increments). As the value of the expression decreases, the object/group will move down (in increments).

This property could be used to display the movement of an elevator.

---

**Note:** Vertical Movement cannot be used if the Vertical Slider is enabled. A group and its objects can be configured with any movement combination (i.e. a group can move vertically while one of its objects rotates, and so on).

---

## Vertical Movement Properties

Objects and groups have the following vertical movement properties:

Property	Description
<b>Movement expression</b>	The value of the expression entered in this field (253 characters maximum) will determine the vertical movement of the object/group. By default, when the expression returns its minimum value, the object/group will shift down to its lowest position. When the expression returns its maximum, the object/group will

Property	Description
	<p>shift up to its highest position. For intermediate values, the object/group will move to the appropriate position between the minimum and maximum offset.</p> <p>To insert a tag or a function, click the <b>Wizard</b> button to the right of this field. This button displays two options: Insert Tag and Insert Function.</p> <p><b>Note:</b> You can also insert Equipment.item references into expression fields using the <b>insert tag</b> option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with available tags.</p>
<b>[Movement expression] Specify range</b>	<p>Select this box to manually specify <b>Minimum</b> and <b>Maximum</b> values for the movement expression, rather than using the default values. For an expression containing an analog variable tag, the defaults are the Engineering Zero and Full Scale values from the last variable tag in the expression. If the analog variable tag does not have Engineering Zero and Full Scale values, the defaults are 0 (zero) and 32000. For expressions without tags, the defaults are 0 (zero) and 100.</p>
<b>[Movement expression] Minimum</b>	<p>Enter the minimum value for the expression. When this value is returned by the expression, the object/group will shift down, by the <b>At minimum</b> offset. You can only enter a value here if you have selected the <b>Specify range</b> box.</p>
<b>[Movement expression] Maximum</b>	<p>Enter the maximum value for the expression. When this value is returned by the expression, the object/group will shift up, by the <b>At maximum</b> offset. You can only enter a value here if you have selected the <b>Specify range</b> box.</p>
<b>[Offset] At maximum</b>	<p>The distance (number of pixels from the original object/group center) that the object/group will shift up when the <b>Movement expression</b> returns its maximum value.</p> <p>You can change the offset value by pressing the up and down arrows to the right of the field, or by entering another value in this field.</p>
<b>[Offset] At minimum</b>	<p>The distance (number of pixels from the original object/group center) that the object/group will shift down when the <b>Movement expression</b> returns its</p>

Property	Description
	<p>minimum value.</p> <p>You can change the offset value by pressing the up and down arrows to the right of the field, or by entering another value in this field.</p> <p><b>Note:</b> You can shift the object/group up at minimum and down at maximum, by entering negative distances in the Offset fields, or by swapping the expression limits (in the <b>Minimum</b> and <b>Maximum</b> fields).</p>

## See Also

[Object Properties - Movement \(Horizontal\)](#)

[Object Properties - Movement \(Rotational\)](#)

### Object Properties - Movement (Rotational)

Objects and groups can be dynamically rotated during runtime, whenever the value of a particular expression changes. By default, as the value of the expression increases, the object/group will rotate clockwise (in increments). As the value of the expression decreases, the object/group will rotate anti-clockwise (in increments).

This property could be used to display an aerial view of the movement of a circular stacker in a coal mining operation.

---

**Note:** Rotational Movement cannot be used if the Rotational Slider is enabled. A group and its objects can be configured with any movement combination (i.e. a group can move vertically while one of its objects rotates, and so on).

## Rotational Movement Properties

Objects and groups have the following rotational movement properties:

Property	Description
<b>Angle expression</b> (253 Chars.)	<p>The value of the expression entered in this field will determine the rotation of the object/group. During runtime, when the expression returns its minimum value, the object/group will rotate to its anti-clockwise limit. When the expression returns its maximum, the object/group will rotate to its clockwise limit. For intermediate values, the object/group will rotate to the appropriate position between the minimum and maximum limits.</p> <p>To insert a tag or a function, click the <b>Wizard</b> button to the right of this field. This button displays two options: Insert Tag and Insert Function.</p>

Property	Description
	<p><b>Note:</b> You can also insert Equipment.item references into expression fields using the <b>insert tag</b> option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with available tags.</p>
<b>[Angle expression] Specify range</b>	<p>Select this box to manually specify <b>Minimum</b> and <b>Maximum</b> values for the angle expression, rather than using the default values. For an expression containing an analog variable tag, the defaults are the Engineering Zero and Full Scale values from the last variable tag in the expression. If the analog variable tag does not have Engineering Zero and Full Scale values, the defaults are 0 (zero) and 32000. For expressions without tags, the defaults are 0 (zero) and 100.</p>
<b>[Angle expression] Minimum</b>	<p>Enter the minimum value for the expression. When this value is returned by the expression, the object/group will rotate anti-clockwise, by the minimum offset. You can only enter a value here if you have selected the <b>Specify</b> range box.</p>
<b>[Angle expression] Maximum</b>	<p>Enter the maximum value for the expression. When this value is returned by the expression, the object/group will rotate clockwise, by the maximum offset. You can only enter a value here if you have selected the <b>Specify</b> range box.</p>
<b>[Angle] At minimum</b>	<p>The anti-clockwise angle (in degrees relative to 0 degrees) that the object/group will rotate when the <b>Angle expression</b> returns its minimum value. You can change the angle by pressing the up and down arrows to the right of the field, or by entering another value in this field.</p>
<b>[Angle] At maximum</b>	<p>The clockwise angle (in degrees relative to 0 degrees) that the object/group will rotate when the <b>Angle expression</b> returns its minimum value. You can change the angle value by pressing the up and down arrows to the right of the field, or by entering another value in this field. <b>Note:</b> You can rotate the object/group clockwise at minimum and anti-clockwise at maximum, by entering negative angles in the Angle fields, or by swapping the expression limits (in the <b>Minimum</b> and <b>Maximum</b></p>

Property	Description
	fields).
<b>[Center axis offset] Express</b>	Click this radio button for the quick and easy way of selecting the point about which the object/group will rotate. The express option gives you the choice of 9 points (Top Left, Bottom Right and so on), which are displayed in the picture field to the right of the dialog. To select one, just click it with your mouse.
<b>[Center axis offset] Custom</b>	<p>Click this radio button to define your own center axis. When you select this radio button, two fields will display to the right, allowing you to plot the position of your center axis. Specify the distance to the right in the first field, and the distance down in the second. The Center axis is plotted based on these two values.</p> <p>For example, if you enter 8 as the horizontal offset, and 13 as the vertical offset, the Center axis will be 8 pixels to the right, and 13 pixels below the center of the object/group.</p> <p>Enter negative values in the offset distance fields to move the Center axis left instead of right, and up instead of down.</p>

## See Also

[Object Properties - Movement \(Horizontal\)](#)

[Object Properties - Movement \(Vertical\)](#)

## Scaling

You can scale objects both vertically or horizontally, depending on the return of an expression or the state of a tag.

### To configure an object or group that changes size:

1. Draw the object (or paste a symbol). The object properties tab dialog will automatically display, unless you have turned off the **Display properties on new** option in the Graphics Builder.
2. Click the **Scaling** tab.
3. Click the **Horizontal** or **Vertical** tab (to the right of the dialog).
4. Enter a **Scaling expression** (the expression that will change the size of the object at runtime).
5. Enter further object property details as necessary, using the **Help** button for detailed information about each field.
6. Click **OK**.

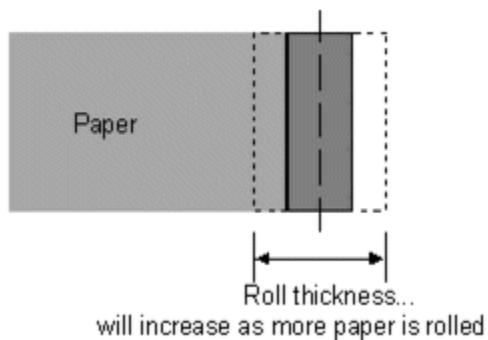
## See Also

- [Object Properties - Scaling \(Horizontal\)](#)
- [Object Properties - Scaling \(Vertical\)](#)
- [Edit Common Object Properties](#)

### Object Properties - Scaling (Horizontal)

The width of an object can be dynamically changed during runtime whenever the value of a particular expression changes. As the value of the expression increases and decreases, the width of the object increases or decreases accordingly as a percentage of the original width; that is, when it was added to the graphics page.

For example, an aerial view of a paper roll (in a paper mill), could display changing roll thickness:

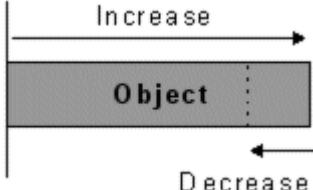
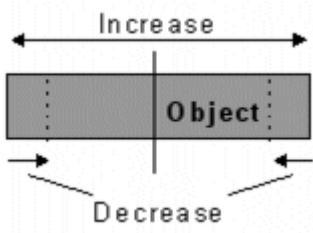
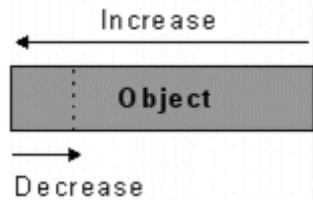


### Horizontal Scaling Properties

Objects have the following horizontal scaling properties:

Property	Description
<b>Scaling expression</b>	<p>The value of the expression entered in this field (253 characters maximum) will determine the horizontal scaling (width) of the object. By default, when the expression returns its minimum value, the object will display at its minimum width (as defined in the Scaling fields below). When the expression returns its maximum value, the object will display at its maximum width (as defined in the Scaling fields below).</p> <p>To insert a tag or a function, click the <b>Wizard</b> button to the right of this field. This button displays two options: Insert Tag and Insert Function.</p> <p><b>Note:</b> You can also insert Equipment.item references into expression fields using the <b>insert tag</b> option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with</p>

Property	Description
	available tags.
<b>[Scaling expression] Specify range</b>	Select this box to manually specify <b>Minimum</b> and <b>Maximum</b> values for the scaling expression, rather than using the default values. For an expression containing an analog variable tag, the defaults are the Engineering Zero and Full Scale values from the last variable tag in the expression. If the analog variable tag does not have Engineering Zero and Full Scale values, the defaults are 0 (zero) and 32000. For expressions without tags, the defaults are 0 (zero) and 100.
<b>[Scaling expression] Minimum</b>	Enter the minimum value for the expression. When this value is returned by the expression, the width of the object will be reduced to its minimum. You can only enter a value here if you have selected the <b>Specify range</b> box.
<b>[Scaling expression] Maximum</b>	Enter the maximum value for the expression. When this value is returned by the expression, the width of the object will be increased to its maximum. You can only enter a value here if you have selected the <b>Specify range</b> box.
<b>[Scaling] At minimum</b>	<p>The minimum width of the object (as a percentage of its original width). The object will be reduced to this width when the <b>Scaling expression</b> returns its minimum value.</p> <p>You can change the percentage by pressing the up and down arrows to the right of the field, or by entering another value in this field. Percentages of greater than 100% can be entered.</p>
<b>[Scaling] At maximum</b>	<p>The maximum width of the object (as a percentage of its original width). The object will grow to this width when the <b>Scaling expression</b> returns its maximum value.</p> <p>You can change the percentage by pressing the up and down arrows to the right of the field, or by entering another value in this field. Percentages of greater than 100% can be entered.</p> <p><b>Note:</b> You can increase the width at minimum, and decrease it at maximum, by swapping the percentages in the Scaling fields (i.e. put the high percentage in the <b>At minimum</b> field, and the low in the <b>At maximum</b>), or by swapping the expression limits (in the <b>Minimum</b></p>

Property	Description
	and <b>Maximum</b> fields).
<b>[Center axis offset] Express</b>	<p>Click this radio button for the quick and easy way of selecting one of three of the object's vertical axes (left, center, and right). These axes appear in the picture field to the right of the dialog.</p> <p>If you choose <b>Left</b>, width changes occur to the right of the object. (i.e., the left edge remains anchored):</p>  <p>If you choose <b>Center</b>, width changes occur equally to both sides. (i.e., the vertical center axis remains anchored):</p>  <p>If you choose <b>Right</b>, width changes occur to the left of the object. (i.e., the right edge remains anchored):</p> 
<b>[Center axis offset] Custom</b>	<p>Click this radio button to define your own center axis. A field appears to the right of the dialog allowing you to specify how far from the object center (in pixels) you would like to place the axis. Although this option gives you the option to place the center axis anywhere on the object, once placed, the scaling process works in exactly the same manner as for the Express option (illustrated above).</p> <p>For example, if you enter 20, the Center axis will be 20 pixels to the right of the object center.</p> <p>Enter a negative value to move the center axis left</p>

Property	Description
	instead of right.

**Note:** If a group and its objects are configured to change size during runtime, the group scaling effects will be combined with the object scaling effects. For example, if a group is configured to double in size at runtime, and one of its objects is configured to halve in size, the object will appear to remain the same size (it halves, then doubles). Remember, however, that the object's position might change as the group gets bigger.

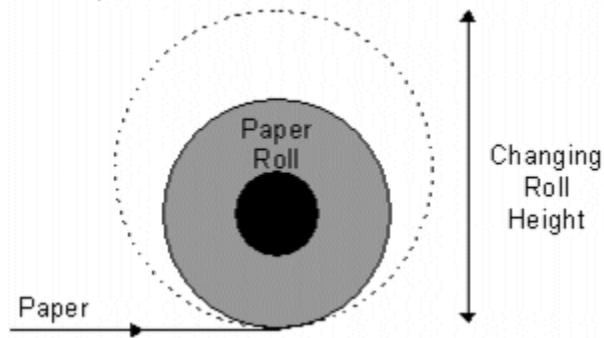
## See Also

[Object Properties - Scaling \(Vertical\)](#)

### Object Properties - Scaling (Vertical)

You can change the height of an object during runtime. As the value of the expression increases or decreases, the height of the object increases or decreases accordingly as a percentage of the original height; that is, when the object was added to the graphics page.

For example, an elevation of a paper roll (in a paper mill), could display changing roll height (and width):

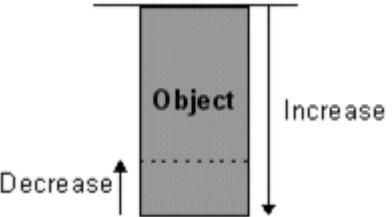
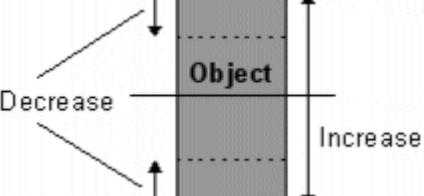
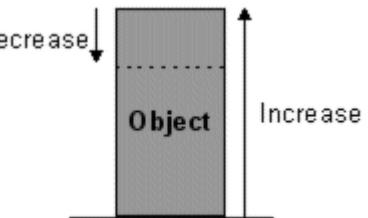


## Vertical Scaling Properties

Objects have the following vertical scaling properties:

Property	Description
<b>Scaling expression</b>	<p>The value of the expression entered in this field (253 characters maximum) will determine the vertical scaling (height) of the object. By default, when the expression returns its minimum value, the object will display at its minimum height (as defined in the Scaling fields below). When the expression returns its maximum value, the object will display at its maximum height (as defined in the Scaling fields below).</p> <p>To insert a tag or a function, click the <b>Wizard</b> button to the right of this field. This button displays two options:</p>

Property	Description
	<p>Insert Tag and Insert Function.</p> <p><b>Note:</b> You can also insert Equipment.item references into expression fields using the <b>insert tag</b> option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with available tags.</p>
<b>[Scaling expression] Specify range</b>	<p>Select this box to manually specify <b>Minimum</b> and <b>Maximum</b> values for the scaling expression, rather than using the default values. For an expression containing an analog variable tag, the defaults are the Engineering Zero and Full Scale values from the last variable tag in the expression. If the analog variable tag does not have Engineering Zero and Full Scale values, the defaults are 0 (zero) and 32000. For expressions without tags, the defaults are 0 (zero) and 100.</p>
<b>[Scaling expression] Minimum</b>	<p>Enter the minimum value for the expression. When this value is returned by the expression, the height of the object will be reduced to its minimum. You can only enter a value here if you have selected the <b>Specify range</b> box.</p>
<b>[Scaling expression] Maximum</b>	<p>Enter the maximum value for the expression. When this value is returned by the expression, the height of the object will be increased to its maximum. You can only enter a value here if you have selected the <b>Specify range</b> box.</p>
<b>[Scaling] At minimum</b>	<p>The minimum height of the object (as a percentage of its original height). The object will be reduced to this height when the <b>Scaling expression</b> returns its minimum value. You can change the percentage by pressing the up and down arrows to the right of the field, or by entering another value in this field. Percentages of greater than 100% can be entered.</p>
<b>[Scaling] At maximum</b>	<p>The maximum height of the object (as a percentage of its original height). The object will grow to this height when the <b>Scaling expression</b> returns its maximum value.</p> <p>You can change the percentage by pressing the up and down arrows to the right of the field, or by entering another value in this field. Percentages of greater than 100% can be entered.</p>

Property	Description
	<p><b>Note:</b> You can increase the height at minimum, and decrease it at maximum, by swapping the percentages in the Scaling fields (i.e. put the high percentage in the <b>At minimum</b> field, and the low in the <b>At maximum</b>), or by swapping the expression limits (in the <b>Minimum</b> and <b>Maximum</b> fields).</p>
<p><b>[Center axis offset] Express</b></p>	<p>Click this radio button to select one of three of the object's horizontal axes (top, middle, and bottom). These axes appear in the picture field to the right of the dialog. Click an axis to select it.</p> <p>If you choose the top, height changes will occur from the top of the object down. (i.e. the top edge will remain anchored):</p>  <p>If you choose the middle, height changes will occur equally above and below the axis. (i.e. the horizontal center axis will remain anchored):</p>  <p>If you choose the bottom, width changes will occur to the top edge of the object. (i.e. the bottom edge will remain anchored):</p> 
<p><b>[Center axis offset] Custom</b></p>	<p>Click this radio button to define your own center axis. A field will display to the right of the dialog, allowing you to specify how far from the object center (in pixels) you would like to place the axis. Although this</p>

Property	Description
	<p>option gives you the freedom to place the center axis anywhere on the object, once placed, the scaling process works in exactly the same manner as for the Express option (illustrated above).</p> <p>For example, if you enter 20, the Center axis will be 20 pixels below the object center.</p> <p>Enter a negative value to move the Center axis up instead of down.</p>

**Note:**

1. If a group and its objects are configured to change size during runtime, the group scaling effects will be combined with the object scaling effects. For example, if a group is configured to double in size at runtime, and one of its object is configured to halve in size, the object will appear to remain the same size (it halves, then doubles). Remember, however, that the object's position might change as the group gets bigger.
2. When the radio buttons in Object Properties - Fill Color (On/Off, Multi-state and so on) are selected, they change the appearance of the right-hand side of the dialog.

**See Also**

[Object Properties - Scaling \(Horizontal\)](#)

**Fill**

Click the **Fill** tab to specify the color which is to fill the object, and the level to which the object will be filled. The fill properties can change dynamically, depending on the return of an expression, or the state of a tag etc. (for instance, you could use this tab to visually reflect tank levels).

The check mark to the left of the Fill tab tells you when a Fill property has been configured. The check marks in the tabs down the right of the dialog indicate which property is configured.

**See Also**

[Fill Color](#)  
[Fill Level](#)  
[Edit Common Object Properties](#)

**Fill Color**

You can control the fill color to use for your objects.

**To configure an object or group with changing fill color:**

1. Draw the object/group (or paste a symbol). The properties tab dialog will automatically display, unless you have turned off the **Display properties on new** option in the Graphics Builder. (For a group, the properties dialog will not display automatically; you need to double-click the group.)

2. Click the **Fill** tab.
3. Click the **Color** tab (to the right of the dialog).
4. Select the type of color change (On/Off, Multi-state and so on).
5. Enter the expression/conditions that will change the object's fill color at runtime.
6. Enter additional object property details as necessary.
7. Click **OK**.

## See Also

[Object Properties - Fill Color \(On/Off\)](#)  
[Object Properties - Fill Color \(Multi-state\)](#)  
[Object Properties - Fill Color \(Array\)](#)  
[Object Properties - Fill Color \(Threshold\)](#)  
[Object Properties - Fill Color \(Gradient\)](#)  
[Edit Common Object Properties](#)

# Object Properties - Fill Color (On/Off)

Objects and groups have the following Fill Color (On/Off) properties:

Property	Description
[Type] On / Off	Select this radio button to fill the object/group with one color when a particular expression is TRUE, and another when it is FALSE. For example, you could fill an object/group with red when a particular variable tag is in alarm, and green when it is not.
[Type] Multi-state	This option is useful when you have several possible conditions, occurring together in different combinations, at different times. Select this option to fill the object/group with a different color for each combination.  For example, three digital variable tags (A,B, and C) can each be ON or OFF at any time. You can fill the object/group with a different color for each ON/OFF combination. In other words, you could use a different fill color for each of the following ON/OFF combinations ABC, ABC, ABC, ABC, ABC, ABC, ABC, ABC.
[Type] Array	The Array option allows you to enter an expression which returns an integer. For each unique integer (from 0 to 255), you can fill the object/group with a

Property	Description
	different color. For example, you could use a different fill color for each threshold of an analog alarm.
<b>[Type] Threshold</b>	Select this radio button to dynamically change the fill color when an expression reaches a specific value (threshold). For example, you might decide that the fill color will change to red when the speed of a motor is greater than or equal to 4500 rpm, and to white when less than or equal to 100 rpm, but remains gray for every speed in between.
<b>[Type] Gradient</b>	Select this radio button to dynamically graduate the fill color, displaying a different color for each unique value returned by a particular expression. This option allows you to select two colors, to be used as the color limits. The color for each value returned is automatically selected from within the range defined by these limits. The result is a fade from one color to another.
<b>ON color when</b>	<p>The color selected as the <b>ON color</b> (below) will be used as the fill color whenever the condition entered here (254 characters maximum) is TRUE. The color selected as the <b>OFF color</b> (below) will be used as the fill color whenever this condition is FALSE. For example, you could fill an object/group with blue when <b>MIX_RUNNING</b> is TRUE, and white when it is FALSE.</p> <p>To insert a tag or a function, click the <b>Wizard</b> button to the right of this field. This button displays two options: Insert Tag and Insert Function.</p> <p><b>Note:</b> You can also insert Equipment.item references into expression fields using the <b>insert tag</b> option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with available tags.</p>
<b>OFF color</b>	<p>The fill color whenever the condition entered above is FALSE. For example, you could fill the object/group with white when <b>MIX_RUNNING</b> is FALSE.</p> <p><b>Note:</b> The color that you select here will change any Fill color specified through Appearance (General) properties tab.</p>
<b>ON color</b>	The fill color whenever the condition entered above is TRUE. For example, you could fill the object/group

Property	Description
	<p>with blue when <b>MIX_RUNNING</b> is TRUE.</p> <p><b>Note:</b> Group fill color is only applied if a fill color is not defined for the individual objects in the group.</p>

## See Also

- [Object Properties - Fill Color \(Multi-state\)](#)
- [Object Properties - Fill Color \(Array\)](#)
- [Object Properties - Fill Color \(Threshold\)](#)
- [Object Properties - Fill Color \(Gradient\)](#)
- [Edit Common Object Properties](#)

# Object Properties - Fill Color (Multi-state)

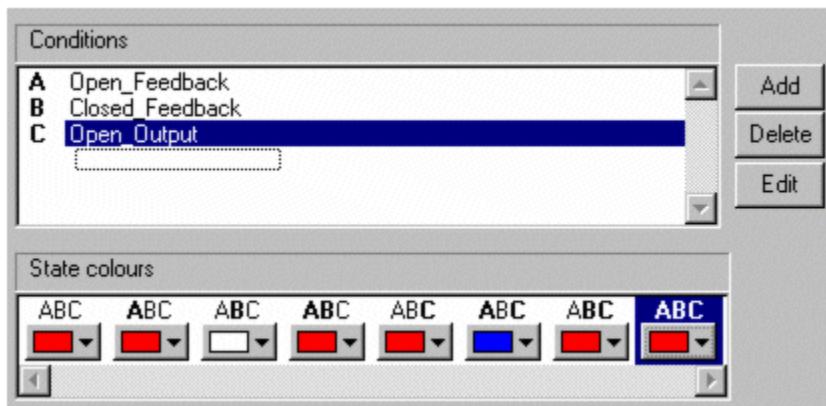
Objects and groups have the following Fill Color (Multi-state) properties:

Property	Description
<b>[Type] On / Off</b>	<p>Select this radio button to fill the object/group with one color when a particular expression is TRUE, and another when it is FALSE. For example, you could fill an object/group with red when a particular variable tag is in alarm, and green when it is not.</p>
<b>[Type] Multi-state</b>	<p>This option is useful when you have several possible conditions, occurring together in different combinations, at different times. Select this option to fill the object/group with a different color for each combination.</p> <p>For example, three digital variable tags (A,B, and C) can each be ON or OFF at any time. You can fill the object/group with a different color for each ON/OFF combination. In other words, you could use a different fill color for each of the following ON/OFF combinations ABC, ABC, ABC, ABC, ABC, ABC, ABC, ABC.</p>
<b>[Type] Array</b>	<p>The Array option allows you to enter an expression which returns an integer. For each unique integer (from 0 to 255), you can fill the object/group with a different color. For example, you could use a different fill color for each threshold of an analog alarm.</p>

Property	Description
[Type] Threshold	Select this radio button to dynamically change the fill color when an expression reaches a specific value (threshold). For example, you might decide that the fill color will change to red when the speed of a motor is greater than or equal to 4500 rpm, and to white when less than or equal to 100 rpm, but remains gray for every speed in between.
[Type] Gradient	Select this radio button to dynamically graduate the fill color, displaying a different color for each unique value returned by a particular expression. This option allows you to select two colors, to be used as the color limits. The color for each value returned is automatically selected from within the range defined by these limits. The result is a fade from one color to another.
Conditions	<p>The conditions you enter here (using a maximum of 128 characters per condition) will occur together in different ways, at different times. You can use each unique combination to force a different fill color for the object/group.</p> <p>To enter a condition, click the relevant line (A, B, C, and so on), click <b>Edit</b>, and type in the condition. You can add more conditions (to a maximum of 5, providing 32 combinations), using the <b>Add</b> button. To insert a tag or function, click the <b>Wizard</b> button. This button displays two options: Insert Tag and Insert Function.</p> <p><b>Note:</b> You can also insert Equipment.item references into expression fields using the <b>insert tag</b> option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with available tags.</p> <p>You can also remove conditions by clicking <b>Delete</b>, but you need to provide at least one condition in this field. Conditions left blank (instead of deleted) are evaluated as false at runtime.</p>
State colors	<p>The fill colors that will be used for each combination of the above conditions.</p> <p><b>Note:</b> The color that you select as <b>ABC</b> (all conditions false) will change any Fill color specified through Appearance (General) properties tab.</p>

## Example

To fill the object/group with a different color each time the status of a valve changes, you could fill out the **Conditions** and **State symbols** fields as follows:



In this example, **Open\_Feedback**, and **Closed\_Feedback** are variable tags representing digital inputs on the valve, and **Open\_Output** is a variable tag representing an output on the valve. So, **ABC** means **Open\_Feedback** is ON, and **Closed\_Feedback** and **Open\_Output** are both OFF. For this combination, the red is used as the fill color to indicate that the valve is not responding to commands, because it is open when it is meant to be closed. The same type of logic applies to the rest of the states.

**Note:** Group fill color is only applied if a fill color is not defined for the individual objects in the group.

## See Also

[Object Properties - Fill Color \(On/Off\)](#)

[Object Properties - Fill Color \(Array\)](#)

[Object Properties - Fill Color \(Threshold\)](#)

[Object Properties - Fill Color \(Gradient\)](#)

## Object Properties - Fill Color (Array)

Objects and groups have the following Fill Color (Array) properties:

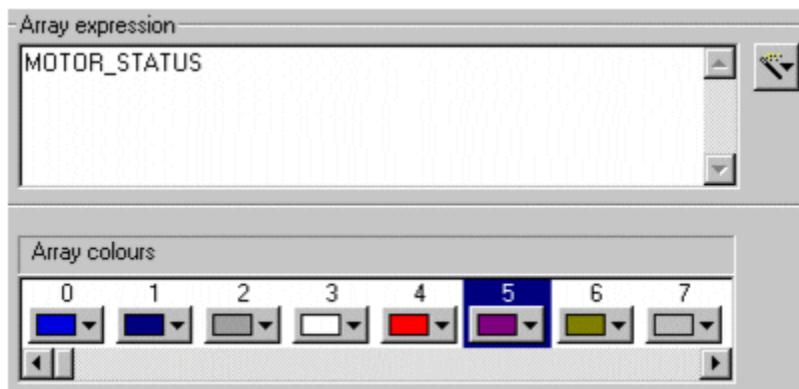
Property	Description
[Type] On / Off	Select this radio button to fill the object/group with one color when a particular expression is TRUE, and another when it is FALSE. For example, you could fill an object/group with red when a particular variable tag is in alarm, and green when it is not.
[Type] Multi-state	This option is useful when you have several possible conditions, occurring together in different combinations, at different times. Select this option to fill the object/group with a different color for each

Property	Description
	<p>combination.</p> <p>For example, three digital variable tags (A,B, and C) can each be ON or OFF at any time. You can fill the object/group with a different color for each ON/OFF combination. In other words, you could use a different fill color for each of the following ON/OFF combinations ABC, ABC, ABC, ABC, ABC, ABC, ABC, ABC.</p>
<b>[Type] Array</b>	<p>The Array option allows you to enter an expression which returns an integer. For each unique integer (from 0 to 255), you can fill the object/group with a different color. For example, you could use a different fill color for each threshold of an analog alarm.</p>
<b>[Type] Threshold</b>	<p>Select this radio button to dynamically change the fill color when an expression reaches a specific value (threshold). For example, you might decide that the fill color will change to red when the speed of a motor is greater than or equal to 4500 rpm, and to white when less than or equal to 100 rpm, but remains gray for every speed in between.</p>
<b>[Type] Gradient</b>	<p>Select this radio button to dynamically graduate the fill color, displaying a different color for each unique value returned by a particular expression. This option allows you to select two colors, to be used as the color limits. The color for each value returned is automatically selected from within the range defined by these limits. The result is a fade from one color to another.</p>
<b>Array expression (128 Chars.)</b>	<p>Enter the expression which is to return an integer. For each value returned, a different color will fill the object/group.</p> <p>If the return value is:</p> <ul style="list-style-type: none"> <li>• Less than 0 (zero), it will be set to 0 (zero), and a runtime hardware alarm will be triggered.</li> <li>• Greater than 255, it will be set to 255, and a runtime hardware alarm will be triggered.</li> <li>• A real (non-integer) number, it will be truncated (for example 8.1 and 8.7 would both be truncated to 8).</li> </ul> <p>To insert a tag or a function, click the <b>Wizard</b> button to</p>

Property	Description
	<p>the right of this field. This button displays two options: Insert Tag and Insert Function.</p> <p><b>Note:</b> You can also insert Equipment.item references into expression fields using the <b>insert tag</b> option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with available tags.</p>
<b>Array colors</b>	<p>The fill colors that will be used for each integer returned by the Array expression entered above (color 0 will be used when the expression returns integer 0, color 1 will be used when integer 1 is returned and so on).</p> <p><b>Note:</b> The color that you select for color 0 (zero) will change any Fill color specified through Appearance - General tab.</p>

## Example

To display different symbols illustrating the various states of a motor, you could fill out the **Array expression** and **Array symbol** fields as follows:



In this example, **MOTOR\_STATUS** is an analog variable tag representing the status of a motor. Each time the motor changes state, an integer is returned (0 = Running, 1 = Starting and so on), and the appropriate color fills the object/group. Color 5 onwards have no bearing on the fill color, because the tag only returns 5 unique integers (0-4).

---

**Note:** Group fill color is only applied if no fill color is defined for the individual objects in the group.

## See Also

[Object Properties - Fill Color \(On/Off\)](#)

[Object Properties - Fill Color \(Multi-state\)](#)

[Object Properties - Fill Color \(Threshold\)](#)

[Object Properties - Fill Color \(Gradient\)](#)

# Object Properties - Fill Color (Threshold)

## Object Properties - Fill Color (Threshold)

Objects and groups have the following fill color (threshold) properties.

Property	Description
[Type] On / Off	Select this radio button to fill the object/group with one color when a particular expression is TRUE, and another when it is FALSE. For example, you could fill an object/group with red when a particular variable tag is in alarm, and green when it is not.
[Type] Multi-state	This option is useful when you have several possible conditions, occurring together in different combinations, at different times. Select this option to fill the object/group with a different color for each combination.  For example, three digital variable tags (A,B, and C) can each be ON or OFF at any time. You can fill the object/group with a different color for each ON/OFF combination. In other words, you could use a different fill color for each of the following ON/OFF combinations ABC, ABC, ABC, ABC, ABC, ABC, ABC, ABC.
[Type] Array	The Array option allows you to enter an expression which returns an integer. For each unique integer (from 0 to 255), you can fill the object/group with a different color. For example, you could use a different fill color for each threshold of an analog alarm.
[Type] Threshold	Select this radio button to dynamically change the fill color when an expression reaches a specific value (threshold). For example, you might decide that the fill color will change to red when the speed of a motor is greater than or equal to 4500 rpm, and to white when less than or equal to 100 rpm, but remains gray for every speed in between.
[Type] Gradient	Select this radio button to dynamically graduate the fill color, displaying a different color for each unique value returned by a particular expression. This option allows you to select two colors, to be used as the color limits. The color for each value returned is

Property	Description
	automatically selected from within the range defined by these limits. The result is a fade from one color to another.
<b>Color expression</b>	<p>The value of the expression entered in this field (128 characters maximum) determines the fill color of the object/group. i.e. When the value of this expression reaches a threshold value (as defined below), fill color will change.</p> <p>To insert a tag or a function, click the <b>Wizard</b> button to the right of this field. This button displays two options: Insert Tag and Insert Function.</p> <p><b>Note:</b> You can also insert Equipment.item references into expression fields using the <b>insert tag</b> option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with available tags.</p>
<b>[Color expression] Specify range</b>	<p>Select this box to manually specify <b>Minimum</b> and <b>Maximum</b> values for the Color expression, rather than using the default values. (Threshold values are percentages of the range between <b>Minimum</b> and <b>Maximum</b>.) For an expression containing an analog variable tag, the defaults are the Engineering Zero and Full Scale values from the last variable tag in the expression. If the analog variable tag does not have Engineering Zero and Full Scale values, the defaults are 0 (zero) and 32000. For expressions without tags, the defaults are 0 (zero) and 100.</p>
<b>[Color expression] Minimum</b>	<p>Enter the minimum value for the expression. In terms of thresholds, the Minimum is <b>0%</b>. You can only enter a value here if you have selected the <b>Specify</b> range box.</p>
<b>[Color expression] Maximum</b>	<p>Enter the maximum value for the expression. In terms of thresholds, the Maximum is <b>100%</b>. You can only enter a value here if you have selected the <b>Specify</b> range box.</p>
<b>Thresholds (%)</b>	<p>The thresholds and their associated colors. A threshold is entered as a percentage of the expression range (the range of values that can be returned by the expression).</p>

## Example

If the expression's minimum is 0 and its Maximum 200, the default thresholds would have the effects described in the table below.

Threshold	Associated Color	Meaning
< 5%	Bright Blue	When the expression returns <b>less than 10</b> , the color fill will be <b>Bright Blue</b> .
< 15%	Blue	When the expression returns <b>less than 30</b> , the color fill will be <b>Blue</b> .
> 85%	Red	When the expression returns <b>greater than 170</b> , the color fill will be <b>Red</b> .
> 95%	Bright Red	When the expression returns <b>greater than 190</b> , the color fill will be <b>Bright Red</b> .

You can add up to 100 threshold color combinations. To add a combination, click **Add** and enter the relevant details. To edit an existing combination, click the relevant line. You can also remove combinations by clicking **Delete**.

Any values not included in a range (for example between 15% and 85% in the example above) produce a static fill color as specified through **Appearance - General** tab.

---

**Note:** Group fill color is only applied if no fill color is defined for the individual objects in the group.

---

## See Also

- [Object Properties - Fill Color \(On/Off\)](#)
- [Object Properties - Fill Color \(Multi-state\)](#)
- [Object Properties - Fill Color \(Array\)](#)
- [Object Properties - Fill Color \(Gradient\)](#)

# Object Properties - Fill Color (Gradient)

## Object Properties - Fill Color (Gradient)

Objects and groups have the following Fill Color (Gradient) properties:

Property	Description
[Type] On / Off	Select this radio button to fill the object/group with one color when a particular expression is TRUE, and

Property	Description
	another when it is FALSE. For example, you could fill an object/group with red when a particular variable tag is in alarm, and green when it is not.
<b>[Type] Multi-state</b>	<p>This option is useful when you have several possible conditions, occurring together in different combinations, at different times. Select this option to fill the object/group with a different color for each combination.</p> <p>For example, three digital variable tags (A,B, and C) can each be ON or OFF at any time. You can fill the object/group with a different color for each ON/OFF combination. In other words, you could use a different fill color for each of the following ON/OFF combinations ABC, <b>A</b>BC, ABC, ABC, <b>A</b>BC, ABC, <b>A</b>BC, <b>A</b>BC.</p>
<b>[Type] Array</b>	The Array option allows you to enter an expression which returns an integer. For each unique integer (from 0 to 255), you can fill the object/group with a different color. For example, you could use a different fill color for each threshold of an analog alarm.
<b>[Type] Threshold</b>	Select this radio button to dynamically change the fill color when an expression reaches a specific value (threshold). For example, you might decide that the fill color will change to red when the speed of a motor is greater than or equal to 4500 rpm, and to white when less than or equal to 100 rpm, but remains gray for every speed in between.
<b>[Type] Gradient</b>	Select this radio button to dynamically graduate the fill color, displaying a different color for each unique value returned by a particular expression. This option allows you to select two colors, to be used as the color limits. The color for each value returned is automatically selected from within the range defined by these limits. The result is a fade from one color to another.
<b>Color expression</b>	The value of the expression entered in this field (128 characters maximum) will determine the fill color of the object/group. By default, when the expression returns its minimum value, the fill color will be the <b>At minimum</b> color (as defined below). When the expression returns its maximum value, the fill color will be the <b>At maximum</b> color (as defined below).

Property	Description
	<p>When the expression returns a value half-way between its minimum and maximum, a color will be selected from the half-way point of the range defined by the <b>At minimum</b> and <b>At maximum</b> colors.</p> <p>To insert a tag or a function, click the <b>Wizard</b> button to the right of this field. This button displays two options: Insert Tag and Insert Function.</p> <p><b>Note:</b> You can also insert Equipment.item references into expression fields using the <b>insert tag</b> option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with available tags.</p>
<b>[Color expression] Specify range</b>	<p>Select this box to manually specify <b>Minimum</b> and <b>Maximum</b> values for the Color expression, rather than using the default values. For an expression containing an analog variable tag, the defaults are the Engineering Zero and Full Scale values from the last variable tag in the expression. If the analog variable tag does not have Engineering Zero and Full Scale values, the defaults are 0 (zero) and 32000. For expressions without tags, the defaults are 0 (zero) and 100.</p>
<b>[Color expression] Minimum</b>	<p>Enter the minimum value for the expression. When this value is returned by the expression, the fill color of the object/group will be the <b>At minimum</b> color. You can only enter a value here if you have selected the <b>Specify range</b> box.</p>
<b>[Color expression] Maximum</b>	<p>Enter the maximum value for the expression. When this value is returned by the expression, the fill color of the object/group will be the <b>At maximum</b> color. You can only enter a value here if you have selected the <b>Specify range</b> box.</p>
<b>At minimum</b>	<p>The fill color of the object/group when the <b>Color expression</b> returns its minimum value.</p> <p><b>Note:</b> The color that you select here will change any Fill color specified through <b>Appearance - General</b> tab.</p>
<b>At maximum</b>	<p>The fill color of the object/group when the <b>Color expression</b> returns its maximum value.</p> <p><b>Note:</b> Group fill color is only applied if a fill color is not defined for the individual objects in the group.</p>

## See Also

- [Object Properties - Fill Color \(On/Off\)](#)
- [Object Properties - Fill Color \(Multi-state\)](#)
- [Object Properties - Fill Color \(Array\)](#)
- [Object Properties - Fill Color \(Threshold\)](#)

### Fill Level

The fill level of an object/group can be changed during runtime, increasing or decreasing dynamically whenever the value of a particular expression changes. As the value of the expression increases and decreases, the fill level will increase and decrease accordingly (as a percentage of the full capacity of the object/group). If the object/group resizes at runtime, the fill level will adjust automatically in order to maintain the correct percentage.

The color that is used is set through either General Appearance, or Color Fill.

This property could be used to display temperature variations. You could even combine the Fill Color and Fill Level properties to produce a thermometer with mercury that rises and changes color with rising temperature.

#### To configure an object or group with changing fill level:

1. Draw the object/group (or paste a symbol). The properties tab dialog will automatically display, unless you have turned off the **Display properties on new** option in the Graphics Builder. (For a group, the properties dialog will not display automatically; you need to double-click the group.)
2. Click the **Fill** tab.
3. Click the **Level** tab (to the right of the dialog).
4. Enter a **Level expression** (the expression that will change the fill level of the object/group at runtime).
5. Enter additional properties as necessary.
6. Click **OK**.

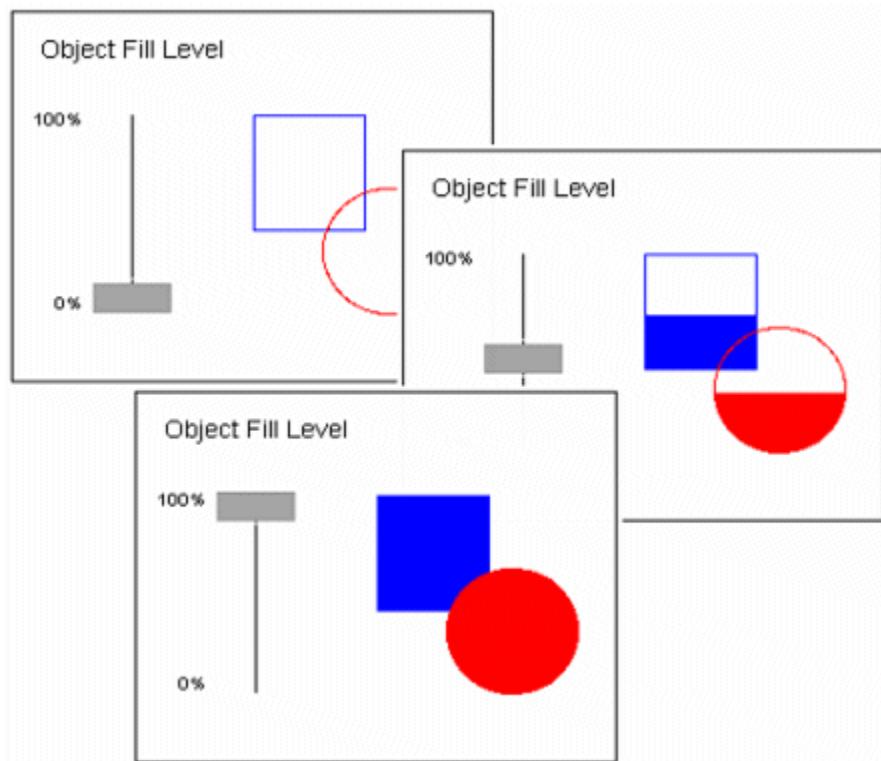
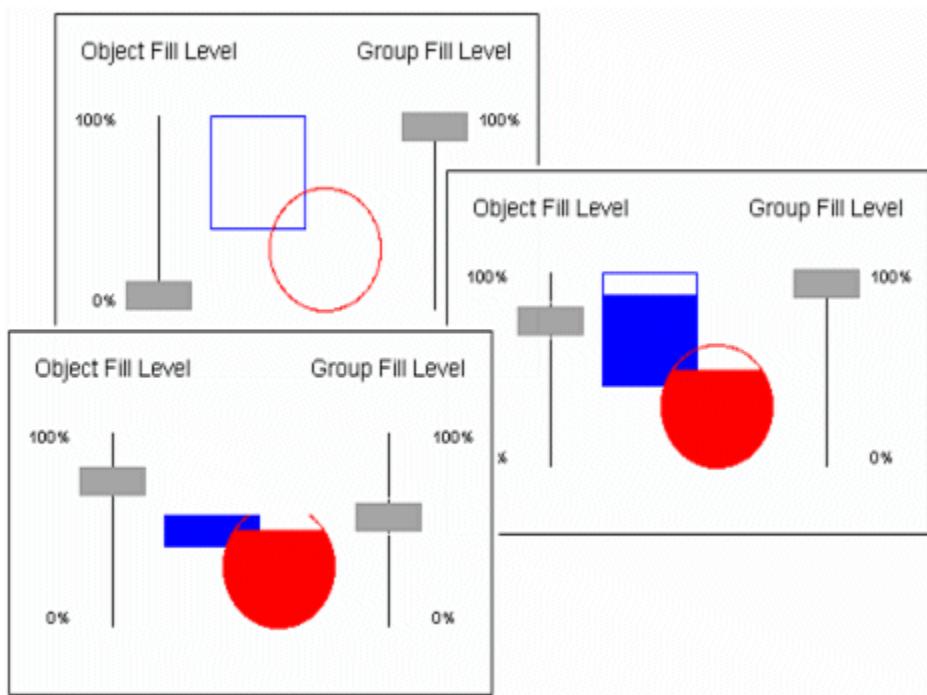
### Fill Level Properties

Objects and groups have the following Fill Level properties:

Property	Description
<b>Level expression</b>	The value of the expression entered in this field (253 characters maximum) will determine the fill level of the object/group. By default, when the expression returns its minimum value, the object/group will be filled to the <b>At minimum</b> level. When the expression returns its maximum value, the object/group will be filled to the <b>At maximum</b> level. When the expression returns a value half-way between its minimum and maximum, the object/group will be filled to half-way between the <b>At minimum</b> and <b>At maximum</b> levels.  To insert a tag or a function, click the <b>Wizard</b> button to the right of this field. This button displays two options:

Property	Description
	<p>Insert Tag and Insert Function.</p> <p><b>Note:</b> You can also insert Equipment.item references into expression fields using the <b>insert tag</b> option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with available tags.</p>
<b>[Level expression]</b> Specify range	<p>Select this box to manually specify <b>Minimum</b> and <b>Maximum</b> values for the Level expression, rather than using the default values. For an expression containing an analog variable tag, the defaults are the Engineering Zero and Full Scale values from the last variable tag in the expression. If the analog variable tag does not have Engineering Zero and Full Scale values, the defaults are 0 (zero) and 32000. For expressions without tags, the defaults are 0 (zero) and 100.</p>
<b>[Level expression]</b> Minimum	<p>Enter the minimum value for the expression. When this value is returned by the expression, the object/group will fill to the <b>At minimum</b> level. You can only enter a value here if you have selected the <b>Specify range</b> box.</p>
<b>[Level expression]</b> Maximum	<p>Enter the maximum value for the expression. When this value is returned by the expression, the object/group will fill to the <b>At maximum</b> level. You can only enter a value here if you have selected the <b>Specify range</b> box.</p>
<b>At minimum</b>	<p>The level to which the object/group will be filled when the <b>Level expression</b> returns its minimum value. For example, if you enter 30, the object/group will be 30% full when the expression returns its minimum value. You can change the percentage by pressing the up and down arrows to the right of the field, or by entering another value in this field.</p>
<b>At maximum</b>	<p>The level to which the object/group will be filled when the <b>Level expression</b> returns its maximum value. For example, if you enter 90, the object/group will be 90% full when the expression returns its maximum value. You can change the percentage by pressing the up and down arrows to the right of the field, or by entering another value in this field.</p>

Property	Description
<b>Fill Direction</b>	The direction in which the color will spread when increasing. There are four options (each represented by an arrow): <b>Up, Down, Left, Right</b> . If you choose Up, the object/group will be filled from the bottom up. If you choose Left, the object/group will be filled from right to left, and so on.
<b>Background color</b>	<p>The color of any unfilled part of the object/group (for example, if the object/group is only 90% full, the unfilled 10% will be displayed using this color). The background is often made transparent. Using transparent, you would see the outline of the object/group, and anything behind the object/group on the page.</p> <p><b>Note:</b> If an object in a group is a slider, it might change the overall size of the group when used at runtime. If it does, the fill level of the group will adjust accordingly.</p>
<b>Group and Object Fill Level: Examples</b>	A group and its objects can be configured with different fill levels. The group fill level, however, is usually thought of as a reveal of the objects in the group. Group fill level and object fill level operate independently of each other; the group fill level just determines how much of the objects display.

**Example 1: The fill level of the objects****Example 2: Group the objects and configure a fill level for the group as well**

In this example, the objects' fill levels can still be adjusted normally. The group's fill level determines how much

of the objects you can see (and how much will be obscured by the groups background color; white, in this case).

## See Also

[Edit Common Object Properties](#)

## Sliders

You can create slider controls to enable an operator to interact with your runtime system by dragging an object on a graphics page.

### To configure a slider:

1. Draw the object/group (or paste a symbol). The properties tab dialog will automatically display, unless you have turned off the **Display properties on new** option in the Graphics Builder. (For a group, the properties dialog will not display automatically; you need to double-click the group.)
2. Click the **Slider** tab.
3. Click the **Horizontal**, **Vertical** or **Rotational** tab (to the right of the dialog).
4. Enter the **Tag** to link to the slider.
5. Enter any additional details.
6. Click **OK**.

## See Also

[Object Properties - Slider \(Horizontal\)](#)

[Object Properties - Slider \(Vertical\)](#)

[Object Properties - Slider \(Rotational\)](#)

[Edit Common Object Properties](#)

## Object Properties - Slider (Horizontal)

Objects and groups can be linked to variable tags in such a way that horizontal sliding of the object/group changes the value of the tag. As the slider moves to the right, the variable tag increases in value. As the slider moves to the left, the variable tag decreases in value. The slider also moves automatically to reflect the changing values of the tag.

---

**Note:** The horizontal slider cannot be used if the rotational slider is enabled, or if horizontal movement is enabled.

---

## Horizontal Slider Properties

Objects and groups have the following horizontal slider properties.

Property	Description
Tag	The value of the tag entered in this field (79 characters)

Property	Description
	<p>maximum) will change when the slider is moved left or right. You can define two slider limits on your graphics page. The object/group will not slide beyond these two points. During runtime, when the slider reaches its left-hand limit (<b>Offset At minimum</b>), the tag value changes to its minimum limit. When the slider reaches its right-hand limit (<b>Offset At maximum</b>), the tag value changes to its maximum limit.</p> <p>To insert a tag, click the <b>Wizard</b> button to the right of this field.</p> <p><b>Note:</b> You can also insert Equipment.item references into expression fields using the <b>insert tag</b> option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with available tags.</p>
<b>[Tag] Continuous update of tag</b>	<p>Select this box if you want the variable tag to be updated continuously while the slider is being moved. If you leave this box unchecked, the tag will only be updated when the slider has been released (i.e. it has been moved, and the operator has released the mouse button).</p>
<b>[Offset] At minimum</b>	<p>The distance (number of pixels from the original object/group center) that the object/group can slide to the left. When it reaches the point defined by this distance, the tag value changes to its minimum limit. You can change the offset value by pressing the up and down arrows to the right of the field, or by entering another value in this field.</p>
<b>[Offset] At maximum</b>	<p>The distance (number of pixels from the original object/group center) that the object/group can slide to the right. When it reaches the point defined by this distance, the tag value changes to its maximum limit. You can change the offset value by pressing the up and down arrows to the right of the field, or by entering another value in this field.</p> <p>You can increase the value of the tag with a left-slide, and decrease it with a right-slide, by entering negative distances in the Offset fields.</p>

**Note:** If an object in a group is a slider, it might change the overall size of the group when used at runtime. If it does, the fill level of the group will adjust accordingly. If a group and one of its objects are both defined as

---

sliders, and they slide in the same direction or one is rotational, the object will take precedence (i.e. only the object will operate as a slider).

---

## See Also

[Object Properties - Slider \(Vertical\)](#)

[Object Properties - Slider \(Rotational\)](#)

### Object Properties - Slider (Vertical)

You can link objects and groups to variable tags in such a way that vertical sliding of the object/group changes the value of the tag. As the slider moves to the up, the variable tag increases in value. As the slider moves to the down, the variable tag decreases in value. The slider also moves automatically to reflect the changing values of the tag.

**Note:** The vertical slider cannot be used if the rotational slider is enabled, or if vertical movement is enabled.

---

## Vertical Slider Properties

Objects and groups have the following vertical slider properties:

Property	Description
<b>Tag</b>	<p>The value of the tag entered in this field (79 characters maximum) will change when the slider is moved up or down. You can define two slider limits on your graphics page. The object/group will not slide beyond these two points. During runtime, when the slider reaches its upper limit (<b>Offset At maximum</b>), the tag value changes to its maximum limit. When the slider reaches its lower limit (<b>Offset At minimum</b>), the tag value changes to its minimum limit.</p> <p>To insert a tag, click the <b>Wizard</b> button to the right of this field.</p> <p><b>Note:</b> You can also insert Equipment.item references into expression fields using the <b>insert tag</b> option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with available tags.</p>
<b>[Tag] Continuous update of tag</b>	Select this box if you want the variable tag to be updated continuously while the slider is being moved. If you leave this box unchecked, the tag will only be updated when the slider has been released (i.e., it has been moved, and the operator has released the mouse button).

Property	Description
<b>[Offset] At maximum</b>	<p>The distance (number of pixels from the original object/group center) that the object/group can slide up. When it reaches the point defined by this distance, the <b>Tag</b> value changes to its maximum limit.</p> <p>You can change the offset value by pressing the up and down arrows to the right of the field, or by entering another value in this field.</p>
<b>[Offset] At minimum</b>	<p>The distance (number of pixels from the original object/group center) that the object/group can slide down. When it reaches the point defined by this distance, the tag value changes to its minimum limit.</p> <p>You can change the offset value by pressing the up and down arrows to the right of the field, or by entering another value in this field.</p> <p>You can increase the value of the tag with a down-slide, and decrease it with an up-slide, by entering negative distances in the Offset fields.</p>

**Note:** If an object in a group is a slider, it might change the overall size of the group when used at runtime. If it does, the fill level of the group will adjust accordingly. If a group and one of its objects are both defined as sliders, the object will take precedence (i.e. only the object will operate as a slider).

## See Also

[Object Properties - Slider \(Horizontal\)](#)

[Object Properties - Slider \(Rotational\)](#)

## Object Properties - Slider (Rotational)

Objects and groups can be linked to variable tags in such a way that rotational sliding of the object/group changes the value of the tag. As the slider rotates clockwise, the variable tag increases in value. As the slider rotates anti-clockwise, the variable tag decreases in value. The slider also moves automatically to reflect the changing values of the tag.

**Note:** The rotational slider cannot be used if either of the other sliders is enabled, or if rotational movement is enabled.

## Rotational Slider Properties

Objects and groups have the following rotational slider properties.

Property	Description
<b>Tag</b>	<p>The value of the tag entered in this field (79 characters maximum) will change as the slider is rotated. You can</p>

Property	Description
	<p>define two slider limits on your graphics page. The object/group will not rotate beyond these two points. During runtime, when the slider reaches its anti-clockwise limit (<b>Offset At minimum</b>), the tag value changes to its minimum limit. When the slider reaches its clockwise limit (<b>Offset At maximum</b>), the tag value changes to its maximum limit.</p> <p>To insert a tag, click the <b>Wizard</b> button to the right of this field.</p> <p><b>Note:</b> You can also insert Equipment.item references into expression fields using the <b>insert tag</b> option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with available tags.</p>
<b>[Tag] Continuous update of tag</b>	<p>Select this box if you want the variable tag to be updated continuously while the slider is being moved. If leave this box unchecked, the tag will only be updated when the slider has been released (i.e. it has been moved, and the operator has released the mouse button).</p>
<b>[Angle] At minimum</b>	<p>Enter an anti-clockwise angle (in degrees relative to 0 degrees). The slider cannot rotate anti-clockwise beyond this limit. When it reaches this limit, the <b>Tag</b> value changes to its minimum limit.</p> <p>You can change the angle value by pressing the up and down arrows to the right of the field, or by entering another value in this field.</p>
<b>[Angle] At maximum</b>	<p>Enter an clockwise angle (in degrees relative to 0 degrees). The slider cannot rotate clockwise beyond this limit. When it reaches this limit, the <b>Tag</b> value changes to its maximum limit.</p> <p>You can change the angle value by pressing the up and down arrows to the right of the field, or by entering another value in this field.</p> <p><b>Note:</b> You can increase the value of the tag with an anti-clockwise slide, and decrease it with a clockwise-slide, by entering negative distances in the Angle fields.</p>
<b>[Center axis offset] Express</b>	<p>Click this radio button for the quick and easy way of selecting the point about which the object/group will rotate. The express option gives you the choice of 9</p>

Property	Description
	points (Top Left, Bottom Right and so on), which are displayed in the picture field on the dialog. To select one, just click it with your mouse.
<b>[Center axis offset] Custom</b>	Click this radio button to define your own center axis. When you select this radio button, two fields will display to the right, allowing you to plot the position of your center axis. Specify the distance to the right in the first field, and the distance down in the second. The Center axis is plotted based on these two values.  For example, if you enter 8 as the horizontal offset and 13 as the vertical, the center axis will be 8 pixels to the right and 13 pixels below the center of the object/group.  Enter negative values in the offset distance fields to move the center axis left instead of right, and up instead of down.

**Note:** If a group and one of its objects are both defined as sliders, the object will take precedence (i.e., only the object will operate as a slider).

## See Also

[Object Properties - Slider \(Horizontal\)](#)

[Object Properties - Slider \(Vertical\)](#)

## Input

Click the **Input** tab to specify the command to be executed, and the message to be logged when an operator clicks on the object. You can also define keyboard commands for the object, and limit their scope with area and privilege settings.

The check mark to the left of the Input tab tells you when an Input property has been configured. The check marks in the tabs down the right of the dialog indicate which property is configured.

## See Also

[Object Touch Commands](#)

[Object Keyboard Commands](#)

[Edit Common Object Properties](#)

## Object Touch Commands

You can assign touch commands to an object or group.

**To assign a touch command to an object or group:**

1. Draw the object/group (or paste a symbol). The properties tab dialog will automatically display, unless you have turned off the **Display properties on new** option in the Graphics Builder. (For a group, the properties dialog will not display automatically; you need to double-click the group.)
2. Click the **Input** tab.
3. Click the **Touch** tab (to the right of the dialog).
4. Enter a command in the command field (the command that will be executed when the object/group is touched at runtime).
5. Enter further details as necessary, using the **Help** button for detailed information about each field.
6. Click **OK**.

**Input Touch Commands - Properties**

The touch property lets you assign commands to the object/group. These commands are then executed when the object/group is touched at runtime (i.e., an operator clicks on the object/group). You can also define messages which will log at these times.

For example, a drive can be jogged by starting it when the mouse button is depressed and stopping it when the mouse button released; variables can be incremented while the mouse button is held, and so on. At the same time, it could log the time and date, and the name of the operator.

Operators without appropriate access requirements cannot touch the object/group at runtime.

Objects and groups have the following input (touch) properties.

Property	Description
Action	There are three actions to which commands can be attached. You can select more than one type of action. Unique commands and log messages can be attached to each action (i.e. you can perform one task on the down action, and another on the up action, and log a separate message for each).
[Action] Up	Select this option if you want a command to be executed (and a unique message to be logged) when the operator positions the mouse pointer over the object/group, and clicks <i>and releases</i> the left mouse button.  As with standard Windows buttons, if the operator moves the cursor away from the object/group before releasing the mouse button, the command isn't executed (unless you also select the <b>Down</b> option).
[Action] Down	Select this option if you want a command to be executed (and a unique message to be logged) when the operator positions the mouse pointer over the object/group, and clicks the left mouse button. The command will execute as soon as the mouse button is

Property	Description
	clicked.
<b>[Action] Repeat</b>	<p>Select this option if you want a command to execute continually (and a unique message to log continually) whenever the operator has the mouse pointer positioned over the object/group, and is holding the left mouse button depressed. If the operator moves the mouse pointer away from the object/group without releasing the mouse button, the command will stop executing, but will start again as soon as the mouse pointer is re-positioned over the object/group. The only exception is when you also have the Down option selected, in which case, the command will execute continually even if the mouse pointer has been moved away from the object/group.</p> <p>To set the delay which precedes the first execution of the command (and the first log of the message), and the delay between each repeat.</p>
<b>Up/Down/Repeat command</b>	<p>The commands (set of instructions) to be executed immediately when the selected action is performed. The command(s) can be a maximum of 253 characters long.</p> <p>To insert a tag or a function, click the <b>Wizard</b> button to the right of this field. This button displays two options: Insert Tag and Insert Function.</p> <p><b>Note:</b> You can also insert Equipment.item references into expression fields using the <b>insert tag</b> option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with available tags.</p>
<b>[Logging] Log Message</b>	<p>The text message sent to the <i>MsgLog</i> field of the Log Device when the selected action is performed by the operator at runtime. The message is plain text, and can include tag name substitutions using Genie or Super Genie syntax. When using Super Genie syntax, the data type needs to be specified. The name of the tag will then be included in the text. The log message can be a maximum of 32 characters long.</p>  <p>To include field data as part of a logged message,</p>

Property	Description
	<p>insert the field name as part of the device format when configuring the device. For instance, in the <b>Format</b> field of the <b>Devices</b> form, you could enter {MsgLog,20} {FullName,15}. This would accommodate the logging of messages such as <b>P2 started by John Smith</b>.</p> <p>The log device to which the message is sent is specified through the General Access tab.</p> <p><b>Note:</b> If the object is part of a Genie or symbol, this property can be defined after the Genie/symbol is pasted onto a page. (Hold down the <b>Control</b> (CTRL) key and double-click the object.) If you define it before pasting (i.e. you define it for the original in the library), you cannot edit it after. Similarly, if the object is part of a template, it can be defined after a page has been created using that template (again, with Control + double-click). If you define it for the actual template, you cannot edit it for pages based on the template.</p>
<b>Repeat rate</b>	<p>This option sets the delay which precedes the first execution of the command(s), and the delay between each subsequent repeat of the command(s).</p> <p>You can change the rate by pressing the up and down arrows to the right of the field, or by entering another value in this field.</p> <p><b>Note:</b> If you define a touch command for an object in a group, the group's touch command will not work.</p>

## See Also

[Edit Common Object Properties](#)

### Object Keyboard Commands

You can assign a keyboard command to an object or group.

#### To assign a keyboard command to an object or group:

1. Draw the object/group (or paste a symbol). The object properties tab dialog will automatically display, unless you have turned off the **Display properties on new** option in the Graphics Builder. (For a group, the properties dialog will not display automatically; you need to double-click the group.)
2. Click the **Input** tab.
3. Click the **Keyboard Commands** tab (to the right of the dialog).
4. Enter the key sequence (see [Define Key Sequences for Commands](#)).

5. Enter a command in the command field (the command that will be executed when the key sequence above is entered by the operator at runtime).
6. Enter additional details as necessary.
7. Click **OK**.

## Input Keyboard Commands - Properties

The keyboard commands properties let you assign keyboard commands to the object/group. A keyboard command is a particular key sequence which executes a command when it is typed in by the operator at runtime. To execute an object/group keyboard command, the operator positions the cursor over the object/group and enters the key sequence using the keyboard.

You can also define a message which will log every time the key sequence is entered.

For example, the operator could change the water level in a tank by placing the mouse over the symbol representing the tank, and typing in the new level. At the same time, a message could be logged, listing the time and date, and the name of the operator.

Operators without the access requirements specified under **Security** (see below) cannot enter keyboard commands for this object/group at runtime.

For a detailed explanation of keyboard command input see the topic Getting Runtime Operator Input in the Cicode Programming Reference.

Property	Description
<b>Key sequence</b>	<p>Enter the key sequences that the operator can enter to execute a command. For example, you might define the key sequence <b>### Enter</b>. During runtime, this key sequence would allow you to type in any three digit number, and click <b>Enter</b> to change variable tag values from mimic pages and so on</p> <p>You can enter as many key sequences as you like. To add a key sequence, click <b>Add</b> and type in the sequence or select one from the menu. To edit an existing sequence, click the relevant line and click <b>Edit</b>. You can also remove key sequences by clicking <b>Delete</b>.</p>
<b>Key sequence command</b>	<p>The commands (set of instructions) to be executed immediately when the selected key sequence is entered. The commands can be a maximum of 253 characters long.</p> <p>To insert a tag or a function, click the <b>Wizard</b> button to the right of this field. This button displays two options: Insert Tag and Insert Function.</p> <p><b>Note:</b> You can also insert Equipment.item references into expression fields using the <b>insert tag</b> option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with</p>

Property	Description
	available tags.
<b>[Security]</b> <b>Same area as object/group</b>	<p>Select this box to assign the keyboard command to the same area as the object/group. Only users with access to this area (and any necessary privileges) will be able to issue this command or log the message. If you want to assign this keyboard command to another area, leave this box unchecked and enter another area below.</p>
<b>[Security]</b> <b>Command area</b>	<p>Enter the area to which this keyboard command belongs. Only users with access to this area (and any necessary privileges) will be able to issue this command or log the message. For example, if you enter Area 1 here, operators need to have access to Area 1 (plus any necessary privileges) to issue this command.</p> <p>Click the menu to the right of this field to select an area, or type in an area number directly.</p> <p><b>Note:</b> If the object is part of a Genie or symbol, this property can be defined after the Genie/symbol is pasted onto a page (<b>Ctrl + double-click</b>). Similarly, if the object is part of a template, this property can be defined after a page has been created using that template (<b>Ctrl + double-click</b>).</p> <p>You can leave this field blank by selecting the <b>Same privilege as object/group</b> box.</p>
<b>[Security]</b> <b>Same privilege as object/group</b>	<p>Select this box to assign the keyboard command the same privilege as the object/group. Only users with this privilege level will be able to issue this command, or log the message. If you want to assign this keyboard command a different privilege, leave the box unchecked and enter another privilege below.</p>
<b>[Security]</b> <b>Privilege level</b>	<p>Enter the privilege level that a user needs to possess to be able to issue this command or log the message. For example, if you enter Privilege Level 1 here, operators need to possess Privilege Level 1 to be able to issue this command. You can also combine this restriction with area restrictions. For example, if you assign the keyboard command to Area 5, with Privilege Level 2, the user needs to be set up with Privilege 2 for Area 5.</p> <p>Click the menu to the right of this field to select a privilege, or type in an area number directly.</p>

Property	Description
	<p><b>Note:</b> If the object is part of a Genie or symbol, this property can be defined after the Genie/symbol is pasted onto a page (<b>Ctrl</b> + double-click). Similarly, if the object is part of a template, this property can be defined after a page has been created using that template (<b>Ctrl</b> + double-click).</p> <p>You can leave this field blank by selecting the <b>Same privilege as object/group</b> box.</p>
<b>[Logging] Log Message</b>	<p>The text message sent to the <i>MsgLog</i> field of the Log Device when the selected action is performed by the operator at runtime. The message is plain text, and can include tag name substitutions using Genie or Super Genie syntax. When using Super Genie syntax, the data type needs to be specified. The name of the tag will then be included in the text. The message can be a maximum of 32 characters long.</p>  <p>If you want to include field data as part of a logged message, you need to insert the field name as part of the device format when you configure the device. For instance, in the <b>Format</b> field of the <b>Devices</b> form, you could enter {MsgLog,20} {FullName,15}. This would accommodate the logging of messages such as <b>P2 started by John Smith</b>.</p> <p>The log device to which the message is sent is specified through the General Access tab.</p> <p><b>Note:</b> If a group and one of its objects are both assigned a keyboard command with the same key sequence, the object's command will take precedence (i.e., the group's command will not execute).</p>

## See Also

[Edit Common Object Properties](#)

## Access

Use the Access tab to assign an area or privilege to an object. Operators without appropriate access rights will not be able to use sliders, object specific keyboard commands, and so on. It also allows you to disable the object under certain runtime circumstances. This means that the object can be embossed, grayed, or even hidden.

You can determine the kind of access you want to have to your objects.

- General Access to Objects
- Disable Access to Objects

## General Access to Objects

Use the General tab to define the general access characteristics of objects.

**To define general access properties for objects:**

1. Double-click the object.
2. Click the **Access** tab.
3. Click the **General** tab.
4. Enter details as necessary, then click **OK**.

Use the Object Properties dialog to set up general identification, security, logging, and privilege parameters for your ActiveX object.

## General Access Properties

Objects and groups have the following general access properties.

Property	Description
[Identification] Object/Group AN	<p>Displays the automatically generated Animation Number of the object/group. The AN (Animation number) uniquely identifies the object/group, and can be used in Cicode functions and so on. This field cannot be edited.</p> <p><b>Note:</b> The Animation name should not be used as a substitute in those Graphics Automation Interface or Cicode Functions that use Animation numbers, unless specified.</p> <p>(Optional) In the <b>Name</b> field, type a name for the graphics object or group. Names need to be unique to the page, symbol, template, super genie, and genie the graphics object may belong to. The animation name should not start with a number or a special character.</p> <p><b>Note:</b> If the object is part of a Genie or symbol, the following properties can be completed after the Genie/Symbol is pasted onto a page (<b>Ctrl</b> + double-click). Similarly, if the object is part of a template, the properties can be defined after a page has been created using that template (<b>Ctrl</b> + double-click).</p>
[Identification] Description	Enter a description of the object/group, and its various functions and so on. This field is purely for the

Property	Description
	entry of information which you consider beneficial to the smooth running and maintenance of your system. It will not affect the way the system runs, and it will not display during runtime.
<b>[Identification] Tool tip</b>	<p>Enter a short, meaningful description (48 characters maximum) of the object/group. During runtime, this description appears when the operator moves the cursor onto the object/group. The message can be plain text, Super Genie syntax or both. When using Super Genie syntax, the data type needs to be included. The name of the tag will then be included in the text.</p> <p>If an object in a group has a tool tip, this object's tool tip will be displayed, and not the group's tool tip. If the object does not have a tool tip, the group tool tip will display. In this instance, if the object is a member of a group, and that group is part of another group, the tool tip for the first group will display.</p> <p>If you place an object behind a group, its tool tip will not display. Remember, however, that group boundaries are rectangular, no matter what shape is formed by the objects in the group. This means that 'blank' spaces between objects in a group are actually part of the group. Even if you can see the individual object, if it is behind the group, its tool tip will not display.</p>
<b>[Security] Same area as page</b>	Select this box to assign the object/group to the same area as the page on which it has been drawn; otherwise, leave it blank, and enter another area in the <b>Object/Group area</b> field (below).
<b>[Security] Object/Group area</b>	Enter the area to which this object/group belongs. Users without access to this area (and any necessary privileges) will not be able to make full use of the object/group. They will not be able to use touch command, keyboard commands, movement, sliders and so on. (In order to avoid confusion for such operators, it is sometimes a good idea to disable the object/group when it is unavailable due to lack of security rights. Disabled objects/groups can be grayed, hidden or embossed.) For example, if you enter Area 1 here, operators need to have access to Area 1 (plus any necessary privileges) to make use of this object/group.

Property	Description
	Click the menu to the right of this field to select an area, or type in an area number directly.
<b>[Security] No privilege restrictions</b>	<p>Select this box to disable privilege restrictions; otherwise, leave it blank, and enter another privilege below. The implications of not assigning a privilege restriction depend upon whether you have used areas in your security setup:</p> <ul style="list-style-type: none"> <li>• <b>No Areas:</b> Every operator has full control of the object/group.</li> <li>• <b>Areas:</b> An operator will only need view access to control the object/group if it does not have privilege restrictions.</li> </ul>
<b>[Security] Privilege level</b>	<p>Enter the privilege level necessary for an operator to use this object/group. Operators without the necessary privileges will not be able to make full use of the object/group. They will not be able to use touch command, keyboard commands, movement, sliders and so on. (To avoid confusion for such operators, disable the object/group when it is unavailable due to lack of security rights. Disabled objects/groups can be grayed, hidden or embossed.) For example, if you enter Privilege Level 1 here, operators need to possess Privilege Level 1 to use of this object/group. You can also combine this restriction with area restrictions. For example, if you assign the object/group to Area 5, with Privilege Level 2, the user needs to have Privilege 2 for Area 5.</p> <p>Click the menu to the right of this field to select a privilege, or type in an privilege number directly.</p> <p><b>Note:</b> If an object is part of a group, users need to have access to the group in order to have access to the object.</p>
<b>[Logging] Log device</b>	<p>This is the device to which messages will be logged for the object/group's keyboard and touch commands. Click the menu to the right of the field to select a device, or type a device name.</p> <p><b>Note:</b> You need to include the <i>MsgLog</i> field in the format of the log device for the message to be sent.</p>

## See Also

[Edit Common Object Properties](#)

### Disable Access to Objects

If you need to, you can disable access to objects.

**To disable access to an object:**

1. Double-click the object.
2. Click the **Access** tab.
3. Click the **Disable** tab.
4. Specify the condition which will disable the object as well as the appearance of the object when disabled.
5. Click **OK**.

### Disable Access Properties

Objects and groups have the following access (disable) properties

Property	Description
<b>Disable when</b>	<p>The object/group will be disabled whenever the expression entered here (254 characters maximum) is true. If the object/group is disabled, the operator will not be able to use any form of input, such as sliders, touch commands, keyboard commands and so on.</p> <p>To insert a tag or a function, click the <b>Wizard</b> button to the right of this field. This button displays two options: <b>Insert Tag</b> and <b>Insert Function</b>.</p> <p><b>Note:</b> You can also insert equipment.item references into expression fields using the <b>insert tag</b> option; however if no equipment has been configured in your system the list will be empty. You will need to configure equipment to populate the list with available tags.</p> <p>There are three ways of indicating a disabled object/group: Embossed, Grayed, and Hidden.</p>
<b>[Disable when] Disable on insufficient area or privilege</b>	The object/group will be disabled if an operator does not meet the area and privilege right defined in the Access.
<b>Disable style:</b>	<ul style="list-style-type: none"><li>• <b>Embossed</b> - When disabled, the object/group will look as if it has been embossed on the graphics page.</li></ul>

Property	Description
	<ul style="list-style-type: none"><li>• <b>Grayed</b> - When disabled, the object/group will be grayed out (all color detail will be removed).</li><li>• <b>Hidden</b> - When disabled, the object/group will be entirely hidden from view.</li><li>• <b>Unchanged</b> - When disabled, the object/group will not change.</li></ul> <p><b>Note:</b> If a group is disabled, objects in that group are also disabled. The disabled style of the group is applied to all the objects within the group. When the group is enabled, each object uses the disabled style of itself.</p>

## See Also

[Edit Common Object Properties](#)

## Metadata

Metadata is a list of names with corresponding values that is attached to an object's animation point. When using any of the [DspAnGetMetadata](#) Cicode functions this Metadata is processed, with the field values retrieved, and set at runtime. At runtime Metadata cannot be added or removed.

Metadata can be implemented when configuring the following animation objects:

- Free hand line
- Straight line
- Rectangle
- Ellipse
- Polygon
- Pipe
- Text
- Number
- Button
- Symbol Set
- Trend
- Paste Symbol
- Active X
- Process Analyst
- Database Exchange Control.

**Note:** Metadata can also be assigned to super genie associations. See [Passing Animation Point Metadata as](#)

---

[Super Genie Associations](#) for more information.

### To add a Metadata entry:

1. Click Add or double click on the row to make it editable.
2. Insert the **Name** of the metadata.
  - To start with an alphanumeric or "\_" character
  - To contain only those characters that are allowed as part of the tag name syntax (excluding the period character '.' and reserved keywords), and not contain any white space.
  - Be no more than 253 characters in length
  - To be unique across a specific animation point.
  - The name cannot be changed at runtime.
  - The name can contain Genie substitutions
3. Insert the corresponding **Value** of the specific metadata entry.
  - The Value can contain Genie substitutions. E.g. %Level%
  - To contain only those characters that are allowed in variable tag names or 'equipment.item' tag references, and not contain any white space.
  - To be no more than 255 characters in length.
  - Literal strings are supported and be specified within single quotes e.g.'Literal'.
- **Note:** At runtime values can be changed, however when the graphics page is closed the values will revert to the original configuration.
4. Click Add to save the row. The next row is highlighted and is now editable. Click on the ESC key to cancel this row or continue to enter name | value pairs until finished.

### To edit a metadata entry:

1. Double-click the cell you want to edit, or select a row and click the edit button.
2. Modify the Name or Value . Click Apply to bring your changes into effect and then OK to close the graphic object properties dialog.

Refer to [Using Metadata](#) for examples on how you can implement metadata in your project.

## See Also

[Edit Common Object Properties](#)

## Using Metadata

The following examples provide a guide into how you can implement metadata within your project. In the first example you will set parameters using metadata, and in the second example see how genie substitutions can be used in metadata.

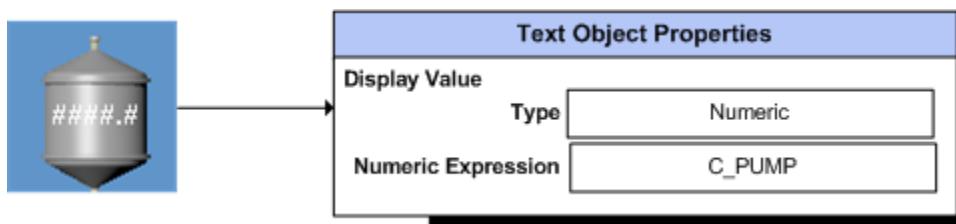
In the first two examples a paint factory needs to control the amount of Cyan, Magenta, Yellow, and Key (Black) (CMYK), which is mixed to create different colors in their color chart.

The third example illustrates how substituting the equipment name, in an equipment.item reference when

configuring a Super Genies lessens the need to define large amounts of metadata in the genie calling it.

## Example 1: Setting Parameters Using Metadata

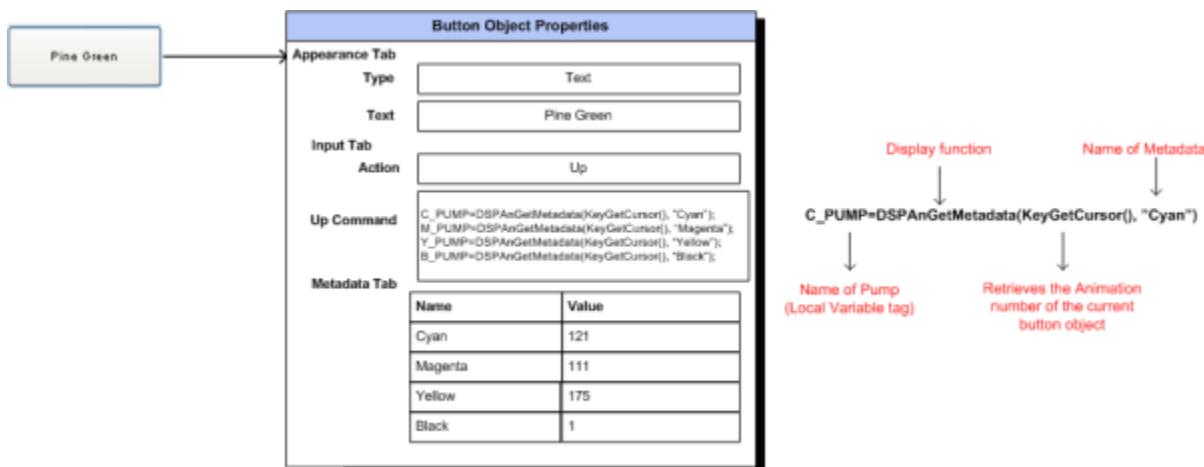
- From the toolbox select the 'number' tool and in the Text Properties dialog configure as below. This is the pump that represents the color Cyan.



- Repeat to create the remaining pumps M\_PUMP, Y\_PUMP, B\_PUMP

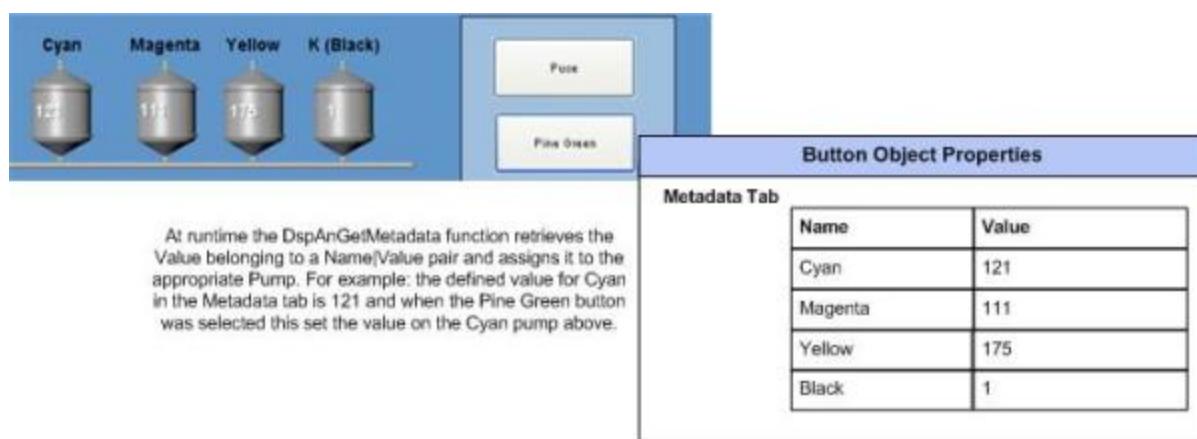
**Note:** Add these pumps to the example project as Local Variable tags

- Create a button object. This button when selected at runtime will set the CMYK values for a specific color e.g Pine Green



**Note:** For Display functions DspAnGetMetadataAt, DspAnSetMetadata, DspAnGetMetadata, and DspAnSetMetadataAt -2 is the default value used to retrieve the unique animation number for the button object. When you need to know the AN that triggered the input/command, the KeyGetCursor() function may be used as it returns the AN where the cursor is currently positioned.

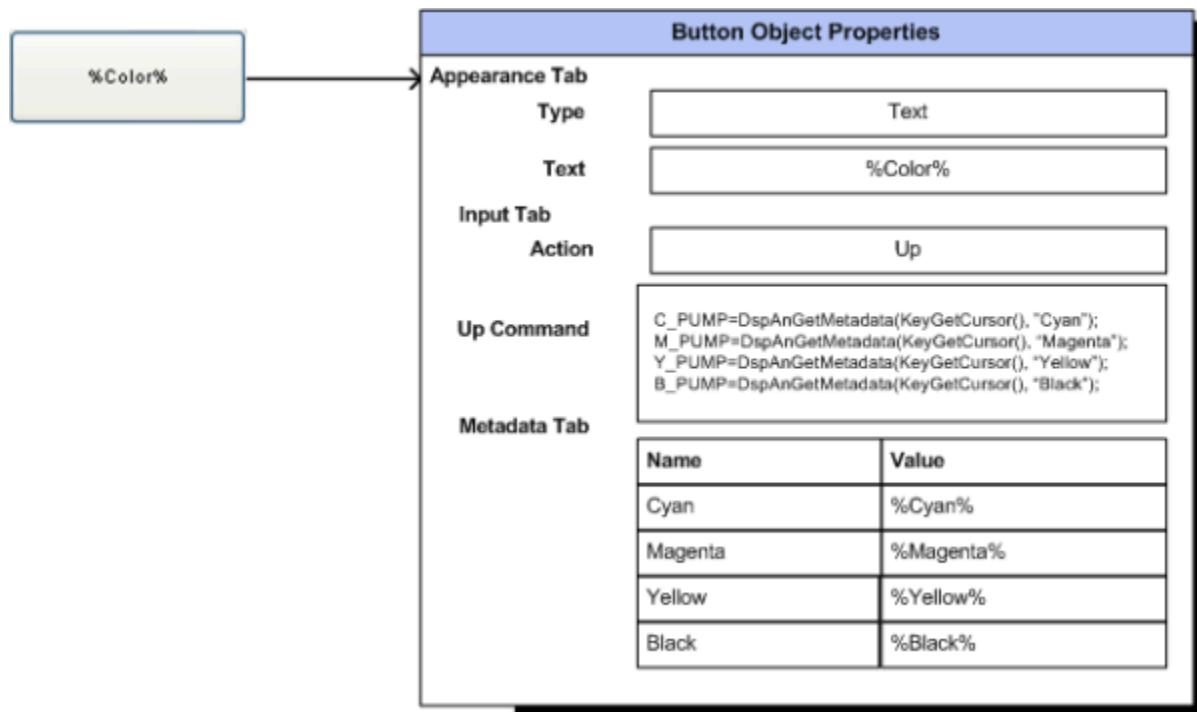
- At runtime when the 'Pine green' button is selected the values of the metadata defined for Cyan, Magenta, Yellow and Black are retrieved and the relevant pumps set.



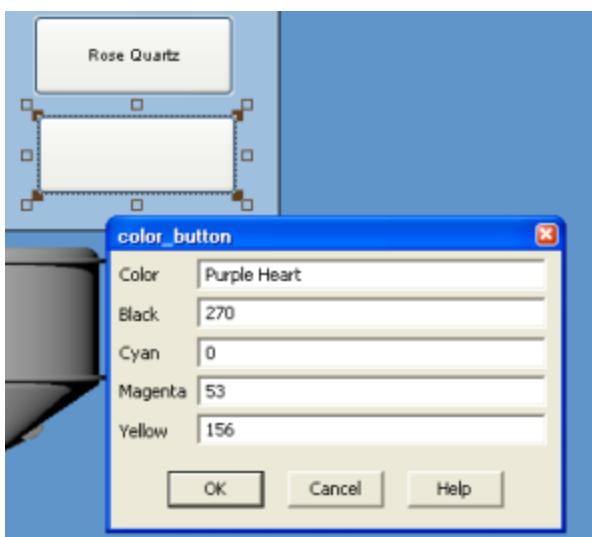
## Example 2: Using Genie Substitutions in Metadata

Instead of creating a new button for every color as you did in the previous example, you can create a genie and save it in the genie library. You can use the genie every time you need to create a new color button. Only needing to configure the name of the button and the CMYK values that belong to it.

1. From File New-> Genie
2. Add a button object onto the page
3. Configure the button – the title of the button and the Value of the Metadata have been defined using genie substitutions e.g %Color% and %Cyan%.



4. Save the genie in the appropriate genie library
5. At design time paste the genie onto the graphics page. A dialog will open prompting you for the name of the button, and the numeric values for Cyan, Magenta, Yellow and Black. In this example the name of the color is "Purple Heart"



6. At runtime when the 'Purple Heart' color button is selected the values of the metadata (that were entered in the genie prompt) are retrieved and the relevant pumps set.

### Example 3: Using Partial Associations in Metadata

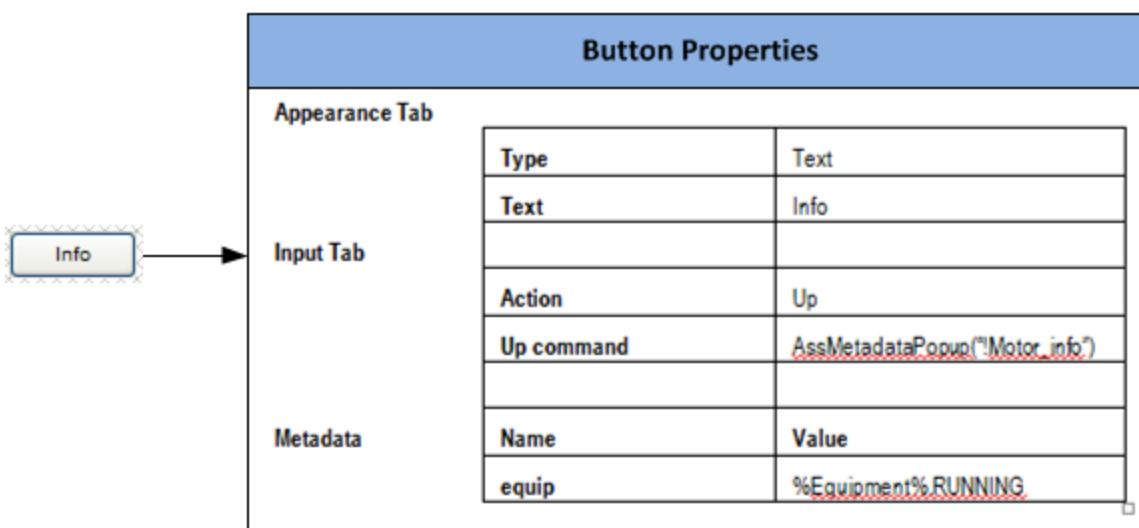
Partial associations, where only part of the variable tag, or equipment.item reference is substituted (with a name) in the Super Genie, can now be used lessening the need to define large amounts of metadata which is not easily maintainable.

In this example you will:

- Create a Genie.
- Create a Super Genie Library object.
- Attach the Super Genie to the Genie.
- Paste onto a graphics page.

#### Create the Genie

1. In Graphics Builder go to File | New | Genie.
2. A new genie will open.
3. From the graphics toolbox select the select the Button tool and click on the page to position it.
4. Draw the button and in the Button properties dialog configure the following:



5. Click **OK** to close the dialog and save the button properties
6. Click the Save tool, or choose **File | Save**.
7. Select the Project and Library in which to store the Genie.
8. Name the Genie e.g. Motor\_Genie and click **OK** to save

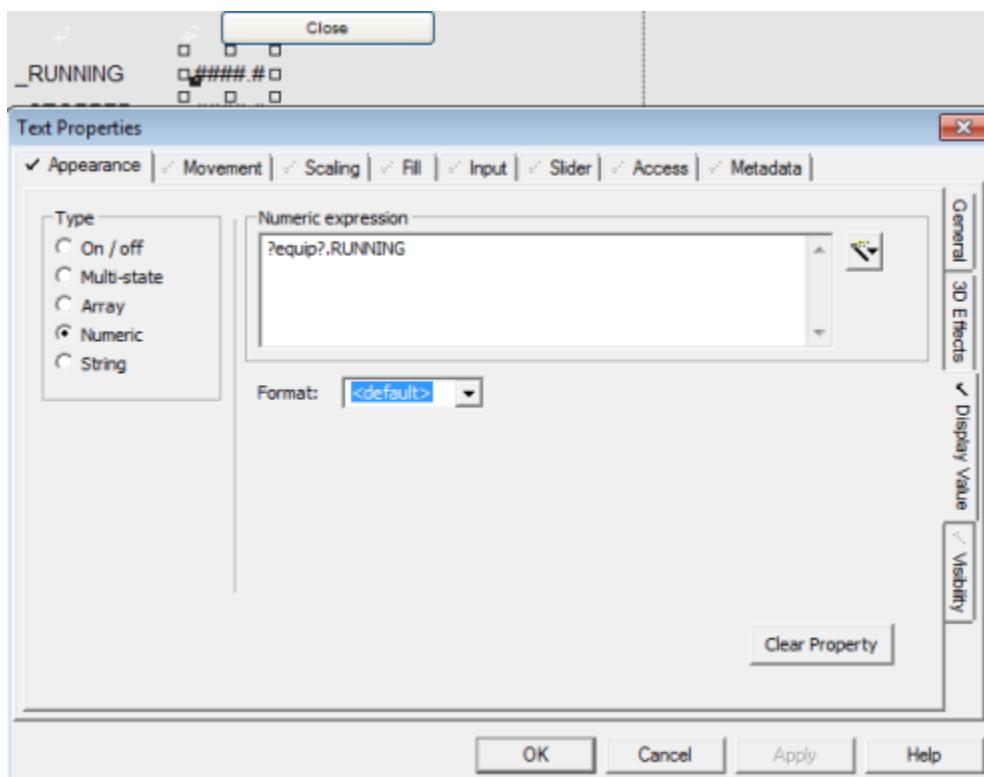
### Create a Super Genie Library Object

Close	
_RUNNING	#####.#
_STOPPED	#####.#
_AUTO	#####.#
_LOCAL	#####.#
_OutOfService	#####.#
_FAIL	#####.#
_ILOCK	#####.#
_STARTCMD	#####.#
_STOPCMD	#####.#
_ILOCKCMD	#####.#
_RESETCMD	#####.#
_FAILSTART	#####.#
_FAILSTOP	#####.#

### To create the Super Genie shown above:

1. From the toolbar select New, or choose File | New.
2. Click the Super Genie option.
3. A blank Super Genie library object page will open.
4. In this example a pop up that displays 13 values will be created.
5. Select the Text tool and click to position the text on the page and to open the Text properties dialog.
6. In the text properties dialog enter \_RUNNING.
7. Repeat steps 5 and 6 for the remainder of the labels.

8. Select the Numeric tool from the graphic toolbox and position next to the first label e.g \_Running.
9. Click to paste onto the page and to open the Text Properties dialog. Configure the Numeric expression as below with ?equip? being the substitution name.



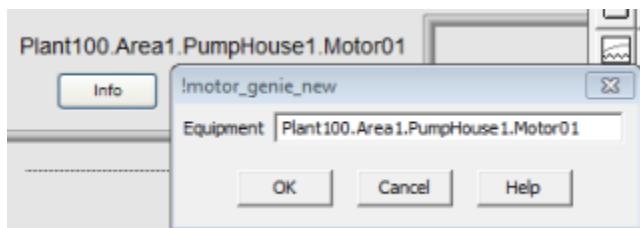
10. Click **OK** to save the text object and close the dialog.
11. Repeat steps 9 and 10 for the remaining equipment.items, matching the label with the end of the expression. For example ?equip?.STOPPED.
12. Click **OK** to apply the changes.
13. Select **File | Save**.
14. In the Save dialog, select the project the page will belong to.
15. Name the Super Genie page (e.g.!Motor\_Info) and click **OK**.
16. Once saved you will need to Attach the Super Genie to the Genie you created previously.

### Attach the Super Genie to a Genie

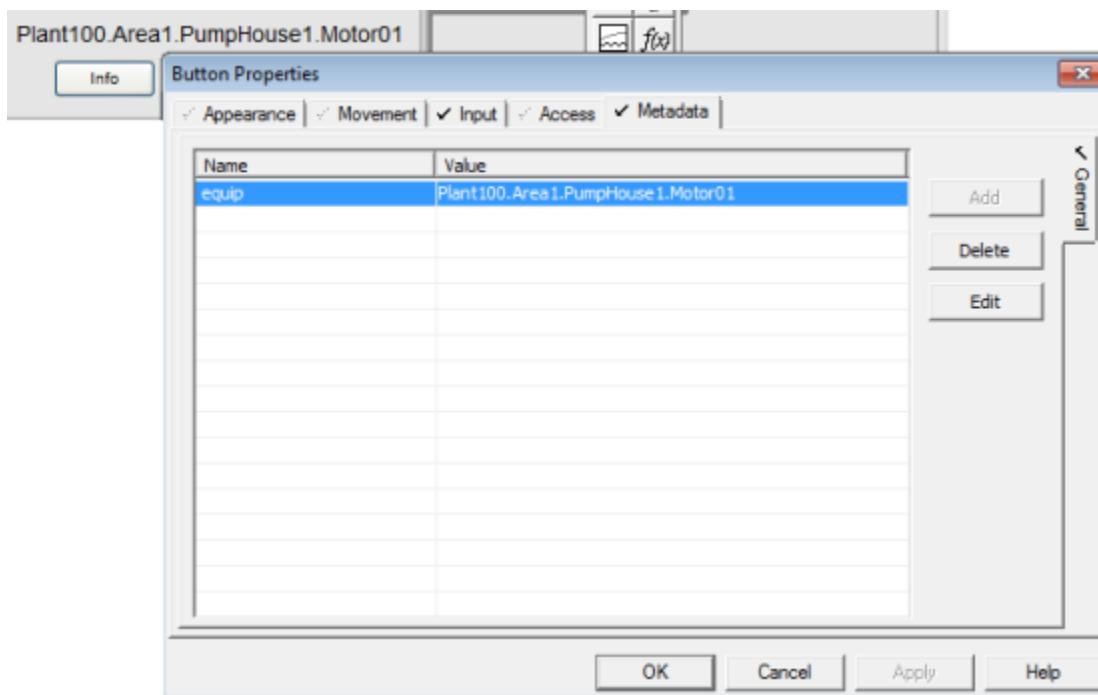
1. Open the Genie.
2. With the Genie open, select **Edit | Attach Super Genie**.
3. Click **Add**, and the Super Genie Dialog box will open.
4. Select the library the Super Genie belongs to.
5. Select the Super Genie to attach e.g. !motor\_info and click **OK**.
6. The Super Genie is added to the Attach Super Genie list.
7. Click **OK** to save the changes, or click **Cancel**.

## Paste onto Graphics Page

1. Create or open an existing graphics page.
2. From the graphics page select Edit | Paste Genie, browse for the genie you created previously and select it.
3. On pasting the genie, the dialog will open prompting you to fill in the 'Equipment' field with the name of equipment for example Plant100.Area1.PumpHouse1.Motor01.



4. The Equipment name entered will replace the value in the metadata tab (in your button as part of the genie). For example %Equipment% becomes Plant100.Area1.PumpHouse1.Motor01.



At runtime, when the 'info' button is pressed the values of the equipment are dynamically generated and assigned to the matching Super Genie Association. E.g. All occurrences of the substitution name ?equip? will be replaced with Plant100.Area1.PumpHouse1.Motor01 (creating the full equipment.item reference) and the values for the equipment.item displayed.

!motor_info	
<b>Close</b>	
_RUNNING	1.00
_STOPPED	0.00
_AUTO	0.00
_LOCAL	0.00
_OutOfService	0.00
_FAIL	0.00
_ILOCK	0.00
_STARTCMD	0.00
_STOPCMD	0.00
_ILOCKCMD	0.00
_RESETCMD	0.00
_FAILSTART	0.00
_FAILSTOP	0.00

## See Also

[Edit Common Object Properties](#)

### Passing Animation Point Metadata as Super Genie Associations

The metadata you define can be assigned to Super Genie associations. The name of the metadata and the name of the association in the Super Genie need to be the same. If the name matches, the value defined for that name | value pair is then inserted into the relevant 'association' field in the Super Genie. At runtime the value is dynamically generated and then displayed on the Super Genie page.

When configuring the object that will call the Super Genie, you can use a Cicode function that will perform the associations individually (DspAnGetMetadataAt()) or at once (AssMetadata()).

The table below outlines the possible display results when passing animation point metadata as Super Genie associations:

where - Y = Defined, N= Not Defined, \* =N/A, and

**VoE** - used when Tag value is not resolved.

**Default Value** is used when the Metadata Value is not defined

**Empty String** is treated as "Not defined"

Animation Point Metadata Pairs		Associations			Display Results at runtime		
Name	Value	Name	Default Value	Value on Error	Tag Resolution	Unresolved Tag	Literal
Y	Y	Y	Y	Y	Tag Value	VoE	Literal
Y	Y	Y	Y	N	Tag Value	#ASC	Literal

Y	Y	Y	N	Y	Tag Value	VoE	Literal
Y	Y	Y	N	N	Tag Value	#ASC	Literal
Y	Y	N	*	*	Tag Value	#ASC	Literal
Y	N	Y	Y	Y	Default Value	VoE	Default Value (literal)
Y	N	Y	Y	N	Default Value	#ASC	N/A
Y	N	Y	N	Y	#ASC	*	*
Y	N	Y	N	N	#ASC	*	*
N	N	N	*	*	#ASC	*	*
N	*	Y	Y	Y	Default Value	VoE	Default Value (literal)
N	*	Y	Y	N	Default Value	#ASC	N/A
N	*	Y	N	Y	#ASC	*	*
N	*	Y	N	N	#ASC	*	*
N	*	N	*	*	#ASC	*	*

## See Also

[Page Properties - Associations](#)

[Super Genies](#)

## Alarm Indicator

An alarm indicator can be used to show the occurrence and status of alarms associated with an object group or Genie. It consists of two main elements:

- Alarm border
- Alarm flag.



For more information, see [Use Alarm Indicators](#).

**Note:** On Composite Genies, alarm indicators are displayed around each piece of related equipment included in the Composite Genie. You can also group individual independent objects, such as pumps, into an equipment group and apply an alarm indicator to the group. In this case, the equipment should be defined as "hidden" with its **Hidden** property set to TRUE.

### To apply an alarm indicator to an object group or Genie:

1. Display the properties for the object group or Genie to which you would like to add an alarm indicator.
2. Click the **Alarm Indicator** tab.
3. Select the **Has Alarm Indicator** check box to enable an alarm indicator.
4. In the **Highest priority alarm from** section, enter the **Equipment** that will be used to determine the highest priority alarm for the alarm indicator.
5. Select the **Include equipment references in count** check box to include equipment references when determining the highest priority alarm for the alarm indicator.
6. Configure the **Border Appearance** and **Flag Appearance** properties (see below).
7. Click **OK**.

When you return to Graphics Builder, the alarm indicator will appear on the graphics page around the object group or Genie.

#### Note:

- Using an Alarm Indicator with a group of Trend objects is not recommended.
- Genies that include legacy objects (from the v3.x/v4.x toolbox), do not support Alarm Indicators. If a Genie contains both a legacy object and a new object, then at runtime the alarm indicator will only be shown around the new object. If a Genie contains only legacy objects, an alarm indicator will not display.

## Alarm Indicator Properties

Property	Description
<b>Has Alarm Indicator</b>	Select this check box to apply an alarm indicator to the object group or Genie.
<b>[Highest priority alarm from]</b> <b>Equipment</b>	Specifies the location in the equipment hierarchy that will be used to determine the highest priority alarm for the alarm indicator. You can choose to include alarm conditions that occur at locations down the equipment hierarchy (referred to as "children") by selecting one of the following options: <ul style="list-style-type: none"><li>• Equipment and Children</li><li>• Equipment Only</li><li>• Children Only.</li></ul> Be aware that any triggering alarms that occur down the equipment hierarchy will not appear in the

Property	Description
	<p>Information Zone when the object is selected (see note below).</p> <p>By default, the page cluster context will be used. However, you can specify a particular cluster using dot (.) notation.</p> <p>You can also use Genie and Super Genie syntax.</p>
<b>[Highest priority alarm from]</b> <b>Include equipment references in count</b>	<p>Select this check box to include equipment references when determining the highest priority alarm for an alarm indicator.</p> <p>Be aware that any triggering alarms that occur in referenced equipment will not appear in the Information Zone when the object is selected (see note below).</p>
<b>[Border Appearance]</b> <b>Is Inside</b>	<p>Select this check box to locate the border within the extent of the object group or Genie. This means no additional space will be required to support a border.</p> <p>If this option is selected, <b>Padding</b> is disabled.</p>
<b>[Border Appearance]</b> <b>Width</b>	<p>Specifies the width of the alarm border (in pixels).</p>
<b>[Border Appearance]</b> <b>Padding</b>	<p>Specifies the amount of space between the extent of the object group or Genie and the inside edge of the alarm border (in pixels).</p>
<b>[Border Appearance]</b> <b>Frame Color</b>	<p>Specifies the color used for the inside and outside edge of the alarm border. Use the drop-down to select a color.</p>
<b>[Flag Appearance]</b> <b>Show Flag</b>	<p>Select this check box to apply a flag to the alarm indicator.</p>
<b>[Flag Appearance]</b> <b>Position</b>	<p>Select the position for the alarm flag on the alarm border. The alarm flag will be attached to the outside edge of the alarm border at the specified location.</p>

**Note:** If you configure an alarm indicator to respond to alarm conditions that occur down the equipment hierarchy ("children") or within referenced equipment, these indirect alarms will not appear in the Information Zone when the object is selected at runtime. If this occurs for an object with an active alarm indicator, use a higher level alarm count or an alarm page to determine the location of the active alarm.

## See Also

[Edit Common Object Properties](#)

## Manipulate Graphics Objects

Objects can be manipulated in various ways, such as moving, resizing, and grouping.

- [Select Objects](#)
- [Move Objects](#)
- [Resize Objects](#)
- [Delete Objects](#)
- [Lock/Unlock Objects](#)
- [Group Objects](#)
- [Copy and Paste Objects](#)
- [Change the Overlap of Objects](#)
- [Align Objects](#)
- [Rotate Objects](#)
- [Mirror Objects](#)
- [Locate an Object](#)
- [Adjust Position and Size Dialog](#)

### Select Objects

You can select a single object or a group of objects at a time.

#### To select a single graphics object:

1. In Graphics Builder, click **Select**.
2. Click the object. The object's sizing handles appear, and the cursor changes from an arrow to a hand while on the object.

---

**Note:** To add other objects to the selection, hold the **Ctrl** key and click each object in turn. To deselect an object from a group selection, while still holding the **Ctrl** key, click the object again. To select every object in the drawing, use **Select All** from the Edit menu. To deselect every object, click anywhere other than on an object.

---

#### To select a group of graphics objects using a marquee box:

1. In Graphics Builder, click **Select**.
2. Click and hold the left mouse button and drag the cursor across the page. This creates a temporary bounding box. Any objects within the box will be selected when you release the mouse button.
3. Release the mouse button.

---

**Note:** To add other objects to the selection, or remove objects from the selection, hold the **Ctrl** key and click each object in turn. To quickly select every object in the drawing, you can use **Select All** from the Edit menu or the keyboard shortcut **Ctrl +A**. To deselect every object, click anywhere other than on an object.

---

## See Also

[Manipulate Graphics Objects](#)

[Move Objects](#)

## Move Objects

You can move an object by doing one of the following:

- Click the object and drag it to a new location.
- Select the object and use the direction keys (left, right, up and down arrows) and diagonal keys (page up, page down, end, and home) on the keyboard to position the object to a new location.
- Use the Adjust Position and Size dialog to move the object to a new location set by X and Y co-ordinates. See [Adjust Position and Size Dialog](#) for more information.

If you move an object as soon as you select it, an outline of the object boundary displays as you move it on your page. If you hold the mouse stationary for 1 second or more before you move the object, the object itself displays as you move it, enabling you to better see the result for more accurate placement.

## See Also

[Manipulate Graphics Objects](#)

[Resize Objects](#)

## Resize Objects

You can re-size objects and manage a node of an object in the following ways.

### To resize an object:

1. Select the object, and then move the cursor over a sizing handle. The cursor changes to a two-sided arrow showing the directions that you can drag the handle to re-size the object.
2. Click and drag the handle to a new location. The object's bounding box appears as you drag.
3. Release the mouse button.

Select a sizing handle at a corner of the object to change the height and width at the same time. If you hold the **CTRL** key while you move a corner sizing handle, the object maintains its aspect ratio (that is, a square remains a square).

Or:

Select the object, and then hold the **SHIFT** key while you press any of the direction keys to resize the object. The selected object is resized in the appropriate direction, anchored to the top-left corner.

If you hold the **CTRL** and **SHIFT** key while you press any of the direction keys, the object maintains its aspect ratio (that is, a rectangle remains a rectangle).

You can also use the Adjust Position and Size dialog to move the object to a new location set by X and Y co-ordinates. See [Adjust Position and Size Dialog](#) for more information.

**To select an object's node:**

1. Select the object. Node selection is only applicable to a Line, Pipe, Polyline, or Polygon object.
2. Position and hold the mouse pointer over the node. The mouse pointer will change to a small cross shape.
3. Click the left mouse button. The color of the selected node will change to the inverse of the background (light on a dark background, dark on a light background).

If you have a node selected and then click another node within the same object, the first node will deselect. To select multiple nodes, hold down the **Ctrl** key and click each node you want to select. (With the **Ctrl** key held down, you can click a previously selected node to deselect it.) Clicking the same node again toggles the selection of the node alternately on and off. To deselect every node, click anywhere other than on a node.

**To move a node of an object:**

1. Select the node(s).
2. Position and hold the mouse pointer over the node. The mouse pointer will change to a small cross shape.
3. Click and hold the left mouse button. The cursor changes to a positioning symbol.
4. Drag the selected node(s) to the desired position.
5. Release the mouse button.

Selecting and moving multiple nodes maintains the aspect ratio of the graphic object between the selected nodes.

**To add a node to an object:**

1. Select the object.
2. Position and hold the mouse pointer directly over the graphic object at the exact point where the new node will be added. The mouse pointer will change to a pointing hand shape.
3. Press **Insert**, or either of the available plus (+) keys.

Depending upon the keyboard you're using, the plus key could be either on the number pad section, or accessed on the main keyboard via the **Shift** key.

**To delete a node from an object**

1. Select the node(s).
2. Press **Delete** or a minus (-) key.

If no nodes are selected, pressing the **Delete** or minus keys deletes the object.

**See Also**

[Manipulate Graphics Objects](#)

[Reshape Objects](#)

[Adjust Position and Size Dialog](#)

## Reshape Objects

Pipe, Polyline, or Polygon objects can be edited to change their shape. Each of these objects consist of a continuous series of lines drawn between structural anchor points called nodes. Nodes are visible when an object is selected. Each node appears as a small square located at specific anchor points along the object. There will be a node located at the start and end of a polyline or pipe, and at every change of direction in an object's shape.

Pipe, Polyline, and Polygon objects can have their shapes changed in many ways. Their nodes can be selected individually or by group and moved to a different position, thus changing the shape of the object. The Pipe, Polyline, and Polygon objects also support node adding and deleting.

Line objects also have nodes, but behave in a more restricted manner than Pipe, Polyline, or Polygon objects.

A straight line can only consist of two nodes, (a start node point and an end node point). These can be individually selected to move the line to a different position, or at least change its direction. The Line object does not support node adding. To achieve the same result as adding a node to a Line object, create a Polyline object instead. Deleting either of the (only two) nodes of a Line object will delete the whole Line object completely.

## See Also

[Manipulate Graphics Objects](#)

[Delete Objects](#)

## Delete Objects

You can delete unwanted objects.

**To delete an object (or a group of objects):**

1. Select the object (or group of objects)
2. Choose **Edit|Delete** or press the **Delete** key (or a minus (-) key).

## See Also

[Manipulate Graphics Objects](#)

[Lock/Unlock Objects](#)

## Lock/Unlock Objects

On complex drawings (with many objects), selecting a discrete group of objects without including every object (in the selected area) can be difficult (for example when an object is hidden by another object). To avoid unintentionally selecting an object, you can 'lock' it in position. When an object is 'locked', it cannot be selected, deleted, moved, or edited. Objects are locked only when the **Edit** menu **Break Lock Mode** option is not selected.

**To lock an object:**

1. Select the object.
2. Choose **Edit | Lock Object**.

**To unlock an object:**

1. Choose **Edit | Break Lock Mode**.
2. Select the object.
3. Choose **Edit | Unlock Object**.

**See Also**

[Manipulate Graphics Objects](#)  
[Group Objects](#)

**Group Objects**

You can group objects to make them easier to manipulate.

**To group objects:**

1. Click **Select**.
2. Select the objects to group, and then click **Group** (or choose **Arrange | Group Objects** or **Ctrl + Shift + G**).

**To ungroup objects:**

1. Click **Select**.
2. Select the objects to group, and then click **Ungroup** (or choose **Arrange | Ungroup Objects** or **Ctrl + Shift + U**).

**To change the properties of a group:**

1. Click **Select**.
2. Double-click the group. The Properties dialog box appears.
3. Change the properties as necessary.
4. Alternatively, choose **Tools | Goto Object**, select the group, and then click **OK**.

**See Also**

[Manipulate Graphics Objects](#)  
[Copy and Paste Objects](#)

**Copy and Paste Objects**

You can copy and paste objects onto other graphics pages. You can use the clipboard to transfer objects between different graphics pages and from other graphics applications.

**To copy an object to the clipboard:**

1. Click **Select**.

2. Select the object (or group of objects).
3. Click **Copy** or choose **Edit | Copy**.

**To paste an object (or group of objects) from the clipboard:**

1. Click **Paste** or choose **Edit | Paste**.

**To cut (remove) an object:**

1. Click **Select**.
2. Select the object (or group of objects).
3. Click **Cut** or choose **Edit | Cut**.

**To cancel your last drawing operation(s):**

1. Click **Undo**, or choose **Edit | Undo**.

You can undo operations performed during the current drawing session apart from edits to bitmaps.

**To cancel the Undo (or Redo) the last drawing operation(s):**

1. Choose **Edit | Redo**.

## See Also

[Manipulate Graphics Objects](#)

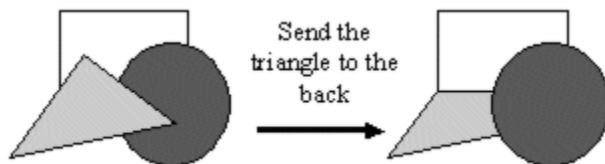
[Change the Overlap of Objects](#)

## Change the Overlap of Objects

In the Graphics Builder, objects can overlap. Where there is an overlap, new objects are placed on top of existing objects. You can move objects backwards and forwards through this display sequence to change the way they overlap. An object can be moved backwards and forwards one step at a time (rather than completely to the back, or completely to the front).

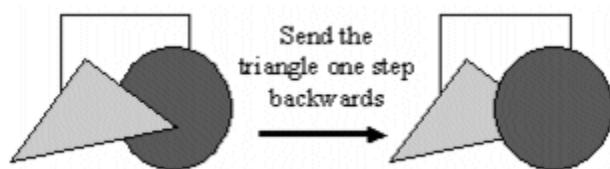
**To position an object (or group of objects) behind other objects so that objects overlap it:**

1. Click **Select**.
2. Select the object (or group of objects).
3. Click **Send to Back** or choose **Arrange | Send to Back**.

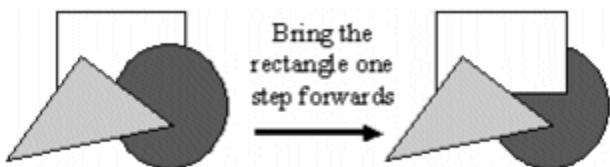


**To send an object (or group of objects) one step backwards:**

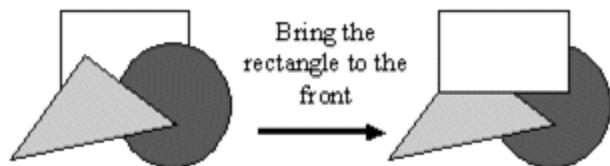
1. Click **Select**.
2. Select the object (or group of objects).
3. Click **Send Backwards** or choose **Arrange | Send Backwards**.

**To bring an object (or group of objects) one step forwards:**

1. Click **Select**.
2. Select the object (or group of objects)
3. Click **Bring Forwards** or choose **Arrange | Bring Forwards**.

**To bring an object to the front:**

1. Click **Select** tool.
2. Select the object (or group of objects).
3. Click **Bring to Front** or choose **Arrange | Bring to Front**.

**See Also**

[Manipulate Graphics Objects](#)

[Align Objects](#)

**Align Objects**

You can precisely align a group of objects vertically, horizontally, or both.

**To align objects:**

1. Click **Select**.
2. Select the objects.
3. Choose **Arrange | Align** or **(Ctrl + Shift + A)**. The Align dialog box appears.

Option	Description	
Vertical	Top	Aligns the top edges of the selected objects
	Center	Vertically aligns the midpoints of the selected objects
	Bottom	Aligns the bottom edges of the selected objects
	Even	Vertically aligns the midpoints of the selected objects with even spacing
	None	Doesn't change the vertical alignment of the selected objects
Horizontal	Left	Aligns the left edges of the selected objects
	Center	Horizontally aligns the midpoints of the selected objects
	Right	Aligns the right edges of the selected objects
	Even	Horizontally aligns the midpoints of the selected objects with even spacing
	None	Doesn't change the horizontal alignment of the selected objects

**See Also**[Manipulate Graphics Objects](#)[Rotate Objects](#)**Rotate Objects**

You can rotate an object 90° right (clockwise) or 90° left (counter-clockwise).

**To rotate an object (or group of objects):**

1. Click **Select**.
2. Select the object(s).
3. Choose **Arrange | Rotate or (Ctrl + Shift + R)**.

**To rotate a text object:**

1. Click **Select**.
2. Select the object(s).
3. Choose **Tools | Convert to Bitmap**.
4. Choose **Arrange | Rotate or (Ctrl + Shift + R)**.
5. Select the direction to rotate the object (or group of objects).

The object(s) are rotated 90 degrees in the direction you select. To rotate the object(s) 180 degrees, click the direction button twice.

**See Also**

[Manipulate Graphics Objects](#)

[Mirror Objects](#)

**Mirror Objects**

You can mirror an object relative to its horizontal or vertical axis.

**To mirror an object (or group of objects) relative to its horizontal or vertical axis:**

1. Click **Select**.
2. Select the object(s).
3. Choose **Arrange | Mirror or (Ctrl + Shift + M)**.
4. Choose the axis about which to mirror the object (or group of objects).

**See Also**

[Locate an Object](#)

[Manipulate Graphics Objects](#)

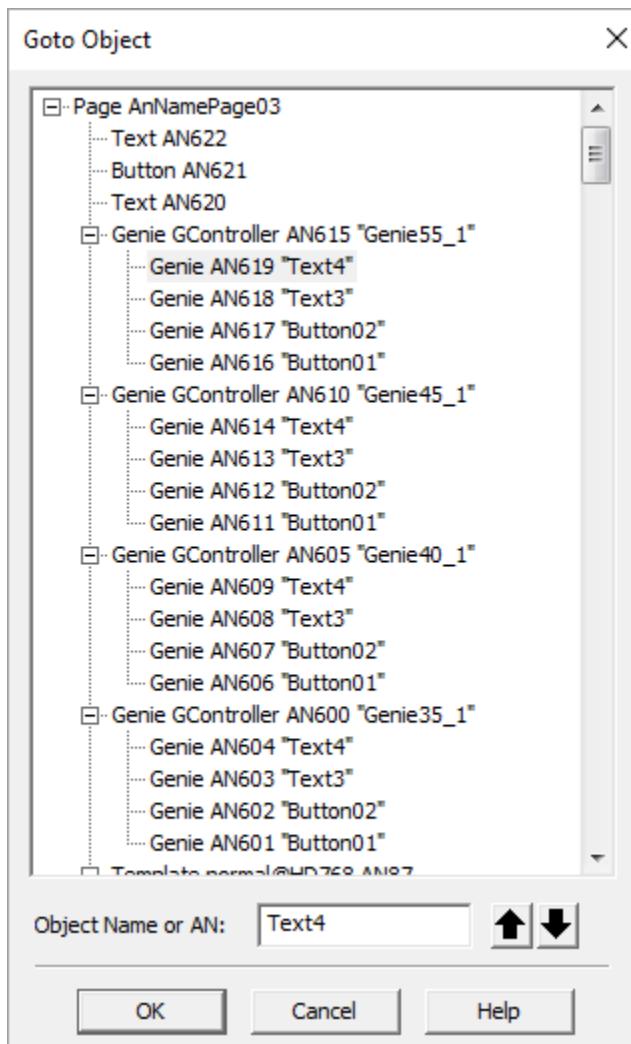
**Locate an Object**

You use the Goto Object dialog box to find, select and access the properties of objects on your current page or on page templates used by the current page. You can select objects, Genies, symbols, and groups on the page (or template) as well as the graphical elements that make up those Genies, symbols, and groups.

**To locate an object (or a group, Genie, symbol, or page template) on the current page and display its**

**properties:**

1. Choose **Tools | Goto Object**.
  2. Locate the object (or group, Genie, symbol, or page template) in the tree structure or type the relevant Animation number or Animation name in the **Object AN** box.
- Objects that are made up of several objects (or other graphical elements) have a plus sign (+) next to them. Click the + sign to see these component objects.
- The Animation name is unique to the page, page template, group, genie and graphics object it belongs to. When you search using a name, up and down arrows are displayed so you can scroll through each instance of the name on the page.



3. Double click the object in the tree structure, or click **OK** to display the object's properties.

**See Also**

[Manipulate Graphics Objects](#)

## Copy an Object to the Library

You can copy an object to the library so that you can use it later on other graphics pages.

### To copy an object to the library (and make it a symbol):

1. Click **Select**.
2. Select the object (or group of objects).
3. Choose **Edit | Copy to Library**. The Copy To dialog box appears.

## Copy To dialog box

This dialog box lets you copy an object (or group of objects) to the library as a symbol.

Feature	Description
Symbol	A name for the symbol.
Library	The project library where the symbol is stored.
Project	The project where the library is stored.
Preview Enable	Displays a thumbnail image of the symbol.

**Note:** To edit the symbol, select it and click **Edit**. To create a new symbol, click **New**.

## New Library dialog box

This dialog box lets you create a new library for the symbol, Genie, or Super Genie. Enter a **Name** for your new library (64 characters maximum).

## Adjust Position and Size Dialog

You can move and resize an object using the Adjust Position and Size Dialog.

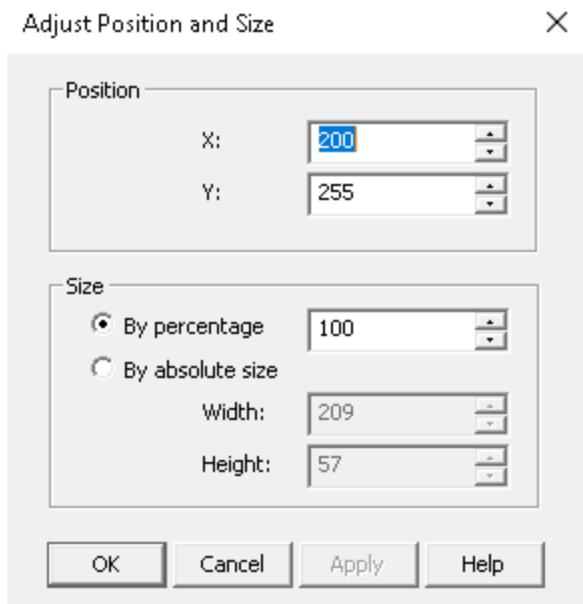
### To move or resize an object:

1. Open Graphics Builder.
2. Select an object or objects on a graphics page.
3. On the **Arrange** menu, select **Adjust Position and Size**.

Or:

Press the **CTRL** plus **Shift** plus **P** keys.

The Adjust Position and Size dialog will appear.



4. Move the object to a new position by adjusting the **X** and **Y** co-ordinates.
5. Resize the object by adjusting the **By percentage** value.

Or:

Adjust the **Width** and **Height** values.

The absolute size of the object is in pixels.

6. Click **Apply** to implement the changes.
7. Click **OK** to close the dialog.

## See Also

[Manipulate Graphics Objects](#)

[Resize Objects](#)

## Symbols

Symbols are a collection of graphics objects that can be used to represent a particular component of a system, from a simple switch to a complex piece of machinery such as a mixer or a tank. They are useful where a component needs to be represented multiple times across a project's graphics pages.

The appearance of a symbol will usually look like the physical piece of equipment it represents so that its meaning is immediately clear to an operator. You can also animate symbols so that they indicate the current operational state of the associated piece of equipment.

You can use symbols in the following ways:

- Paste a symbol to a graphics page, template, Genie, Super Genie or another symbol (see [Paste a Symbol](#))
- Paste a collection of symbols to a page as a "symbol set" that changes appearance in response to condition changes (see [Add a Symbol Set](#))
- Save a graphics object or grouped graphics object to a library as a symbol, allowing it to be reused (see [Save](#))

[a Symbol to a Library\)](#)

- Link a symbol to a library symbol so that any changes to the library are automatically instantiated (see [Paste a Symbol](#)).

A comprehensive range of symbols are installed with Plant SCADA. These symbols are stored in several libraries in the "Include" system project. To browse the symbols associated with the active project, use the **Libraries** view in the **Visualization** activity (see [Browse Libraries in Plant SCADA Studio](#)).

**Note:** To display the properties for one of the grouped objects that form a symbol, hold the Control (CTRL) key down and double-click the specific object. Alternatively, you can use the Goto Object tool, accessible via the **Tools** menu (see [Locate an Object](#)).

## See Also

[Create a Symbol](#)

[Open an Existing Symbol](#)

## Create a Symbol

You can create a new symbol from scratch, or you can create a symbol from objects selected on a graphics page.

### To create a new symbol:

1. Open Graphics Builder.
2. On the **File** menu, select **New**.

Or:

Select the **New** button.



3. On the New dialog box, click **Symbol**.

The symbol will open in Graphics Builder ready for editing. The small cross that appears in the middle of the workspace represents the symbol's anchor point.

4. Edit and save the symbol (see [Save a Symbol to a Library](#)).

### To create a symbol from selected objects:

1. Open the page that contains the required object(s).
2. Select the object(s) you would like to save as a symbol.
3. From the Edit menu, select **Copy to Library**. The Copy To dialog box will appear.
4. Edit the dialog using the steps described in [Save a Symbol to a Library](#).

**Note:** To delete a symbol from the library, select the symbol name and click **Delete**.

## See Also

[Open an Existing Symbol](#)

[Paste a Symbol](#)

## Open an Existing Symbol

**To open an existing symbol for editing:**

1. Open Graphics Builder.
2. On the **File** menu, select **Open**.  
Or:  
Select the **Open** button.



3. On the Open dialog box, select the **Symbol** tab.
4. Select the **Project** where the symbol is stored.
5. Select the **Library** where the symbol is stored.
6. Select the **Symbol** you would like to open.
7. Click **OK**.

You can now edit and save the symbol (see [Save a Symbol to a Library](#)).

---

**Note:** To delete a symbol from the library, select the symbol name and click **Delete**.

---

## See Also

[Paste a Symbol](#)

## Save a Symbol to a Library

You can store frequently used objects or groups of objects (including bitmap objects) in a library as [Symbols](#).

**To save a symbol to a library:**

1. Create a symbol using one of the methods described in [Create a Symbol](#).
2. On the **Copy To** or **Save As** dialog box, select the **Project** in which the symbol is to be stored.
3. Select the **Library** in which the symbol is to be stored.
4. Enter a **Name** for the symbol.
5. To display a preview of the symbol, select the **Enable** check box.
6. Click **OK**.

When you save an object in a library, the current properties of that object are saved with it. When you paste it as a symbol to a graphics page, they are used as defaults for the symbol.

---

**Note:** Pasted symbols have different **Appearance** properties to those of normal objects as you can only specify a visibility property.

---

## See Also

[Open an Existing Symbol](#)

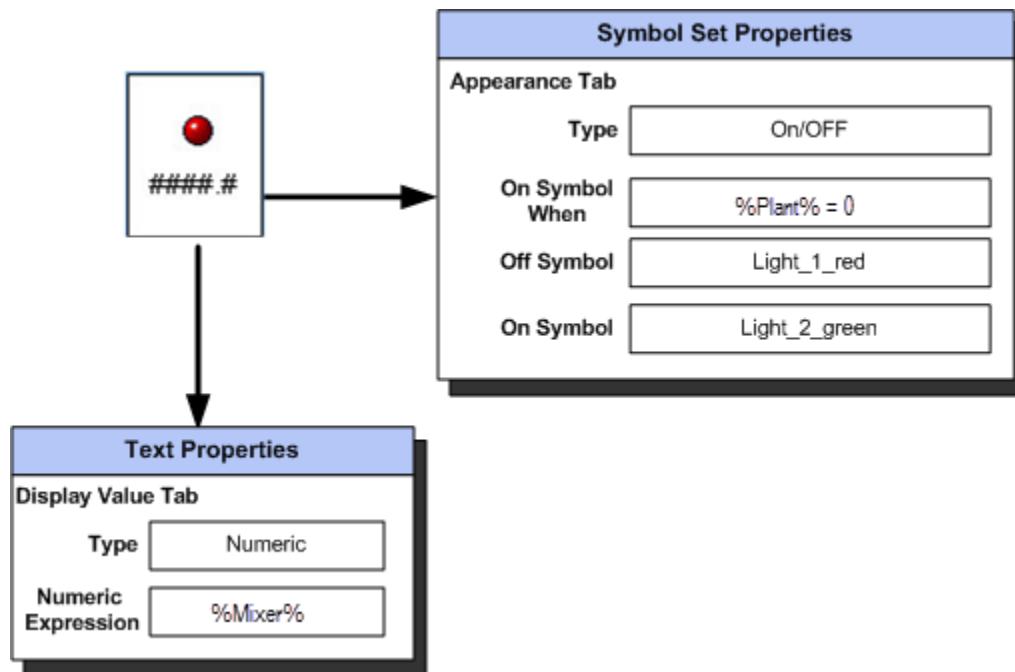
[Paste a Symbol](#)

## Genies

A Genie is a graphic object (or group of related graphic objects) that you configure and save in a Genie library for reuse in your project. When configuring the Genie, you use a substitution for part of or all of an object's expression. This means when you use the Genie, you only need to configure these substitutions and not all of the properties for the object or objects within the Genie.

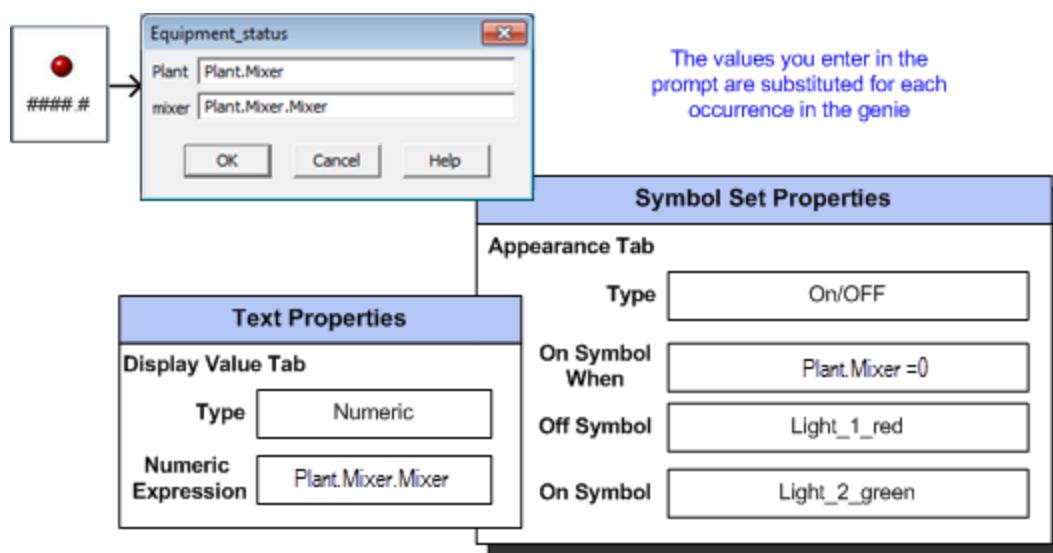
**Note:** To browse the Genies associated with the active project, use the **Libraries** view in the **Visualization** activity (see [Browse Libraries](#) in Plant SCADA Studio).

In the illustration below observe how part of the expression in the 'On Symbol When' field has been substituted with %Plant%. The percentage symbols (%%) denote that "Plant" is a Genie substitution (see [Define Genie Substitutions](#)).



When the Genie is pasted onto the graphics page a dialog will open, prompting you to configure the substitutions used in the Genie. The properties of a Genie can also be accessed by double clicking on the Genie.

In the illustration below observe how the field names in the Status Indicator match the 'substitutions' used when creating the Genie. For the Genie substitution 'Plant', an equipment reference is entered in the prompt.



During compilation the equipment, 'equipment.item' or variable tag you entered into the prompt replace each substitution in the Genie. When you reuse this Genie in the project, you can enter different references to customize each use of the Genie on a page.

**Note:** To display the properties of the objects in a Genie (instead of the Genie Properties), select the Genie and hold the Control (CTRL) key down and double-click the object. The object dialog will open with the properties of the object read-only.

## See Also

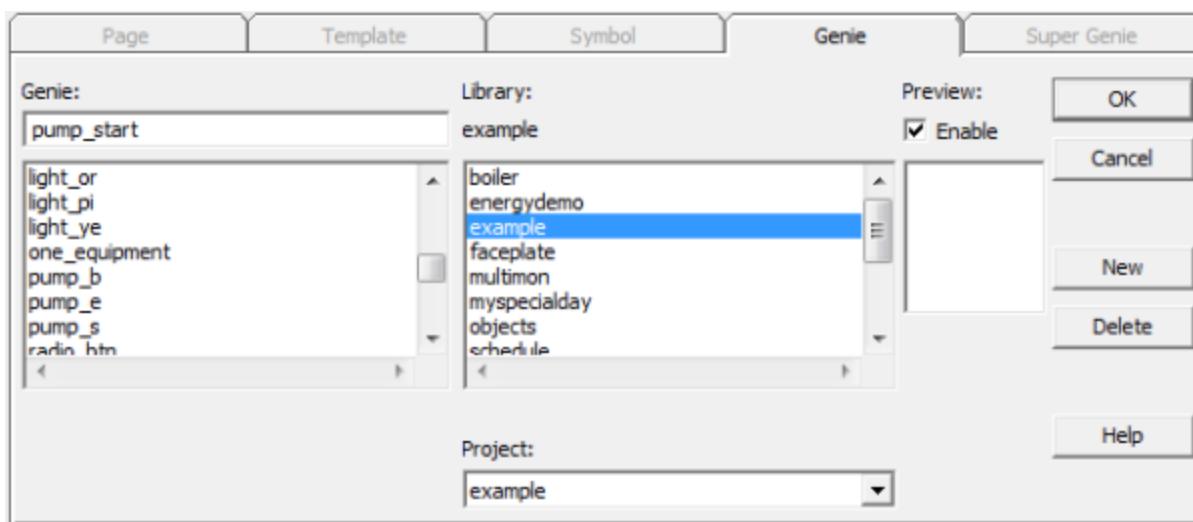
- [Create a New Genie](#)
- [Paste a Genie](#)
- [Configure a Genie](#)
- [Dynamically Instantiate a Genie at Runtime](#)

## Create a New Genie

Configuring a new Genie is similar to creating a page with graphical objects, but with genies there is no background. Typically you create a new Genie using the Graphics Builder, add the objects, define the Genie substitutions, and save the Genie in a Genie library.

### To create a new Genie:

1. Open Graphics Builder.
2. From the **File** menu select **New**.
3. Click **Genie**.
4. Create your Genie object, and define the required substitutions (see [Define Genie Substitutions](#)).
5. Click the **Save** button on the Graphics Builder toolbar, or choose **Save** from **File** menu.



6. Select the **Project** and **Library** in which you would like to store the Genie.
7. Enter a name for the Genie in **Genie** field.
8. Click **OK**.

To create a new library for the Genie, click **New**. The New Library dialog will display. In the field provided, name the new library and click **OK**. The new library will be added to the library list. You can now [Paste a Genie](#) on a graphics page.

## See Also

[Define Genie Substitutions](#)

[Use Genie Substitutions in Templates](#)

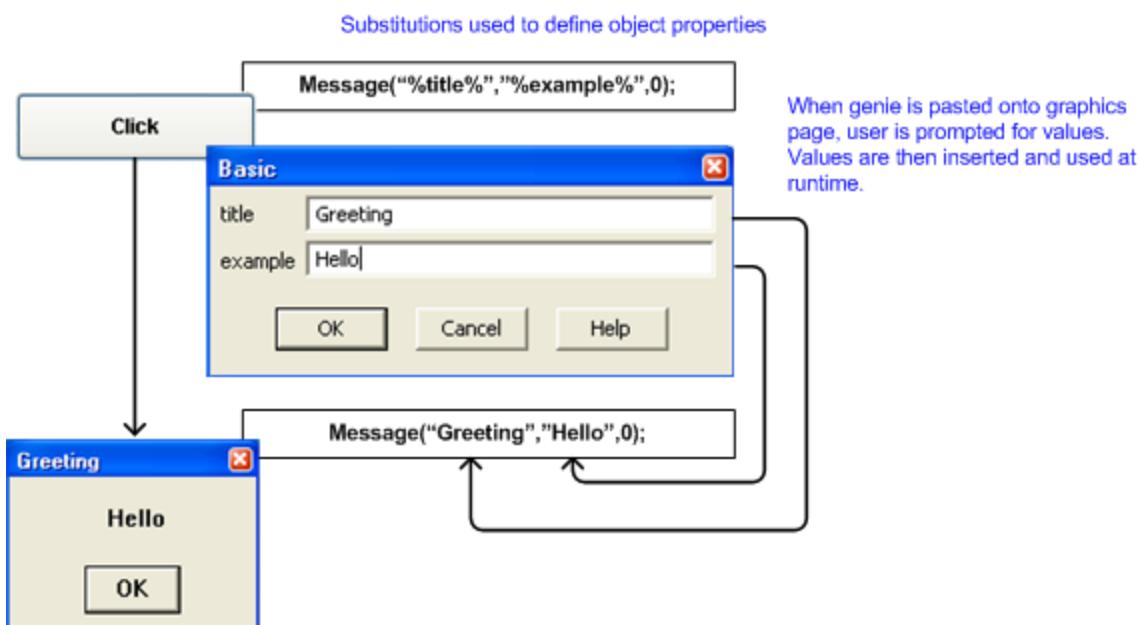
[Use Equipment.Item References with Genies](#)

[Use Structured Tags with Genies](#)

[Use the IFDEF Macro](#)

## Define Genie Substitutions

A Genie substitution can be a number or name you use to define an object's expression. Used as a placeholder the expression is replaced with a real value when you compile the project.



To define a Genie, you use substitution strings in the properties of the objects that will be specific to each instance. You can use substitutions for any expression in an object (or in a group of objects). To specify an expression as a substitution, enclose the placeholder text between percentage (%) characters. For example:

%title% %example%

---

**Note:** Area is a reserved name in Plant SCADA, thus the use of %AREA% within a Genie substitution is not supported. See Reserved Words for more information.

For static STRING substitutions, enclose the %% characters in quotation marks, for example:

"%title%" "%example%"

This will stop Plant SCADA from reading the placeholder text as a tag.

Only fields that accept expressions can have Genie substitutions; however any expression can be substituted, including constants or labels, or 'equipment.item' tag references.

You can also define substitutions for variables that do not exist in the current project. By using the IFDEF function, you can hide the variables at runtime. Refer to [Use the IFDEF Macro](#) for more information.

## See Also

[Use Genie Substitutions in Templates](#)

[Use Equipment.Item References with Genies](#)

[Use Structured Tags with Genies](#)

[Use the IFDEF Macro](#)

## Use Genie Substitutions in Templates

When you create a new custom template and add objects to the template, you can use Genie substitutions directly in the expressions of the object.

Configure the template using substitution strings (enclosed between percentage characters '% %') for the relevant properties of each object.

When finished save the new template.

When you subsequently create a new page based on the template, double-click the template (typically a graphics object on the template), and a dialog will appear prompting you for the values for the substitutions used in the template.

## See Also

[Create a New Genie](#)

[Define Genie Substitutions](#)

[Use Equipment.Item References with Genies](#)

[Use Structured Tags with Genies](#)

[Use the IFDEF Macro](#)

## Use Equipment.Item References with Genies

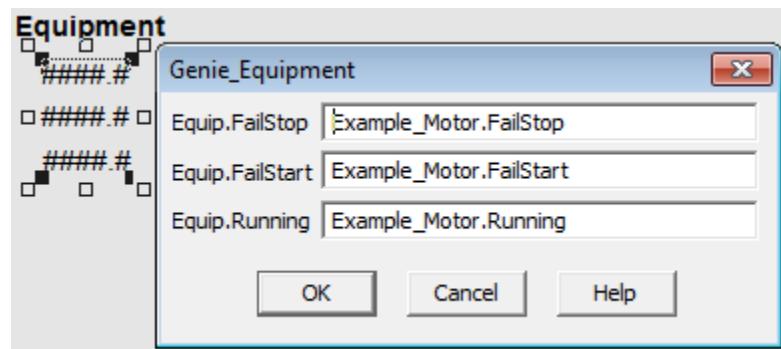
When you define a Genie, you can use full or partial Genie substitutions to generate the complete 'equipment.item' reference when the Genie is used.

If you needed to define a Genie to display the values of the equipment "Example\_Motor.FailStop", "Example\_Motor.FailStart" and " Example\_Motor.Running", you could configure a Genie as follows:

Numeric expression	%Equip.FailStop%
Numeric expression	%Equip.FailStart%
Numeric expression	%Equip.Running%

By using full substitutions, each numeric expression is a separate Genie substitution.

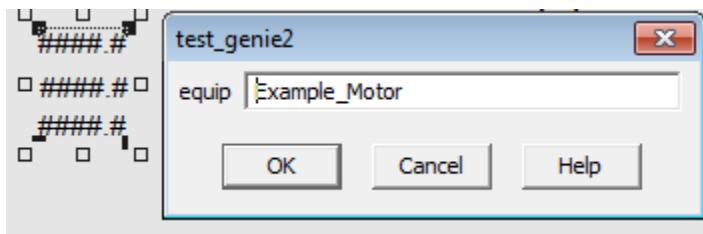
When the Genie is pasted onto a graphics page you would be prompted to enter the 'equipment.item' reference for each expression.



Using a partial substitution where only part of the 'equipment.item' reference is substituted, you could configure the numeric expressions as follows:

Numeric expression	%Equip%.FailStop
Numeric expression	%Equip%.FailStart
Numeric expression	%Equip%.Running

When the Genie is pasted onto a graphics page you would be prompted to enter just one 'equipment.item' reference (for example, "Example\_motor").



The partial substitution is then used in the Genie to create the complete 'equipment.item' reference for all three objects.

## See Also

[Create a New Genie](#)

[Define Genie Substitutions](#)

[Use Genie Substitutions in Templates](#)

[Use Structured Tags with Genies](#)

[Use the IFDEF Macro](#)

## Use Structured Tags with Genies

When you define a Genie, you can add a prefix or suffix to a Genie property to generate the complete tag when the Genie is used. For example, you could define a Genie property as follows:

`%tag%_PV`

If you then use "DEV1" for the tag, the Genie will generate the complete tag:

`DEV1_PV`

This is also the case if you reference the variable tag using the associated equipment and item name:

`%equipment.item%_PV`

If you use "PUMP.SPEED" for equipment.item, the Genie will generate the complete tag:

`PUMP.SPEED_PV`

You can add extra information at the beginning (prefix), or on the end (suffix) of the Genie property, or use both a prefix and suffix in the same Genie property. For example, if you have defined a loop controller with three bar graphs (created using the fill property in a rectangle) to display the tags DEV1\_PV, DEV1\_SP and DEV1\_OP, you can configure a Genie as follows:

Level expression	<code>%PV_Tag%</code>
Level expression	<code>%SP_Tag%</code>
Level expression	<code>%OP_Tag%</code>

This configuration means each rectangle has a separate Genie tag.

When you configure the Genie (with the Genie dialog), you have to enter three separate tags: DEV1\_PV, DEV1\_SP and DEV1\_OP. However, if you use structured tags, you can configure the rectangles as follows:

Level expression	%Tag%_PV
Level expression	%Tag%_SP
Level expression	%Tag%_OP

In this case, you only have to enter one tag (DEV1) to generate six objects. The Genie automatically concatenates DEV1 with either \_PV, \_SP, or \_OP, depending on where the tag is substituted. As well as a reduction in configuration time, this Genie is easier to maintain.

**Note:** The above example illustrates the power of Genies. The more complex and the greater number of objects in a Genie, the greater the advantage of using structured tags. You can also make complex Genies by using multiple variables for a Genie property. For example, "%Level%\_TIC\_%Occ%\_PV" or any combination of prefix, suffix and number of Genie variables.

## See Also

[Create a New Genie](#)

[Define Genie Substitutions](#)

[Use Genie Substitutions in Templates](#)

[Use Equipment.Item References with Genies](#)

[Use the IFDEF Macro](#)

## Use the IFDEF Macro

Genies are customizable, reusable objects that are even more flexible when used with the IFDEF macro. With the IFDEF macro, during compilation it is able to detect if a Genie substitution is valid or not. If the substitution is not valid you are able to hide it from the page, thus making your Genies more generic and reusable in more contexts.

**Note:** In Plant SCADA, using the IFDEF macro is considered an advanced configuration scenario.

The expression entered in the **Hidden When** field of an object's property is used to determine if the object will be visible on the page at runtime. The expression evaluates to either TRUE or FALSE and the object is hidden when the expression is TRUE.

You define the variable tag and conditions under which the object is hidden by entering an IFDEF statement into the **Hidden When** field when you configure the object. The IFDEF statement is evaluated by the compiler and the value of the resulting expression or variable tag will determine whether or not the object is hidden.

This can significantly reduce the number of necessary Genies, as the configuration engineer does not need to generate several smaller Genies to cater to operations driven by a slightly different range of tags.

The IFDEF statement consists of three arguments:

```
IFDEF (<"Tag Reference">, <Result value if tag defined>, <Result value if tag undefined>)
```

The first includes a tag reference (variable tag name or 'Equipment.Item'). If the variable tag is defined in the tag database at project compilation, the IFDEF statement is replaced in the Expression field by the second argument. If the variable tag is undefined, the Expression field will contain the third argument, displaying a value of a tag that may not be defined in your project.

## Example - Hidden When

```
IFDEF("Bit_1", 0, 1)
```

In the above example, if Bit\_1 is defined in the tag database, the value in the **Hidden When** field will be 0. If Bit\_1 is undefined, the value will be 1. Since the object is hidden when the value is TRUE, the object will be hidden when BIT\_1 is undefined (i.e. when the **Hidden When** field contains 1).

## Example - Display Value

```
IFDEF("Bit_2", "NA")
```

If the second argument is omitted, as in Example 2, the variable tag specified in the first argument is used. If Bit\_2 is defined, therefore, the display value will contain Bit\_2. If the tag is not defined, then NA would be displayed.

If Bit\_2 is undefined, the Hidden When expression evaluates to 1 (TRUE) and the object is hidden.

### To enter an IFDEF statement in the Hidden When Field:

1. Double-click the graphics object for which you want to edit the field.
2. Select the **Appearance** tab.
3. Click the **Hidden When** field and enter the IFDEF statement.
4. Click **OK**.

## See Also

[Create a New Genie](#)

[Define Genie Substitutions](#)

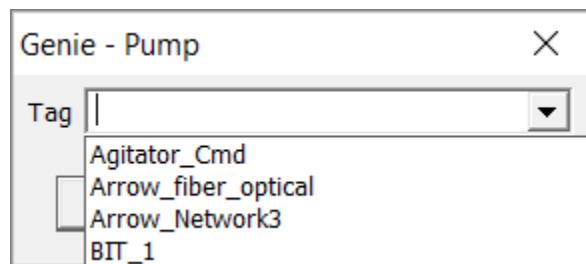
[Use Genie Substitutions in Templates](#)

[Use Equipment.Item References with Genies](#)

[Use Structured Tags with Genies](#)

## Genie Forms

The Genies that are included with Plant SCADA often have drop-down lists that list the available variable tags. However, the forms on the custom Genies only have blank fields to enter the required information.



Customized drop-down lists are created by using Genie Forms. A Genie Library is a file with a .ctm extension and each Genie Library can have an associated form file (for example, Drives.ctm would have Drives.frm). The Genie form (.frm) file should reside in the same directory as the Genie Library (.ctm) file. A form definition for each Genie in the associated library can be provided. If one is not provided, a default form is created.

A Genie form follows the following format:

```
FORM "<genie>" [, "<form title>"];
<field number>: "<field name>", <field size> [, readonly/readwrite [, "<list form>", <"list
field>"[, "<list filter>"]]];
NORMAL
"<prompt> {<field number> }"
```

## Example

This is the form file for the standard Genie - Pump

```
FORM "Pump_East", "Genie - Pump";
1: "Tag", 32, readwrite, "Variable Tags", "NAME", "TYPE=DIGITAL";
NORMAL
"Tag{ 1 }"
```

## Genie Form Format

The Genie Form format is divided into three distinct sections:

- [Genie Definition and Title](#)
- [Record Definition](#)
- [Form Definition](#)

### Genie Definition and Title

This section defines the look of the Genie form.

Field	Definition
Form Keyword	Start the Genie form definition with the keyword FORM. FORM
Name	This is the Name of the Genie as it is specified in the Genie library. FORM "Pump_East"
Genie Title Bar	Text that will appear in the Genie Form title bar. FORM "Pump_East", "Genie - Pump"; <b>Note:</b> To enhance debugging of the project, it may be useful to set the Genie Title Bar to <libraryname>.<geniename>.

## See Also

[Genie Forms](#)

## Record Definition

The Record Definition section defines each of the fields that will appear on the Genie Forms.

Field	Definition
Field Number	Number for each field: 1:
Field Name	Field Name inserted between quotation marks " ". This is the word that appears between the percentage marks in the Genie. For example, %Tag%. 1: "Tag"
Field Size	Variable tags may be up to 79 characters in length. However, for display purposes it may be desirable to display fewer characters on the form field. 1: "Tag", 32
Write Access	The two options for this are <b>readwrite</b> and <b>readonly</b> . 1: "Tag", 32, readwrite

If a dropdown list is not going to be used (for example, formatting the field for privileges), this is enough to define the field. Finish the line with a semicolon (;).

However, if the field is being defined for something that will extract values from a .dbf file, additional definition is needed.

Field	Definition
List Form	The next field identifies the project database name, as defined in BIN\citect.frm, which contains the options to go into the list. The name of the list form needs to have quotation marks. 1: "Tag", 32, readwrite, "Variable Tags"
List Field	This field contains the values that will appear in the dropdown list. It also needs to be specified within quotation marks. 1: "Tag", 32, readwrite, "Variable Tags", "NAME"
List Filter	This is the format of the List Filter <field> = <filtertext>, where: <field> is the field name to filter by. This defaults to the field name specified as the List Field, so <field> = may be left out if filtering by the List Field. <filtertext> is the text to search for in the field. Only records that have the specified filter text in the specified field will be added to the dropdown list.

Field	Definition
	<p>A <b>Type</b> filter could be used for a Genie that is an on/off switch and will only apply to digital tags. If this is the case, the filter would be <b>TYPE=DIGITAL</b>.</p> <p><i>1: "Tag", 32, readwrite, "Variable Tags", "NAME", "TYPE=DIGITAL";</i></p> <p>If the list is going to be filtered on an extension, a wildcard (*) is used. For example, to list only the _CMD tags:</p> <p><i>1: "Tag", 32, readwrite, "Variable Tags", "NAME", "*_CMD";</i></p> <p>When a list is filtered this way, the dropdown list will remove the filter characters, that is, _CMD, from the end of the tag name. This is desirable when using structured tag names as the Genie will probably add the extension.</p> <p>It is also possible to filter for both type and extension. The following example shows a list that filters for an INTEGER type and a _CMD extension :</p> <p><i>1: "Tag", 32, readwrite, "Variable Tags", "NAME", "TYPE=INT, NAME=*_CMD";</i></p>

## Form Definition

This section defines the look of the Genie form.

Field	Definition
Normal Keyword	<p>Starts with the Keyword NORMAL.</p> <p>NORMAL</p>
Field Prompt	<p>The string that describes the form. The string consists of a prompt for the field.</p> <p>NORMAL</p> <p>"TAG</p>
Field Number	<p>The field itself is indicated by the Field Number enclosed in braces. Spaces need to be entered between the braces to display the size of the field. This string is copied across to the form and since it is a string it needs to be entirely enclosed in quotes.</p> <p>NORMAL</p> <p>"TAG {1 }"</p>

**Note:** The Form Definition may span multiple lines. Check that there is only one pair of quotation marks.

## Paste a Genie

A Genie can be pasted on any graphics page.

1. Open Graphics Builder.
2. Click the **Paste Genie** button in the objects toolbox, or select **Paste Genie** from the **Edit** menu.
3. In the Paste Genie dialog, select the library to which the required Genie belongs.
4. Select the Genie from the **Genie** list and click **OK**. You can also double-click the thumbnail of the Genie.  
The Genie is pasted onto the graphics page. A dialog will open, prompting you to configure the properties of the Genie (see [Genie Parameters Dialog Box](#)).
5. Enter the required values, and click **OK** to close the dialog.

To reuse this Genie in the project, you would enter a different title and text when prompted.

**Note:** To display the properties of the individual objects in a Genie (instead of the Genie Properties), hold the Control (CTRL) key down and double-click the object. The properties will be read-only.

## See Also

[Configure a Genie](#)

## Genie Parameters Dialog Box

The Genie Parameters Dialog Box displays the configuration fields associated with a Genie.

If the Genie was user created, the dialog will display the fields defined to support the required Genie substitutions (see [Define Genie Substitutions](#)).

If the Genie was provided in a Plant SCADA system project, select the Genie from one of the lists below for a description of the fields.

## Situational Awareness System Projects

Library	Genies
sa_controls	Alarm List Genie Alarm List Vertical Genie Generic List Genie Generic List Vertical Genie List Row Genie Navigation Zone Button Genie (HD1080) Navigation Zone Button Genie (UHD4K) Navigation Zone Tab Genie (HD1080) Navigation Zone Tab Genie (UHD4K) Navigation Zone Tab Bar Genie

Library	Genies
	Tree View Genie Tree View Item Genie
sa_controls_common	Button Base Genie Button Label Genie Check Box Genie Scroll Bar Horizontal Genie Scroll Bar Vertical Genie Toolbar Button Base Genie Toggle Button Genie Tab Base Genie Tab Genie
sa_common	Clock Timer Genie Control Mode Genie Label Genie MEO Genie Meter Value Genie Out Of Service (OOS) Genie Output Bar Genie Output Value Genie Process Variable (PV) Genie Selection Adorner Genie Selection Adorner Auto Genie Status Indicator Genie.
sa_faceplate	Faceplate Button Genie Faceplate Command Button Genie Limit Label Genie Local Remote Indicator Genie MEO Selector Genie Output Track Genie Output Valve Genie Selection Adorner Genie
sa_filter	Item Acknowledged Genie (HD1080) Item Acknowledged Genie (UHD4K) Item Base Genie Item Priority Genie (HD1080)

Library	Genies
	Item Priority Genie (UHD4K) Item Shelved Genie (HD1080) Item Shelved Genie (UHD4K) Item State Genie (HD1080) Item State Genie (UHD4K) Item Unacknowledged Genie (HD1080) Item Unacknowledged Genie (UHD4K).
sa_misc	Direction Arrow Genie
sa_navigation	Link Down Genie Link Left Genie Link Right Genie Link Up Genie Static Link Left Genie Static Link Right Genie
sa_priorities	Disabled Normal Genie Disabled Small Genie Priority 1 Normal Genie Priority 1 Small Genie Priority 2 Normal Genie Priority 2 Small Genie Priority 3 Normal Genie Priority 3 Small Genie
sa_workspace	Pane Genie

See [Situational Awareness System Projects](#).

## Library\_Controls Project

Library	Genies
lib_controls	AlarmTable Calendar Data Browse Table EquipTree Scrollbar_Horz Scrollbar_Vert

Library	Genies
	Slider SQL Table Tab Table Row Table TagTable Tree

See [Library Controls Include Project](#).

For more information, see [Configure a Genie](#).

## See Also

[Genies](#)

## Configure a Genie

All Genies need to be configured for them to be functional at runtime. Depending on the type of Genie, you may need to perform one or more of the following steps.

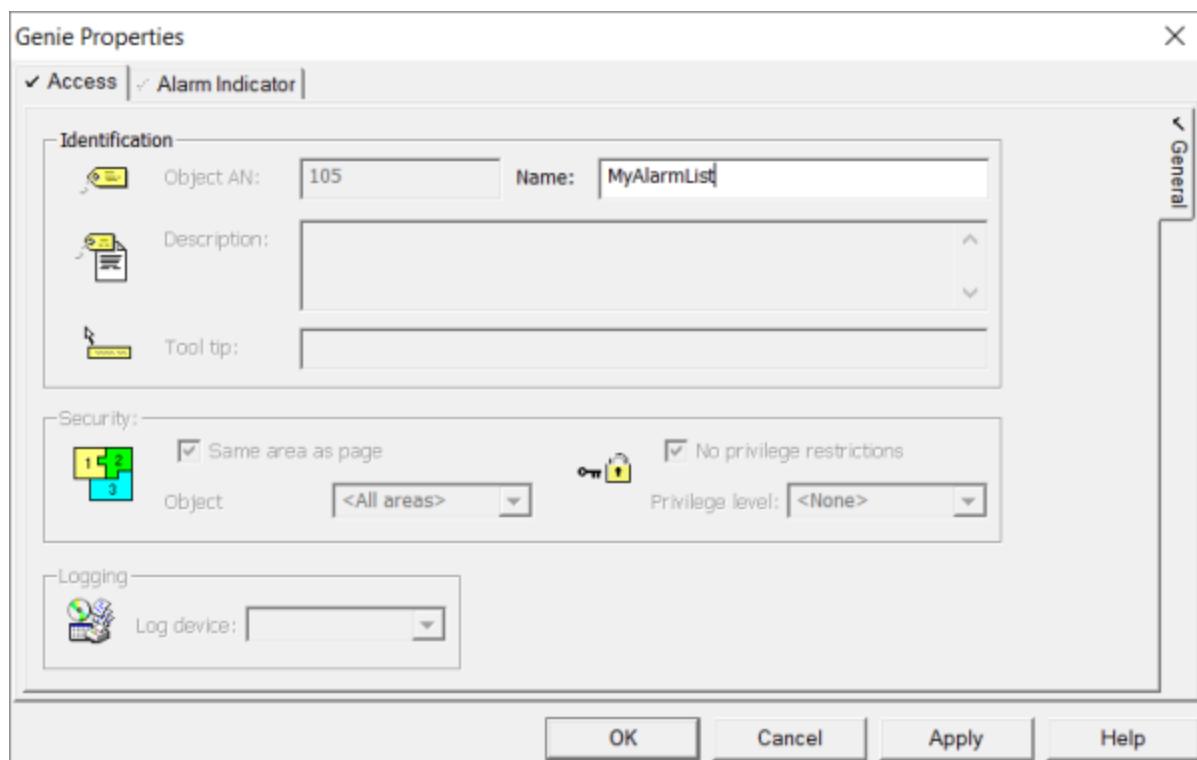
---

**Note:** To display the properties of the individual objects in a Genie (instead of the Genie Properties), hold the Control (CTRL) key down and double-click the object. The properties will be read-only.

---

- **Set Genie Properties**

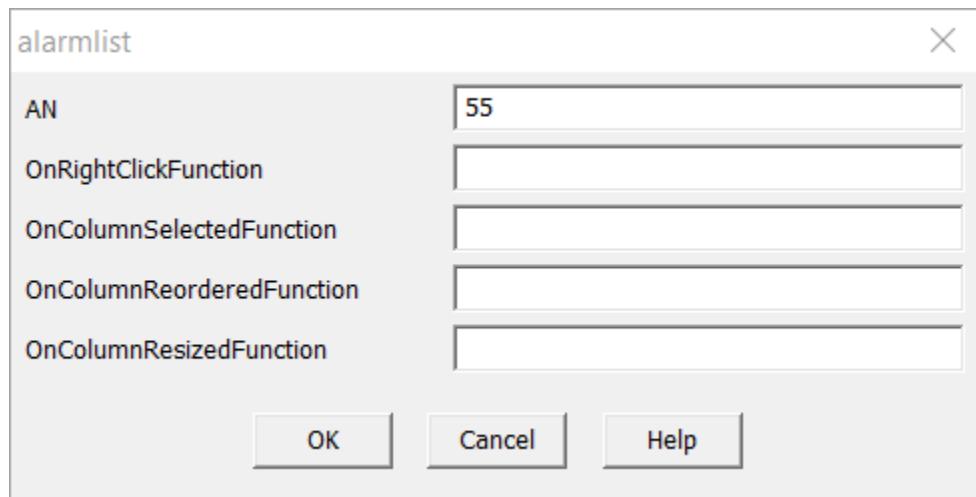
Select the Genie and from the right-click menu, select **Properties**. Set the required properties for the Genie. The image below shows the properties for the Alarm List Genie.



- **Set Genie Parameters**

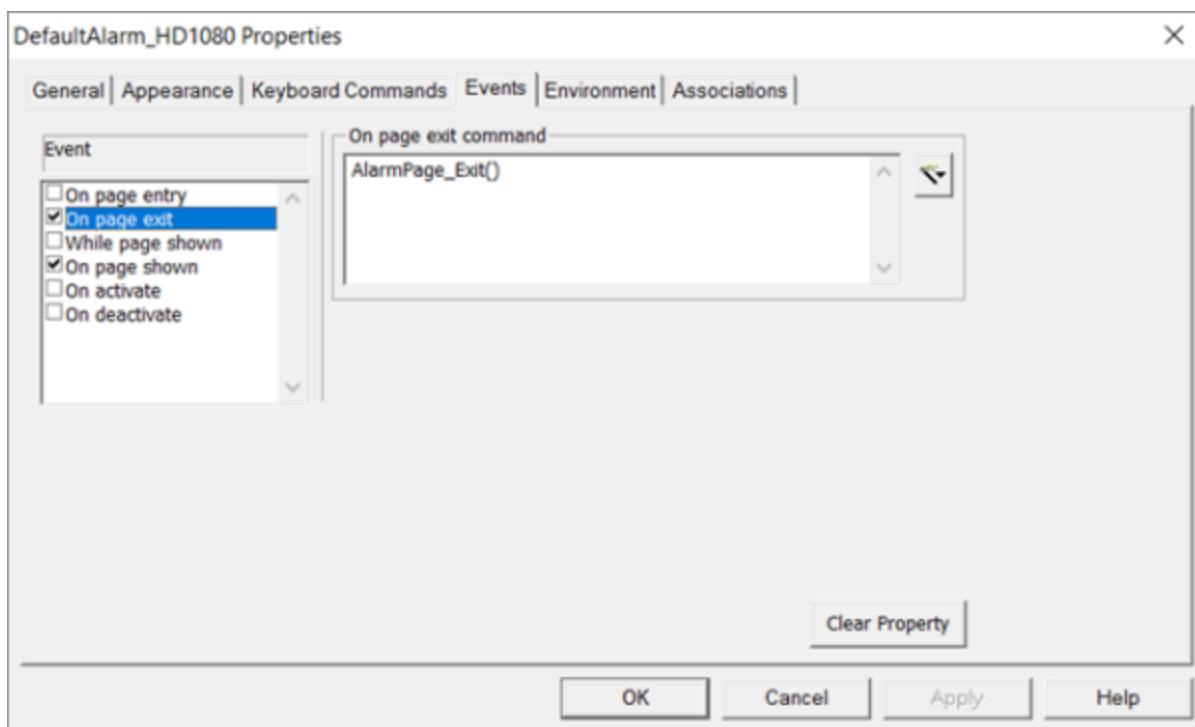
As soon as you insert a Genie, the [Genie Parameters Dialog Box](#) is displayed. You can also open this dialog box by double-clicking on the Genie, or from the right-click menu by selecting the **Genie Parameters** option.

As a default, the parameters are configured to perform certain operations. For example, the *OnColumnResizedFunction* parameter for the AlarmList Genie allows the user to re-size columns at runtime. The functionality provided by the parameters can be extended through user Cicode functions. The image below shows the parameters for the AlarmList Genie.



- **Set Page Properties**

Right-click on the page, and select **Page Properties**. On the **Events** tab, select the required events and specify the Cicode command for the event. There may be some events that need to be configured for the Genie to function correctly at runtime. The set of events to be configured differs for each Genie.



If you modify a Genie after you have used it in your project, occurrences of the Genie may be automatically updated throughout the project.

If you modify a Genie when the project is running in the background, you need to select one of the [Update Pages](#) commands to see the changes in the runtime project. If a runtime page containing the Genie is displayed when the change is made, it will not be updated until after you exit then re-display it.

You can access a list of Genies provided in Plant SCADA's system projects in the topic [Genie Parameters Dialog Box](#).

## See Also

[Paste a Genie](#)

## Dynamically Instantiate a Genie at Runtime

Use the [DspSym](#) Cicode function with a Genie to dynamically display content that is only generated at runtime. DspSym can be used with or without an expression.

At runtime when the page is refreshed or the operator clicks away from the page the instantiated Genie is deleted.

---

**Note:** ActiveX objects and controls (such as Process Analyst, Database Exchange Control and Scheduler) are not supported. If used and the "I want to use this Genie with Cicode Function DspSym" is selected, then on saving the page the following error message will be displayed:

*Genie cannot be saved.*

*Genies cannot contain ActiveX controls when configured to be used with the cicode function DspSym.*

*You must remove all ActiveX controls to continue.*

---

To save the page, deselect the checkbox or remove or delete the ActiveX controls used in the Genie.

**To edit the properties of the Genie:**

1. Open Graphics Builder.
2. Click the **Open** button on the toolbar, or select **Open** from **File** menu.
3. Select the **Genie** tab.
4. Select the **Project** and **Library** in which the Genie is stored.
5. Select the **Genie**.
6. Click **OK**.
7. Right click on the page and select **Page Properties**
8. Select the option "I want to use this Genie with Cicode Function DspSym"
9. Click **OK**
10. If the Genie or an object within the genie uses substitution '%%' syntax in the expression, replace the substitution with the full expression so it can be used at runtime. If you do not edit the substitution a compile error message may be displayed.

**Note:** You can define [Metadata](#) instead of using the full expression.

---

**See Also**

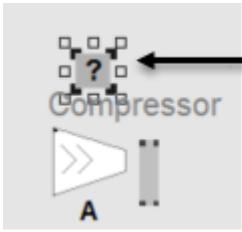
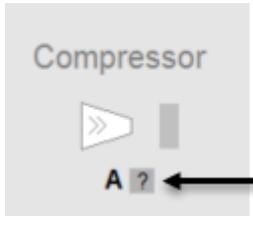
[Configure a Genie](#)

## Composite Genies

A Composite Genie is a collection of individual Genies assembled to form a single object. The individual objects and their properties are defined in an XML template file along with layouts for the collection, that is, the Composite Genie. You can insert multiple instances of the composite object on to a graphics page, and specify different parameter settings including values, alignment and display options for each instance to tailor the Composite Genie to suit your requirements.

For example, you want to build a mirrorable compressor object that is capable of optionally displaying a label, a mode indicator and a meter alongside the symbol. Previously, you could either create separate genies for each of the permutations, or you could build a single genie that used visibility to show/hide elements at runtime.

With Composite Genies, you can simply select the [XML template](#) for a compressor, which provides a set of presentation options for you to configure (label, orientation, etc). All you need to do is set values for these options and insert the object on to a page. See [Insert a Composite Genie](#) for more information.

Without Composite Genies	Using Composite Genies
<p>Instead of inserting and configuring individual Genies:</p>  <p>You need to change the layout and alignment of these Genies manually.</p>	<p><a href="#">Insert a Composite Genie:</a></p>  <p>All components laid out neatly with alignment and layout options that engineers can customize.</p>
<p>If you add related equipment indication to your compressor Genie above, the alignment of the objects needs to be adjusted to accommodate the new Genie:</p>  <p><b>Alignment of newly added sub-equipment</b></p>	<p>With a Composite Genie, simply bring up the <a href="#">presentation options</a> and select to add an output bar resulting in:</p>  <p><b>Newly added sub-equipment aligned automatically</b></p>

With the use of Composite Genies:

- The size of the object libraries can be reduced significantly as only a single Composite Genie needs to be maintained as opposed to maintaining multiple permutations of each object used in graphics pages.
- Engineers can get a clearer picture of their page design because the Composite Genie displays only those parameters that have been selected.
- A richer options interface for parameters is available. Using the XML template, library designers can enrich the options interface that allows operators to select from different layouts, modes, presents parameters taking into account dependencies, naming rules and so on.

As with Genies and Symbols, Composite Genies can be inserted for use in a graphics page, template or super genie. You can perform the following operations:

- [Insert a Composite Genie](#)
- [Edit/Update a Composite Genie](#)
- [Delete Unused Composite Genie Instances](#)
- [Delete a Composite Genie](#)

## Insert a Composite Genie

A Composite Genie can be inserted into any graphics page, template or Super Genie. It cannot be inserted into a Genie or symbol.

**Note:** The appearance of Composite Genies can be affected by different DPI settings. It is recommended that when running Graphics Builder, the **Scale and Layout** setting should be set to "100%" in the Windows™ **Display** settings. After changing the Scale and Layout setting for a computer, you should restart it.

**To insert a Composite Genie:**

1. Open Graphics Builder.
2. Click the **Paste Composite Genie** button in the Objects toolbox.



The Open dialog box is displayed.

3. Select the XML template for the Composite Genie from the list of templates.

**Note:** The template file that you select needs to be located in the Composite Genies folder in the current project folder (or one of its included projects).

4. Click **OK**. The Presentation Options dialog box is displayed.

**Presentation Options - Meter**

Template ID - {1e49c603-fcbc-4913-abd1-97920283c6bd}

The following options are available for this composite genie:

Search Options	
Meter Type	Level
Equipment Name	LevelMeter
Equipment Item Prefix	
Size	Small
Orientation	Vertical
Display Label	Above
Label	Meter
Display Alarm Indicator	<input type="checkbox"/>
Display Status Indicator	None
Display Process Variable	<input checked="" type="checkbox"/>
Display Control Mode	<input checked="" type="checkbox"/>
Display Controller Output	<input checked="" type="checkbox"/>
Display Setpoint	<input checked="" type="checkbox"/>

**Meter Type**  
Controls the visual presentation for this meter on the graphics page.

Show only parameters with invalid input

OK CANCEL

5. Type or select the values for the parameters displayed.

**Note:** Depending upon the options you choose, some items may be hidden or displayed. For example, on the Meter Composite Genies, when you select the **Display Control Output** check box, the **Display Trend** and **Trend Type** options are displayed.

6. Use the **Search** box to search for parameters in the dialog box. Type the parameter name or part of the parameter name to locate a parameter.

**Note:** Clear the Search box before clicking **OK** on the Presentation Options dialog box.

7. Use the **Show only parameters with invalid input** option to display options for which incorrect or no values have been selected or specified.

8. Click **OK** to insert the Composite Genie, or **Cancel** to close the Presentation Options dialog box without inserting the Composite Genie.

**Note:** You cannot rotate or mirror a Composite Genie that has been inserted on a page.

## See Also

[Composite Genies](#)

[Edit/Update a Composite Genie](#)

[Delete Unused Composite Genie Instances](#)

[Delete a Composite Genie](#)

## Edit/Update a Composite Genie

To edit the parameter options for an existing Composite Genie:

1. Open Graphics Builder.
2. Click to select the inserted Composite Genie.
3. Right-click the Composite Genie, and select **Edit Composite Genie** from the context menu. The Presentation Options dialog box is displayed with the Composite Genie instance name and template ID.
4. Edit the parameters as required.
5. Click **OK**.

Double-clicking on the template ID selects the ID. You can copy the ID and use it to locate duplicate instances of the template in Windows Explorer.

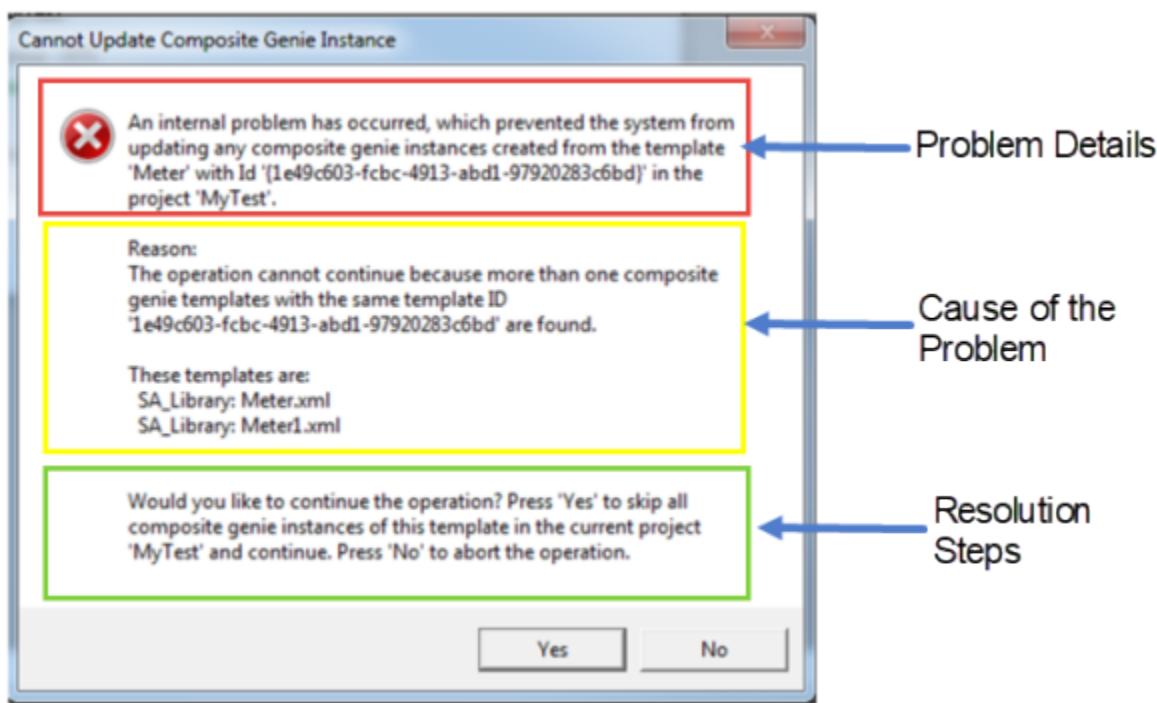
If you modify a Composite Genie after you have used it in your project, all instances of the Composite Genie may be automatically updated with the latest version of the template. Changes including updates to labels, genies and layout will be propagated to all instances. If a template is up to date, associated instances will not be updated.

**Note:** You should not modify the properties of the folder in which Composite Genie instances are stored (%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\User\<Project Name>\\_CompositionCaches). You should not manually delete or modify files in this folder. Doing so many result in errors when inserting Composite Genies on graphics pages.

An error message is displayed if a problem is encountered during the update process. The message details the cause and the options available for resolving the issue. An error may occur if a Composite Genie does not exist or

if the project is erroneous.

For example, if you create a copy of an existing template, but retain the same GUID in the XML file of the copied template, you will see the following message:



## Update a Particular Instance

Updating a particular instance of a Composite Genie allows you to verify the changes to a Composite Genie without affecting other instances of the object on other pages.

### To update a particular instance of a Composite Genie:

1. Open Graphics Builder.
2. Update the Composite Genie template as required.
3. Double-click the instance represented by the template. The Presentation Options dialog box is displayed with the modified parameters. For example if you have added a new parameter, the Presentation Options dialog should show the parameter.
4. Click **OK**. The changes including updates to labels, genies and layout will be propagated to this instance of the Composite Genie.
5. Save the page.

---

**Note:** Before you apply the changes to all instances of Composite Genies across all pages, verify the changes using the instructions above.

---

## Update All Instances

### To update instances of Composite Genies:

1. Open Graphics Builder.
2. On the **Tools** menu, select **Update Pages**. This command will replace an existing Composite Genie with its newest version available. This command will only affect the Active project and its includes. Compile the project for the changes to take effect for runtime.

---

**Note:** This change cannot be reversed.

---

**Note:** If a runtime page contains an updated Composite Genie instance, an online change will update runtime. Refer to [Online Changes](#) for more information.

---

## See Also

[Composite Genies](#)

[Insert a Composite Genie](#)

[Delete Unused Composite Genie Instances](#)

[Delete a Composite Genie](#)

## Delete Unused Composite Genie Instances

When you add a graphic object as a Composite Genie on a page, a Composite Genie instance that is uniquely identified by its presentation options, is automatically created in the project folder. This establishes a reference from the graphic object to this Composite Genie instance. There can be multiple graphic objects that refer to the same instance of a Composite Genie if their presentation options are the same. Therefore, deleting a graphic object does not delete the associated Composite Genie instance.

Over time there can be a number of Composite Genie instances that are not associated with any graphic object. These unused Composite Genie instances can slow down operations and occupy project disk space. You can use the **Pack Libraries** option to remove unused instances.

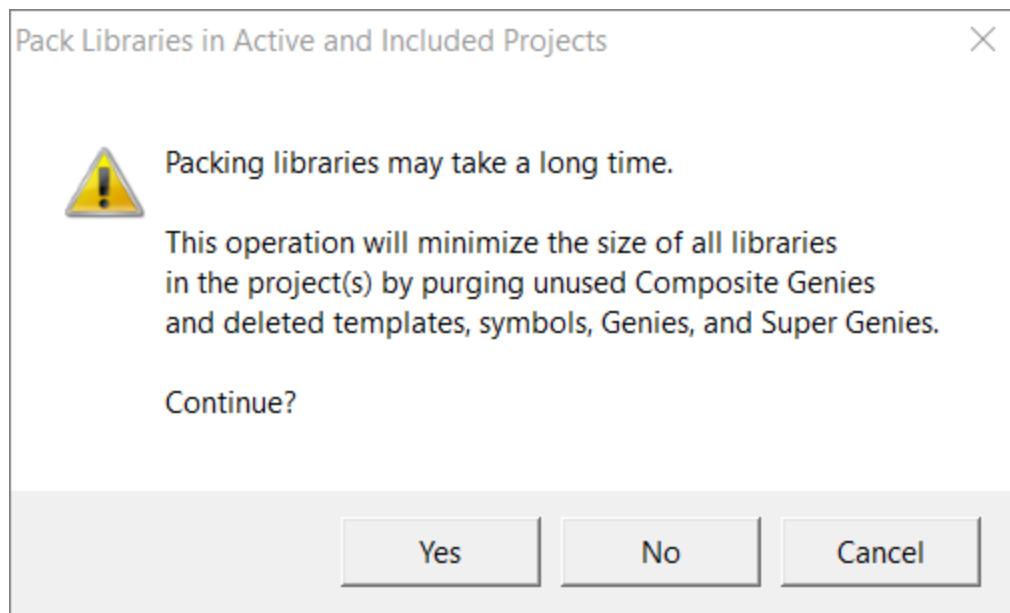
---

**Note:** You should not modify the properties of the folder in which Composite Genie instances are stored (%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\User\<Project Name>\\_CompositionCaches). You should not manually delete or modify files in this folder. Doing so may either prevent your pages from opening or cause your Composite Genies to be displayed incorrectly.

---

### To delete unused instances:

1. Open Graphics Builder.
2. On the **Tools** menu, select **Pack Libraries in Active Projects** or **Pack Libraries in Active and Included Projects**. The following message is displayed.



3. Click **Yes**. All unused instances of Composite Genies will be deleted.

## See Also

[Composite Genies](#)

[Insert a Composite Genie](#)

[Delete Unused Composite Genie Instances](#)

[Delete a Composite Genie](#)

## Delete a Composite Genie

### To delete a Composite Genie:

1. Open the page in which the Composite Genie has been inserted.
2. Click to select the Composite Genie.
3. Press the **Delete** key. This will remove the Composite Genie from the page.
4. Save the page.

## See Also

[Composite Genies](#)

[Insert a Composite Genie](#)

[Edit/Update a Composite Genie](#)

[Delete Unused Composite Genie Instances](#)

## Composite Genie Templates

Plant SCADA provides a wide range of out-of-the box Composite Genies in the form of XML templates. An XML template contains the definition of a Composite Genie. A library engineer may modify an existing XML template or create new templates defining their own Composite Genies. It is recommended that you make a copy of the template before you modify it so that templates are not overwritten if you re-install Plant SCADA.

---

**IMPORTANT:** If you make a copy of a template, you need to assign a new GUID to the template and all the compositions in the template. For more information, see [Visual Template](#) and [Compositions](#).

---

XML templates for Composite Genies are located in the %PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\User\<project name>\ Composite Genies folder.

The **CompositionTemplate.xsd** file, located in the %PROGRAMFILES(X86)%\AVEVA Plant SCADA\Bin, contains the schema for the Composite Genie XML templates. This file describes the elements and attributes that may be contained in an XML template for a Composite Genie, and the order in which the elements and attributes need to be added to the XML template. The XML template file is validated against the XSD file. When you modify an XML template or create a new one, it is recommended that you use a good XML editor that supports schema error detection so that errors are highlighted and can be addressed easily.

Errors may be encountered if the XML file for a Composite Genie does not conform to the definitions in the underlying XSD. Alternatively, errors may be encountered if there are duplicate IDs for elements, undefined conditions, and so on. In both cases, errors will be reported with details of the line number and position of the error when an operator inserts the Composite Genie on to a page. For more details, see [Edit/Update a Composite Genie](#).

An XML template for a Composite Genie contains the following elements:

- [Visual Template](#)
- [Parameters](#)
- [Content Items](#)
- [Compositions](#)
- [Conditions](#)
- [Alarm Indicators](#)

Each of these elements is explained in detail in the sections below. It is recommended that you review these sections before you modify existing XML templates or create new templates.

### Visual Template

This section comprises standard XML declarations. In addition, this section contains the GUID, which is a unique identifier for the XML template.

---

**Note:** If you modify an existing template or create a new Composite Genie template, you should validate the XML file by loading the schema definition file (CompositionTemplate.xsd which is in the <Plant SCADA installation folder>\bin folder) into your XML editor.

---

If you create your own Composite Genie, you will need to create a GUID for it. Several online tools are available for generating a GUID.

**To generate a GUID:**

1. Search for the term "generate GUID" in a browser using any search engine. A list of websites that can generate a GUID is displayed.
2. Access any one of the sites.
3. Select the format for the GUID. The GUID format should include braces and hyphens and should be in upper case. For example, {1E49C603-FCBC-4913-ABD1-97920283C6BD}.
4. Generate the GUID.

**Example**

```
<?xml version="1.0" encoding="utf-8"?>
<VisualTemplate
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="CompositionTemplate.xsd"
    Guid="{1E49C603-FCBC-4913-ABD1-97920283C6BD}"
    Name="Meter"
    Thumbnail="encodedstring">
```

**See Also**

[Parameters](#)  
[Content Items](#)  
[Compositions](#)  
[Prerequisites](#)  
[Container](#)  
[Conditions](#)  
[Alarm Indicators](#)

**Parameters**

The <Parameter> element defines the attributes that will be displayed as Presentation Options to the user when they insert a Composite Genie on to a graphics page. It is comprised of the following attributes for each parameter:

Attribute	Description
Name	Unique name for the parameter.
Label	Display name for the parameter.
Type	Data type of the parameter – Boolean, Integer or String.

Attribute	Description
MinLength	Minimum number of characters that the parameter value needs to contain.
MaxLength	Maximum allowable characters for the parameter value
MinValue	Minimum value that can be specified for the parameter
MaxValue	Maximum allowable value for the parameter
DefaultValue	Default value set for the parameter
NamingRules	Naming rules can be used to force the user to enter parameter values in a particular format. For example, to use equipment naming rules, set this attribute to "Equipment". Refer to the Equipment Naming Rules and Tag Naming Rules sections for more information.
Description	Description of the parameter. This appears at the bottom of the Parameter Options dialog box when the operator clicks to specify the value for a parameter. It is recommended that you enter a description for the parameter in order to help engineers to specify valid values for the parameters.
Values	<p>List of selectable values if an option is to be displayed as a dropdown list in the Parameter Options dialog box. To create a list of values, each value needs to be declared with a unique ID and Label. The label will appear as one of the values the user can select. For example, if you need an option titled "Orientation" with the values "Vertical" and "Horizontal" available for selection, the following code needs to be included in the Parameters element:</p> <pre data-bbox="871 1486 1527 1592">&lt;/Parameter&gt; &lt;Parameter Name="Orientation" Label="Orientation" Type="String" DefaultValue="Vertical"&gt;   &lt;Values&gt;     &lt;Value Id="1" Label="Vertical"/&gt;     &lt;Value Id="2" Label="Horizontal"/&gt;   &lt;/Values&gt; &lt;/Parameter&gt;</pre>

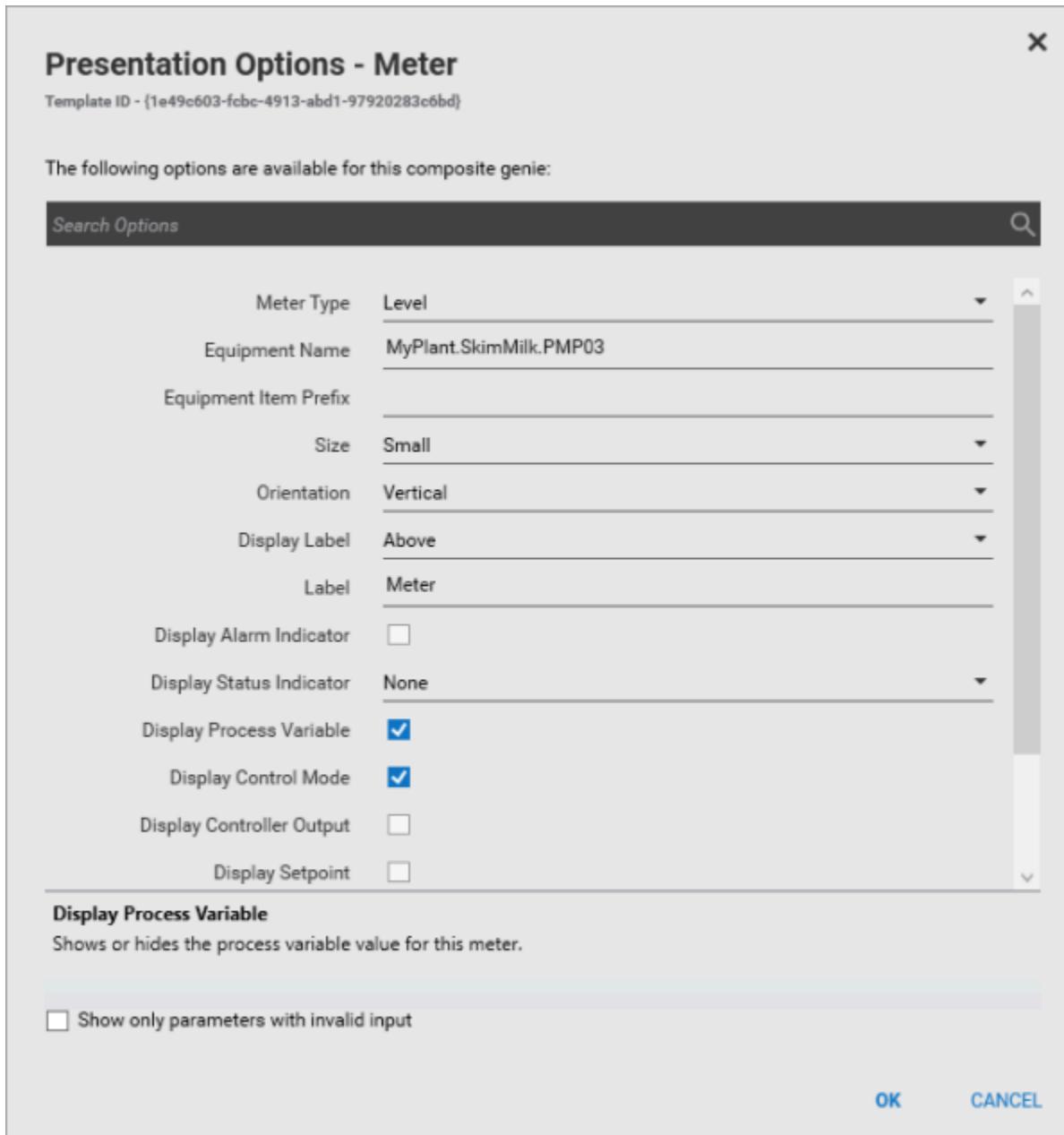
Parameter definitions can include the `<Dependency>` element to show or hide parameters. This element can use Global, Parameter or Composite conditions to show or hide a parameter based on the operator's selection of an option or value. For more information about conditions, refer to [Conditions](#).

## Example - Simple

This is a simple example where a dependency has been defined using the `<Dependency>` element to show the **Label** parameter only if the operator selects the **Display Label** checkbox. This is achieved through the use of a [Conditions](#). You can also use Parameter and/or Composite conditions within the `<Dependency>` element.

```
</Parameter>
<Parameter Name="DisplayLabel" Label="Display Label" Type="Bool" DefaultValue="true"/>
<Parameter Name="Label" Label="Label" Type="String" MinLength="1" MaxLength="79"
DefaultValue="Meter" Description="Enter the common name used to help identify the reading">
<Dependency>
  <ParameterCondition Name="DisplayLabel" Comparison="IsEqualTo" Value="true"/>
</Dependency>
</Parameter>
```

The above code will display the following presentation options.



## Example - Complex

This example provides a slightly more complex scenario with parameters to be displayed based on the operator's selection of value. Therefore, a list of values and a dependency need to be defined.

- Refer to the XML code for the "Number of sub equipment" parameter. This parameter can take the values 0, 1, 2, 3, 4 or 5. The <Values> attribute defines this list of selectable values (see the sample code below).

```
<Parameter Name="MEO" Label="Number of equipment" Type="Int" DefaultValue="1"
Description="If the object will represent multiple pieces of equipment, select
how many pieces of equipment the object group will include. When the object displays,
it will include a corresponding number of MEOs.">
  <Values>
    <Value Id="1" Label="0"/>
    <Value Id="2" Label="1"/>
    <Value Id="3" Label="2"/>
    <Value Id="4" Label="3"/>
    <Value Id="5" Label="4"/>
    <Value Id="5" Label="5"/>
  </Values>
</Parameter>
```

- Depending upon the value the operator selects for this parameter, the Sub Equipment parameter(s) will be displayed. For example, if the operator selects 0 for the "Number of sub equipment" parameter, the Sub Equipment parameter will not be displayed, if the user selects 1, a single Sub Equipment parameter with the label Sub Equipment #1 Name will be displayed, if the user selects 2, two Sub Equipment parameters with the labels Sub Equipment #1 Name and Sub Equipment #2 Name will be displayed and so on. The user can choose up to 5 pieces of sub equipment.

To be able to select 5 pieces of sub equipment, a parameter each needs to be defined for each piece of sub equipment. In addition, each of these parameters needs to have a dependency defined within it (see below for sample code that shows the definition for two sub equipment pieces).

- A dependency is defined using the <Dependency> attribute. For Sub Equipment 1, global condition "Show\_MEO1" needs to be satisfied. For detailed information about conditions, see the Conditions topic.

```
<Parameter Name="MEO_Equip1" Label="Sub Equipment #1 Name" Type="String" MinLength="1"
MaxLength="79" DefaultValue="" NamingRules="Equipment" Description="Enter a name for the
first piece of equipment in the group.">
  <Dependency>
    <GlobalConditionRef Id="Show_MEO2" />
  </Dependency>
</Parameter>
<Parameter Name="MEO_Equip2" Label="Sub Equipment #2 Name" Type="String" MinLength="1"
MaxLength="79" DefaultValue="" NamingRules="Equipment" Description="Enter a name for the
second piece of equipment in the group.">
  <Dependency>
    <GlobalConditionRef Id="Show_MEO2" />
  </Dependency>
</Parameter>
```

## Presentation Options - Drive

Template ID - {e91e1230-c421-4dec-b7cf-f6bc99b978df}

The following options are available for this composite genie:

Search Options

Equipment Name	Drive
Drive Type	Compressor
Size	Small
Orientation	Right
Display Label	None
Display Alarm Indicator	<input type="checkbox"/>
Display Status Indicator	None
Number of Equipment	1
Display Compact Running State	<input type="checkbox"/>
Display Control Mode	<input checked="" type="checkbox"/>
Display Meter	<input type="checkbox"/>
Display Output Bar	<input type="checkbox"/>

**Drive Type**  
Controls the visual presentation for this drive on the graphics page.

Show only parameters with invalid input

[OK](#)   [CANCEL](#)

## See Also

[Visual Template](#)

[Content Items](#)

[Compositions](#)

[Prerequisites](#)

[Container](#)

[Conditions](#)

[Alarm Indicators](#)

## Alarm Indicators

The <AlarmIndicators> element is an optional element for displaying alarm indicators for equipment. It comprises the following child elements:

Element	Description	Attribute	Description
EquipmentLink	Defines each piece of related equipment for which an alarm needs to be displayed.	Id	A unique string identifier with which <AlarmIndicator> elements in compositions can refer to the forward-declared equipment link.
		TemplateParameter	Name of the template parameter. The value of this is used as the equipment expression of the alarm indicator. The template parameter needs to be defined in the <Parameters> element in the same document.
		IncludeEquipmentReference	Set this to 'true' to include equipment references when determining the highest priority alarm for the alarm indicator. Set this to 'false' if you do not want to include equipment references.
		Hierarchy	Node in the equipment hierarchy that will be used to determine the highest priority alarm for the alarm indicator. You can set this to include equipment that occur at nodes down the equipment hierarchy (referred to as "children"). Set this to one of the following: EquipmentOnly EquipmentAndChildren ChildrenOnly

BorderStyle	Defines the alarm display settings.	Id  Width  Padding  IsInside	A unique string identifier with which <AlarmIndicator> elements in compositions can refer to the forward declared border style.
			Sets the width of the alarm border (in pixels).
			Sets the amount of space (in pixels) between the extent of the object group or Genie and the inside edge of the alarm border.
			Set this to 'true' to place the border within the extent of the object group or Composite Genie. Note that you cannot set a padding in this case.

**Note:** An alarm indicator is applied to all equipment items within the Composite Genie. The items are automatically grouped and the alarm border is applied to the composite if there are more than two visible items in the container.

## Example

```
<AlarmIndicators>
  <!-- Here optional attributes are set to explicit value deliberately so that the template can still stick with the requirement even when application default behavior changes in the future. -->
  <EquipmentLinks>
    <EquipmentLink Id="EM001" TemplateParameter="EM01_Equip1" IncludeEquipmentReference="false" Hierarchy="EquipmentOnly" />
    <EquipmentLink Id="EM002" TemplateParameter="EM02_Equip2" IncludeEquipmentReference="false" Hierarchy="EquipmentOnly" />
    <EquipmentLink Id="EM003" TemplateParameter="EM03_Equip3" IncludeEquipmentReference="false" Hierarchy="EquipmentOnly" />
    <EquipmentLink Id="EM004" TemplateParameter="EM04_Equip4" IncludeEquipmentReference="false" Hierarchy="EquipmentOnly" />
    <EquipmentLink Id="EM005" TemplateParameter="EM05_Equip5" IncludeEquipmentReference="false" Hierarchy="EquipmentOnly" />
  </EquipmentLinks>
  <!-- DefaultStyle will be implicitly applied. Use explicit 'BorderStyle' attribute in <AlarmIndicator> element if a different border style is desired. -->
  <BorderStyles>
    <BorderStyle Id="DefaultStyle" Width="2" Padding="0" IsInside="false" IsDefault="true" />
    <BorderStyle Id="OuterInsidePaddingStyle" Width="2" Padding="1" IsInside="false" />
    <BorderStyle Id="InsideStyle" Width="2" IsInside="true" />
  </BorderStyles>
</AlarmIndicators>
```

## See Also

- [Visual Template](#)
- [Parameters](#)
- [Content Items](#)
- [Compositions](#)
- [Prerequisites](#)
- [Container](#)

## Content Items

The <ContentItems> element defines the library objects (Genies) that are available to be included in compositions for the Composite Genie. It contains one or more <ContentItem> child elements, each of which comprises the following attributes:

Attribute	Description
Id	Unique numeric identifier within the <ContentItems> section.  <b>Note:</b> You should not use the same numeric Id for more than one <ContentItem> elements. Doing so would result in an error when the system reads this visual template document.
Type	This is set to the type of the object that makes up the Composite Genie and should not be modified.  <b>Note:</b> In this version Composite Genies constitute only Genies. So, Type is set to "Genie" in the XML templates.
Project	Name of the project which contains the referred library object. The project needs to be either the same as the project in which the visual template resides or one of its include projects.  <b>Note:</b> If the project is renamed, the Genie will not be available for use unless this attribute is modified in the XML template. Keep this in mind when renaming a project.
Library	Name of the object library to which this item belongs.
Item	Name of the item to be included in the Composite Genie.

**Note:** You need to explicitly define each and every item that you want to include in your compositions.

## Example

Each line in the code below defines an item, in this case a Genie, that will be available to be included in a composition.

```
<ContentItems>
<Content Id="01" Type="Genie" Project="SA_Library" Library="sa_common" Item="selection_adorner_auto" />
<Content Id="02" Type="Genie" Project="SA_Library" Library="sa_common" Item="label_12_c" />
<Content Id="03" Type="Genie" Project="SA_Library" Library="sa_common" Item="label_12_l" />
<Content Id="04" Type="Genie" Project="SA_Library" Library="sa_common" Item="label_15_c" />
<Content Id="05" Type="Genie" Project="SA_Library" Library="sa_common" Item="label_15_l" />
<Content Id="06" Type="Genie" Project="SA_Library" Library="sa_common" Item="clocktimer" />
<Content Id="07" Type="Genie" Project="SA_Library" Library="sa_common" Item="delta" />
<!-- Meter Types - Small -->
<Content Id="011" Type="Genie" Project="SA_Library" Library="sa_meter" Item="level_128_h" />
<Content Id="012" Type="Genie" Project="SA_Library" Library="sa_meter" Item="level_128_v" />
...

```

## See Also

[Visual Template](#)  
[Parameters](#)  
[Compositions](#)  
[Prerequisites](#)  
[Container](#)  
[Conditions](#)  
[Alarm Indicators](#)

## Compositions

The <Composition> element details the “composition” of the Composite Genie, that is, the possible layouts, parameter conditions and parameters that constitute the Composite Genie. It contains multiple elements each with a set of attributes. Typically, the XML template for a Composite Genie is made up of several compositions to represent the different layouts available. Each composition can define only one layout. The template needs to have at least one composition.

Attribute	Description
Composition Id	Unique identifier for the composition. This is an ID that is generated using the GUID Generator tool.

## Example

```
<Compositions>
  <Composition Id="{0C2B5CD5-F0FD-4021-8461-2864AFE24226}" >
```

## See Also

[Visual Template](#)  
[Parameters](#)  
[Content Items](#)  
[Prerequisites](#)  
[Container](#)  
[Conditions](#)  
[Alarm Indicator](#)

## Prerequisites

Within a composition, one or more prerequisites may be defined within the <Prerequisite> element. Prerequisites are made up of conditions, which are evaluated at runtime. If a prerequisite is true, the corresponding composition will be applied. Otherwise, the next composition will be evaluated, and a

composition will be applied only when the prerequisite is satisfied. Prerequisites are applied sequentially. For more information, refer to the Conditions topic.

**Note:** When defining conditions, it is recommended that you start with the most complex condition first and then add conditions in decreasing order of complexity. This is so that the most specific composition is selected for the Composite Genie.

Attribute	Description
GlobalConditionRef	Used to refer to a predefined Global condition
ParameterCondition	Used to specify conditions by comparing a template parameter value to a predefined value.
CompositeCondition	Used to combine more than one condition with either AND or OR relation specified in its attribute. Its child conditions can be made up of predefined Global conditions <GlobalConditionRef Id="..." />, a Parameter condition and nested Composite conditions.

## Example

```
<GlobalCondition Id="Show_LabelSmall">
  <CompositeCondition Operator="And">
    <ParameterCondition Name="DisplayLabel" Comparison="IsNotEqualTo" Value="None" />
    <GlobalConditionRef Id="IsSmall" />
  </CompositeCondition>
</GlobalCondition>
<GlobalCondition Id="Show_LabelLarge">
  <CompositeCondition Operator="And">
    <ParameterCondition Name="DisplayLabel" Comparison="IsNotEqualTo" Value="None" />
    <GlobalConditionRef Id="IsLarge" />
  </CompositeCondition>
</GlobalCondition>
```

## See Also

[Visual Template](#)  
[Parameters](#)  
[Content Items](#)  
[Container](#)  
[Conditions](#)  
[Alarm Indicator](#)

## Container

The Container element is mandatory, and contains the details of the layout, Plant SCADA content items (library objects) to be included in the Composite Genie, their alignment and other display options such as margins (left,

right, top and bottom).

Attribute	Description	Attribute	Description
Layout		Hotspot	Specifies the page co-ordinates where the Composite Genie is placed when inserted on to a page.
Container	Defines the layout options for the Composite Genie including the alignment, margin and stacking options.	Margin	Positioning (left, top, right and bottom margins) of the object on the page relative to the alignment.
		Layout	Specifies how the container is laid out - Stacked Horizontal, Stacked Vertical or Overlaid. For more information, refer to the More about Layouts section.
		VisibleWhen	Used to show/hide the container based on a Global condition. The container will be visible only when the condition is satisfied.
		CreateGroup	Determines whether the contents of the container will be a group. When the contents of a container are in a group, the placement of the contents are automatically adjusted to accommodate the largest object/text when the container is re-sized and also at runtime if animations are included. By default, the contents of a container are placed in a group. If the contents are not grouped, each item needs to be re-sized and placed manually. You can use the

			<p>GoTo Object dialog to view the hierarchy of items in a container.</p> <p><b>Note:</b> The size of a container is defined by the biggest object in the container.</p>
	AnimationName		<p>Sets the animation name to apply to the object. This is unique to the layout.</p> <p><b>Note:</b> An animation name should begin with a letter and contain only letters, numbers and the following symbols: _, %, \$, @ and #. It should not exceed 26 characters.</p>
	HorizontalAlignment		<p>Sets the alignment of the container to display horizontally within its parent.</p>
	VerticalAlignment		<p>Sets the alignment of the container to display vertically within its parent.</p>
	ZIndex		<p>Determines the position of the container when multiple containers are overlaid. The object with the lowest ZIndex is stacked on top of the pile while the object with the highest ZIndex is placed at the bottom of the stack of containers. The ZIndex is relative within a container whether or not objects in the container are grouped. For more information, refer to the More about Layouts section.</p>
AlarmIndicator	Sets the alarm indicator	VisibleWhen	Alarm Indicator will be

	that needs to be displayed for this composition.		displayed if the condition specified here is satisfied.
		EquipmentLink	Reference to the Alarm Indicator to be displayed as defined in the <AlarmIndicators> element.
		BorderStyle	Reference to the border style to be used for the Alarm Indicator as defined in the <AlarmIndicators> element.
Contents	Defines the Plant SCADA content items that are available within this composition.	Item ID	Reference to the Plant SCADA content item such as a Genie. Each item included here should have been included in the <ContentItems> element.
		HorizontalAlignment	Whether the item will be left, right or center-aligned.
		VerticalAlignment	Whether the item will be top, middle or bottom aligned.
		Margin	Left, top, bottom and right margins. For information, see the More about Margins topic.
		VisibleWhen	Item will be displayed if the condition specified here is satisfied.
		AnimationName	Name of the item as configured in the General Access properties. <b>Note:</b> An animation name should begin with a letter and contain only letters, numbers and the following symbols: _, %, \$, @ and #. It should not exceed 26 characters.

		ZIndex	Determines the position of the item when multiple items are overlaid. The item with the lowest ZIndex is placed on top of the stack.
Parameters	For each item that you add to a composition, you need to link it to a parameter defined in the XML template. This is done so that the value of the parameters can be passed into the substitution in the content item (for example, %Equipment%) through the Composite Genie.	Name	Name of the substitution parameter specified without the % (for example, "Equipment" for %Equipment% substitution). This name is case sensitive.
		TemplateParameter	Name of parameter as specified in the Parameter element, the value of which is passed into the substitution in the content item.

## Example - Layout

This example shows the Hotspot co-ordinates for positioning the Composite Genie at a particular location on the page.

```
<Layout Hotspot="10,10">
```

## Example - Container

This example shows the code for a container, which includes:

- layout options for the container and conditions when it is to be displayed (using the VisibleWhen attribute)
- content items and conditions when they are to be displayed (using the VisibleWhen attribute)
- parameters to be passed into template parameters

```
<Container Layout="StackedHorizontal" HorizontalAlignment="Center" AnimationName="MEO1Horz_Small">
  <AlarmIndicator VisibleWhen="Show_MEOAlarm" EquipmentLink="MEO1" />
  <Content VisibleWhen="Show_ControlMode" ItemId="031" Margin="0,-4,2,-1" VerticalAlignment="Center">
    <Parameters>
      <Parameter Name="Equipment" TemplateParameter="MEO_Equip1"/>
    </Parameters>
  </Content>
  <Content ItemId="081" Margin="0,0,0,0" VerticalAlignment="Center">
    <Parameters>
      <Parameter Name="Equipment" TemplateParameter="MEO_Equip1"/>
    </Parameters>
  </Content>
</Container>
```

## See Also

[Visual Template](#)

[Parameters](#)

[Content Items](#)

[Prerequisites](#)

[Conditions](#)

[Alarm Indicator](#)

## More About Margins and Alignment

Margins and alignment determine the positioning of a Composite Genie. They can be used in conjunction to set the [More About Layouts](#) of Composite Genies.

## Margins

Margins may be specified for a container or a content item within a container. Margins can be used to set the distance between an object and its child, and determine the positioning of an object relative to its parent.

Margins are specified as a set of four numbers separated by a comma. For example, 2, 0, 1, 3. Here 2 is the Left margin, 0 is the Top margin, 1 is the right margin and 3 is the bottom margin. The values for margins may be the same or different depending upon your requirements. Margins are specified in pixels.

Margins are applied to the container as a whole, and/or to each individual item. For example, a container margin of 2 pixels for the left, top, right and bottom can be defined with:

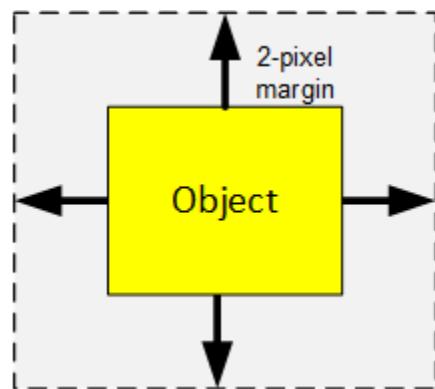
```
<Container Margin="2,2,2,2" Layout="StackedVertical" HorizontalAlignment="Left" AnimationName="Label">
```

This will apply a 2-pixel margin to the group of items within the container.

Margins can be applied to an item within the container as follows:

```
<Content ItemId="1" Margin="2,2,2,2">
```

This will apply a 2-pixel margin on all sides of the object that has the ItemID 1.

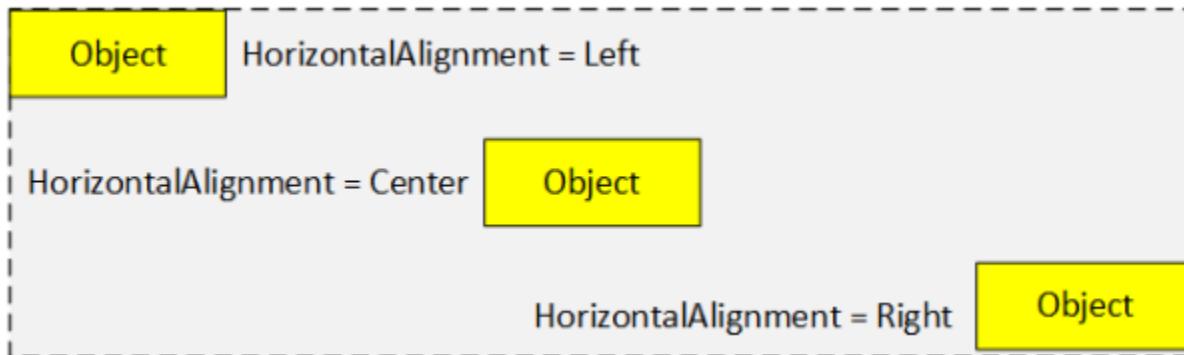


## Alignment

### HorizontalAlignment

The HorizontalAlignment attribute sets the alignment that will be applied to child objects. It can set to:

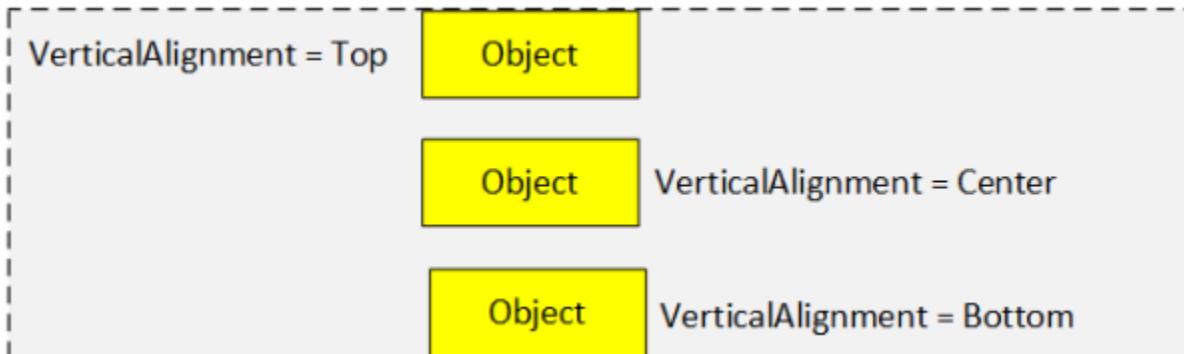
- Left - Child objects are aligned to the left of the parent object's allocated space.
- Center - Child objects are aligned to the center of the parent object's allocated space.
- Right - Child objects are aligned to the right of the parent object's allocated space.



### VerticalAlignment

The VerticalAlignment attribute sets the alignment that will be applied to child objects. It can be set to:

- Top - Child objects are aligned to the top of the parent object's allocated space.
- Center - Child objects are aligned to the center of the parent object's allocated space.
- Bottom - Child objects are aligned to the bottom of the parent object's allocated space.



## See Also

[More About Layouts](#)

### More About Layouts

Layouts can be defined at the container level or at the object level. It is recommended that layout attributes be defined for the parent object with respect to the positioning required for the child object as shown in the examples below.

**Note:** It is recommended that you group objects if you have changing animations, size or positions at runtime.

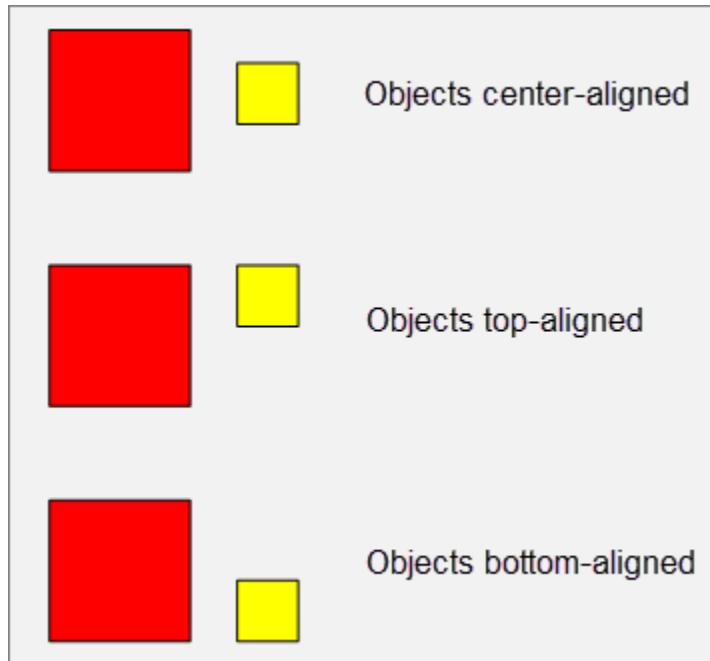
Layouts are of the following types:

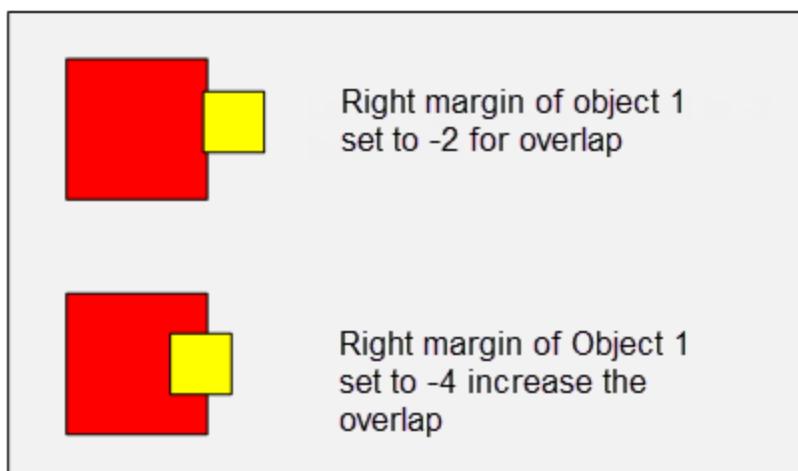
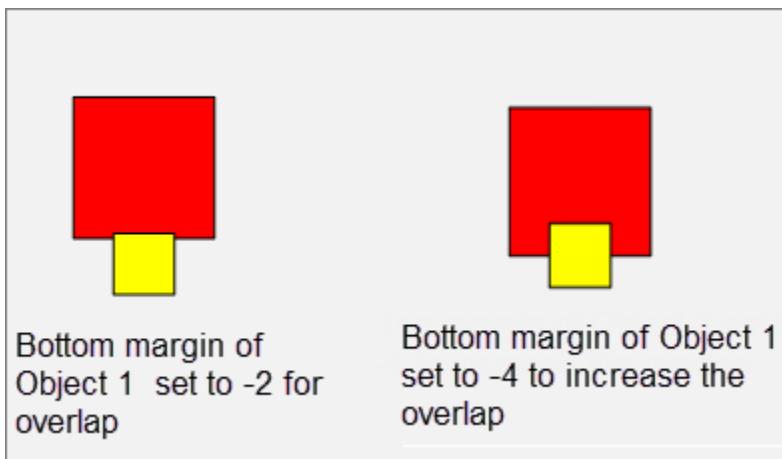
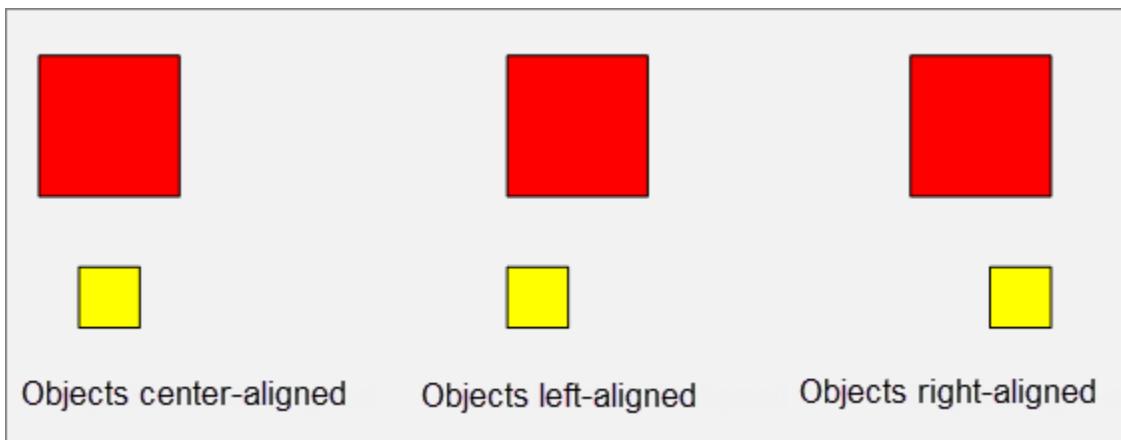
- **StackedHorizontal:** Objects are aligned from left to right. By default, the objects as a group, are center

aligned. To top- or bottom-align the objects within the group, set the VerticalAlignment attribute to "Top" or "Bottom", respectively. This needs to be set in in the XML template. If you have two objects in a group, you need to specify the vertical alignment only for the second item. The second item will then be aligned relative to the first one. Use the Margin attribute to control overlap of objects. A negative margin setting will position an item closer to the object next to it.

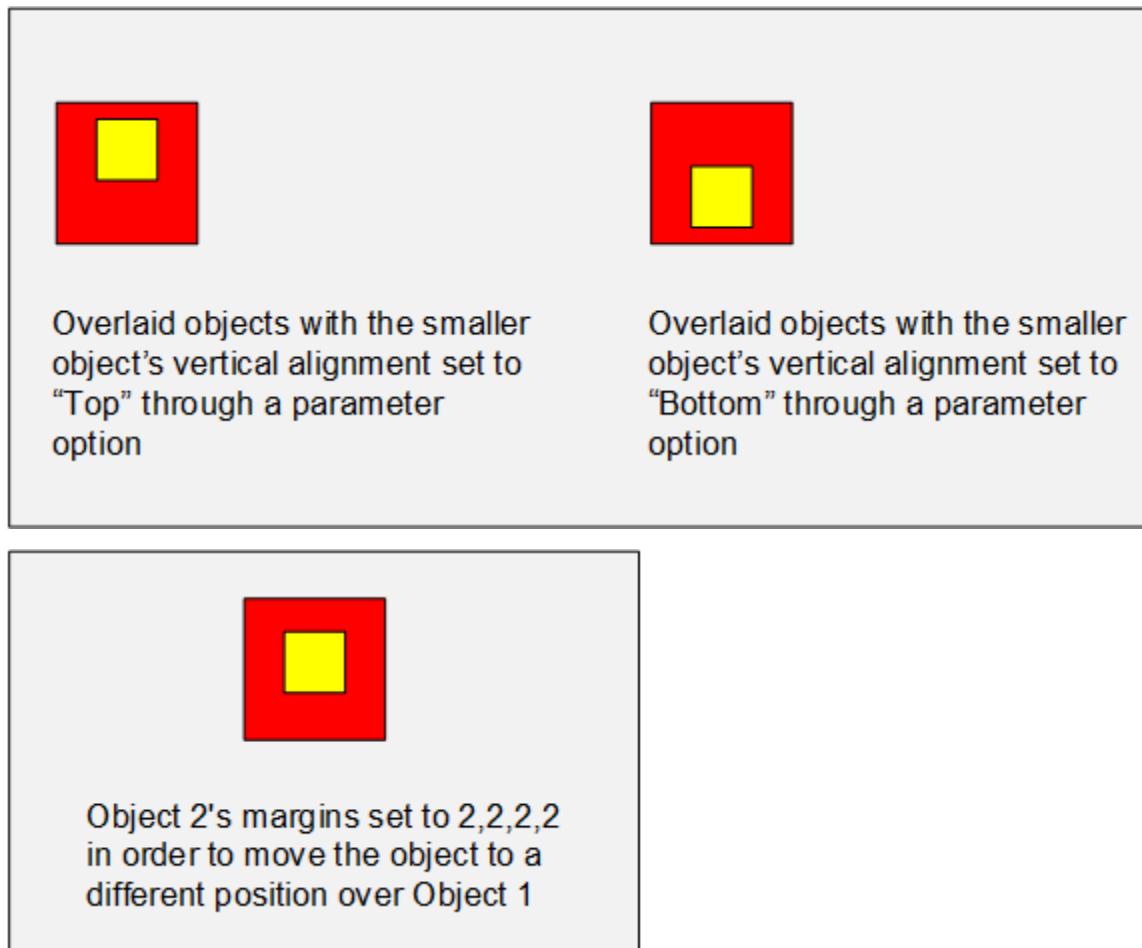
- **StackedVertical:** Objects are aligned from top to bottom. By default, the objects, as a group, are center aligned. To left- or right-align the objects within the group, set the HorizontalAlignment attribute to "Left" or "Right", respectively. If you have two objects in a group, you need to specify the horizontal alignment only for the second object. The second item will then be aligned relative to the first object. This needs to be set in in the XML template. Use the Margin attribute to control overlap of objects. A negative margin setting will position an item closer to the item below it.
- **Overlaid:** All objects are placed at the same spot, and the order in which they are placed is controlled by the value of the ZIndex. The object with the lowest ZIndex will be at the top of the overlaid stack. Bigger objects can be stacked at the bottom and overlaid with smaller objects with the smallest object on the top.

### Example: StackedHorizontal



**Example: StackedVertical**

## Example: Overlaid



## More About Alarm Indicators

Alarm indicators may be applied to a Composite Genie as a group or to an individual Genie within the composite. Alarm indicators can be applied in one of two ways:

- Using the pre-defined alarm indicators from the <AlarmIndicators> attribute. For more information see, [Alarm Indicators](#).
- Making a declaration within a composition using the <AlarmIndicator> attribute (see the example below).

## Example

In the examples below, we have defined alarm indicators within the composition instead of using the pre-defined indicators.

In the example below, the alarm indicator has been defined for the container using the <AlarmIndicator> attribute. The alarm indicator will be visible around the Composite Genie with the specified width and padding when the condition is satisfied.

```
<Container Layout="StackedVertical" AnimationName="MEO1_Small">
<AlarmIndicator VisibleWhen="Show_MEOAlarm" width="2" Padding="1" IsInside="false" />
<Content ItemId="081" Margin="0,0,0,0" VerticalAlignment="Center">
  <Parameters>
    <Parameter Name="Equipment" TemplateParameter="MEO_Equip1"/>
  </Parameters>
</Content>
```

You can also define an alarm indicator for a specific content item (Genie) using the `<AlarmIndicator>` attribute. The alarm indicator will be visible only around the content item with the specified width and padding when the "VisibleWhen" condition you have specified is satisfied.

**Note:** If a condition is not defined, the alarm indicator will be displayed regardless of user input.

## Conditions

Conditions can be used for defining dependencies in a composition. For example, if a parameter needs to be displayed based on the value selected for another parameter or a composition needs to be applied based on a set of conditions. Conditions are of the following types:

- Global Conditions
- Parameter Conditions
- Composite Conditions

The order in which conditions are defined is important because only a condition defined later can use conditions defined before it. This is to avoid cyclical reference. For example, you have defined Global condition "MEO" first followed by condition "SmallMeter". The "SmallMeter" condition can use "MEO" as a Parameter condition, but the "MEO" condition cannot use the "SmallMeter" condition (shown in Example 1 below).

Before going through details about the types of conditions, it is important to understand operators that can be used to define conditions. The section below provides details of the different types of operators.

## Operators

Conditions can be defined with the help of operators. Operators are of two types:

- Logical – This include "AND" and "OR". These operators can be used when defining Composite conditions.
- Comparison – The following comparison operators are available to perform comparisons for string, integer and Boolean values:

Operator	Used for Comparing Values of Data Types...
IsEqualTo	Boolean, String and Integer
IsNotEqualTo	Boolean
IsGreaterThan	Integer
IsGreaterThanOrEqualTo	
IsLessThan	

Operator	Used for Comparing Values of Data Types...
IsLessThanOrEqualTo	
StartsWith	String
EndsWith	
Contains	

## Global Conditions

Global conditions are conditions that need to be used frequently within an XML template. They are defined once and can be used anywhere within an XML template to incorporate parameter dependencies or selection of a composition. Global conditions can also be used:

- to pre-define the components of the Composite Genie that can be enabled or disabled
- in determining parameter dependency, and can be used in one or more compositions within the XML template.

They may include nested Parameter as well as Composite conditions (shown in Example 2 below).

Global conditions are defined using the GlobalCondition element, which comprises the following attributes:

Attribute	Description
Name	User friendly name for the condition, which needs to be unique.

### Example

In this example, the Global condition "Show\_LabelPadding" is met when the one of the predefined Global conditions in the Composite condition is met.

```
<GlobalCondition Id="Show_LabelPadding">
  <CompositeCondition Operator="Or">
    <GlobalConditionRef Id="IsLabelSideDiff" />
    <GlobalConditionRef Id="IsNotModeCompact" />
    <GlobalConditionRef Id="IsDrivePump" />
    <GlobalConditionRef Id="IsDriveBorePump" />
    <GlobalConditionRef Id="IsDriveBlower" />
  </CompositeCondition>
</GlobalCondition>
```

### Example

This example shows a nested condition that can be used to evaluate a Global condition.

```
<GlobalCondition Id="IsModeCompact">
  <CompositeCondition Operator="Or">
    <!--First Sub-condition of Composite Condition -->
    <ParameterCondition Name="MEO" Comparison="IsEqualTo" Value="0" />
    <!--Second Sub-condition of Composite Condition, which is a Composite condition joined by an "AND" operator -->
    <CompositeCondition Operator="And">
      <ParameterCondition Name="MEO" Comparison="IsEqualTo" Value="1" />
      <ParameterCondition Name="DisplayMEOCompact" Comparison="IsEqualTo" Value="true" />
    </CompositeCondition>
  </CompositeCondition>
</GlobalCondition>
```

## Parameter Conditions

Parameter conditions can be used in a composition within the Prerequisite element to determine the composition that will be selected based on user input and to toggle visibility of a parameter. Parameter conditions are also used to define Global conditions. The attributes of a Parameter condition include:

Attribute	Description
Name	User friendly name for the condition, which needs to be unique.
Value	Expected value of the parameter against which the condition will be evaluated.

### Example

```
<ParameterCondition Name="Meter" Value="true" />
```

## Composite Conditions

Composite conditions are conditions that are joined together with the AND or OR operator and function as a group. Composite conditions can be made up of Global or Parameter conditions.

Attribute	Description
Operator	This can be set to "AND" or "OR" and determines how the conditions needs to be evaluated.

### Example

In the sample code, two parameter conditions are joined together using the AND operator.

```
<CompositeCondition Operator="And">
  <ParameterCondition Name="MEO" Comparison="IsGreaterThan" Value="0"/>
  <ParameterCondition Name="MEO" Comparison="IsLessThanOrEqualTo" Value="4"/>
</CompositeCondition>
```

## See Also

- [Visual Template](#)
- [Parameters](#)
- [Content Items](#)
- [Compositions](#)

Prerequisites

Container

Alarm Indicators

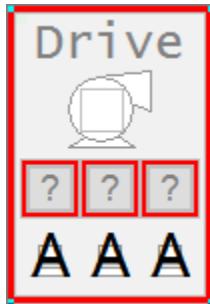
## Alarm Indicators and Alarm Flags

Alarm indicators and alarm flags are defined using Global conditions, and are included within a composition to be made visible to the user via the Presentation Options dialog box.

Given below is the code snippet for the Global condition that defines an alarm indicator.

```
<GlobalCondition Id="Show_AlarmBorder">
  <ParameterCondition Name="DisplayAlarm" Comparison="IsEqualTo" Value="true" />
</GlobalCondition>
```

Alarm indicators can be displayed on a Composite Genie and/or the related equipment associated with it (shown below).



Given below is the code snippet for the Global condition that defines an alarm flag and its positioning.

```
<GlobalCondition Id="Show_AlarmFlagTopLeft">
  <ParameterCondition Name="DisplayAlarmFlag" Comparison="IsEqualTo" Value="Top Left" />
</GlobalCondition>
<GlobalCondition Id="Show_AlarmFlagTopCenter">
  <ParameterCondition Name="DisplayAlarmFlag" Comparison="IsEqualTo" Value="Top Center" />
</GlobalCondition>
<GlobalCondition Id="Show_AlarmFlagTopRight">
  <ParameterCondition Name="DisplayAlarmFlag" Comparison="IsEqualTo" Value="Top Right" />
</GlobalCondition>
<GlobalCondition Id="Show_AlarmFlagMiddleLeft">
  <ParameterCondition Name="DisplayAlarmFlag" Comparison="IsEqualTo" Value="Middle Left" />
</GlobalCondition>
<GlobalCondition Id="Show_AlarmFlagMiddleRight">
  <ParameterCondition Name="DisplayAlarmFlag" Comparison="IsEqualTo" Value="Middle Right" />
</GlobalCondition>
<GlobalCondition Id="Show_AlarmFlagBottomLeft">
  <ParameterCondition Name="DisplayAlarmFlag" Comparison="IsEqualTo" Value="Bottom Left" />
</GlobalCondition>
<GlobalCondition Id="Show_AlarmFlagBottomCenter">
  <ParameterCondition Name="DisplayAlarmFlag" Comparison="IsEqualTo" Value="Bottom Center" />
</GlobalCondition>
<GlobalCondition Id="Show_AlarmFlagBottomRight">
  <ParameterCondition Name="DisplayAlarmFlag" Comparison="IsEqualTo" Value="Bottom Right" />
</GlobalCondition>
```

Alarm flags need to be included in a composition within the **Alarm Indicators** for them to be displayed in the Presentation Options dialog box.

```
<AlarmIndicator VisibleWhen="Show_AlarmBorder" EquipmentLink="Equip" BorderStyle="MainStyle" FlagStyle="Hidden">
  <Triggers>
    <Trigger When="Show_AlarmFlagTopLeft" FlagStyle="TopLeft" />
    <Trigger When="Show_AlarmFlagTopCenter" FlagStyle="Top" />
    <Trigger When="Show_AlarmFlagTopRight" FlagStyle="TopRight" />
    <Trigger When="Show_AlarmFlagMiddleLeft" FlagStyle="Left" />
    <Trigger When="Show_AlarmFlagMiddleRight" FlagStyle="Right" />
    <Trigger When="Show_AlarmFlagBottomLeft" FlagStyle="BottomLeft" />
    <Trigger When="Show_AlarmFlagBottomCenter" FlagStyle="Bottom" />
    <Trigger When="Show_AlarmFlagBottomRight" FlagStyle="BottomRight" />
  </Triggers>
</AlarmIndicator>
```

The triggers are used to define FlagStyle conditionally. When none of the When conditions is satisfied, the FlagStyle is set to the fallback value specified by `<AlarmIndicator FlagStyle="<fallback_value>">` element.

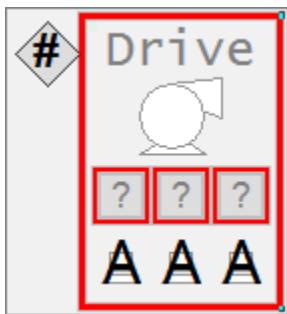
You can also set up the FlagStyle attribute within `<AlarmIndicator>` without Triggers. The FlagStyle attribute is "Hidden" by default if it is not specified.

An alarm flag can be positioned at one of the following locations:

- None: Alarm flag is not displayed. This is the default.
- Top Left
- Top Center
- Top Right
- Center Left
- Center Right
- Bottom Left
- Bottom Center
- Bottom Right

---

**Note:** If you change the size of an alarm flag and run the Update Pages command, the Composite Genie instances are re-sized to accommodate the size of the alarm flag.



## Super Genies

Super Genie library objects are non-instantiated pages that are linked to a Genie. When you save your Super Genie as a library object, it is not saved as a normal page and thus is not instantiated by default in your project. A Super Genie library object is only instantiated when it has been attached to a Genie and the Genie is placed on a graphics page in your project.

You can configure the Super Genie library object to display static and dynamic content using association Cicode functions. See [Dynamic Associations](#) for more information.

**Note:** To browse the Super Genies associated with the active project, use the **Libraries** view in the **Visualization** activity (see [Browse Libraries](#) in Plant SCADA Studio).

## See Also

[Super Genie Library Objects and the Page Properties Associations Tab](#)

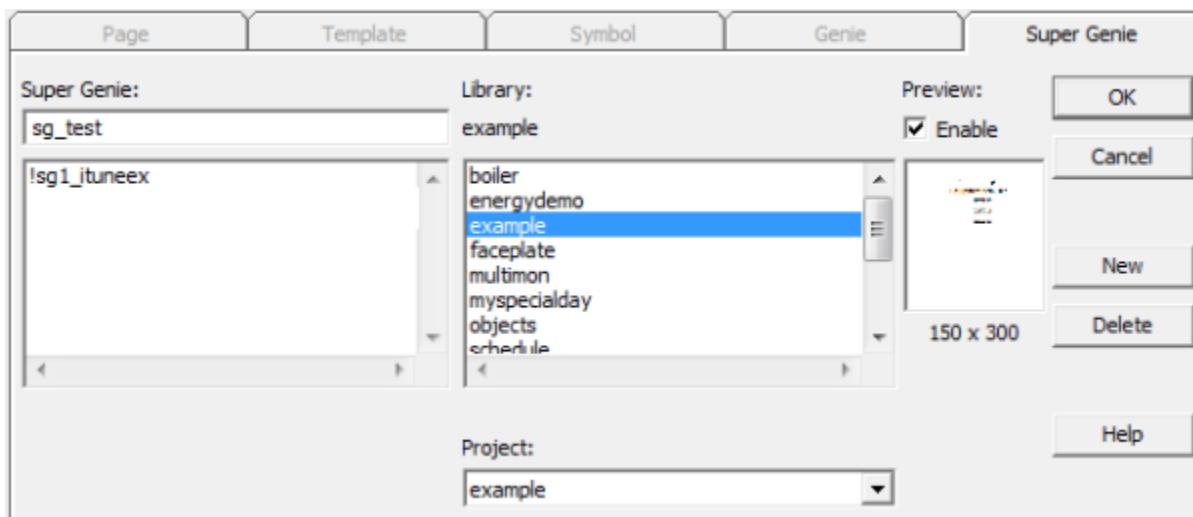
## Create a New Super Genie

You can configure and save a Super Genie page as a library object. If you save your Super Genie as a library object it is not saved as a normal page and thus is not instantiated by default in your project. A Super Genie library object is only instantiated when it has been attached to a Genie and the Genie is placed on a page in your project. Once the library object is instantiated the Super Genie is available for use in your project.

**Note:** If you configure a Super Genie in this way and name the page with an ! prefix, then you need to select **List System Pages** from the Graphics Builder Options menu to show the page with the other pages, so you can then select it and edit the page. Otherwise it will be hidden.

### To create a new Super Genie:

1. Open Graphics Builder.
2. From the toolbar select the **New** button, or select **New** from the **File** menu.
3. Click the Super Genie option. A blank Super Genie page will open.
4. Create your Super Genie as you would any graphics page.



5. From the toolbar select **Save**, or select **Save** from the **File** menu.
6. The Save as dialog opens. The Super Genie tab is active, with no other options available.
7. Enter a name for the Super Genie in the **Super Genie** field (for example, "!sg2\_test").
8. Select the **Library** and **Project** in which you would like to store the Super Genie.

Like Genie libraries, Super Genies libraries are global and can be used between projects.

9. Click **OK**.

**Note:** Save it in a Super Genie library using an exclamation mark (!) prefix. This keeps the pages hidden in the

---

configuration environment (they are visible only if attached to a Genie).

---

## See Also

[Super Genies](#)

[Dynamic Associations](#)

## Super Genie Library Objects and the Page Properties Associations Tab

When you define a Super Genie, you are actually creating a Super Genie template, similar to a page template. When the Super Genie is instantiated in the project, this template is used to create a new Super Genie page. At this point, any associations saved with the template are copied across to the Super Genie page. However, if subsequent changes are made to the page association definitions of the template (under page properties), the corresponding associations of the Super Genie page are not updated.

After the Super Genie page content has been modified, the Super Genie page created from the template needs to be updated. In the Graphics Builder select **Tools | Update Pages** (see [Update Pages](#)).

If it is not updated, then your page may display out-of-date content.

## Attach a Super Genie to a Genie

### To attach a Super Genie to a Genie:

1. Open Graphics Builder.
2. With the Genie open, select **Attach Super Genie** from the **Edit** menu.
3. Click **Add**. The Super Genie Dialog box will open.
4. Select the library to which the Super Genie belongs.
5. Select the Super Genie to attach (for example, "!PopUp\_SG") and click **OK**.  
The Super Genie is added to the Attach Super Genie list.
6. Click **OK** to save the changes, or click **Cancel**.

---

**Note:** A Super Genie can be attached to more than one Genie.

---

## See Also

[Edit a Super Genie Library Object](#)

[Edit a Super Genie Page](#)

[Dynamic Associations](#)

## Edit a Super Genie Library Object

If you modify a Super Genie while its project is running in the background, you will be prompted to 'Update Pages'. This will update your changes in the runtime environment. If a runtime page containing the Super Genie is displayed when the change is made, it will not be updated until after you exit the page and display it again.

For those pages created from a Super Genie, if the parameter [CtDraw.RSC]AllowEditSuperGeniePage is set to 0 (the default), a message will be displayed that will help prevent you from editing the Super Genie page directly. Instead you will need to edit the Super Genie Library object that created that page.

If the parameter [CtDraw.RSC]AllowEditSuperGeniePage is set to 1, a message will display that asks for confirmation before allowing you to edit the page directly.

---

**Note:** Changes made directly to a page (created from a Super Genie) will be overridden when you 'Update Pages'.

---

### To open an existing Super Genie library object:

1. Open Graphics Builder.
2. Click the **Open** toolbar button, or select **Open** from the **File** menu.
3. Select the **Super Genie** tab.
4. Select the **Project** and **Library** in which the Super Genie is stored.
5. Select the **Super Genie**.
6. Click **OK**.

### To delete a Super Genie library object:

1. Open Graphics Builder.
2. Click the **Open** toolbar button, or select **Open** from the **File** menu.
3. Select the **Super Genie** tab.
4. Select the **Project** and **Library** in which the Super Genie is stored.
5. Select the **Super Genie** and click **Delete**.  
A confirmation message will display.
6. Click **OK** to delete the library object.

Deleting a Super Genie that has been attached to a Genie will not cause that Genie to become inoperable at runtime.

---

**Note:** If you delete a Super Genie that is in use, the object configured to call the page at runtime will become inoperable.

---

## See Also

[Super Genies](#)

## Edit a Super Genie Page

If you modify a Super Genie while its project is running in the background, you will be prompted to 'Update Pages'. This will update your changes in the runtime environment. If a runtime page containing the Super Genie is displayed when the change is made, it will not be updated until after you exit the page and display it again.

### To open an existing Super Genie page:

1. Go to Graphics Builder.

2. Click the **Open** toolbar button, or select **Open** from the **File** menu.
3. Select the **Pages** tab.
4. Select the **Project** to which the page belongs.
5. Select the **Super Genie**.
6. Click **OK**.

#### To delete a Super Genie page from the project:

1. Click the **Open** toolbar button, or select **Open** from the **File** menu.
2. Select the **Pages** tab.
3. Select the **Project** to which the page belongs.
4. Select the **Super Genie** and click **Delete**.
5. A confirmation message will display.
6. Click **Yes** to delete the page and its records from the project. Click **No** to delete the page from the project only. Click **Cancel** to cancel the operation.

**Note:** If you delete a Super Genie page that is in use, the object configured to call the page at runtime will become inoperable.

#### See Also

[Super Genies](#)

## Web Content

Plant SCADA allows you to natively incorporate web-based content into your runtime system. This is achieved by adding a Web Content object to a page in Graphics Builder (see [Add a Web Content Object](#)).

Each instance of a Web Content object has a **URL** property that allows you to specify the content that is displayed when the host graphics page is launched. If required, you can update the URL at runtime using the Cicode function [DspWebContentSetURL](#).

### **WARNING**

#### **LOSS OF CONTROL**

- Do not use the Plant SCADA Web Content control to display an untrusted or potentially unsafe web site, as this may leave your runtime system vulnerable to malicious activity.
- Do not enter a URL in the Web Content control that will allow users to navigate to an untrusted or potentially unsafe web site (for example, a search engine).

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Before you implement the Web Content control as part of your runtime system, you should carefully consider the security implications of the content that will be accessible. If using an internet-based website or application, you will be externally exposing your SCADA system and network.

When open at runtime, the Web Content browser operates in the same way as a typical web browser. However, the following functionality is not supported:

- Signing in to Google sites is not supported.
- A current Windows user login cannot be used for Single Sign-On support.
- Links that launch a popup window or a new tab will open the destination in the current browser window.
- Download links will not work.
- Upload links will not work (including drag-and-drop uploads).
- Javascript functions that open prompt or confirm dialogs (`window.alert`, `window.prompt` and `window.confirm`) will not work.
- Dialogs that prompt a user to confirm if they want to leave a page (specifically those generated by the Javascript `window.onbeforeunload` event handler) will not appear. Navigation will proceed immediately.
- HTTP Basic Authentication is not supported (the login dialog does not display).
- DRM-protected video streams are not supported.
- You cannot choose to accept the risks and continue to a site that has certificate issues (for example, if a certificate has expired). This means you cannot connect to an Industrial Graphics page using the server's IP address instead of a hostname.

When a Web Content control is displayed, any user-specific data (such as cookies, storage and browsing history) is stored in the profile folder for the current Windows user (for example, `C:\Users\<Windows User Name>`). The following sub-directory is used:

`\AppData\Local\AVEVA Plant SCADA\WebContent\Profiles\<User>\`

Where `<User>` is the name of the user that is currently logged in to Plant SCADA. Some changes to the user name may be required to create a valid path.

If no one is currently logged in to Plant SCADA, this directory changes to:

`\AppData\Local\AVEVA Plant SCADA\WebContent\Profiles\$ViewOnly\`

Any data that is not user-specific is stored in the following directory:

`\AppData\Local\AVEVA Plant SCADA\WebContent\CommonData`

If you want to automatically delete the application data associated with a Web Content user when they log out, you can use the parameter [\[Security\]DeleteWebContentProfile](#).

The current browser session will be closed when you navigate away from a graphics page that hosts a web content object.

---

**Note:** If a second (or subsequent) instance of the Plant SCADA runtime client is launched on a computer by the same user, the process ID (PID) of the client process will be included in the path used by the Web Content objects in the second (or subsequent) instances of the client. For example:

`\AppData\Local\AVEVA Plant SCADA\WebContent.<PID>`

When this occurs, all browser instances will use incognito mode. The Profiles sub-directory will not be created, and any user-specific data will not be stored. The PID folder will be deleted when the client process exits.

---

You can use the INI parameter [\[Debug\]ChromiumDevToolsPort](#) to enable debugging for a Web Content browser.

---

**Note:** You need to consider which users will be logging in to the runtime system and if they need their Web Content user data protected. For more information, see [Securing Web Content User Data](#).

## See Also

[Graphics Objects](#)

[Graphics Object Types](#)

## Securing Web Content User Data

When you include a [Web Content](#) object in a project, you need to consider which users will be logging in to the runtime system and if they have any requirements to protect their user data (such as cookies, storage and browsing history).

Plant SCADA's runtime security system supports two types of user account:

- Windows™ users (see [Integrate Windows™ User Groups](#)).
- User defined in a Plant SCADA project (see [Add a Plant SCADA User](#)).

The following information describes the security implications for these two types of users.

- **Plant SCADA Users**

When a Plant SCADA user logs in and opens a page containing a Web Content object, their user data will end up in an individual folder. However, this folder is stored under the user account of the Windows user that is currently logged into the operating system.

For example:

C:\Users\<OS Windows User>\AppData\Local\AVEVA Plant SCADA\WebContent\Profiles\<Plant SCADA User>\

If any users are able to access the file system, they could potentially view the user data files of other Plant SCADA users.

- **Windows Users**

When a Windows user logs in and opens a page containing a Web Content object, their user data will end up in an individual folder. However, this folder is stored under the user account of the Windows user that is currently logged into the operating system.

For example:

C:\Users\<OS Windows User>\AppData\Local\AVEVA Plant SCADA\WebContent\Profiles\<Windows User>\

If any users are able to access the file system, they could potentially view the user data files of other Windows users.

This is not a concern if the Windows user attempting to access Plant SCADA is also logged into the operating system using their own account. This way, their user data folder will be secured within their individual Windows user account.

---

**Note:** If a Web Content object hosts a website or application that stores sensitive user data, it is recommended that you use Windows user accounts to secure Plant SCADA runtime. Each user also needs to be logged into the local operating system before they use the same account to log in to Plant SCADA.

## See Also

[Graphics Objects](#)

## Graphics Object Types

# Dynamic Associations

Dynamic associations are placeholders you can use on a graphics page to link to equipment, 'equipment.item' and/or variable tag references via association Cicode functions at runtime.

**Note:** Prior to Plant SCADA 2015, dynamic associations were referenced within Super Genie pages. With the introduction of equipment, 'equipment.item' references and partial associations, the concept is now referred to as dynamic associations.

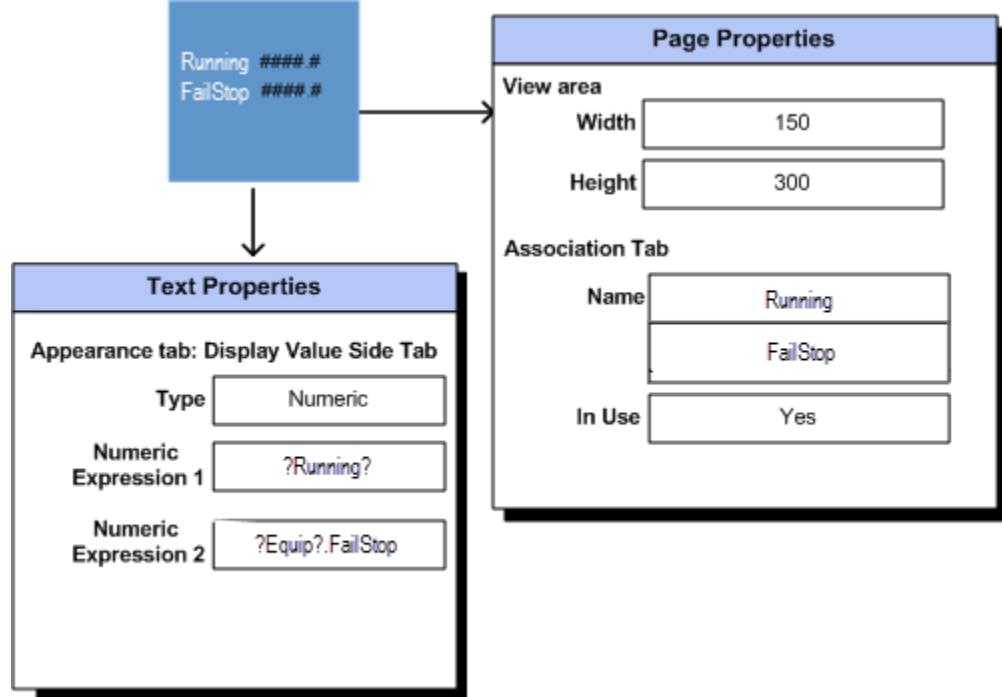
When you replace part or all of a command or expression with a placeholder, the placeholder is called an association.

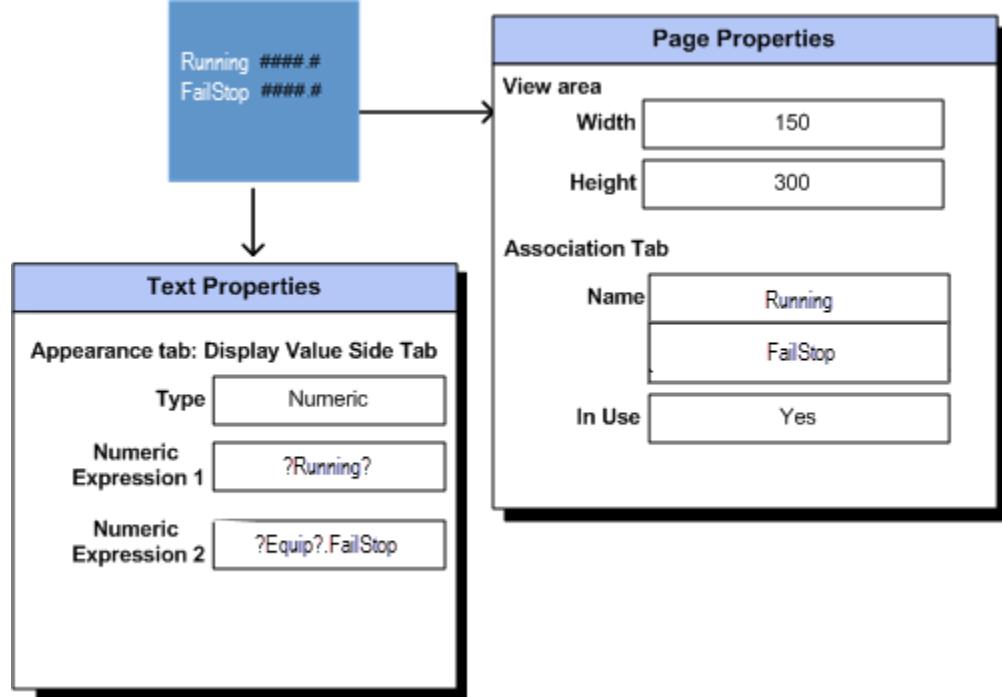
You can use associations in any graphics object properties that accept tags, commands and expressions. You can also use associations in:

- log messages for object touch and keyboard commands
- tool tips
- page keyboard commands
- the comment for Trend objects.

However, you cannot use the dynamic association syntax in a report, alarm, trend, or background Cicode function.

In the illustration below, Numeric Expression 1 has been replaced with a placeholder '?Running?'. The question marks ('? ?') denote it is a dynamic association.

Only part of Numeric Expression 2 has been replaced with the placeholder '?Equip?'.  




## See Also

- [Define Dynamic Associations](#)
- [Use Equipment References with Dynamic Associations](#)
- [Use Structured Tags with Dynamic Associations](#)
- [Link Dynamic Associations](#)
- [Dynamic Associations and Areas](#)
- [Use Constants with Dynamic Associations](#)
- [Use Arrays and Array Offsets with Dynamic Associations](#)

## Define Dynamic Associations

To define a dynamic association, enclose the association name between question mark (?) characters. You can also include the data type, though this is optional.

?<Association>?

?<Data Type> <Association>?

Everything between the question mark characters will be replaced at runtime with the associated value. The association may be referenced as many times as necessary on a graphics page.

---

**Note:** If you have upgraded and historically used numbers in your associations, these will continue to work as numbers are regarded the same as a name.

The data types for an association allow you to pass tags of the following types:

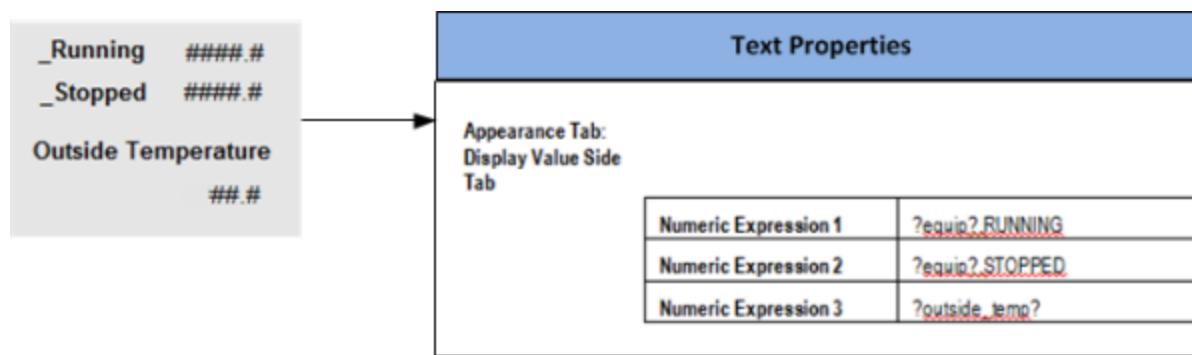
- BYTE
- BCD
- DIGITAL
- INT
- UINT
- LONG
- LONGBCD
- STRING
- REAL

When the <Data\_Type> is not specified, Plant SCADA will automatically use the relevant typecast variable type at runtime.

---

**Note:** Explicit type casting to BCD, BYTE, LONGBCD, UINT and ULONG types is not supported for the associated tag. Use automatic data type associations for these types. If using automatic data type associations, be aware that it may be more difficult to decipher error logs.

In the following example an association has been used in numeric expressions 1 to 3. The names used are "equip" and "outside\_temp". As they are associations, the names have been placed within "? ?".



For Numeric expression 1 and 2, only part of the expression has been associated with a placeholder. For numeric expression 3, the whole expression has been replaced with an association.

This means at runtime, a variable tag, equipment name or 'equipment.item' reference will replace the associations "?equip?" and "?outside\_temp?" in the expression.

You can also use the association with prefixes, suffixes and logical operators. For example:

- ?INT equip?.Running
- Plant100.?INT equip?.Running
- ?INT 1?\_PV , TAG\_?INT 1?, TAG\_?INT 1?\_PV
- (NOT(?30?)OR(?52?)OR(?152?)) AND(?54?)
- (NOT ?30? OR ?52? OR ?152?) AND ?54?).

**Note:** When using logical operators, use brackets or spaces in the dynamic associations. If brackets or spaces are not used, the compiler will produce an "Incompatible type" error message.

This will be expanded to the complete reference at runtime. For example, an association of "?equip?.Running" when associated with equipment named "Pump1" at runtime, will become "Pump1.Running".

**Note:** Tag elements (for example, Field, Override Mode or Control Mode) and Tag Items (for example, V, Q, T) are not supported for partial associations in dynamic associations.

## See Also

[Use Equipment References with Dynamic Associations](#)

## Use Equipment References with Dynamic Associations

When defining dynamic associations, you can use full or partial associations to generate the complete 'equipment.item' reference when a page is displayed at runtime.

For example, you can define dynamic associations to display the values of the following equipment from the Example Project:

- Example\_Motor.FailStop
- Example\_Motor.FailStart
- Example\_Motor.Running.

To do this, you can configure object properties as follows:

Numeric expression	?Equip.FailStop?
Numeric expression	?Equip.FailStart?
Numeric expression	?Equip.Running?

For each numeric expression, a full association has been used as a dynamic association.

When configuring an object to call the dynamic association page, each 'equipment.item' reference will be passed to the dynamic association using an association Cicode function. Each dynamic association needs to be configured in the object.

```
Ass(-2, "Equip.FailStop", "Example_Motor.FailStop"); Ass(-2, "Equip.FailStart",
"Example_Motor.FailStart"); Ass(-2, "Equip.Running", "Example_Motor.Running");
WinNewAt("equipSg",100,100,0);
```

Using a partial association (where only part of the 'equipment.item' reference is used), you can configure the numeric expressions as follows:

Numeric expression	?Equip?.FailStop
Numeric expression	?Equip?.FailStart
Numeric expression	?Equip?.Running

When configuring the object to call the dynamic association page, only one 'equipment.item' reference needs to be passed to the dynamic association using an association Cicode function.

```
Ass(-2, "Equip", "Example_Motor"); WinNewAt("equipSg",100,100,0);
```

## See Also

[Create a Dynamic Association Page](#)

[Display a Dynamic Association Page](#)

## Use Structured Tags with Dynamic Associations

Dynamic associations support direct concatenation of the tag with other information (as do Genies). For example, the following is valid:

```
?INT 1?_PV , TAG_?INT 1?, TAG_ ?INT 1?_PV
```

## See Also

[Link Dynamic Associations](#)

## Link Dynamic Associations

Plant SCADA allows you to link dynamic associations. Linking refers to a situation where an existing set of associations are passed onto a new page.

To do this, configure the graphics object's input command to use the AssChain Cicode functions when displaying

the related pop up.

## See Also

[Dynamic Associations and Areas](#)

## Dynamic Associations and Areas

When you display a dynamic association, the area of the page is inherited from its parent. For example, if the parent page is in area 1, when you display a dynamic association it will also be area 1. This allows you to call the same dynamic association from different pages in different areas.

The inherited area may be avoided by defining a specific area for the dynamic association. This means, every instance of the dynamic association will have the same area, despite the area inherited from its parent.

Dynamic associations will only inherit areas if their area is blank.

## See Also

[Use Constants with Dynamic Associations](#)

## Use Constants with Dynamic Associations

You can use the following types of constants with dynamic associations:

- STRING
- INTEGER
- DIGITAL
- REAL
- LONG.

To pass a constant you need to format the argument in the association function to include a single quote on either side. For example, to pass the constant data **1.2345** into a dynamic association, you would call the association function like this:

```
Ass(hWin, sArg, "'1.2345'");
```

To pass a variable tag or 'equipment.item' tag reference, you don't need the single quotes. For example, to pass variable tag **TAG1** into a dynamic association, you would call the association function as follows;

```
Ass(hWin, sArg, "TAG1");
```

Or if using 'equipment.item' to reference the variable tag:

```
Ass(hWin, sArg, "Pump.Speed");
```

The ability to pass constants into dynamic associations is restricted in that the constant association can be used where you can enter a normal Cicode tag. For example, a keyboard command, a symbol address field, and so on.

## See Also

[Use Arrays and Array Offsets with Dynamic Associations](#)

## Use Arrays and Array Offsets with Dynamic Associations

Dynamic association can accept array elements or entire arrays as substitution. Passing an element of an array is achieved by referencing the element, as shown here:

```
AssPopUp("MyPopUp", "DigArray[42]");
```

To pass an entire array to a dynamic association, only the array name is used. For example:

```
AssPopUp("MyPopUp", "DigArray");
```

## Use Arrays with Dynamic Associations

To pass an entire array to a dynamic association, configure it to accept the array instead of a single value. Use the following syntax for the dynamic association substitution string:

```
?<Data Type><Substitution String>? [<element>]
```

Only arrays of data type DIGITAL, INT, REAL, and LONG are supported.

## Use Array Offsets with Dynamic Associations

You can use array offsets with a tag that is an array.

For example, you may have a tag (named *Tag1*) defined as an array of four elements. When using the GENERIC protocol, the tag has an address I1[4]:

Tag1 = I1[4] ->	I1
	I2
	I3
	I4

Tag1 represents the registers I1, I2, I3 and I4. When Tag1 is used with the Ass() Cicode function it behaves as follows:

- If you associate a value using Ass(hWin, "X", "Tag1", 0) then the substitution string ?X?[0] gives the value in I1.
- If you associate a value using Ass(hWin, "X", "Tag1", 0) then the substitution string ?X?[2] gives the value in I3.
- If you associate a value using Ass(hWin, "X", "Tag1[0]", 0) then the substitution string ?X?[2] gives the value in I3.
- If you associate a value using Ass(hWin, "X", "Tag1[1]", 0) then the substitution string ?X?[2] gives the value in I4. This is because the two offsets are added together to determine the final offset within the array variable.
- If you associate a value using Ass(hWin, "X", "Tag1[2]", 0) then the substitution string ?X?[2] gives the error

#ERR. This is because the sum of the two array offsets gives a position (Tag1[4]) that is outside the bounds of the array.

You can also use array offsets with a tag that is not an array.

For example, you may have a tag (named *Tag2*) defined as a single value. When using the GENERIC protocol, the tag has an address I1:

Tag2 = I1 ->	I1
--------------	----

When Tag2 is used with the Ass() Cicode function, it behaves as follows:

- If you associate a value using Ass(hWin, "X", "Tag2", 0) then the substitution string ?X?[0] gives the value in I1. This is because a non-array tag is equivalent to an array with one element.
- If you associate a value using Ass(hWin, "X", "Tag2[0]", 0) then the substitution string ?X?[0] gives the value in I1. This is because a non-array tag is equivalent to an array with one element.
- If you associate a value using Ass(hWin, "X", "Tag2", 0) then the substitution string ?X?[2] gives the notification #ERR. This is because a non-zero offset cannot be applied to a non-array tag.
- If you associate a value using Ass(hWin, "X", "Tag2[0]", 0) then the substitution string ?X?[2] gives the notification #ERR. This is because a non-zero offset cannot be applied to a non-array tag.
- If you associate a value using Ass(hWin, "X", "Tag2[1]", 0) then the substitution string ?X?[2] gives the notification #ERR. This is because a non-zero offset cannot be applied to a non-array tag.
- If you associate a value using Ass(hWin, "X", "Tag2[2]", 0) then the substitution string ?X?[2] gives the notification #ERR. This is because a non-zero offset cannot be applied to a non-array tag.

## See Also

[Dynamic Associations](#)

## Create a Dynamic Association Page

You can configure any graphics page (normal or Super Genie) to use dynamic associations. This allows the page to be reused in different contexts as the actual data that is used is assigned at runtime when the page displays.

### To create a dynamic association page:

1. Open Graphics Builder.
2. From the toolbar select **New**.

Or:

From the **File** menu, select **New**.

3. On the New dialog box, select **Page** or **Super Genie**.

If you select Page, the Page Template dialog box will display. Choose the required **Template** and click **OK**.

4. Add the required objects to the page.
5. Right-click on an object and select **Properties**.
6. Configure the dynamic associations in the required property fields.
7. Click **OK** to apply the changes.

8. From the **File** menu select **Save**.
9. In the Save dialog, select the project to which the page will belong.
10. Name the page and click **OK**.

## See Also

[Display a Dynamic Association Page](#)

## Display a Dynamic Association Page

To display a page that uses dynamic associations, you need to add an object to a graphics page that is configured to call the page. For example, you could use a new menu item, a button, or a symbol.

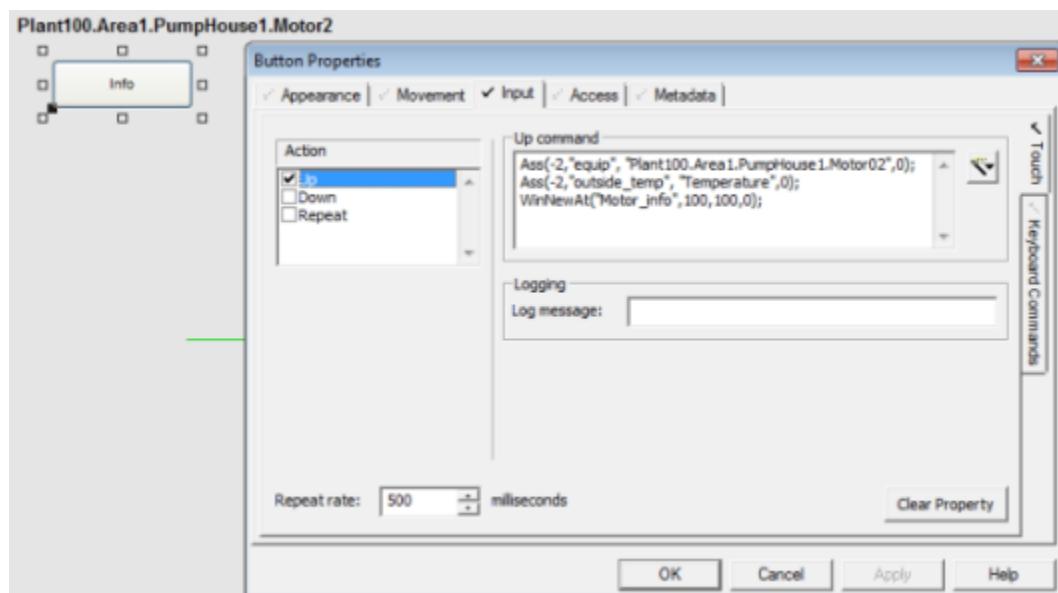
When configuring the object, you use one of the Association Cicode functions. At runtime, this links the associations with the equipment, 'equipment.item' or variable tag references.

If you use the association Cicode function, you need to use it once for each value you want to pass to the Super Genie.

### To use a Dynamic Association:

1. Open Graphics Builder.
2. Open the page to which you would like to add an object that calls the dynamic association page.
3. Place and configure an input object on the page. This object will process the dynamic associations.

This is illustrated in the screen shot below, where a button was placed on the graphics page that will process two associations before displaying the popup page.



The **Up** input command for the button has been configured as follows:

```
Ass (-2,"equip","Plant100.Area1.PumpHouse1.Motor2",0);
Ass(-2,"outside_temp","Temperature",0); WinNewAt("Motor_info",100,100,0);
```

Where:

- "equip" is the name of the substitution and "Plant100.Area1.PumpHouse1.Motor2" is the equipment

name assigned to the association.

- "outside\_temp" is the name of the substitution and "Temperature" is the tag name assigned to the association.
  - "Motor\_info" is the name of the dynamic association page.
4. Save the graphics page.

## See Also

[Dynamic Associations](#)

## Animation Points

A graphics page in Plant SCADA is typically made up of several objects. Each point on a graphics page where an object is displayed is called an Animation Point. When you add a graphics object to your page, it is automatically allocated an animation number (AN) to represent the associated animation point.

Animation Numbers can be linked to static items (such as a bitmap or an object without an animation expression), or animated items where an animation expression is defined.

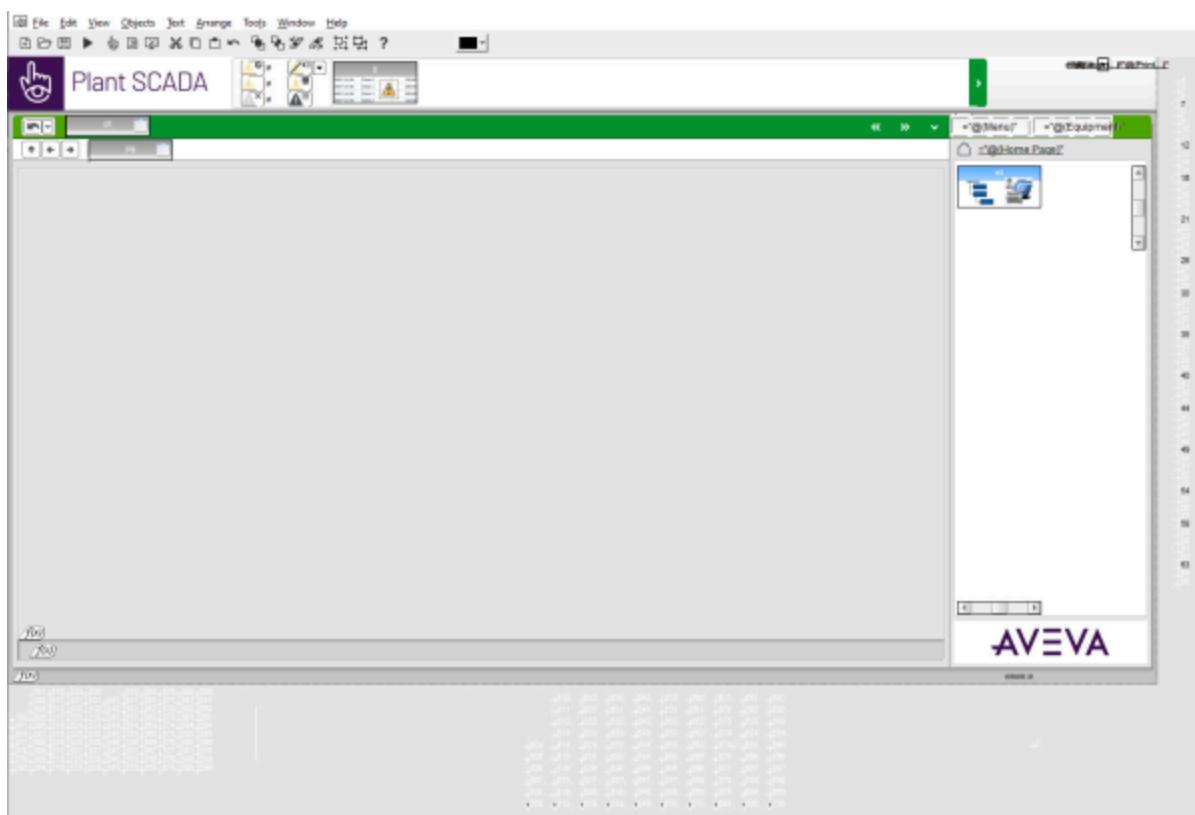
---

**Note:** The number of graphic objects that you can use is limited by the performance of your computer, and the number (and complexity) of the animation expressions used. A good rule of thumb is to try and keep the number of objects (and hence ANs) to less than 3000.

---

AN's are automatically generated for graphics objects; however the AN's 1 and 2 are used for automatically displaying system information such as messages and keyboard entry. In the case of the SxW and Tab Style templates, a large number of ANs are reserved for future development. These reserved ANs are listed around the edges of a page when it is displayed in Graphics Builder.

- [Animation Numbers on a SxW Page](#)



- Animation Numbers on a Tab Style Page



Genies are a combined set of one or more graphics objects. This means each Genie results in two or more ANs and animation records being saved to the project configuration; one for the Genie and one for at least one object within it. The total number of Genies that can be used on a single page is determined by the number of graphics objects that are defined within each Genie, and the total number of ANs and animation records.

You can add individual animation points to a graphics page by using the **Cicode Object** tool (add a Cicode Object

without a command).

When configuring the graphics object on the page you can name the object and use that name to retrieve the object at runtime. See [Naming Graphic Objects](#) for more information.

**Note:** You can locate any object on a page by referencing its AN (Animation Number or Name). To locate an object using its AN or Name, use the **Goto Object** tool (accessible via the **Tools** menu). See [Locate an Object](#).

## See Also

[Naming Graphic Objects](#)

[Symbols](#)

## Naming Graphic Objects

In Plant SCADA animation numbers are not static, meaning the number allocated to the object may change to the next available animation number on the graphics page. For example; a symbol comprised of two objects may use AN's 4 and 5; however when the symbol is pasted onto the graphics page, these AN's may already be in use. In this case the AN's would change to the next two available AN's. This may make it difficult to reference the object at runtime, as the AN does not uniquely identify the object. To make it easier to reference an object at runtime you can now name individual graphics objects within a Page, Template, Super Genie, Genie and Symbol. Graphic objects include: AN, Line, Rectangle, Pipe, Symbol, Symbol Set, Group, and Trend. (See [General Access to Objects](#) for more information).

The name of the object is unique to the name space to which the object belongs to. For example;

- an object on a page <Graphics Object Name>
- an object in a group <Group Name>.<Graphics Object Name>
- an object in a genie <Genie Instance Name>.<Graphics Object Name>

This means if you have a text object named Text4 within a symbol, the full name is actually <SymbolName>.Text4. As a result, you could potentially have multiple text objects named Text4 on the page, with the name given to the symbol making it unique. The animation name is static, thus you can search for the object at runtime using the name.

**Note:** You can locate any object on a page by referencing its AN (Animation Number or Name). To locate an object using its AN or Name, use the **Goto Object** tool (accessible via the **Tools** menu). See [Locate an Object](#).

## See Also

[Symbols](#)

## Interaction with Graphics Pages at Runtime

To enable an operator to interact with a graphics page at runtime, you can use the following commands and controls:

- **Touch commands** — triggered via a mouse click, or similar (see [Touch Commands](#))

- **Keyboard commands** — triggered via a keyboard key, or sequence of keys (see [Keyboard Commands](#))
- **Slider controls** — change the value of an analog variable by dragging an object on the graphics page (see [Slider Controls](#)).

You can assign privileges to commands and controls, and send a message to the command log each time an operator issues a command.

## Touch Commands

You can assign touch commands to the objects on a graphics pages. Touch commands allow an operator to send commands to the runtime system by clicking (with the mouse or similar) on an object on the graphics page. (For buttons, the command can be executed by highlighting the button with the cursor keys on the keyboard and pressing Enter.)

You can define several commands for an object, one command to execute when the operator clicks on the object, another for when the operator releases the mouse button, and another to operate continuously while the operator holds the mouse button down.

For example, a drive can be jogged by starting it when the mouse button is depressed and stopping it when the mouse button released; variables can be incremented while the mouse button is held, and so on.

---

**Note:** You can define a disable condition for any object on a page (including buttons). When the condition is active, the operator cannot interact with the object.

---

## See Also

- [Object Touch Commands](#)
- [Edit Common Object Properties](#)

## Slider Controls

Slider controls allow an operator to change the value of an analog variable by dragging an object on a graphics page. Sliders also move automatically to reflect the value of the associated variable tag.

To configure a slider, you associate a variable with an object. When the operator moves the object, the value of the variable will change. Objects can be set up to slide vertically and/or horizontally, or they can be rotated.

A slider is configured via the **Slider** tab on the Object Properties dialog box. Slider properties can be configured to work in tandem with other object properties, such as Fill Color, Movement, and Scaling.

## See Also

- [Sliders](#)

## Configuring Commands - Frequently Asked Questions

### Q: How do I start a motor with the keyboard?

**A:** Define a Page Keyboard command that sets the value of the digital variable to 1, for example:

- Key Sequence: **ENTER** or **F5**
- Command: **Conv\_Motor = 1**

**Q: How do I start a motor with a button?**

**A:** Define a Button command that sets the value of the digital variable to 1 (for example **Conv\_Motor = 1**).

**Q: How do I adjust a setpoint from a keyboard entry?**

**A:** Define a Page Keyboard command to set the setpoint to a new value, for example:

- Key Sequence: **##### ENTER**
- Command: **SP1 = Arg1**

**Q: How do I enter a command that is bigger than the width of the field?**

**A:** Use an Include file or write a Cicode function and call that function. For details, see [Using Include \(Text\) Files](#) and [Writing Functions](#) respectively.

## AVEVA™ Industrial Graphics

AVEVA Industrial Graphics is a thin-client solution for the delivery of HTML5 graphics and functional content to desktop and mobile browsers. Content is created using the Plant SCADA Industrial Graphics Editor.

Plant SCADA supports two types of Industrial Graphics:

- **Pages** — used to host a set of symbols.
- **Symbols** — Industrial Graphics objects that are stored within libraries.

These are managed in Plant SCADA Studio, along with Industrial Graphics **Menus** and **Styles** (see [Design an Industrial Graphics Application](#)).

The following tools and components are also provided with Plant SCADA to create and run Industrial Graphics applications.

### Industrial Graphics Editor

Pages and symbols are edited using the AVEVA Industrial Graphics Editor. This tool allows you to create basic elements, such as rectangles, lines, and text elements that can support real-time animations. You can also embed symbols to develop complex graphics with common components, and incorporate symbols from the Industrial Graphics libraries provided with Plant SCADA.

### Industrial Graphics Server

AVEVA Industrial Graphics are accessed by clients via an Industrial Graphics Server. It authenticates clients details and provides access to the graphical and functional content. See [Install an Industrial Graphics Server](#).

For runtime operation, it is recommended that you set up a dedicated Industrial Graphics Server. Use Plant SCADA's **Deployment** functionality to remotely manage and deploy project versions to the Industrial Graphics Server. To do this, you need to set up the host computer as a deployment client (see [Deployment](#)).

With any update method of deployment (Prompt, Notify or Force) the Industrial Graphics Web Server

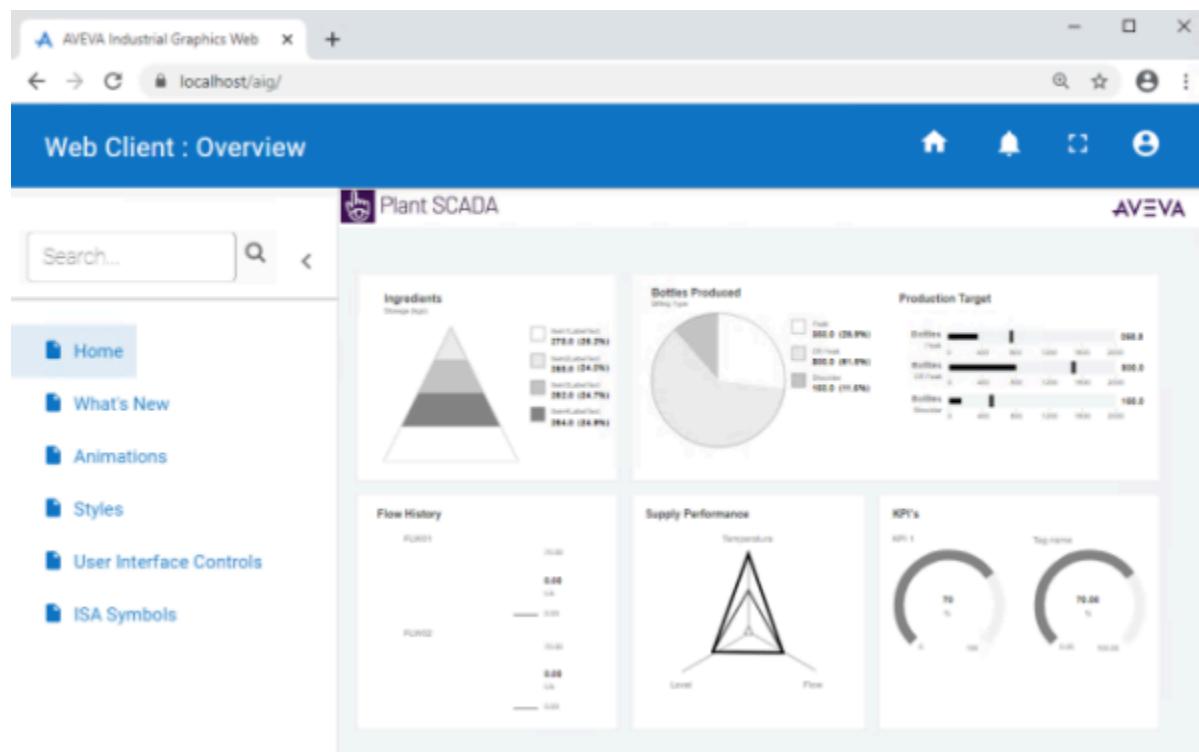
automatically activates the newly deployed project and version. When this occurs, the client shows content from the most recent deployment and informs you of the update with a notification.

If Runtime Manager is installed on the same machine and you are using the "Prompt" or "Notify" update method for deployment, then you have to restart the Runtime Manager to activate the newly deployed project and version. For more information refer to [Specify the Update Method](#).

**Note:** If you would like to review an Industrial Graphics application during development, you need to have an Industrial Graphics Server (connected to a System Management Server) installed locally on your development workstation.

## Industrial Graphics Client

An Industrial Graphics application can be viewed in an HTML browser on any computer or device that is authenticated by the Industrial Graphics Server. When connected, you will see the pages you have created, arranged according to the menu structure defined in the Visualization activity (see [Use the Industrial Graphics Web Client](#)).



The page displayed above is an example of an Industrial Graphics Web Client application that you can access in the ExampleSA project.

If you would like to view this content on a development workstation with a locally installed Industrial Graphics Server, run the ExampleSA project and direct a browser to the following address:

- <https://localhost/aig>

To view the project on a remote Industrial Graphics Server you need to:

1. Set up an Industrial Graphics Server (see [Install an Industrial Graphics Server](#)).
2. Deploy the ExampleSA project to the Industrial Graphics Server.

3. Connect a remote browser to the Industrial Graphics Server.

For a full description of how to build an Industrial Graphics application and run the Web Client, see [Design an Industrial Graphics Application](#).

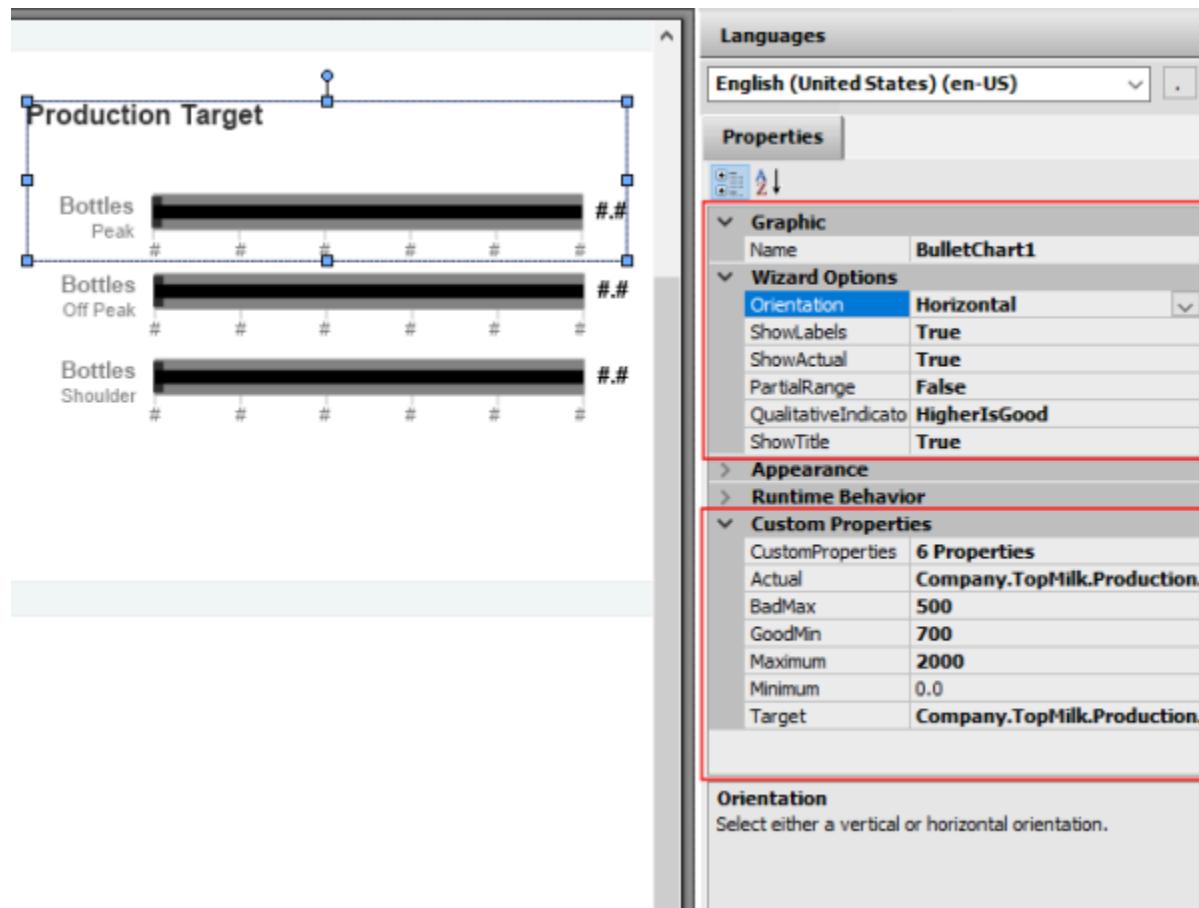
**Note:** The Industrial Graphics Web Client does not use indicators such #COM, #BAD or a dithered background to show the quality or status of device communications. For Industrial Graphics applications, quality and status is indicated by using a status element, or by applying styles to primitive objects that reflect the quality and/or status of I/O communications. For more information, see [Configure Style Overrides for Quality and Status Elements](#).

## IndustrialGraphics\_Include Project

The IndustrialGraphics\_Include project is a system project that is installed with Plant SCADA. It contains a default set of Industrial Graphics which includes charting and dashboard symbols as well as ISA symbols. These can only be embedded within Industrial Graphics pages and symbols.

The included symbols have been designed to support Industrial Graphic Styles, which allow you to centrally manage the appearance of elements, status animations and numeric values via the **Standards | Styles** view in Plant SCADA Studio (see [Styles](#)).

The charting symbols also offer many configuration options when selected within the Industrial Graphics Editor. These are accessible via the **Wizard Options** and **Custom Properties** sections of the Properties grid.



Help is provided for a selected option at the bottom of the Property grid.

If you would like to use these symbols when creating Industrial Graphics content, you need to add **IndustrialGraphics\_Include** to your project as an included project (see [Add an Included Project](#)). It is not automatically included when you create a project using a Situational Awareness, SxW, Tab Style or any other starter project.

---

**Note:** The **IndustrialGraphics\_Include** project is a system include project, and only appears in the Projects tree if the **System Projects** filter is applied to the Projects activity.

---

Licensing for AVEVA Industrial Graphics is managed using the AVEVA Enterprise Licensing system, a common platform for AVEVA Enterprise Software product licenses. For more information, see [AVEVA™ Enterprise Licensing](#).

## Design an Industrial Graphics Application

Creating an application that displays in an Industrial Graphics Web Client involves the stages described below.

---

**Note:** If you would like to review the output of a design during development, you need to have an Industrial Graphics Server (connected to a System Management Server) installed locally on the development workstation. While this is suitable for the design process, it is recommended that you deploy the project to a dedicated Industrial Graphics Server for runtime operation.

---

### Project Setup

You can add Industrial Graphics to any new or existing Plant SCADA project. However, if you want to make use of the default set of Industrial Graphics symbols provided with Plant SCADA, you need to add the system project **IndustrialGraphics\_Include** as an included project.

The symbols contained in this project will then be available to embed within your Industrial Graphics pages and symbols.

### Design

1. Create the required pages and symbols. The pages you create can be used to host a set of symbols.
  - Pages are accessible via the **Pages** view in Plant SCADA Studio (see [Browse Industrial Graphics Pages in Plant SCADA Studio](#))
  - Symbols are accessible via the **Libraries** view Plant SCADA Studio (see [Browse Industrial Graphics Libraries in Plant SCADA Studio](#)).

From these views you can create and open items in the Industrial Graphics Editor. You can also create new Libraries to organize your symbols.

2. In the Industrial Graphics Editor, configure any animations so that they point to the required tags and equipment.items.
3. Configure the application **Menu**.

The **Visualization** activity can be used to build a hierarchical menu that an operator can use to access your Industrial Graphic pages in a client. See [Configure a Menu for Industrial Graphics Pages](#).

4. If required, you can modify the **Styles** used for elements, status animations and numeric values.

Styles define a set of visual properties for text, lines, graphic outlines, and interior fill. They allow you to establish consistent visual standards. They are defined in Plant SCADA Studio's **Standards** activity. See [Styles](#).

---

**Note:** It is recommended that you review the list of supported graphical elements and limitations before you start creating an application. See [Supported Graphical Elements and Known Limitations](#).

---

## Compile the Project

1. To build your Industrial Graphics application, compile the project.
2. Run the project to engage live I/O data.

## Iterative Development

To review your design on the development workstation (using a local Industrial Graphics Server) enter the following address in a browser:

- `https://localhost/aig`

Be aware, however, that the connected System Management Server may be configured to use a non-default port (see [Advanced Configuration for a System Management Server](#)). This changes the port the AIG server uses to listen, which means you need to include the current **HTTPS Port** setting in the address as follows:

- `https://localhost:<HTTPS Port value>/aig`

You can continue to build the application and test the output in the browser. However, you will need to recompile the project to view all changes, particularly if you have added or deleted embedded symbols.

## Deploy

1. Follow the normal [deployment](#) procedure to deliver a project version to the deployment client located on the Industrial Graphics Server.
2. To see live data, make sure your Plant SCADA I/O servers are running with your project.  
After the application is deployed, it will be available to any computer or device that can access the Industrial Graphics Server.

## View the Industrial Graphics Web Client in a Browser

Before you can view the Web Client in a browser, you need to consider the following:

- **Authentication** — does the current client user have appropriate access?

The person logging in to the Industrial Graphics application needs to be part of a domain group associated with the "Industrial Graphics Users" or "Industrial Graphics RW Users" security role. See [Secure the Industrial Graphics Web Client](#) for more information.

The Windows security group their account is a member of should also be assigned to a Role within the Plant SCADA project to allow authorization by the I/O server. See [Roles](#).

- **Licensing** — is the Industrial Graphics Server appropriately licensed?

If the Industrial Graphics Server cannot acquire a license, it will run in Single Session Mode. This provides access to a single user on a first-come-first-serve basis. However, if more users require access the server will need an available license. See [AVEVA™ Enterprise Licensing](#).

You can then point the browser to one of the following addresses:

- `http://<IPAddress>/aig`
- Or:
- `http://<NodeName>/aig`

Where `<IPAddress>` or `<NodeName>` corresponds to the computer where the application is deployed.

As with a development workstation, if the connected System Management Server is configured to use a non-default port (see [Advanced Configuration for a System Management Server](#)) you will need to include the current **HTTPS Port** setting in the address as follows:

- `https://<IPAddress:<HTTPS Port value>/aig`
- `https://<NodeName:<HTTPS Port value>/aig`

## See Also

[Use the Industrial Graphics Web Client](#)

[Enable Tag Writes for Industrial Graphics Applications](#)

## Enable Tag Writes for Industrial Graphics Applications

You can use an AVEVA™ Industrial Graphics application to write to variable tags in a Plant SCADA project. However, to restrict access to this functionality, it is not enabled by default.

To allow tag writes to occur from an Industrial Graphics client, there are three things you need to consider.

1. The "Industrial Graphics R/W Users" security role on the Industrial Graphics Server.

This security role is created locally by Plant SCADA when you install an Industrial Graphics Server on a computer. It is used to manage access for client application users that require read/write access to a Plant SCADA runtime system.

You need to confirm that users of an Industrial Graphics application are part of this group. See [Configure User Access for an Industrial Graphics Web Client](#).

2. Configuration of your variable tags.

Your variable tags need to be configured to support writes from one or more Plant SCADA roles.

To do this, you need to set the **Write Roles** property for each variable tag that will have writes enabled. The Property Grid allows you to select one of the roles configured in your Plant SCADA project, or you can manually enter a comma-separated list to include multiple roles. See [Add a Variable Tag](#).

---

**Note:** Plant SCADA runtime does not support online changes for variables, which means any changes you make to the Write Roles setting for a variable tag will not be implemented until you restart the I/O server.

3. The roles in your Plant SCADA project.

The user that is currently logged in to the Industrial Graphics application needs to be part of a Windows™ domain group that is associated with the role specified in the **Write Roles** property.

For more information on how to associate a Windows domain group with a **role**, see [Integrate Windows™ User Groups](#).

To cover the requirements of points 1 and 3, it is recommended that you establish a Windows domain group for any Industrial Graphics users that require write access to your runtime system. You can then associate this group with the relevant role in your Plant SCADA project, and with the **Industrial Graphics R/W Users** security role on the Industrial Graphics Server.

**Note:** If you have renamed any Roles in your project, you should also restart the I/O server to synchronize the changes at runtime, particularly if you use the Cicode function [UserUpdateRecord](#) to recompile a local project configuration.

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

The status of write operations is not reflected by the status element or quality styles applied to elements. Design your pages so that an operator can determine if a value they have selected/entered has been set or rejected by displaying the current value.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

A computer that hosts an Industrial Graphics Server must be configured to use an English regional locale. This ensures that any values with unit separators are interpreted correctly when entered by an operator.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** When a write occurs for a tag that is a byte data type, you need to confirm that the value is within the range 0-255, otherwise an incorrect value may be written to the tag. It is recommended you use range checking to avoid an unexpected value being written to a tags that are a byte data type.

## **See Also**

[Design an Industrial Graphics Application](#)

[Use the Industrial Graphics Web Client](#)

## **Create Industrial Graphics**

Industrial Graphics are graphics you can create to visualize data in an HMI/SCADA system.

The Industrial Graphic Editor is used to create Industrial Graphics from basic elements, such as rectangles, lines, and text elements. You can also use the Industrial Graphic Editor to embed and configure an Industrial Graphic from the Graphic Toolbox library of graphics.

After you create an Industrial Graphic, you can embed it into another graphic or an HMI system window and use it at run time.

You can embed an Industrial Graphic in a template or instance of an object providing several ways to visualize object-specific information quickly and easily. Embedding a graphic in a template means that you can update one graphic and cascade the changes throughout your application.

Depending on your development requirements, you can select where and how to store industrial graphics.

- Create and store graphics as a standard set that you can re-use, such as a generic valve graphic.
- Store graphics as templates if you want to use the graphics in multiple instances at run time.
- Store graphics for use in a specific application.

## In This Chapter

[The Industrial Graphic Editor](#)

[Embedding Graphics](#)

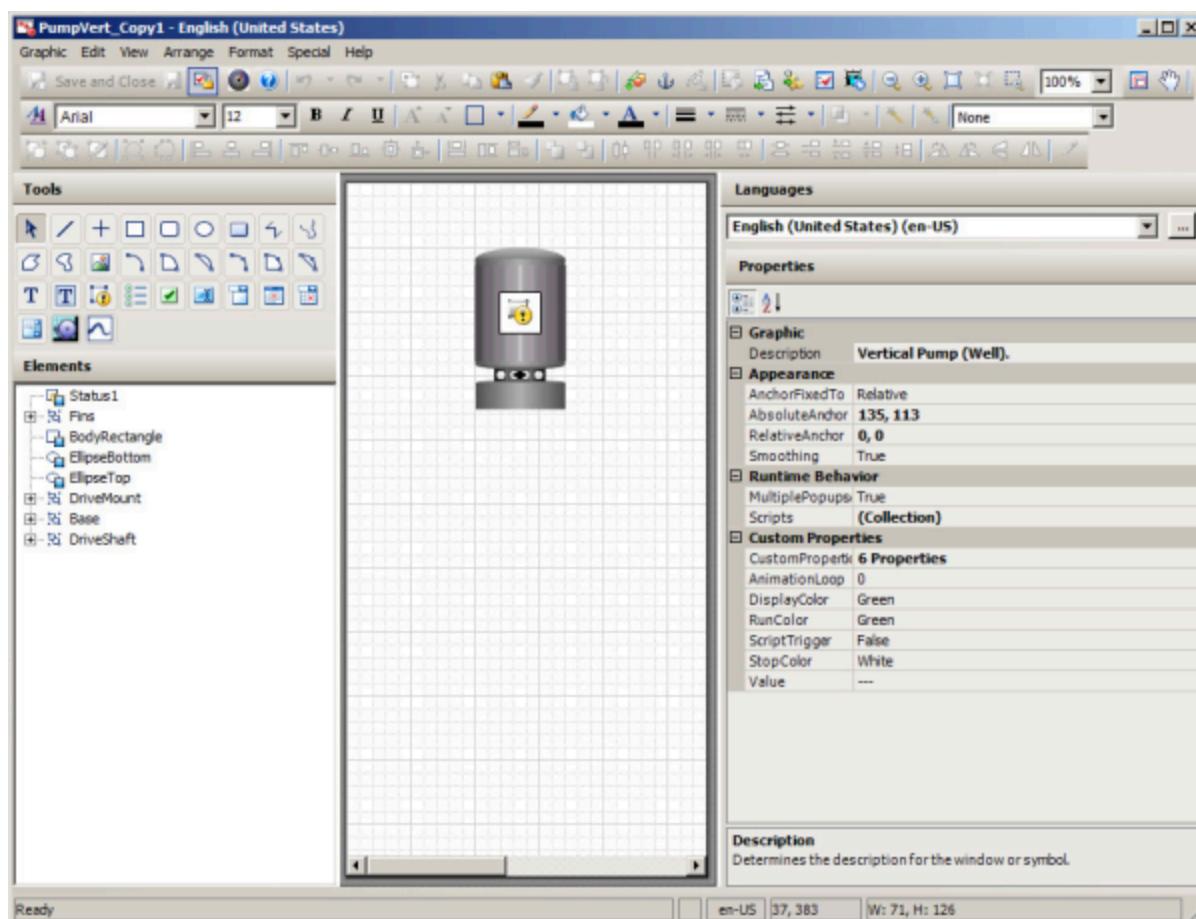
## The Industrial Graphic Editor

The Industrial Graphic Editor allows you to create an Industrial Graphic. It also allows you to open an existing graphic created in your HMI/SCADA software for editing.

To create an Industrial Graphic in the editor, select a basic graphical object, called an element, from the Tools panel and place it on the drawing area, called the canvas. Typical elements are lines, rectangles, ellipses, curves, and so on.

Then, change the appearance of drawn elements by accessing their properties directly or by graphically manipulating them. You can also change the appearance of an embedded Industrial Graphic by configuring the associated custom properties. Finally, you can configure animations for the elements or the graphics.

After you open the Industrial Graphic Editor, you will see the various tools and palettes to create and customize graphics.



The Industrial Graphic Editor includes the following areas:

- **Tools Panel:** A collection of elements you use to create your graphic.
- **Canvas:** The area in which you place and edit elements to create a graphic.
- **Elements List:** List that displays named elements on the canvas in a hierarchical view.
- **Language Selector:** List that displays the configured languages for the graphic. For more information, see .
- **Properties Editor:** Shows the properties belonging to one or more currently selected elements.
- **Animation Summary:** Area that shows you a list of animations belonging to the currently selected element. It is only visible if an element is selected.
- **Symbol Wizard Editor:** The Symbol Wizard Editor is a feature of the Industrial Graphic Editor to create graphics containing multiple visual and functional configurations called Symbol Wizards. For more information, see [Creating Multiple Configurations of a Graphic](#).

**Note:** The Industrial Graphics Editor (and its associated Styles and Languages dialogs) are only available in English, French, German, Simplified Chinese and Japanese.

## Related Topics

[Create Industrial Graphics](#)

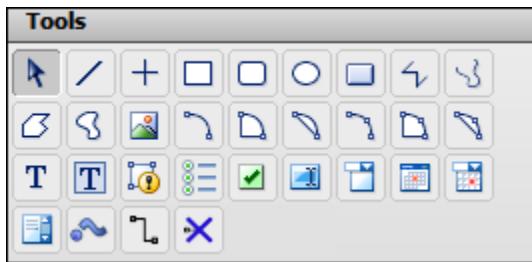
[Tools Panel](#)

[Elements List](#)

[Properties Editor](#)  
[Animations Summary](#)  
[Canvas](#)  
[Elements](#)  
[Properties](#)  
[Animations](#)

## Tools Panel

The Tools panel contains elements you can select to create your graphic on the canvas.



The Tools panel includes:

- Basic objects, such as lines, rectangles, polygons, arcs, and so on.
- A pointer tool to select and move elements on the canvas.
- Windows controls, such as combo boxes, calendar controls, radio button groups, and so on.
- A status element that you can use to show quality and status of a selected equipment.item.

For more conceptual information, see [Elements](#).

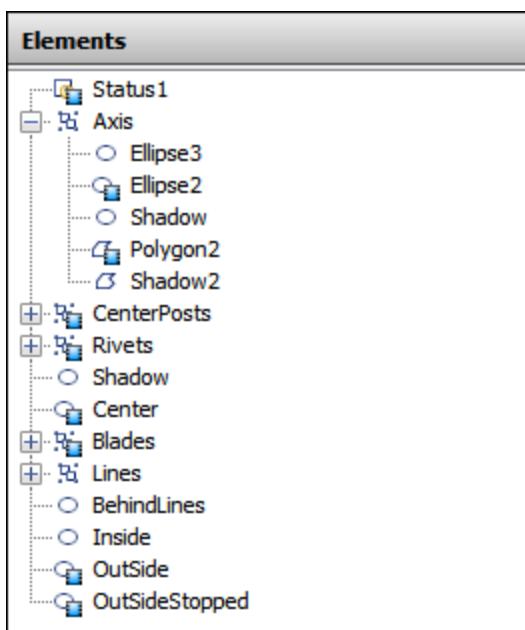
For more information on how to use elements, see [Working with Graphic Elements](#).

## Related Topics

[The Industrial Graphic Editor](#)

## Elements List

The Elements List is a list of all elements on the canvas.



The Elements List is particularly useful for selecting one or more elements that are visually hidden by other elements on the canvas. Use the Elements List to:

- See a list of all elements, groups of elements, and embedded graphics on the canvas.
- Select elements or groups of elements to work with them.
- Rename an element or a group of elements.

---

**Caution:** If you rename an element or a group, the animation references to it do not automatically update. You must manually change all animation links referencing the old name. For more information, see [Substituting References in Elements](#).

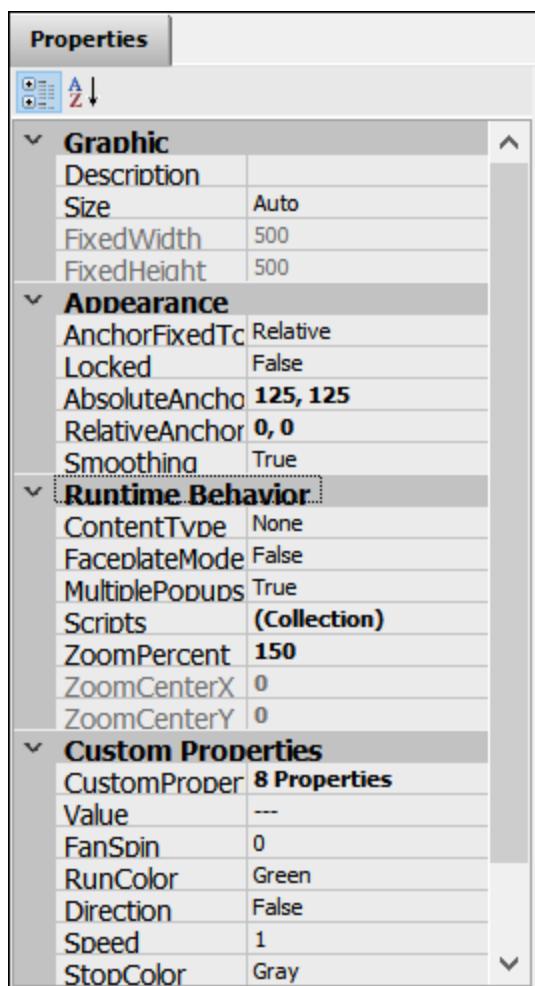
---

## Related Topics

[The Industrial Graphic Editor](#)

## Properties Editor

Use the Properties Editor to view and set properties for the selected element or group of elements.



For more conceptual information about element properties, see [Properties](#).

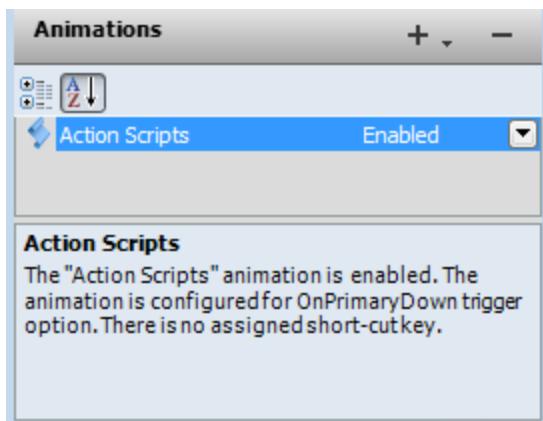
For more information on how to use element properties, see [Editing Common Properties of Elements and Graphics](#).

## Related Topics

[The Industrial Graphic Editor](#)

## Animations Summary

Use the Animations summary to review, select, and configure the animation behavior of a selected element.



For an overview of the different animation types, see [Animation Types](#).

For more information on how to use the animations, see [Animating Graphic Elements](#).

## Related Topics

[The Industrial Graphic Editor](#)

## Canvas

The canvas is your drawing area. Use it like any image editing software by drawing elements and changing them to suit your requirements.

## Related Topics

[The Industrial Graphic Editor](#)

## Elements

You use elements to create a graphic. The Industrial Graphic Editor provides the following:

- Basic elements such as lines, rectangles, ellipses, arcs, and so on
- Status element to show a quality status icon
- Windows controls, such as combo boxes, calendar controls, radio button groups, and so on

You can create the following from existing elements on the canvas:

- Groups
- Path graphics

You can embed the following on the canvas:

- Imported Client Controls.

## Related Topics

[The Industrial Graphic Editor](#)

[Basic Elements](#)

[Status Element](#)

[Windows Common Controls](#)

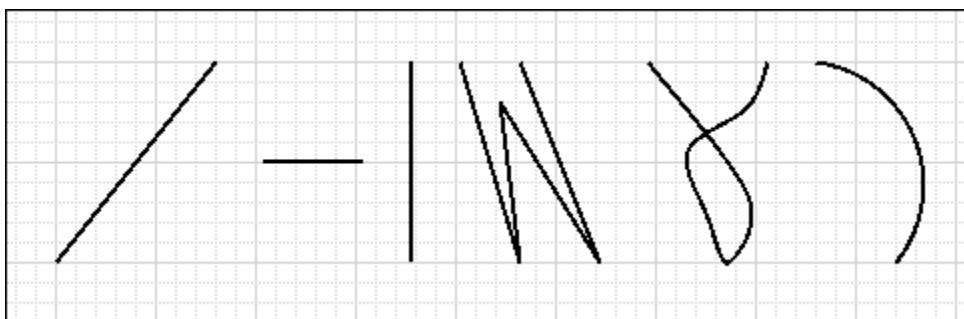
[Groups](#)

[Path Graphics](#)

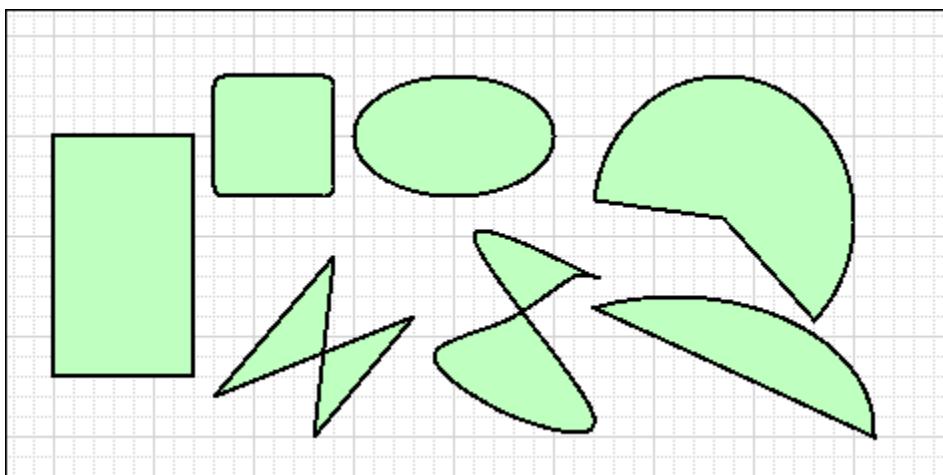
## Basic Elements

You can use the following basic elements to create a graphic:

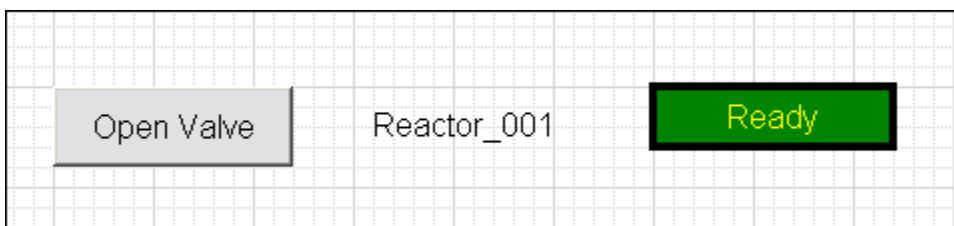
- Open elements, such as lines, H/V lines, polylines, curves, and arcs.



- Closed elements, such as rectangle, rounded rectangle, ellipse, polygon, closed curve, pie, and chord. You can draw arcs, pies, and chords from two points or from three points.



- Text elements, such as buttons, text, and text boxes.



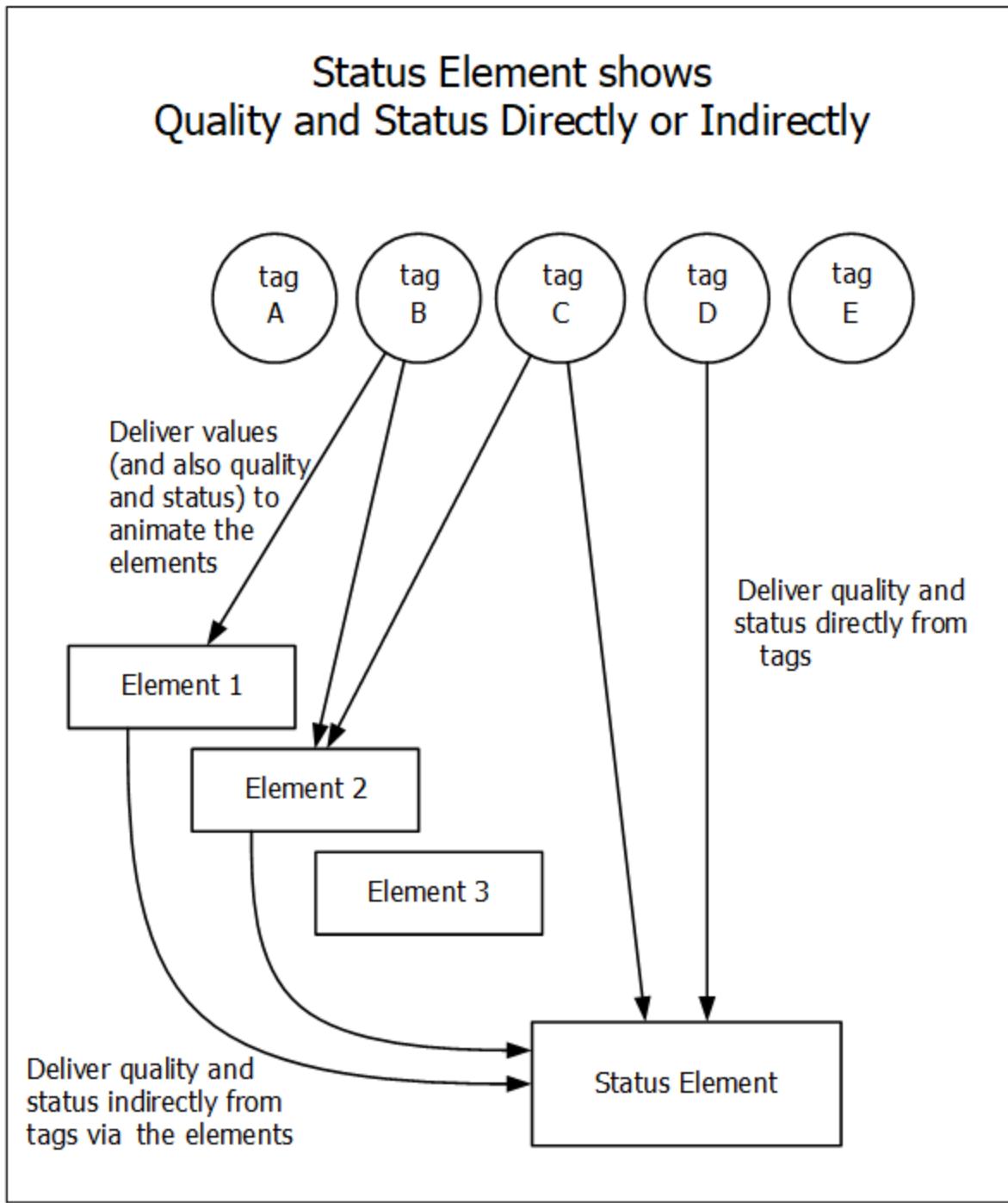
## Related Topics

[Elements](#)

### Status Element

The status element provides a graphical representation of the communications status of an equipment.item or a tag, and the data quality of the equipment.item or tag's value. Use the status element to monitor and indicate communications status and data quality of:

- All equipment.items or tags used in one or more specified animated elements at the same hierarchical level.
- One or more specified equipment.items or tags.



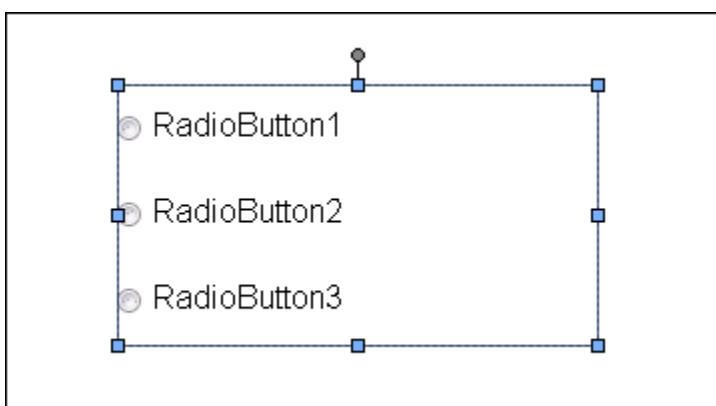
## Related Topics

[Elements](#)

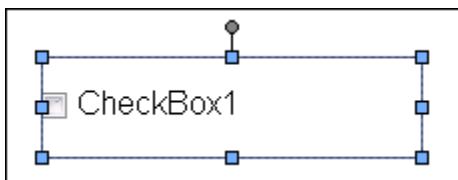
## Windows Common Controls

Using Windows common controls, you can add extended user interaction to your graphic. You can use:

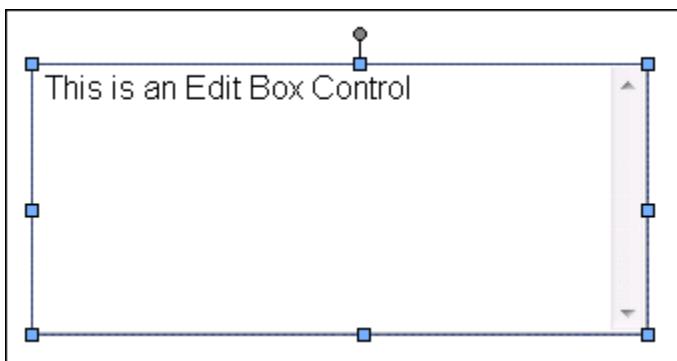
- A radio button group to select an option from a mutually exclusive group of options.



- A check box to add a selectable option.



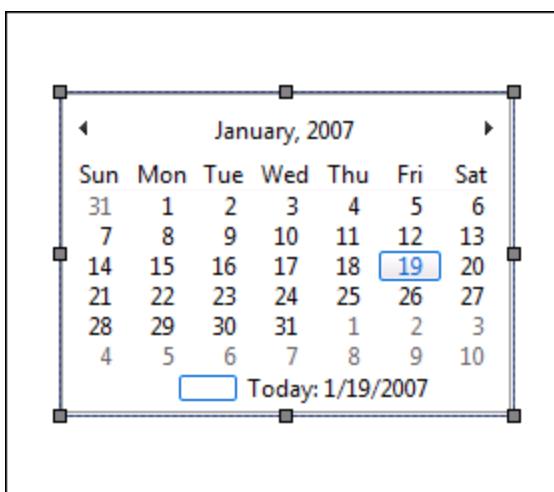
- An edit box to add an entry box for text.



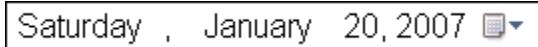
- A combo box to select an option from a drop-down list.



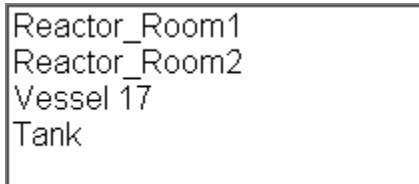
- A calendar to use a date selection control.



- A date and time picker to select a date and time in a compact format.



- A list box to select one or more options from a list.



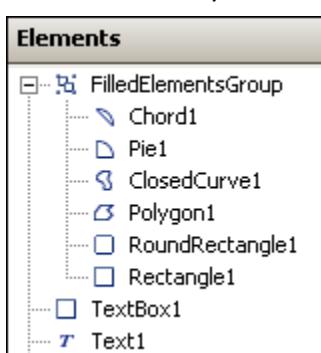
## Related Topics

[Elements](#)

## Groups

Grouping enables you to combine elements as a unit. Groups can contain elements and other groups.

Groups are shown in the Elements List with a default name, such as Group1. They are shown as a branch in the element hierarchy.



For example, you can create a series of elements that model a valve in your facility. When the valve has all the properties and animations you want, you can group the elements together.

You can then work with the elements as one set of elements or, by selecting the elements in the Elements List,

you can work with the individual elements in the group without having to break the group. This is called inline editing.

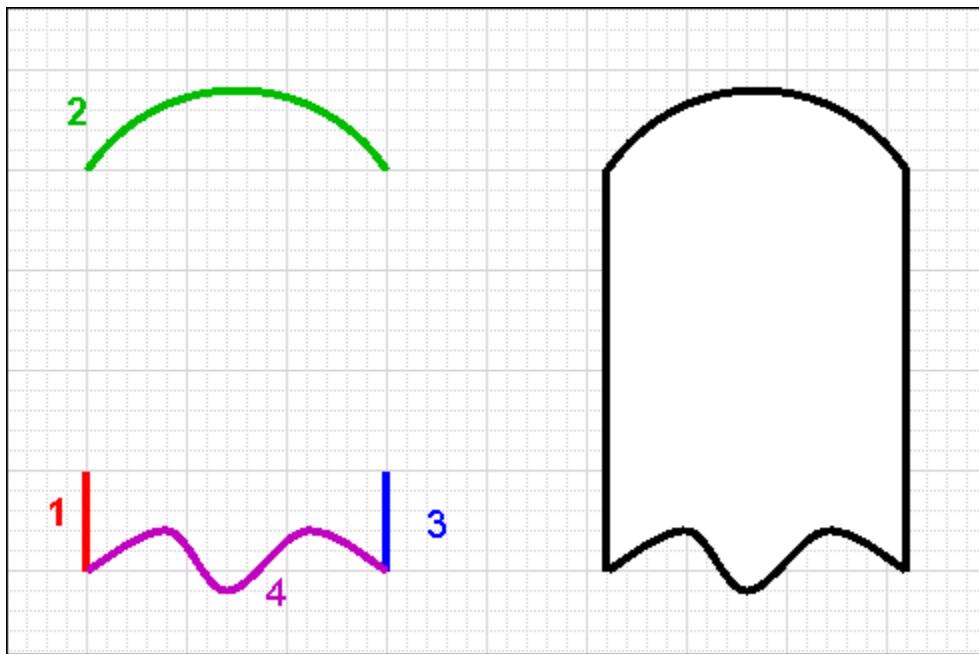
Another advantage of inline editing is that you can easily select an individual element graphically without having to know its element name.

## Related Topics

[Elements](#)

## Path Graphics

Path graphics are elements that combine selected open elements, such as lines, H/V lines, polylines, curves, and arcs, into a single closed graphic element.



A path graphic depends on the:

- Order in which you drew the elements. Each element is linked to the next element by z-order. The z-order of the elements is the order shown in the Elements List.
- Direction in which you drew its elements. The ending point of one element is connected to the starting point of the next element.

The properties of the elements contained within a path graphic are retained. When you break a path graphic, the elements it contains appear as they did before you created the path graphic.

A path graphic has the same properties as a rectangle, ellipse, or polygon. These properties are lost when you break the path.

## Related Topics

[Elements](#)

## Properties

Properties determine the appearance and behavior of an element or the graphic. For example, the width property determines the width in pixels of the selected element.

There are two types of properties:

- Predefined properties
- Custom properties

## Related Topics

[The Industrial Graphic Editor](#)

[Predefined Properties](#)

[Custom Properties](#)

[Properties of Groups](#)

### Predefined Properties

Properties are specific to the selected element and can vary between elements of different types. All elements have the following property categories:

- Graphic - the name of the element (or group)
- Appearance - element dimension, location, rotation, transparency, and locked status

You can view specific properties for a specific kind of element or group by clicking a drawing tool and drawing an element.

You set properties at design time. Some properties can be read or written to at run time, such as X, Y, Width, Height, Visible, and so on. The element type determines which properties are available and can be read or written at run time.

## Related Topics

[Properties](#)

### Custom Properties

Use custom properties to extend the functionality of a graphic. A custom property can contain:

- A value that can be read and written to.
- An expression that can be read.
- An object equipment.item that can be read and written to if the equipment.item allows being written to.
- A property of an element or graphic.
- A custom property of a graphic.
- A reference to a tag.

For example, for a tank graphic called TankSym you can create a custom property called TankLevel that is calculated from an equipment.item reference to Tank\_001.PV. You can then reference the tank level by TankSym.TankLevel.

Custom properties appear in the Properties Editor when no elements are selected. You can edit default initial values of custom properties in the editor directly or use the **Edit Custom Properties** dialog box to do so.

For more information, see [Using Custom Properties](#).

## Related Topics

[Properties](#)

### Properties of Groups

Groups have their own properties you can view and set in the Properties Editor. For most properties, changing group properties indirectly affects the properties of its contained elements.

You can change the following group properties:

- Name (Name)
- Position (X, Y)
- Size (Width, Height)
- Orientation (Angle)
- Point of Origin (AbsoluteOrigin, RelativeOrigin)
- Transparency (Transparency)
- Locked (Locked)
- Enablement (Enabled)
- Tab Order (TabOrder)
- Tab Stop (TabStop)
- Single Object Treatment (TreatAsIcon)
- Visibility (Visible)

## Related Topics

[Properties](#)

[Changing/Renaming a Group Name](#)

[Changing the Position of a Group](#)

[Changing the Size of a Group](#)

[Changing the Orientation of a Group](#)

[Changing the Transparency of a Group](#)

[Locking the Group](#)

[Run-Time Properties of a Group](#)

[Renaming a Group or its Elements](#)

# Changing/Renaming a Group Name

If you change the group name, it has no affect on the contained elements. The contained elements keep their name.

If you rename an element or a group, the animation references to it are not automatically updated. You must manually change all animation links referencing the old name. For more information, see [Substituting References in Elements](#).

## Related Topics

[Properties of Groups](#)

# Changing the Position of a Group

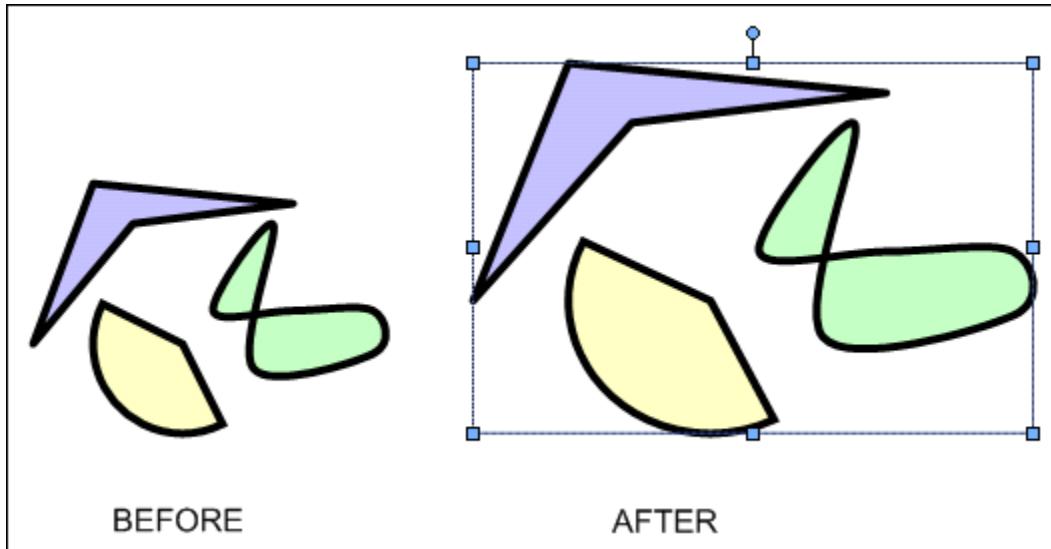
If you change the position of the group, all contained objects are moved with the group. They maintain the relative position to each other, but their absolute positions change.

## Related Topics

[Properties of Groups](#)

# Changing the Size of a Group

If you change the size of the group, all contained objects are resized proportionally.

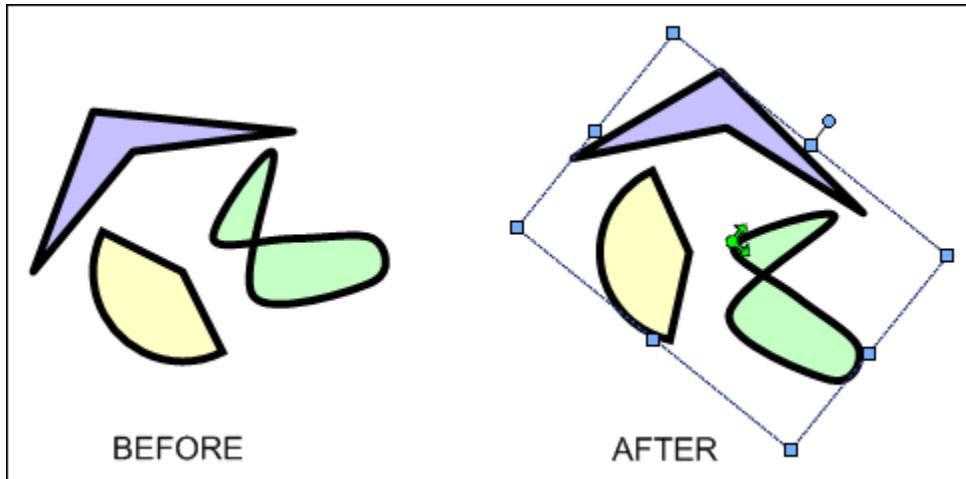


## Related Topics

[Properties of Groups](#)

# Changing the Orientation of a Group

If you change the angle of the group, all contained objects are rotated with the group around the origin of the group, so that the group remains visually intact.



## Related Topics

[Properties of Groups](#)

# Changing the Transparency of a Group

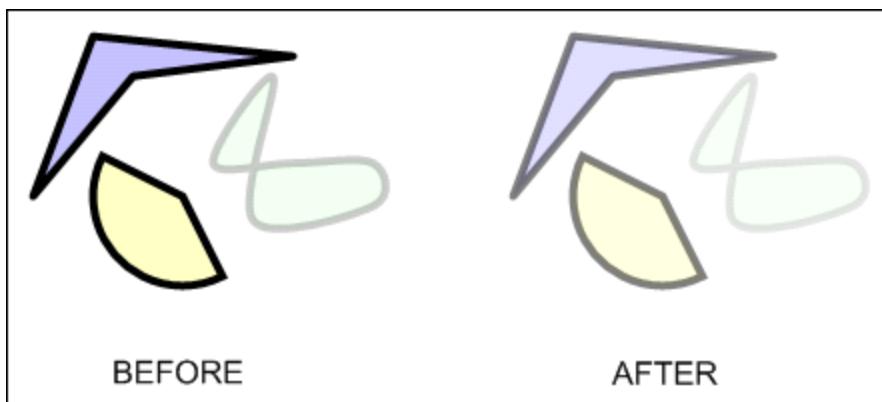
If you increase the transparency of the group, all contained objects appear more transparent, but their own transparency property values do not change. If you change their transparency values, it is in relation to the transparency level of the group.

For example, if you add an element with 80 percent transparency to a group, and then apply 50 percent transparency to the group, the element appears to have 90 percent transparency.

This is calculated as follows:

$$1 - (1 - 0.8) * (1 - 0.5) = 0.9$$

The transparency property values, however, stay unchanged at 80 percent for the element and 50 percent for the group.



## Related Topics

[Properties of Groups](#)

## Locking the Group

If you lock the group, it has no effect on the contained elements. You can still edit the contained elements in inline editing mode. You cannot move, resize, or rotate the group.

## Related Topics

[Properties of Groups](#)

## Run-Time Properties of a Group

If you change the run-time properties of a group, the elements do not inherit the property value of the group, but they do inherit the behavior of the group.

For example, if you create a group from elements, some of which have their visibility set to true and some to false, then set the group visibility to false, ALL elements in that group are invisible.

However the Visible property values of the contained elements still maintain their original values (true or false).

## Related Topics

[Properties of Groups](#)

## Renaming a Group or its Elements

If you rename an element or a group, the animation references to it are not automatically updated. You must manually change all animation links referencing the old name. For more information, see [Substituting References in Elements](#).

## Related Topics

[Properties of Groups](#)

## Animations

You can use animations to bind the run-time behavior and appearance of elements to equipment.items and tags, custom properties, and other element's properties.

For example, you can bind the vertical fill of a rectangle to a tag or equipment.item that contains the current level of a tank.

Animations are specific to the selected element and vary between elements of different types.

## Related Topics

[The Industrial Graphic Editor](#)

[Animation Types](#)

[Data Sources for Animations](#)

[Animation Capabilities of Groups](#)

[Animation States](#)

## Animation Types

There are two types of animations:

- Visualization animations determine the element's appearance, such as blinking, fill style, percent fill horizontal, value display, and so on.
- Interaction animations determine the element's behavior, such as horizontal sliders, user input, and so on.

There are visualization and interaction animations that are specific to certain elements. For example, the DataStatus animation is specific to the Status element. Element-specific animations also determine element behavior and appearance.

You can configure the following common animation types:

Animation Type	Description
Visibility	Shows or hides the element depending on a value or an expression.
Fill Style	Specifies the interior fill style depending on a discrete or analog expression or one or more conditions.
Line Style	Specifies the style and pattern of the element line depending on a discrete or analog expression or one or more conditions.
Text Style	Specifies the style of the element text depending on a

<b>Animation Type</b>	<b>Description</b>
	discrete or analog expression or one or more conditions.
<b>Blink</b>	Sets the element to blink invisibly or with specified colors depending on a discrete value or expression.
<b>Element Style</b>	Defines a set of visual properties that determine the appearance of text, lines, graphic outlines, and interior fill shown in Industrial graphics.
<b>% Fill Horizontal</b>	Fills the element with color partially from left to right or vice versa, depending on an analog value or expression.
<b>% Fill Vertical</b>	Fills the element with color partially from top to bottom or vice versa, depending on an analog value or expression.
<b>Location Horizontal</b>	Positions the element with a horizontal offset depending on an analog value or expression.
<b>Location Vertical</b>	Positions the element with a vertical offset depending on an analog value or expression.
<b>Width</b>	Increases or decreases the element width depending on an analog value or expression.
<b>Height</b>	Increases or decreases the element height depending on an analog value or expression.
<b>Point</b>	Changes the X and Y coordinate values of one or more selected points on a graphic or graphic element.
<b>Orientation</b>	Rotates the element by an angle around its center point or any other point depending on an analog value or expression.
<b>Value Display</b>	Shows a discrete, analog, string value, time value, name or expression.
<b>Tooltip</b>	Shows a value or expression as a tooltip when the mouse is moved over the element.
<b>Disable</b>	Disables the element's animation depending on a Boolean value or expression.
<b>User Input</b>	Enables the run-time user to type a Boolean, analog, string, time or elapsed time value that is then assigned to an equipment.item.

Animation Type	Description
<b>Slider Horizontal</b>	Enables the run-time user to drag the element left or right and write back the offset to an analog equipment.item.
<b>Slider Vertical</b>	Enables the run-time user to drag the element up or down and write back the offset to an analog equipment.item.
<b>Pushbutton</b>	Writes predetermined values to Boolean or analog references when the user clicks on the element.
<b>Action Scripts</b>	Runs an action script when the run-time user clicks on the element.
<b>Show Symbol</b>	Shows a specified graphic at a specified position when the run-time user clicks on the element.
<b>Hide Symbol</b>	Hides a specified graphic when the run-time user clicks on the element.

## Related Topics

[Animations](#)

### Data Sources for Animations

The data used for animations can come from various sources. You can configure the animation to point at these sources. Animation data can come from:

- Tag values.
- Tag extensions.
- Equipment and equipment.items.
- Predefined properties of an element or graphic.
- Custom properties of a graphic.

---

**Note:** Arrays are not supported.

---

When entering an expression or reference for an animation, text auto-complete is supported.

For the top level entries, suggestions will include up to 10000 items from the full list of clusters, top-level equipment, variable tags and alarm tags (presented as a combined list). Each subsequent level is defined by a entering a period (.). The suggested list is filtered based on what you type.

## Related Topics

[Animations](#)

## Animation Capabilities of Groups

By default, a group of elements has limited animation capabilities of its own. For a group you can configure the following animations:

- Blinking
- Enabling/disabling
- Vertical and horizontal location
- Orientation
- Height and width
- Visibility

However, you can set the TreatAsIcon property value to True. The group is then treated as a single object and you can configure more animations. These animations take precedence over animations defined for the elements within the group.

## Related Topics

[Animations](#)

## Animation States

Some animations have multiple configuration panels.

A state selection panel appears, where you can select the animation state. Depending on what you select, the configuration panel is populated differently. The animation state can be a:

- Data type, where the animation is tied to a specific data type.
- Truth table, where the animation is tied to a set of Boolean conditions.

## Related Topics

[Animations](#)

[Data Type Animation State](#)

[Truth Table Animation State](#)

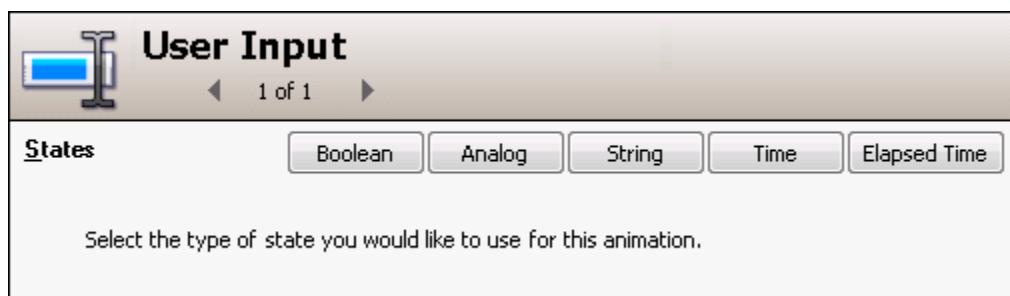
# Data Type Animation State

Certain animations support configuration of one or more data types. In the configuration panel of an animation, select the data type you want to configure, such as:

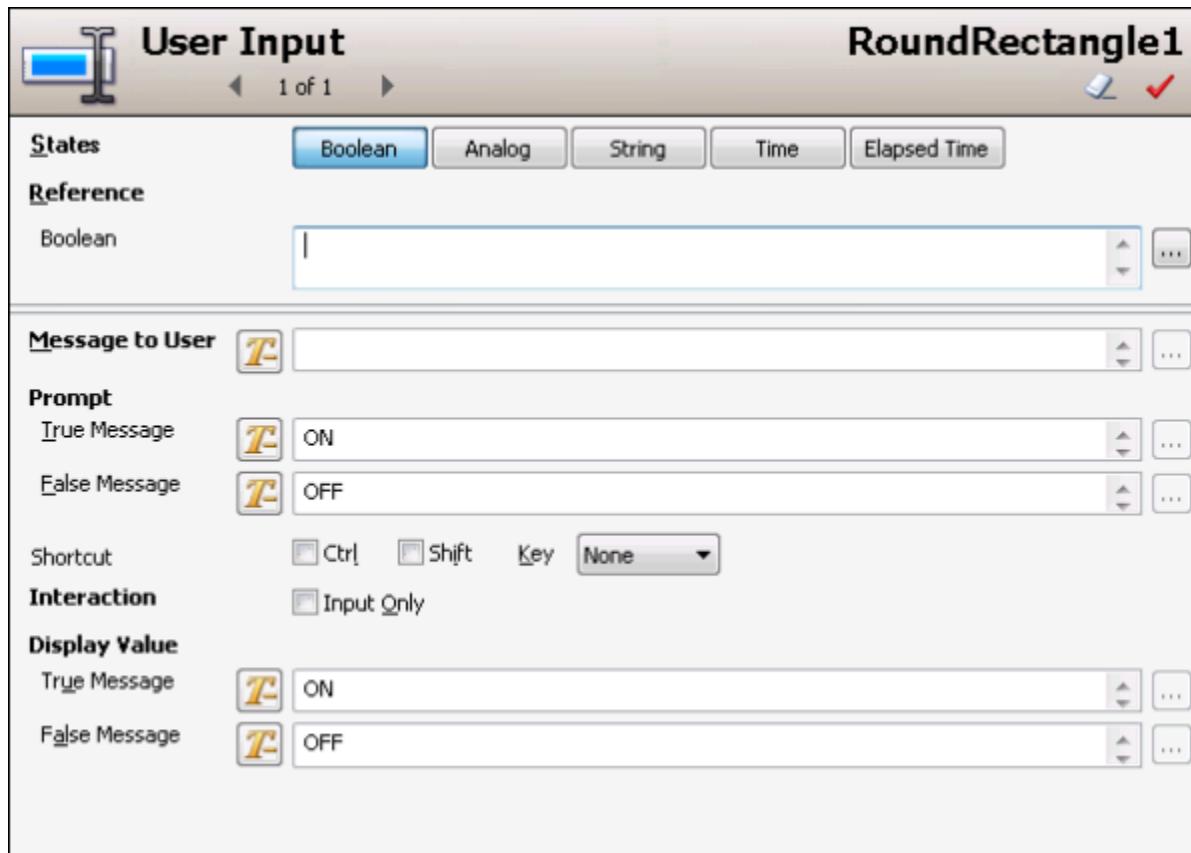
- Boolean
- Analog
- String

- Time
- Elapsed Time
- Name

For example, if you select the **User Input** animation link, the **User Input** state selection page appears on the right in the **Edit Animations** dialog box.



A configuration panel appears below the **States** buttons. For example, a configuration panel that is specific to the user input of a Boolean value.



## Related Topics

[Animation States](#)

# Truth Table Animation State

Certain animations support the configuration of a truth table. The truth table is a collection of up to 100 Boolean conditions you can configure to determine the output.

You can configure the default appearance for the case that none of the conditions are fulfilled.

The conditions are evaluated from top to bottom of the list. When the first true condition is met, its assigned appearance is the one used and the condition evaluation stops.

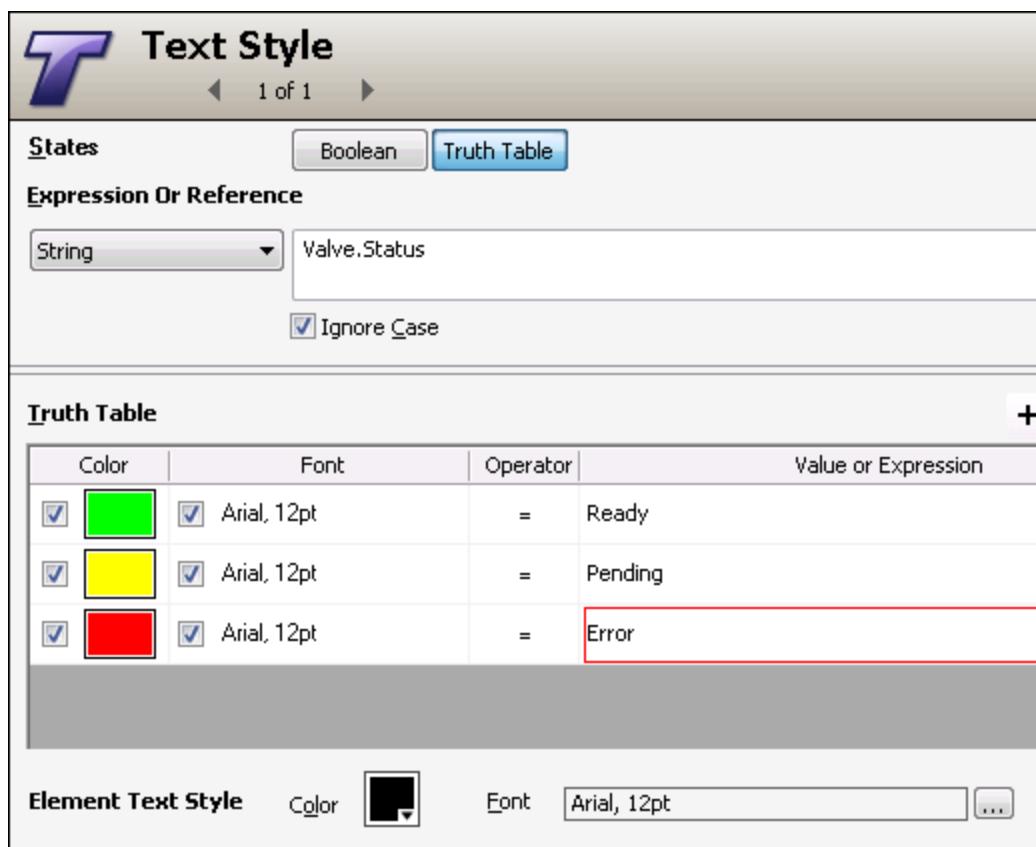
For example, you want a text animation to use a different text color depending on the value of a string equipment.item, such as a status indicator.

Status indicator	Text color
Ready	Green
Pending	Yellow
Error	Red

If you select the **Text Style** animation link, the **Text Style** state selection page appears on the **Edit Animations** dialog box.



You can click the **Truth Table** button to configure conditions for the appearance of the text style.



By default the text color is black if none of the conditions are fulfilled at run time.

## Related Topics

[Animation States](#)

## Embedding Graphics

You can embed graphics into other graphics. Embedding graphics enables you to rapidly develop more complex graphics with common components.

For example, you can create a single tank graphic, then embed it multiple times in another graphic to create a graphic representing a collection of tanks.

There is no limit to the number of levels of embedding.

Embedded graphics appear in the Elements List. The default name is the same as the source graphic, followed by a numeric sequence.

## Related Topics

[Appearance of Embedded Graphics](#)

[Size Propagation and Anchor Points](#)

[Embedding Graphics within Graphics](#)

## Appearance of Embedded Graphics

Embedded graphics appear in the Elements List. The default name is the same as the source graphic, followed by a numeric sequence.

## Related Topics

[Embedding Graphics](#)

## Size Propagation and Anchor Points

An anchor point controls how changes in graphic size are propagated to embedded instances. By default, the anchor point of the graphic is the center point of all elements on the canvas.

This can be done graphically on the canvas, or by setting anchor position properties in the Properties Editor.

There are two types of anchors:

- Use the `AbsoluteAnchor` property to specify its position as absolute coordinates.
- Use the `RelativeAnchor` property to specify its position as coordinates relative to the graphic center.

When you embed a graphic, the embedded graphic inherits the anchor point in relation to its own center point.

You can also set the `AnchorFixedTo` property. When you make changes to the graphic that affects its size, the `AnchorFixedTo` property determines if the absolute position or relative position of the anchor point is recalculated. This property can have following values:

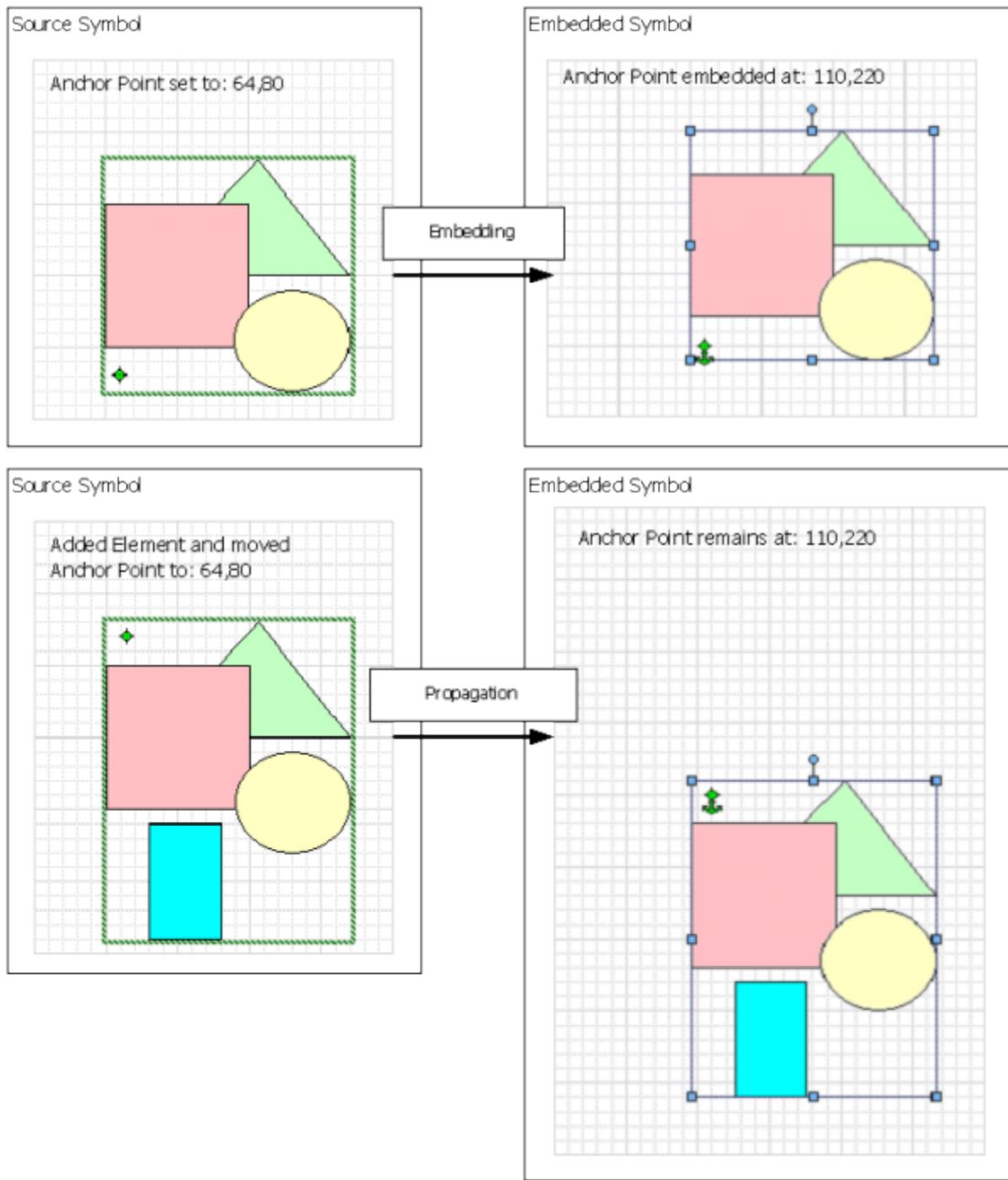
- **Absolute:** The absolute anchor point position is unchanged, and the relative anchor point position is recalculated.
- **Relative:** The absolute anchor point position is recalculated, and the relative anchor point position is unchanged.

---

**Note:** When you change the `AbsoluteAnchor` property, the `AnchorFixedTo` property is set to the value `Absolute`. When you change the `RelativeAnchor` property, the `AnchorFixedTo` property is set to the value `Relative`.

---

You can change the position of the anchor point of the graphic. This affects the position of the embedded instances. The anchor points of the embedded instances, however, remain unchanged.



**Note:** You can change the anchor point of an embedded graphic. This moves the embedded graphic. It does not change the anchor point position in relation to the graphic. You can resize or rotate the embedded graphic. The anchor point moves in relation to the embedded graphic. You can also use the AnchorPoint property in the Properties Editor to change the position.

## Related Topics

[Embedding Graphics](#)

[Embedding Graphics within Graphics](#)

## Creating Multiple Configurations of a Graphic

The Symbol Wizard Editor is a feature of the Industrial Graphic Editor to create multiple configurations of a graphic. A graphic configuration represents different visual or functional variations of a graphic.

Graphic configurations are created using layers containing associated graphic elements, custom properties, and named scripts. Based on graphic properties and possible values of these properties, rules are applied that specify when a layer is part of a graphic configuration.

### Related Topics

[Understanding Visual and Functional Graphic Configurations](#)

### Understanding Visual and Functional Graphic Configurations

Standard Industrial Graphics show reasonably realistic views of process objects. These graphics can be modified with the Symbol Wizard to incorporate multiple visual configurations in a graphic.

You can also use Symbol Wizards to create functional properties.

### Related Topics

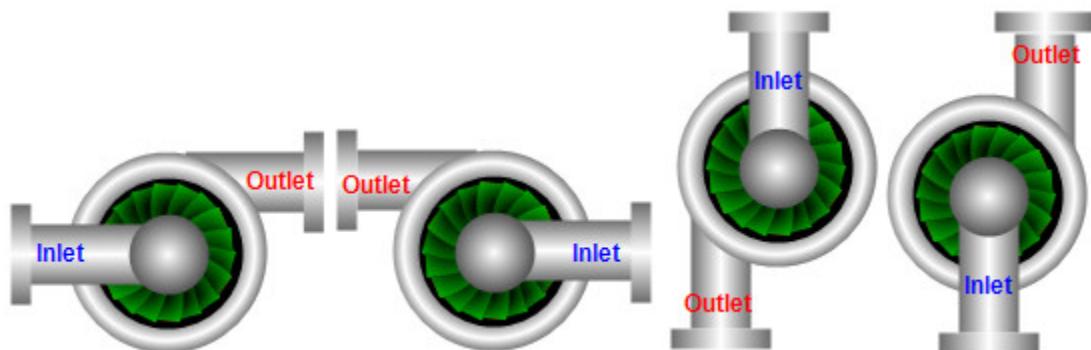
[Creating Multiple Configurations of a Graphic](#)

[Visual Graphic Configurations](#)

[Functional Graphic Configurations](#)

### Visual Graphic Configurations

Using an example of a centrifugal pump with separate inlet and outlet pipes, there are four practical visual configurations. The pump's blade housing is common and appears in all possible configurations. But, the pump's inlet and outlet pipes can be placed at the left or right in a horizontal direction or at the top or bottom when the pump is oriented vertically.



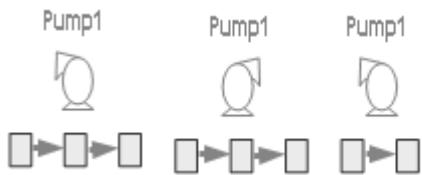
Orientation is the visual property that identifies the different configurations of a pump graphic. The equipment.items associated with the Orientation property are left, right, top, and bottom.

## Related Topics

[Understanding Visual and Functional Graphic Configurations](#)

### Functional Graphic Configurations

You can also use Symbol Wizards to create functional properties. For example, a multi-stage pump symbol could be built with a Wizard Option to select either a five-stage, three-stage, or single stage pump in addition to a visual Orientation property to select left or right pump configurations.



## Related Topics

[Understanding Visual and Functional Graphic Configurations](#)

## Using the Industrial Graphic Editor

### In This Chapter

[Showing, Hiding and Adjusting Panels](#)

[Panning and Zooming the Canvas](#)

[Configuring Designer Preferences](#)

[Using the Symbol Wizard Editor](#)

### Showing, Hiding and Adjusting Panels

You can edit graphics using the Industrial Graphic Editor. Depending on where the graphic is contained, you can start the Industrial Graphic Editor from your HMI/SCADA software:

You can:

- Show and hide Industrial Graphic Editor panels to allocate more space on the canvas.
- Pan and zoom the canvas to make finer or more granular adjustments to elements.
- Place a grid on the canvas surface to align elements more precisely.

You can hide the Properties Editor and Animation Summary to allocate more space on the canvas.

### To show or hide the Properties Editor and Animation Summary panels

- Do either of the following:
  - Press ALT + ENTER.
  - On the **View** menu, click **Properties**.

You can also adjust the size of the Elements List and Properties Editor.

### To adjust the size of panels

1. Drag the dividing line between the panels to specify the new panel size.
2. Release the mouse button and the panels are resized.

## Related Topics

[Using the Industrial Graphic Editor](#)

## Panning and Zooming the Canvas

You can pan and zoom the canvas to make finer visual adjustments to the elements or to get a better overview of a large graphic.

Use the Pan and Zoom toolbar to pan and zoom.



## Related Topics

[Using the Industrial Graphic Editor](#)

[Panning](#)

[Zooming](#)

## Panning

You can use the Pan functions of the Pan and Zoom toolbar to do the following:

- Use the **Pan and Zoom** window to select which part of the canvas appears on the screen.
- Grab the canvas with the Hand tool and move it (Pan).

You can also use the scroll wheel of the mouse to pan up and down in the current canvas display.

## Related Topics

[Panning and Zooming the Canvas](#)

[Using the Pan and Zoom Window to Pan](#)

[Using the Hand Tool to Pan](#)

[Using the Mouse Scroll Wheel to Pan](#)

# Using the Pan and Zoom Window to Pan

Use the **Pan and Zoom** window to pan the canvas area.

### To use the Pan and Zoom window for panning

1. On the Pan and Zoom toolbar, click the **Pan and Zoom** window icon. The **Pan and Zoom** window appears.
2. In the **Pan and Zoom** window, move the mouse within the red rectangle. The pointer hand icon appears.
3. Click and hold the left mouse button down.
4. Drag the mouse. The red rectangle moves with the mouse.
5. Release the mouse button. The area shown in the canvas is changed accordingly.

### Related Topics

[Panning](#)

## Using the Hand Tool to Pan

Use the Hand tool to pan the canvas area. This is equivalent to picking up the canvas and moving it so that the visible canvas area changes.

### To use the Hand tool to pan



1. On the **Pan and Zoom** toolbar, click the **Pan** icon.
2. Move the mouse over the canvas. The Hand tool pointer appears.
3. Click the canvas to grab the canvas and keep the mouse button down.
4. Move the mouse to change the area of canvas that is shown.
5. Release the mouse button.

### Related Topics

[Panning](#)

## Using the Mouse Scroll Wheel to Pan

You can use the mouse scroll wheel to:

- Pan up or down.
- Pan 360 degrees.

### To use the mouse scroll wheel to pan up or down

1. Click the canvas so that no elements are selected.
2. Move the mouse scroll wheel:
  - Forward to pan up.

- Backward to pan down.

### To use the mouse scroll wheel to pan in any direction

1. Click the canvas so that no elements are selected.
2. Click the mouse scroll wheel. The pointer appears in 360 degrees scroll mode.
3. Move the mouse. The visible area of the canvas is panned accordingly.
4. When you are done, click the canvas.

## Related Topics

[Panning](#)

### Zooming

Use the Pan and Zoom toolbar to:

- Zoom in on a specified point to magnify the current elements.
- Zoom out from a specified point.
- Zoom to the default zoom factor (100 percent).
- Zoom so that the currently selected element is shown across the available canvas area or zoomed to the maximum value of 500 percent.
- Zoom in on an area of the canvas using a "rubber band" selection with your mouse.
- Specify or select a zoom factor.

You can also use the CTRL key and the scroll wheel of the mouse to zoom in and zoom out the current canvas view.

## Related Topics

[Panning and Zooming the Canvas](#)

[Zooming In to a Specified Point](#)

[Zooming Out from a Specified Point](#)

[Zooming to the Default Zoom Value](#)

[Zooming a Selected Element](#)

[Zooming a Specified Area](#)

[Selecting or Specifying a Zoom Value](#)

[Using the Pan and Zoom Window to Pan](#)

[Using the Mouse Scroll Wheel for Zooming](#)

# Zooming In to a Specified Point

You can zoom in by 25 percent of the default scale to any specified point on the canvas.

## To zoom in to a specified point



1. Click the Zoom In icon in the toolbar.
2. Move the mouse over the canvas. The Zoom In pointer appears.
3. Click the canvas to where you want to zoom in. The canvas is zoomed in at the specified point.

## Related Topics

[Zooming](#)

# Zooming Out from a Specified Point

You can zoom out by 25 percent of the default scale from any specified point on the canvas.

## To zoom out to a specified point

1. Click the Zoom Out icon in the toolbar.
2. Move the mouse over the canvas. The Zoom Out pointer appears.
3. Click the canvas from where you want to zoom out. The canvas is zoomed out from the specified point.

## Related Topics

[Zooming](#)

# Zooming to the Default Zoom Value

You can reset the zoom to the default zoom value 100 percent.

## To reset the zoom to the default zoom value



- Click the Zoom to Normal icon in the toolbar. The canvas zoom is reset to its default.

## Related Topics

[Zooming](#)

# Zooming a Selected Element

You can zoom one or more selected elements so that they appear as large as possible in the allocated canvas area. This is useful when you want to make fine adjustments to one or more elements.

## To zoom a selected element

1. Select the elements you want to zoom.
2. Click the Zoom To Selection icon in the toolbar. The visible canvas is zoomed so that the selected elements appear as large as possible.

## Related Topics

[Zooming](#)

# Zooming a Specified Area

You can zoom a specified area by using the "rubber band" selection method.

## To zoom a specified area

1. Click the Rubber Band Zoom icon.
2. Move the mouse over the canvas. The Rubber Band pointer appears.
3. Move the mouse to the top left corner of the area you want to zoom.
4. Hold the left mouse button down and then drag the mouse to the bottom right corner of the area you want to zoom.
5. Release the mouse button. The area is zoomed to the entire canvas area.

## Related Topics

[Zooming](#)

# Selecting or Specifying a Zoom Value

You can select a defined zoom value or type a zoom value. Valid values are 25 percent to 500 percent.

## To select or specify a zoom value

- On the Zoom and Pan toolbar, do one of the following:
  - Click the zoom value list and select a zoom value.
  - Click the zoom value in the zoom value list, type a valid value, and then click **Enter**.

## Related Topics

[Zooming](#)

# Using the Pan and Zoom Window to Change the Zoom

Use the **Pan and Zoom** window to change the zoom of the canvas.

---

**Note:** You can also use the **Pan and Zoom** window to "scroll" to a different part of the canvas. This is called panning. For more information, see [Panning](#).

---

### To use the Pan and Zoom window for zooming

1. On the Zoom and Pan toolbar, click the **Pan and Zoom** window icon. The **Pan and Zoom** window appears.
2. In the **Pan and Zoom** window, move the mouse over a corner or an edge of the red rectangle.
3. Click and hold the left mouse button down. The corresponding resize pointer appears.
4. Drag the mouse. The red rectangle changes size proportionally.
5. Release the mouse button. The zoom of the canvas is changed accordingly.

## Related Topics

[Zooming](#)

# Using the Mouse Scroll Wheel for Zooming

You can use the mouse scroll wheel to zoom the canvas area. The canvas is then zoomed on the midpoint of all selected elements or, if none are selected, on the midpoint of the canvas.

### To use the mouse scroll wheel for zooming

- Press and hold the CTRL key and move the scroll wheel:
  - Forward to zoom in by a factor of 25 percent of the default zoom value.
  - Backward to zoom out by a factor of 25 percent of the default zoom value.

## Related Topics

[Zooming](#)

## Configuring Designer Preferences

Use the **Designer Preferences** dialog box to set Industrial Graphic Editor preferences. Preferences can be configured for the following:

- Grid Settings - The grid helps you precisely place and move elements on the canvas.
- Canvas Settings - The settings for the appearance of the graphic on the canvas can also be configured.
- Graphics Performance Index Warning window visibility
- Image Editor selection

### To open the Designer Preferences dialog window

1. Open the Industrial Graphic Editor.
2. On the **View** menu, click **Preferences**. The **Designer Preferences** dialog box appears.

### To configure Grid Settings

1. Click the box next to the **Grid color** label. The **Select Grid Color** dialog box appears. For more information, see [Setting a Solid Color](#).
2. In the **Grid size** box, type a value from 1 to 100 to specify the distance in pixels between each line in the grid.
3. In the **Major subdivisions** box, type a value from 1 to 10 to specify the number of major subdivisions of the grid. Major subdivisions are emphasized lines that visually create larger grid cell blocks.
4. Clear or select the **Grid visible** check box to hide or show the grid.
5. Clear or select the **Snap to grid** check box. With the snap-to-grid option set, when you move elements or groups on the canvas they are moved to the closest grid intersection. If this option is not set, you can move the elements freely to any location on the canvas.

### To configure Canvas Settings and graphic appearance

1. Click the box next to the **Background Color** label. The **Select Canvas Color** dialog box appears. For more information, see [Setting a Solid Color](#).
2. Clear or select the **Symbol Smoothing** check box. If this option is not set, lines drawn on the canvas may show jagged edges. With this option set, lines drawn on the canvas show smooth edges.
3. Clear or select the **Show Anchor** check box. With this option selected, the graphic displays anchor icons, if anchors were created in the graphic.

### To select an Image Editor

- Choose the graphic editing tool from the **Image Editor** menu. If you select **Choose Custom Editor**, the **Select Image Editing Application** window appears so you can make a selection.

### To save your settings as default settings

1. Click **Save as Default**.
2. Click **Apply**.
3. Click **OK**.

## Related Topics

[Using the Industrial Graphic Editor](#)

## Using the Symbol Wizard Editor

The Symbol Wizard Editor in the Industrial Graphic Editor allows you to create graphics with multiple configurations called Symbol Wizards. They create Symbol Wizards with the Symbol Wizard Editor. You can embed Symbol Wizards and use the Symbol Wizard Editor to select the configuration needed for an application.

Start creating a multi-configuration graphic by opening a graphic element or graphic in the Industrial Graphic Editor. Show the Symbol Wizard Editor by clicking the Symbol Wizard icon from the Industrial Graphic Editor's menu bar, selecting it as an option of the **View** menu, or pressing the Alt+W key combination.

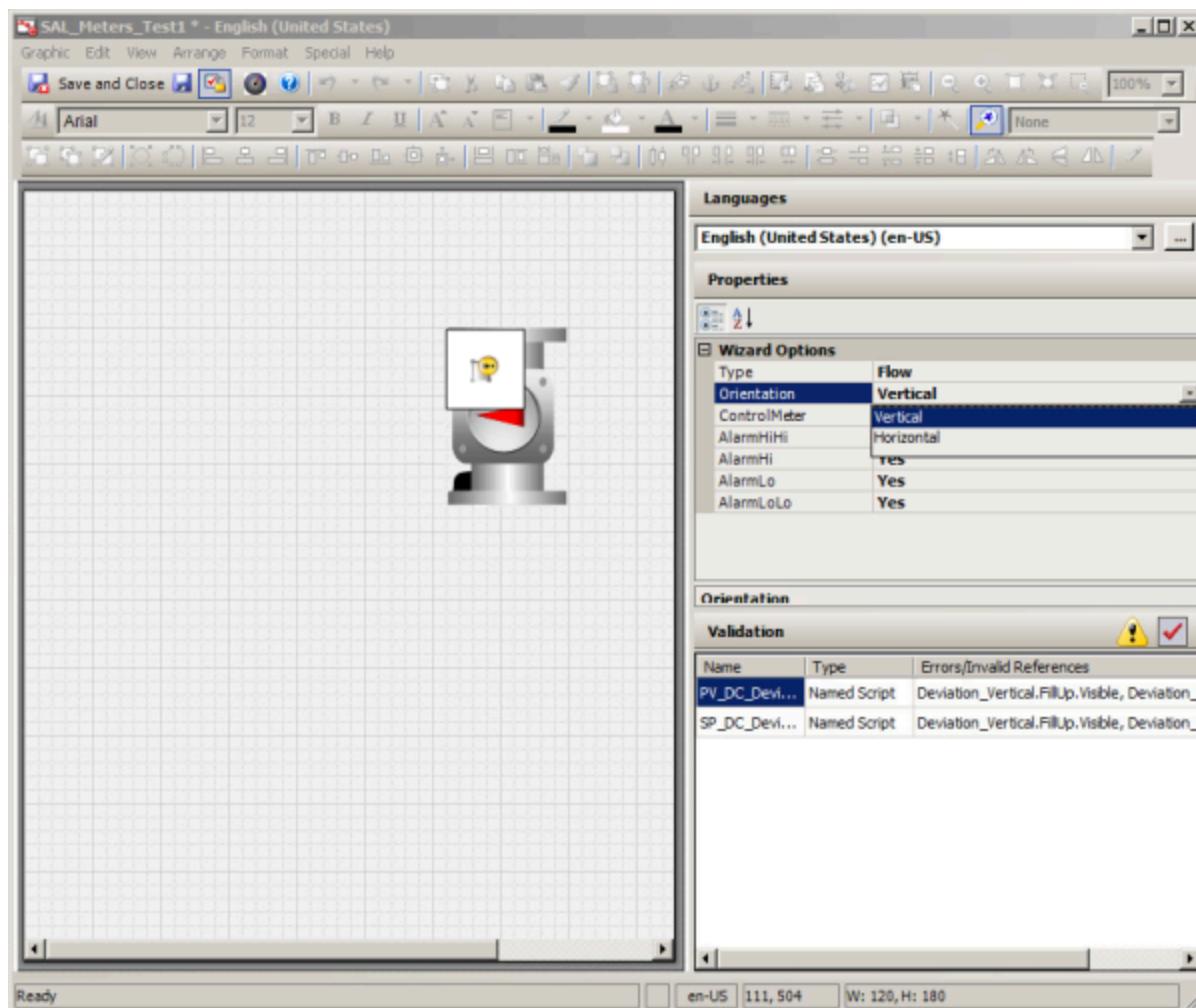
The Industrial Graphic Editor window updates to show tabbed Symbol Wizard panes at the left of the window. The top pane shows the graphic elements, named scripts, and custom properties of the graphic in separate views.

The bottom pane shows tabbed **Options** and **Layers** panes. The **Options** pane shows a hierarchical list of Choice Groups, Choices, and Options that define graphic properties and the possible values associated with each property. The **Layers** pane includes a list of defined graphic layers. Beneath each layer, separate folders contain the graphic's elements, custom properties, and named scripts associated with each layer. You can add, edit, or delete items associated with the **Options** and **Layers** panes.

Symbol Wizard **Option Properties** or **Layer Properties** panes appear to the right of the canvas area in the Industrial Graphic Editor window after selecting items from the **Options** or **Layers** panes.

Both properties pane shows the name of the selected item and any rule associated with the item. If a Choice Group is selected from the **Options** pane, the **Options Properties** pane also shows the default value of the Choice Group and a **Description** field.

After creating the configurations of a graphic with the Symbol Wizard Editor, use the Symbol Wizard Preview to verify that all configurations are correct. The Symbol Wizard Preview can be opened by clicking it from the menu bar, selecting it as an option of the **View** menu, or pressing the Alt+P key combination.

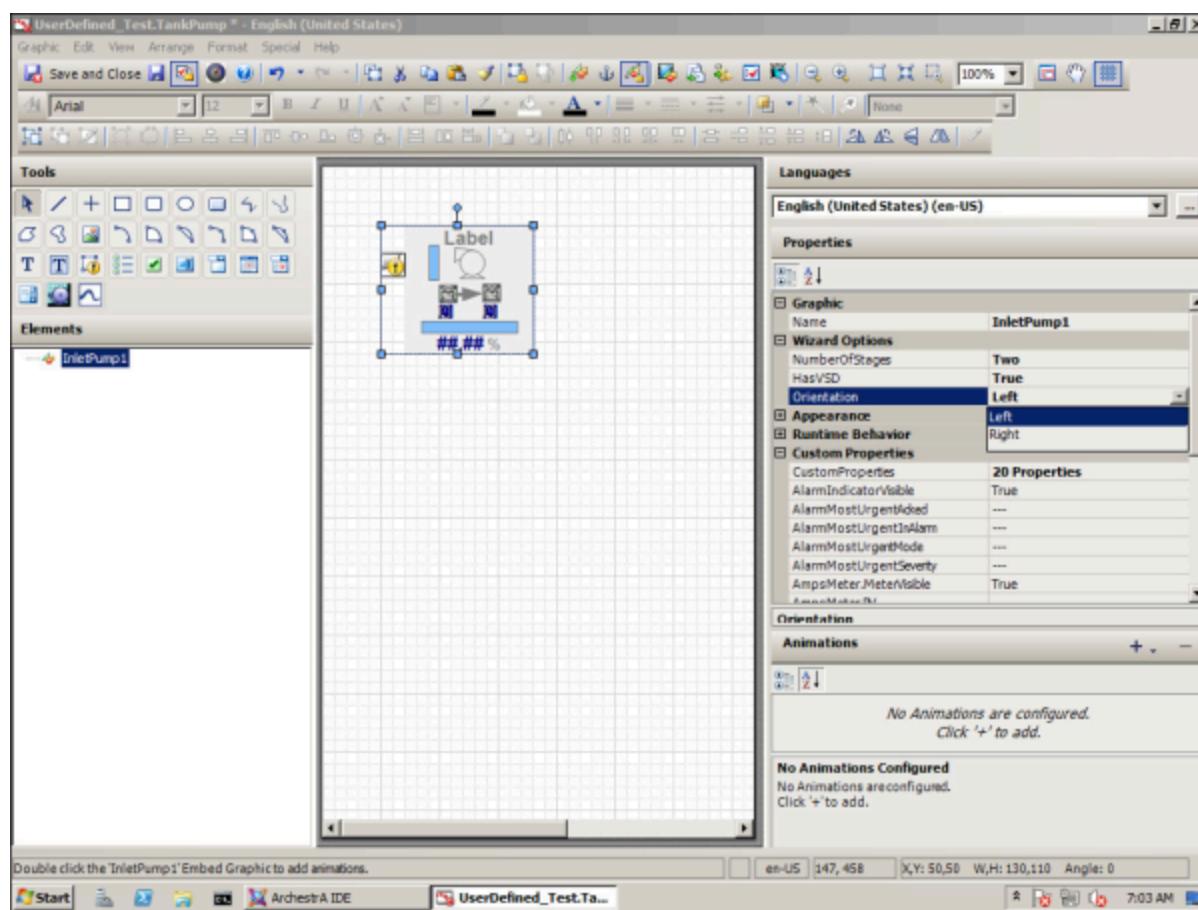


After opening Symbol Wizard Preview, the **Properties** pane shows **Wizard Options**, which includes drop-down menus to select options to show the different configurations created for the graphic. As options are selected, the graphic is updated to show the selected configuration.

The **Validation** pane shows any script or custom property errors within the graphic. Selecting a listed error from the **Validation** pane shows the **Custom Properties** or **Scripts** dialog box to identify and correct an error within the graphic.

After verifying that all graphic configurations are correct, save the graphic into the Graphic Toolbox. For more information about the Symbol Wizard tasks completed by a user, see [Designing a Symbol Wizard](#).

To create an application containing Symbol Wizards, add a graphic to an equipment reference or create a new graphic from the Graphic Toolbox. Then, embed a Symbol Wizard. The graphic appears with the default configuration selected.



The **Wizard Options** pane shows a set of drop-down lists with configuration options. Select options from the drop-down lists to change the graphic's configuration to meet the needs of an application. Finally, edit and update the custom properties and named scripts that are associated with the multi-configuration graphic. For more information about the Symbol Wizard tasks, see [Using Symbol Wizards in an Application](#).

## Related Topics

[Using the Industrial Graphic Editor](#)

## Working with Graphic Elements

### In This Chapter

[About Graphic Elements](#)

[Drawing and Dragging Elements](#)

[Import an SVG as an Industrial Graphic](#)

[Editing Element Properties](#)

[Selecting Elements](#)

[Inline Editing](#)

[Copying, Cutting, and Pasting Elements](#)

- [Moving Elements](#)
- [Aligning Elements](#)
- [Adjusting the Spacing between Elements](#)
- [Resizing Elements](#)
- [Adjusting the z-Order of Elements](#)
- [Rotating Elements](#)
- [Moving the Origin of an Element](#)
- [Add Connectors Between Graphic Elements](#)
- [Flipping Elements](#)
- [Locking and Unlocking Elements](#)
- [Making Changes Using Undo and Redo](#)
- [Working with Groups of Elements](#)
- [Using Path Graphics](#)

## About Graphic Elements

This section explains how to work with the common features of graphic elements. For information about features specific to certain elements such as element properties, see [Editing Graphic-Specific and Element-Specific Properties](#).

Graphic elements are basic shapes and controls you can use to create a graphic to your specifications. You can:

- Draw an element by selecting an element from the Tools panel, placing it on the canvas, and then configuring its properties.

Select one or more elements on the canvas with the mouse or from the **Element** list.

- Edit certain elements in a special way called inline editing.
- Copy, cut, paste, and duplicate elements.
- Move elements around on the canvas.
- Align elements to each other.
- Change the spacing between elements.
- Resize elements.
- Change the z-order of elements to change which elements appear on top of others when they overlap.
- Rotate elements.
- Change the origin of elements to specify around which point the elements are rotated.
- Flip elements on their horizontal or vertical axis.
- Lock elements to stop them being moved or changed.
- Undo and redo any number of changes made previously to the graphic.
- Create groups of elements to bind them together.
- Create a path graphic from multiple open line elements.

## Related Topics

[Working with Graphic Elements](#)

### Drawing and Dragging Elements

You can create elements such as lines, curves, circles, squares, and so on. You can combine these elements to create complex drawings of all the equipment in your manufacturing environment.

After you draw an element, you can modify its properties. For more information about modifying properties, see [Editing Element Properties](#).

Regardless of the kind of element you are drawing, drawing each kind of element is very similar.

After you draw an element, the pointer tool is selected again by default. To draw multiple elements of the same type, double-click the element in the Tools panel. It remains selected after you draw your first element of that type. You can press the ESC key to return to the pointer tool again.

If you draw or drag an element outside of the visible canvas area to the right or bottom, horizontal and/or vertical scroll bars appear but the visible area does not follow the mouse. You can later use the scroll bars to scroll the canvas and see the element you drew or moved.

## Related Topics

[Working with Graphic Elements](#)

[Drawing Rectangles, Rounded Rectangles, Ellipses, and Lines](#)

[Drawing Polylines, Polygons, Curves, and Closed Curves](#)

[Drawing 2-Point Arcs, 2-Point Pies and 2-Point Chords](#)

[Drawing 3-Point Arcs, 3-Point Pies, and 3-Point Chords](#)

[Placing and Importing Images](#)

[Drawing Buttons](#)

[Placing Text](#)

[Drawing Text Boxes](#)

[Drawing Status Elements](#)

[Drawing User Interface Common Controls](#)

[Dragging Elements](#)

### Drawing Rectangles, Rounded Rectangles, Ellipses, and Lines

You can draw rectangles, rounded rectangles, ellipses, and lines on the canvas.

#### To draw a rectangle, rounded rectangle, ellipse, or line

1. Click the appropriate icon in the **Tools** panel.
2. Click the canvas and drag the shape of the element on the canvas.
3. When you are done, release the mouse button.

## Related Topics

[Drawing and Dragging Elements](#)

### Drawing Polylines, Polygons, Curves, and Closed Curves

You can draw polylines, polygons, curves, and closed curves on the canvas.

If you are drawing a closed element, the element automatically closes when you are done drawing.

#### To draw a polyline, polygon, curve, or closed curve

1. Click the appropriate icon in the **Tools** panel.
2. Click the canvas where you want to start the element.
3. Click the next point for the element.
4. Continue clicking until you have all the points you require.
5. When you are done, right-click.
6. You can change the shape of these elements anytime by editing their control points. For more information, see [Editing Control Points](#).

## Related Topics

[Drawing and Dragging Elements](#)

### Drawing 2-Point Arcs, 2-Point Pies and 2-Point Chords

You can draw 2-point arcs, 2-point pies, and 2-point chords on the canvas.

If you are drawing a closed element, the element automatically closes when you are done drawing.

#### To draw a 2-point arc, 2-point pie, or 2-point chord

1. Click the appropriate icon in the Tools panel.
2. Click the canvas where you want to start the element and hold the mouse button.
3. Drag the mouse to where you want the element to end.
4. When you are done, release the mouse button.
5. You can change the shape of these elements anytime by editing their control points. For more information, see [Editing Control Points](#).

## Related Topics

[Drawing and Dragging Elements](#)

### Drawing 3-Point Arcs, 3-Point Pies, and 3-Point Chords

You can draw 3-point arcs, 3-point pies and 3-point chords on the canvas.

If you are drawing a closed element, the element automatically closes when you are done drawing.

### To draw a 3-point arc, 3-point pie, or 3-point chord

1. Click the appropriate icon in the Tools panel.
2. Click the canvas where you want to start the element.
3. Click the canvas in two other places to define the element.
4. You can change the shape of these elements anytime by editing their control points. For more information, see [Editing Control Points](#).

## Related Topics

[Drawing and Dragging Elements](#)

## Placing and Importing Images

You can place an image element on the canvas and import an image into it.

### To draw an image

1. Click the image icon in the **Tools** panel.
2. Click the canvas and drag the shape of the image element.
3. Release the mouse button. The **Open** dialog box appears.
4. Browse to the image file, select it, and then click **Open**. The image file is loaded into the image element.

## Related Topics

[Drawing and Dragging Elements](#)

## Drawing Buttons

You can draw a button on the canvas. You can configure a button with a text label or an image.

For more information on how to configure a button with an image after drawing it on the canvas, see .

### To draw a button

1. Click the button icon in the Tools panel.
2. Click the canvas and drag the shape of the button element.
3. Release the mouse button. The button text appears in edit mode.
4. Type a text label for the button and click **Enter**.

## Related Topics

[Drawing and Dragging Elements](#)

## Placing Text

You can place text on the canvas.

The text element has no border and no background fill. The text does not wrap. When you type the text, the size of the Text element expands.

You can also drag the handles of the Text element to resize it.

### To place text

1. Click the text icon in the **Tools** panel.
2. Click the canvas where you want to place the text.
3. Type the single line of text you want.
4. When you are done, do one of the following:
  - Click **Enter** to type a new line of text. This new line is a new element.
  - Click the canvas outside the text element.

## Related Topics

[Drawing and Dragging Elements](#)

## Drawing Text Boxes

You can draw text boxes on the canvas. Text boxes can have borders and background fill.

You can also configure the text to wrap in the text box. For more information, see [Wrapping Text in Buttons](#).

### To draw a text box

1. Click the text box icon in the **Tools** panel.
2. Click the canvas where you want to place the text box.
3. Drag a rectangle on the canvas.
4. Release the mouse button. The text appears in edit mode.
5. Type a text label for the text box, and then click **Enter**.

## Related Topics

[Drawing and Dragging Elements](#)

## Drawing Status Elements

Use the status element to indicate specific quality and status conditions of equipment.items.

### To draw status elements

1. Click the status icon in the **Tools** panel.
2. Click the canvas where you want to place the status element.
3. Drag a rectangle on the canvas.

4. Release the mouse button.

## Related Topics

[Drawing and Dragging Elements](#)

### Drawing User Interface Common Controls

Draw user interface common controls on the canvas to add additional functionality to your graphic. Each of the controls has specific behavior when it is drawn. For example, you can change the width of a combo box, but not the height.

#### To draw a windows control

1. Click the appropriate common interface control icon in the **Tools** panel.
2. Click the canvas where you want to place the control.
3. Drag a rectangle on the canvas.
4. Release the mouse button.

## Related Topics

[Drawing and Dragging Elements](#)

### Dragging Elements

After you draw elements on the canvas, you can drag them to a new position.

#### To drag elements on the canvas

1. Select one or more elements.
2. Click one of them and hold the mouse button down.
3. Drag the mouse to the new position.
4. Release the mouse button.

## Related Topics

[Drawing and Dragging Elements](#)

### Import an SVG as an Industrial Graphic

The Industrial Graphic Editor supports the importing of Scalable Vector Graphics (SVG) as Industrial Graphics. You can perform the following:

- Import an SVG into the Industrial Graphic Editor. The graphic elements automatically get converted to an Industrial Graphic, which can include many primitives.  
Insert the SVG while using the **Image** icon from the **Tools** panel of the Industrial Graphic Editor.

- Use SVG for the **UpImage** and **DownImage** on a button. For more information, see [Configuring Buttons with Images](#).
- Use SVG for the **Quality & Status** display icons under **Styles** configuration.

A list of supported SVG elements can be found at <https://github.com/svg-net/SVG>.

#### To import an SVG and insert as an Industrial Graphic:

1. In the Industrial Graphic Editor, on the **Graphic** menu, click **Import SVG**.
2. Browse for the required SVG and click **Open**.

---

**Note:** You can import multiple SVG files at once.

---

3. To insert the SVG file as Industrial Graphics, click anywhere on the canvas.

OR

1. Open the SVG in a notepad and copy all the content.

---

**Note:** Make sure the format of the SVG is proper.

---

2. Either right-click in the Industrial Graphic Editor canvas and select **Paste SVG**, or, on the **Graphic** menu, select **Paste SVG**.
3. Click on the canvas where you want to paste the SVG.

OR

- Drag and drop the SVG to the canvas.

You can drag and drop multiple SVG files at once.

---

**Note:** You can drag and drop SVG files only if User Account Control (UAC) is turned off. Before turning off UAC, be sure to check your organization's security policy, as turning it off could cause a security issue. If you try to drag and drop an SVG file to the canvas with UAC turned on, a forbidden icon is displayed.

---

When importing an SVG, be aware of the following:

- If the imported SVG is bigger than the view area, then the SVG will be resized accordingly by maintaining the aspect ratio.
- If there are any unsupported characters in the name of the SVG or primitive, it will be converted to “\_”.
- The SVG element name is used to name the corresponding primitive. If a SVG element does not have a name, then Industrial Graphic primitive type is used to name the primitive. For example, if the SVG group name is "Header", the primitive name in Graphic Editor after import will also be Header. If the SVG group has no name, the same will be named as <primitive type>\_<number>, that is group\_01.
- Only three colors are supported in gradients. If the SVG file has more than three colors for primitive, the first, last, and middle colors are considered.

The middle color is calculated by the formula  $middle\ color = (int)((colors.Length) / 2)$ . That is, if there are four colors, then it will consider the second color as middle color.

For a list of supported and unsupported elements, see [SVG Limitations](#).

## SVG Limitations

The Industrial Graphic Editor includes support for SVG graphics. However, this support has some limitations. The table below shows the SVG graphic support status for each type of element.

Element Type	Element	Support
Animation elements	animate	NO
Animation elements	animatecolor	NO
Animation elements	animatemotion	NO
Animation elements	animatetransform	NO
Animation elements	set	NO
Basic shapes	circle	YES
Basic shapes	ellipse	YES
Basic shapes	line	YES
Basic shapes	polygon	YES
Basic shapes	polyline	YES
Basic shapes	rect	YES
Container elements	a	YES
Container elements	defs	YES
Container elements	g	YES
Container elements	glyph	YES
Container elements	marker	NO
Container elements	mask	NO
Container elements	missing-glyph	YES
Container elements	pattern	NO
Container elements	svg	YES
Container elements	switch	YES
Container elements	symbol	YES
Descriptive elements	desc	YES

Element Type	Element	Support
Descriptive elements	metadata	YES
Descriptive elements	title	YES
Filter Elements	filter	NO
Filter light source elements	feDistantLight	NO
Filter light source elements	fePointLight	NO
Filter light source elements	feSpotlight	NO
Filter Primitive Elements	feBlend	NO
Filter Primitive Elements	feColorMatrix	NO
Filter Primitive Elements	feComponentTransfer	NO
Filter Primitive Elements	feComposite	NO
Filter Primitive Elements	feConvolveMatrix	NO
Filter Primitive Elements	feDiffuseLighting	NO
Filter Primitive Elements	feDisplacementMap	NO
Filter Primitive Elements	feFlood	NO
Filter Primitive Elements	feFuncA	NO
Filter Primitive Elements	feFuncB	NO
Filter Primitive Elements	feFuncG	NO
Filter Primitive Elements	feFuncR	NO
Filter Primitive Elements	feGaussianBlur	NO
Filter Primitive Elements	feImage	NO
Filter Primitive Elements	feMerge	NO
Filter Primitive Elements	feMergeNode	NO
Filter Primitive Elements	feMorphology	NO
Filter Primitive Elements	feOffset	NO
Filter Primitive Elements	feSpecularLighting	NO
Filter Primitive Elements	feTile	NO

Element Type	Element	Support
Filter Primitive Elements	feTurbulence	NO
Gradient elements	linearGradient	Partial *
Gradient elements	radialGradient	Partial *
Graphics elements	circle	YES
Graphics elements	ellipse	YES
Graphics elements	image	YES
Graphics elements	line	YES
Graphics elements	path	YES
Graphics elements	polygon	YES
Graphics elements	polyline	YES
Graphics elements	rect	YES
Graphics elements	text	YES
Graphics elements	use	YES
Shape elements	circle	YES
Shape elements	ellipse	YES
Shape elements	line	YES
Shape elements	path	YES
Shape elements	polygon	YES
Shape elements	polyline	YES
Shape elements	rect	YES
Structural elements	defs	YES
Structural elements	g	YES
Structural elements	svg	YES
Structural elements	symbol	YES
Structural elements	use	YES
Text content child elements	altglyph	YES

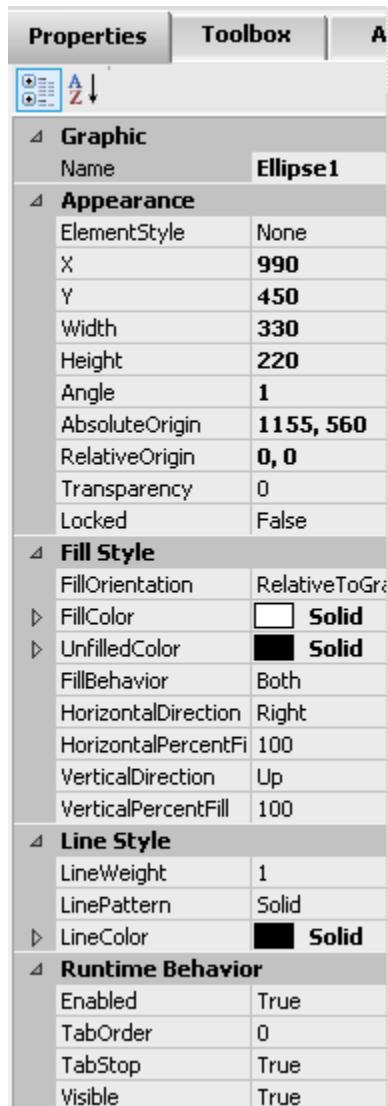
Element Type	Element	Support
Text content child elements	textpath	YES
Text content child elements	tref	YES
Text content child elements	tspan	YES
Text content elements	altGlyph	YES
Text content elements	text	YES
Text content elements	textPath	YES
Text content elements	tref	YES
Text content elements	tspan	YES
Uncategorized elements	altglyphdef	NO
Uncategorized elements	altglyphitem	NO
Uncategorized elements	clippath	NO
Uncategorized elements	color-profile	NO
Uncategorized elements	cursor	NO
Uncategorized elements	font	YES
Uncategorized elements	font-face	YES
Uncategorized elements	font-face-format	YES
Uncategorized elements	font-face-name	YES
Uncategorized elements	font-face-src	YES
Uncategorized elements	font-face-uri	YES
Uncategorized elements	foreignobject	NO
Uncategorized elements	glyph	NO
Uncategorized elements	glyphref	NO
Uncategorized elements	hkern	NO
Uncategorized elements	mpath	NO
Uncategorized elements	script	NO
Uncategorized elements	stop	NO

Element Type	Element	Support
Uncategorized elements	style	YES
Uncategorized elements	view	NO
Uncategorized elements	vkern	NO

(\*) Gradients support up to 3 colors

## Editing Element Properties

You can control the appearance of an element, a group of elements, or multiple elements with functions on the toolbar and properties shown in the **Properties Editor** of the Industrial Graphic Editor.



Often you can edit an element by changing the values of its properties instead of using the mouse to perform the same function. This is useful when you want very exact editing, such as when you want to resize an element to a specific width.

The **Properties Editor** shows the properties common to all selected elements.

- Read-only properties appear in grey.
- Non-default values appear in bold.

---

**Note:** The **Properties Editor** not only supports values, but also allows input of color, font, and file information in the respective dialog boxes.

---

Properties are organized in categories so you can find them more easily. The following table shows the categories:

Property Category	Purpose
<b>Graphic</b>	Element name or other describing identifiers
<b>Appearance</b>	Element style, location, size, orientation, offset, transparency and locked status
<b>Fill Style</b>	Any parameters related to the fill appearance of the element
<b>Line Style</b>	Any parameters related to the appearance of element lines
<b>Text Style</b>	Any parameters related to the appearance of element text
<b>Runtime Behavior</b>	Element visibility, tab order and any other element behavior at run time
<b>Custom Properties</b>	Additional user-defined properties you can associate with any element

## Related Topics

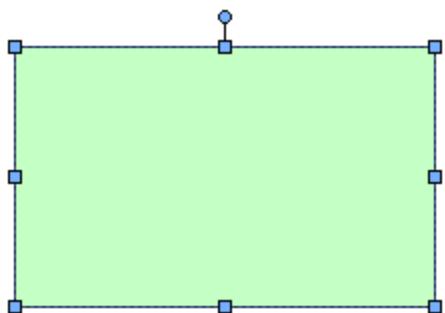
[Working with Graphic Elements](#)

## Selecting Elements

You can select one or more graphic elements from the Industrial Graphic Editor canvas by:

- Clicking on them with the mouse.
- Dragging a lasso around them with your mouse.
- Selecting them with a menu option or with a shortcut key.
- Selecting them by name from the **Elements** list.

When you select an element, handles appear around the border of the element that can be moved to control the size of the element. You can change the orientation of a graphic element by moving the handle connected to the top center border handle.



When you select multiple elements, the last selected element is the primary element. All other previously selected elements are secondary elements.

Selected Element	Description
Primary Element	Appears with color-filled handles. Behaves as an active selected element. Is the point of reference for all operations, such as aligning or spacing multiple selected elements.
Secondary Elements	Appear with white handles. Behave as inactive selected elements. Follow the edits made to the primary element.

To select a group, you need to click one of the elements contained in the group.

## Related Topics

- [Working with Graphic Elements](#)
- [Selecting Elements by Mouse Click](#)
- [Selecting Elements by Lasso](#)

[Selecting All Elements](#)

[Selecting Elements Using the Elements List](#)

[Unselecting Elements](#)

## Selecting Elements by Mouse Click

You can select one or more elements by pressing Shift + clicking. This is particularly useful for selecting multiple elements that are not necessarily all included in a specified rectangular area on the canvas.

### To select an element or multiple elements by mouse click

1. On the canvas, click an element. It becomes selected.
2. To select further elements, press Shift+ click. The other elements become selected.

---

**Note:** You can see in the Elements List which elements are selected.

---

## Related Topics

[Selecting Elements](#)

## Selecting Elements by Lasso

You can select one or more elements by lassoing them with your mouse. This is useful for selecting multiple elements within a specified rectangular area on the canvas.

### To select elements by lasso

1. On the canvas, click outside any element and hold the mouse button down.
2. Drag the mouse so that the lasso wraps around all elements that you want to select.
3. When you are done, release the mouse button. The elements that are fully enclosed within the lasso are selected.

## Related Topics

[Drawing and Dragging Elements](#)

## Selecting All Elements

You can select all elements using the Select All function.

### To select all elements

- On the **Edit** menu, click **Select All**. All elements on the canvas are selected.

---

**Note:** You can also press the F2 key to select all elements.

---

## Related Topics

[Drawing and Dragging Elements](#)

## Selecting Elements Using the Elements List

You can use the Elements List to select any elements on the canvas. The Elements List is particularly useful for selecting elements behind other elements.

The Elements List shows which elements are currently selected. The primary selected element appears by default in dark blue, the secondary selected elements appear by default in light blue.

---

**Note:** The color setting of the Elements List depends on the setting for the **Selected Items** option in the operating system's **Display Properties Appearance** panel.

### To select elements using the Elements List

1. In the **Elements List**, select the element name.
2. To select multiple elements, Ctrl + click the other elements.

## Related Topics

[Selecting Elements](#)

## Unselecting Elements

You can unselect one or more selected elements. You can do this by clicking on them individually on the canvas or in the Elements List.

If you want to remove the selected elements in a specified rectangular area, you can use the lasso.

### To unselect elements individually

1. Do one of the following:
  - Shift + click the selected element on the canvas.
  - Ctrl + click the selected element name in the Elements List.
2. Repeat the previous step for all elements you want to unselect.

### To unselect elements from a specified rectangular area

1. Shift + click the canvas outside of any element.
2. Drag the mouse so that the lasso surrounds the elements that you want to unselect.
3. Release the mouse button. The selected elements within the lasso are unselected, and the selected elements outside the lasso remain selected.

## Related Topics

[Selecting Elements](#)

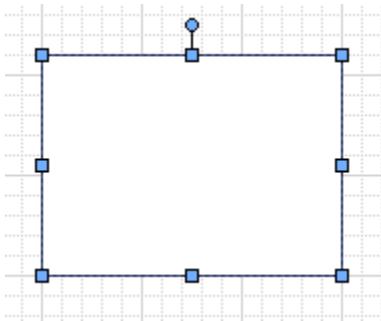
## Inline Editing

After you place graphic elements on the Industrial Graphic Editor canvas, you can edit them by selecting them and clicking on them again. This is called inline editing. The following elements support inline editing.:

Element	Use inline editing to
Button, text, text box	Edit the text.
Polyline, polygon, curve, closed curve	Edit the control points.
2-point arc, 2-point pie, 2-point chord, 3-point arc, 3-point pie, 3-point chord	Edit the start and sweep angles.
Group	Edit the individual elements and groups contained in the group.
Path	Edit the control points.

### To edit elements with inline editing

1. Select an element by clicking on it or selecting it from the **Elements** list. Element handles appear around the border of the element.



2. Click the element again to begin inline editing.
  - For buttons, text, and text boxes, the text is selected and you can type new text.
  - For polylines, polygons, curves, and closed curves, the control points of the element appear. Move a control point to change the shape of the element. You can also add and delete control points. For more information, see [Adding and Removing Control Points](#).
  - For arcs, pies, and chords, the handles for the start angle and sweep angle appear. Move a handle to change the start angle and sweep angle.
  - For groups, the group handle is replaced with a shaded outline. You can select individual elements and groups within the group to edit and move them.
3. Click the canvas outside the element to exit from inline editing.

### Related Topics

[Working with Graphic Elements](#)

### Copying, Cutting, and Pasting Elements

After you draw elements, you have the same cut, copy, and paste options available to you as in any other Windows application. However, some of these options behave differently in the Industrial Graphic Editor.

You can also duplicate elements. Duplicating elements lets you quickly make copies of existing selected elements without first copying or cutting. You can duplicate one or more selected elements at the same time.

When you copy or duplicate elements, all of its properties are copied with the element. If you do not want the properties to be identical, you must change the properties after you copy.

Locked grouped elements and the path element behave differently when you copy or duplicate them.

If you copy or duplicate:

- A set of elements that are locked, the copy is not locked.
- Grouped elements, the copy is still grouped.
- A path element, the copy is also a path.

## Related Topics

[Working with Graphic Elements](#)

[Copying Elements](#)

[Cutting or Deleting Elements](#)

[Duplicating Elements](#)

## Copying Elements

After you select an element, you can copy it by using menu options or you can Ctrl + click.

### To copy one or more elements

Do any of the following:

- Select one or more elements to be copied on the canvas. On the **Edit** menu, click **Copy**. On the **Edit** menu, click **Paste**. The paste pointer appears. Click the canvas where you want to place the copy.
- Ctrl + click an element.
- Select one or more elements to be copied on the canvas. Press Ctrl + C. Press Ctrl + V. The paste pointer appears. Click the canvas where you want to place the copy.

## Related Topics

[Copying, Cutting, and Pasting Elements](#)

## Cutting or Deleting Elements

You can cut elements or groups or you can delete them. Cutting lets you select elements or groups and remove them from the canvas. You can paste the removed elements or groups.

Deleting elements or groups deletes them from the canvas. You cannot paste deleted elements or groups.

### To cut one or more elements

- Select one or more elements, and then do one of the following:
  - On the **Edit** menu, click **Cut**.

- Press Ctrl + X.

### To cut and paste elements on the canvas

1. Select the element or group.
2. On the **Edit** menu, click **Cut**.
3. Do one of the following:
  - Click **Paste** on the **Edit** menu.
  - Press Ctrl + V.
4. Click the canvas location where you want the element or group to be placed.

### To delete an element or a group

1. To remove the element or group and **not** use it in the future, select the element or group.
2. Do one of the following:
  - Click **Delete** on the **Edit** menu.
  - Press Delete on your keyboard.

## Related Topics

[Copying, Cutting, and Pasting Elements](#)

## Duplicating Elements

Duplicating elements enables you to select an element or elements and quickly make copies of them.

You can also specify the amount of overlap when you duplicate.

### To duplicate elements

1. Select one or more elements.
2. Do one of the following:
  - a. Click **Duplicate** on the **Edit** menu. The selected element is duplicated and appears offset to the original element.
  - b. Press Ctrl + D. The selected element is duplicated and appears offset to the original element.
  - c. Ctrl + click one of the selected elements to duplicate all selected elements. You can keep the mouse button down and drag them to the new position on the canvas.

### To set the overlap when you duplicate

1. Duplicate an element or elements. The element is copied overlapping the original.
2. Move the duplicated element to the location relative to the original. For example, move the duplicated element five grid spaces above the original element.
3. Duplicate the element again. The new duplicate is placed in the same offset you specified in the preceding step. For example, five grid spaces above the original element.

## Related Topics

[Copying, Cutting, and Pasting Elements](#)

## Moving Elements

After you create elements, you can move them to the location you want on the canvas.

You can move elements or groups by dragging them to the new location or you can open the properties for the element or group and change the X and Y properties.

If you turned on snap to grid, moving an element or group with the mouse snaps the element or group to the grid.

If you move an element or group by specifying X and Y coordinates, it does not snap to the grid.

You can move an element or group vertically or horizontally using the keyboard.

### To move an element or group using the mouse

1. Select the element or group you want to move.
2. Drag the elements or group to the new location.

### To move an element or group by specifying the X and Y properties

1. Select the element or group you want to move.
2. In the Properties Editor, expand **Appearance**.
3. Do the following:
  - In the **X** box, type the new X location.
  - In the **Y** box, type the new Y location.
4. Click in the canvas or click ENTER.

### To move an element or group vertically or horizontally using the mouse

1. Shift + click to select the element or group you want to move.
2. Drag the elements or group to the new location.

### To move an element or group vertically or horizontally using the keyboard

1. Select the element or group you want to move.
2. Do one of the following:
  - Press the Up or Down arrow keys to move the element or group vertically by one unit in the grid.
  - Press the Left or Right arrow keys to move the element or group horizontally by one unit in the grid.

---

**Note:** You can move the element or group by two units in the grid by additionally pressing the Shift key, by four units by additionally pressing the Ctrl key, and by 10 units by additionally pressing both keys.

### To move multiple elements or groups

1. Select the elements and/or groups.

2. Move them as you would with one single element. The elements are moved together and maintain their spatial relationship when moving.

## Related Topics

[Working with Graphic Elements](#)

## Aligning Elements

After you draw elements, you can align them:

- Horizontally so that their top or bottom sides or their center points are horizontally aligned.
- Vertically so that their left, right, or center points are vertically aligned.
- So that their center points are on top of each other.
- So that their points of origin are on top of each other.

When you align elements, the secondary elements are moved so that they align with the primary element. For more information about primary and secondary elements, see [Selecting Elements](#).

## Related Topics

[Working with Graphic Elements](#)

[Aligning Elements Horizontally](#)

[Aligning Elements Vertically](#)

[Aligning Elements by their Center Points](#)

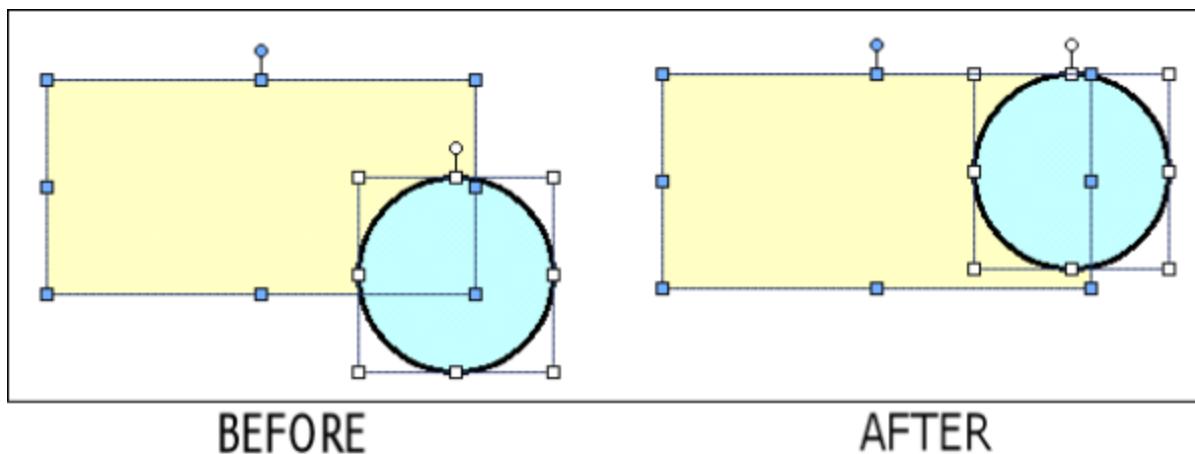
[Aligning Elements by their Points of Origin](#)

## Aligning Elements Horizontally

You can align multiple elements by their top or bottom sides or horizontally on their middle points.

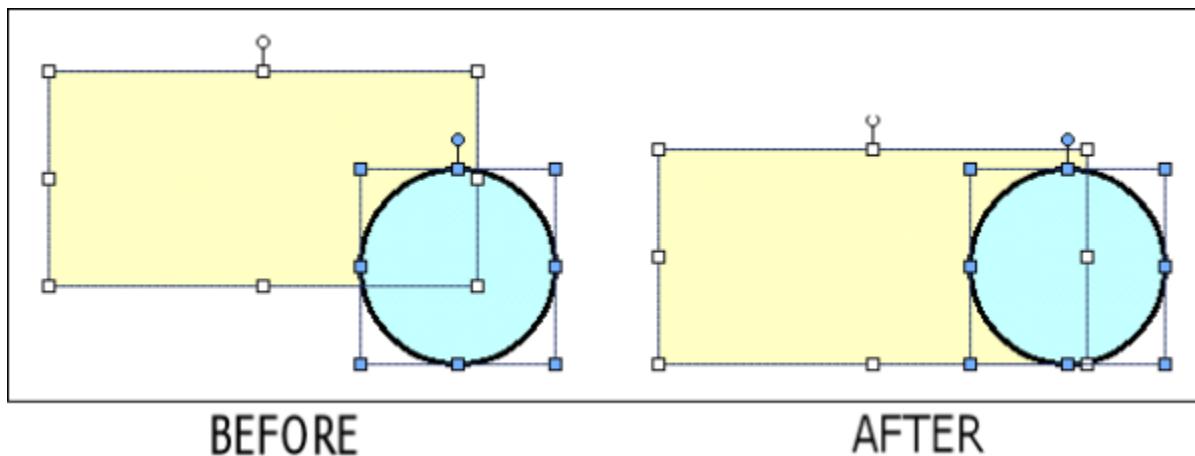
### To align elements by their top sides

1. Select all elements that you want to align. The element you want to align all other elements to needs to be the primary element.
2. On the **Arrange** menu, point to **Align**, and then click **Align Top**. The secondary elements are moved so that their top sides are aligned with the top side of the primary element.



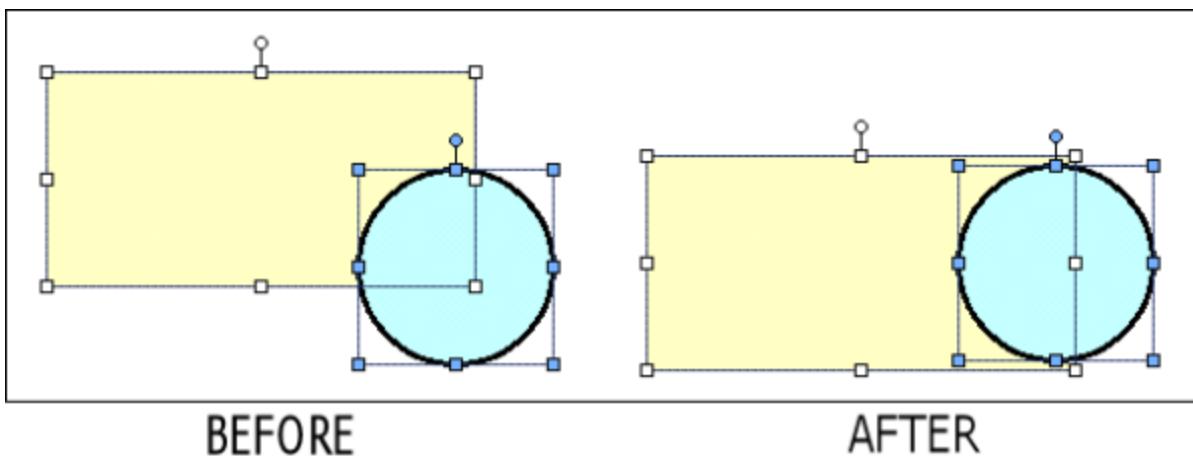
### To align elements by their bottom sides

1. Select all elements that you want to align. The element you want to align all other elements to needs to be the primary element.
2. On the **Arrange** menu, point to **Align**, and then click **Align Bottom**. The secondary elements are moved so that their bottom sides are aligned with the bottom side of the primary element.



### To align elements horizontally by their center points

1. Select all elements that you want to align. The element you want to align all other elements to needs to be the primary element.
2. On the **Arrange** menu, point to **Align**, and then click **Align Middle**. The secondary elements are moved vertically so that their center points are aligned with the center point of the primary element.



## Related Topics

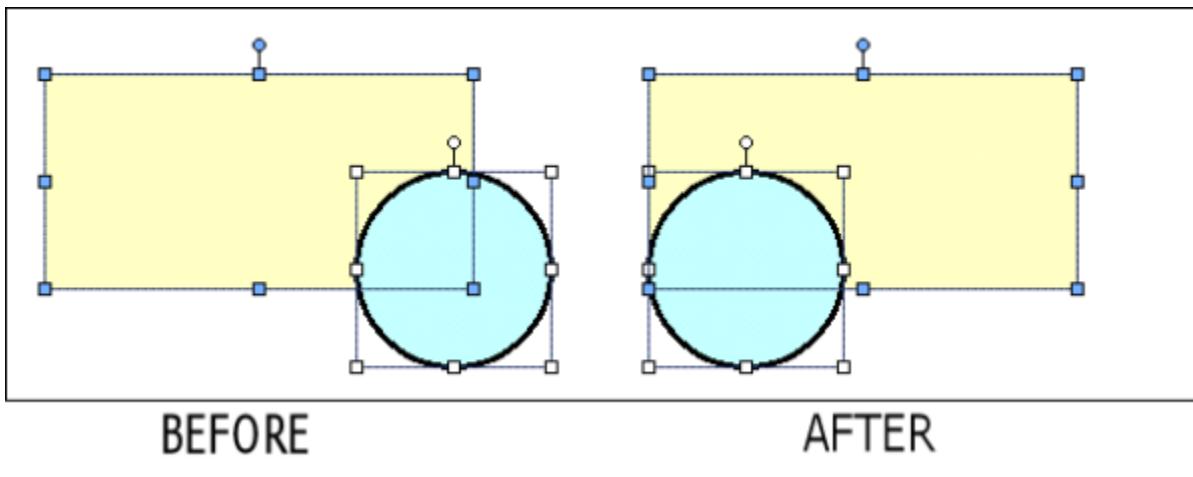
[Aligning Elements](#)

### Aligning Elements Vertically

You can vertically align multiple elements on the left, right, or their center points.

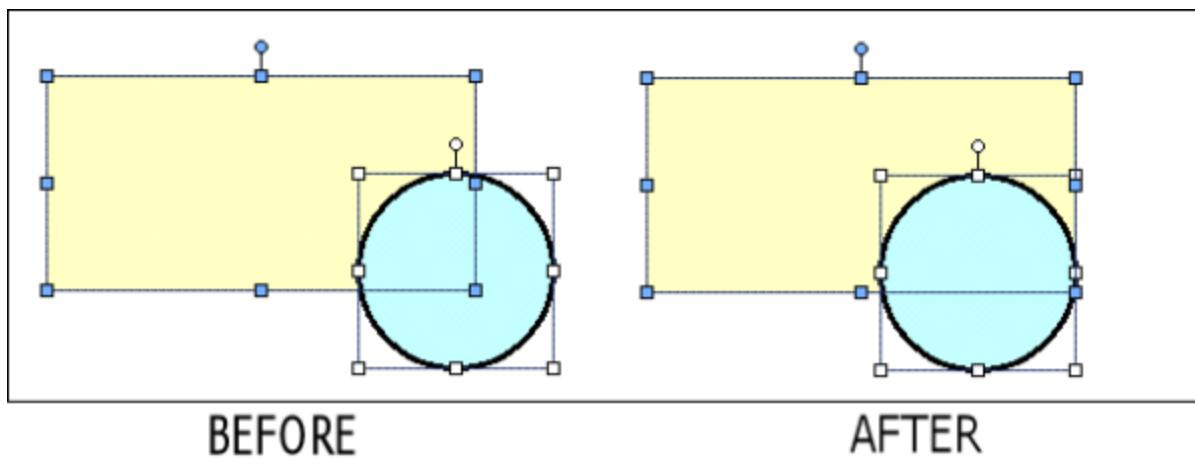
#### To align elements by their left sides

1. Select all elements that you want to align. The element you want to align all other elements to needs to be the primary element.
2. On the **Arrange** menu, point to **Align**, and then click **Align Left**. The secondary elements are moved so that their left sides are aligned with the left side of the primary element.



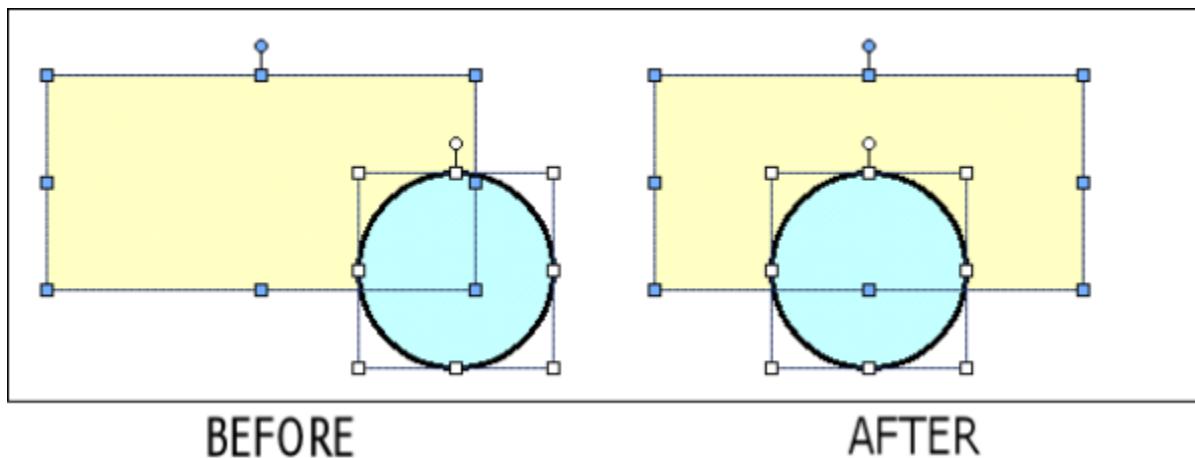
#### To align elements by their right sides

1. Select all elements that you want to align. The element you want to align all other elements to needs to be the primary element.
2. On the **Arrange** menu, point to **Align**, and then click **Align Right**. The secondary elements are moved so that their right sides are aligned with the right side of the primary element.



### To align elements vertically by their centers

1. Select all elements that you want to align. The element you want to align all other elements to needs to be the primary element.
2. On the **Arrange** menu, point to **Align**, and then click **Align Center**. The secondary elements are moved horizontally so that their center points are aligned with the center point of the primary element.



## Related Topics

[Aligning Elements](#)

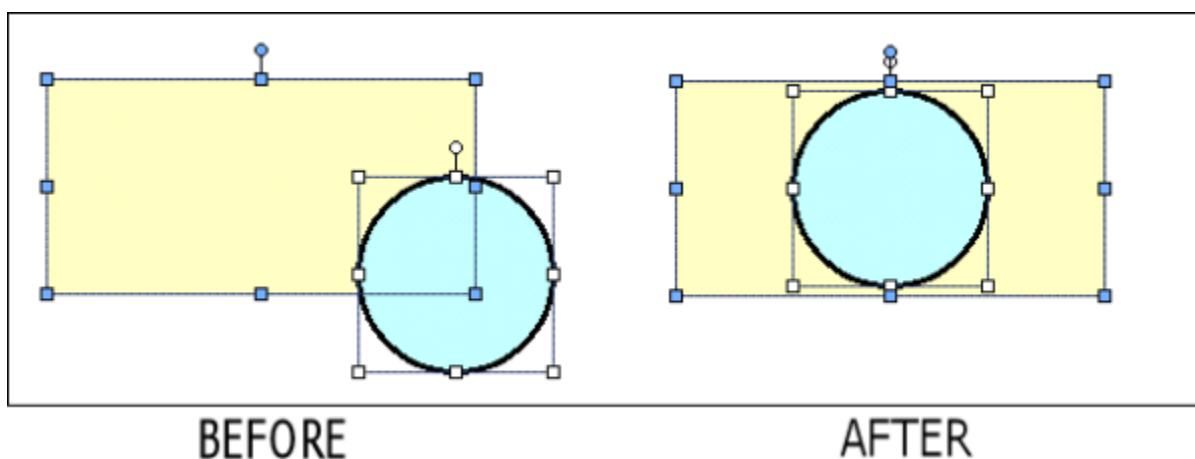
### Aligning Elements by their Center Points

You can align elements by their center points. The center point of one or more elements is the point halfway between the horizontal and vertical boundaries.

### To align elements on their center points

1. Select all elements that you want to align. The element you want to align all other elements to needs to be the primary element.
2. On the **Arrange** menu, point to **Align**, and then click **Align Centers**. The secondary elements are moved so

that their center points are placed on top of the center point of the primary element.



## Related Topics

[Aligning Elements](#)

### Aligning Elements by their Points of Origin

You can align elements by their points of origin. By default, the element's center point is the point of origin. But, an element's center point can be changed. The center point is the anchor point of an element to the canvas.

#### To align elements on their points of origin

1. Select all elements that you want to align. The element you want to align all other elements to needs to be the primary element.
2. On the **Arrange** menu, point to **Align**, and then click **Align Origins**. The secondary elements are moved so that their points of origins are placed on top of the point of origin of the primary element.

## Related Topics

[Aligning Elements](#)

### Adjusting the Spacing between Elements

You can adjust the space between elements according to specific rules.

You can adjust the spacing between elements in the following ways:

- Horizontally - moves the selected elements left or right without changing the vertical positions.
- Vertically - moves the selected elements up or down without changing the horizontal positions.
- Distribution - moves the selected elements so that their center points are distributed in equal distance to each other.
- Equal spacing - moves the selected elements so that the distance between their edges is equal.
- Increase spacing - moves all selected elements one pixel further away from each other. The primary element

does not move.

- Decrease spacing - moves all selected elements one pixel closer toward each other. The primary element does not move.
- Remove spacing - removes all space between selected elements so that their edges touch.

## Related Topics

[Working with Graphic Elements](#)

[Distributing Elements](#)

[Making Space between Elements Equal](#)

[Increasing Space between Elements](#)

[Decreasing Space between Elements](#)

[Removing All Space between Elements](#)

## Distributing Elements

You can distribute elements so that their center points are distributed in equal distance to each other.

### To distribute elements horizontally

1. Select at least three elements.
2. On the **Arrange** menu, point to **Space**, and then click **Distribute Horizontal**. The selected elements are distributed horizontally.

### To distribute elements vertically

1. Select at least three elements.
2. On the **Arrange** menu, point to **Space**, and then click **Distribute Vertical**. The selected elements are distributed vertically.

## Related Topics

[Adjusting the Spacing between Elements](#)

## Making Space between Elements Equal

You can space elements so that the distances between their boundaries are equal.

The difference between making space between elements equal and distributing them is that making space equal uses the boundaries of the elements, whereas distributing uses the center points. Both do not necessarily lead to the same result.

### To make the horizontal space between elements equal

1. Select at least three elements.
2. On the **Arrange** menu, point to **Space**, and then click **Make Horizontal Space Equal**. The selected elements are moved so that the horizontal spaces between their boundaries are equal.

## To make the vertical space between elements equal

1. Select at least three elements.
2. On the **Arrange** menu, point to **Space**, and then click **Make Vertical Space Equal**. The selected elements are moved so that the vertical spaces between their boundaries are equal.

## Related Topics

[Adjusting the Spacing between Elements](#)

### Increasing Space between Elements

You can increase space between elements equally.

The primary element does not move. All secondary elements are moved away from the primary element.

#### To increase the horizontal space between elements

1. Select at least two elements.
2. On the **Arrange** menu, point to **Space**, and then click **Increase Horizontal Spacing**. The selected elements are moved so that the horizontal space between them is increased by one pixel.
3. Repeat the previous step to move the selected elements further away from each other.

#### To increase the vertical space between elements

1. Select at least two elements.
2. On the **Arrange** menu, point to **Space**, and then click **Increase Vertical Spacing**. The selected elements are moved so that the vertical space between them is increased by one pixel.
3. Repeat the previous step to move the selected elements further away from each other.

## Related Topics

[Adjusting the Spacing between Elements](#)

### Decreasing Space between Elements

You can decrease space between elements equally.

The primary element does not move. All secondary elements move toward the primary element. You can move them until the left sides of all elements are aligned.

#### To decrease the horizontal space between elements

1. Select at least two elements.
2. On the **Arrange** menu, point to **Space**, and then click **Decrease Horizontal Spacing**. The selected elements are moved so that the horizontal space between them is decreased by one pixel.
3. Repeat the previous step to move the selected elements closer toward each other.

## To decrease the vertical space between elements

1. Select at least two elements.
2. On the **Arrange** menu, point to **Space**, and then click **Decrease Vertical Spacing**. The selected elements are moved so that the vertical space between them is decreased by one pixel.
3. Repeat the previous step to move the selected elements closer toward each other.

## Related Topics

[Adjusting the Spacing between Elements](#)

## Removing All Space between Elements

You can remove all space between selected elements so that their boundaries touch.

The primary element does not move. All secondary elements move toward the primary element. You can move them until the left and right sides of all secondary elements are aligned.

## To remove all horizontal space between elements

1. Select all elements between which you want to remove the space.
2. On the **Arrange** menu, point to **Space**, and then click **Remove Horizontal Spacing**. The horizontal space between all selected elements is removed, so that their boundaries touch.

## To remove all vertical space between elements

1. Select all elements between which you want to remove the space.
2. On the **Arrange** menu, point to **Space**, and then click **Remove Vertical Spacing**. The vertical space between all selected elements is removed, so that their boundaries touch.

## Related Topics

[Adjusting the Spacing between Elements](#)

## Resizing Elements

You can resize selected elements by:

- Dragging the handles of a single element to increase or decrease its horizontal or vertical size.
- Changing the Width and Height properties of one or more elements using the Properties Editor.
- Proportionally resizing multiple elements.
- Making multiple objects the same width and/or height.

Some elements cannot be resized or can only be resized in certain directions, such as the Calendar control or DateTime Picker. If the primary element has such restrictions, then any secondary elements resize proportional to the change in primary element's size and do not resize independently.

## Related Topics

- [Working with Graphic Elements](#)
- [Resizing a Single Element with the Mouse](#)
- [Resizing Elements by Changing Size Properties](#)
- [Resizing Elements Proportionally](#)
- [Making Elements the Same Width, Height, or Size](#)

### Resizing a Single Element with the Mouse

You can resize a single selected element with the mouse.

You can resize most elements to any given width and height, or to a fixed width to height ratio.

#### To resize a single selected element with the mouse

1. Select an element. The handles of the selected element appear.
2. Drag one of the handles. The object is resized while you drag.
3. Release the mouse button.

#### To resize a single selected element with the mouse and keeping a fixed width/height ratio

1. Select an element. The handles of the selected element appear.
2. Press and hold the Shift key.
3. Drag one of the handles. The object is resized while you drag, the width/height ratio stays unchanged.
4. Release the mouse button and Shift key.

## Related Topics

- [Resizing Elements](#)

### Resizing Elements by Changing Size Properties

You can resize one or more elements by changing the width and/or height property of the selected elements.

#### To resize elements by changing their size properties

1. Select one or more elements.
2. In the Properties Editor, type a value for **Width** and for **Height**. The selected elements are resized accordingly.

## Related Topics

- [Resizing Elements](#)

## Resizing Elements Proportionally

You can resize multiple elements proportionally on the canvas. One element is the primary element you can use to resize. The secondary elements resize proportionally to the change of the primary element.

### To resize elements proportionally

1. Select multiple elements.
2. Drag one of the handles of the primary element. The secondary elements are resized accordingly by the same percentage.
3. Release the mouse button.

For example, assume the primary element is 100 pixels wide and 50 pixels high. A secondary element is 200 pixels wide and 20 pixels high.

You drag the handle of the primary element so that it is 120 pixels wide (20 percent increase) and 100 pixels high (100 percent increase).

Then the secondary element is resized to 240 pixels wide (20 percent increase of the original width of 200 pixels) and 40 pixels high (100 percent increase of the original width of 20 pixels).

## Related Topics

[Resizing Elements](#)

## Making Elements the Same Width, Height, or Size

You can make elements the same width, height, or size.

### To make elements the same width

1. Select at least two elements. The primary element is the element that provides the target width for all elements.
2. On the **Arrange** menu, point to **Size**, and then click **Make Same Width**. The width of the secondary elements are resized to the same width as the primary element.

### To make elements the same height

1. Select at least two elements. The primary element is the element that provides the target height for all elements.
2. On the **Arrange** menu, point to **Size**, and then click **Make Same Height**. The height of the secondary elements are resized to the same height as the primary element.

### To make elements the same size

1. Select at least two elements. The primary element is the element that provides the target size for all elements.
2. On the **Arrange** menu, point to **Size**, and then click **Make Same Size**. The size of the secondary elements are resized to the same size as the primary element.

## Related Topics

[Resizing Elements](#)

### Adjusting the z-Order of Elements

The z-order of elements specifies which element appears on top of other elements when the elements overlap on the canvas. The z-order also determines how the elements of a path graphic connect.

When you place new elements on the canvas, they are placed at the top and can cover all other elements.

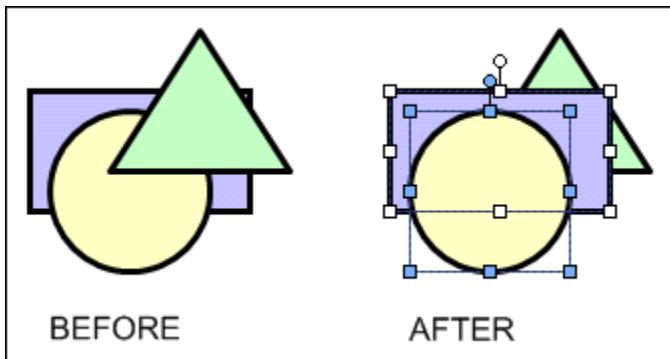
However, you might want to bring certain elements forward so that they are always visible or overlap certain other elements. Or you may want to use a large background element behind all other elements. You can:

- Bring one or more elements to the very front.
- Send one or more elements to the very back.
- Bring one or more elements one level forward.
- Send one or more elements one level backward.

You can use the Elements List to see or change the z-order of the elements.

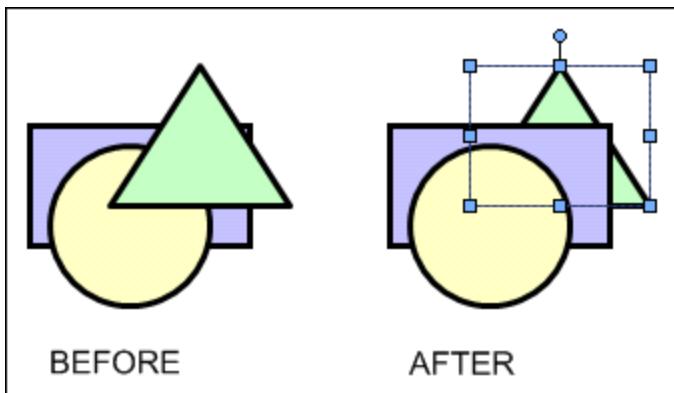
#### To bring selected elements to the front

- On the **Arrange** menu, point to **Order**, and then click **Bring To Front**. The selected elements are brought to the front. They do not change their relative z-order.

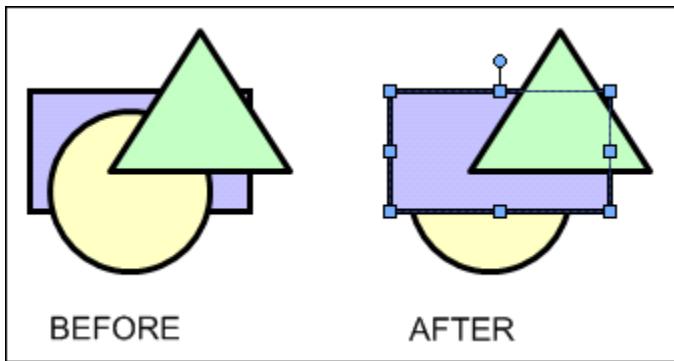


#### To send selected elements to the back

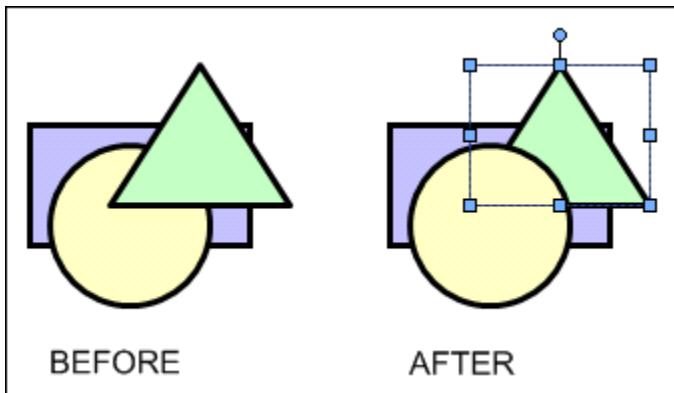
- On the **Arrange** menu, point to **Order**, and then click **Send To Back**. The selected elements are sent to the back. They do not change their relative z-order.

**To bring selected elements one level forward**

- On the **Arrange** menu, point to **Order**, and then click **Bring Forward**.

**To send selected elements one level backward**

- On the **Arrange** menu, point to **Order**, and then click **Bring Backward**.

**Related Topics**

[Working with Graphic Elements](#)

## Rotating Elements

You can rotate elements to any orientation (0 - 359 degrees):

- Graphically with the rotation handle.
- Numerically by typing the orientation angle in the Properties Editor.
- By rotating in 90 degree increments in a clockwise or counter-clockwise direction.

The element rotates around its point of origin. By default, the point of origin is in the center of the element. You can move the point of origin to any other location, even outside of the object itself. To change the point of origin, see [Moving the Origin of an Element](#).

## Related Topics

[Working with Graphic Elements](#)

[Rotating Elements with the Mouse](#)

[Rotating Elements by Changing the Angle Property](#)

[Rotating Elements by 90 Degrees](#)

### Rotating Elements with the Mouse

You can rotate one or more elements with the mouse. If you select multiple elements, you can rotate the primary element. The secondary elements rotate in unison with the primary element.

You can rotate elements:

- Freely in the range 0 to 359 in integer degrees.
- In multiples of 15 degrees.
- In multiples of 45 degrees.

You can rotate an element with the rotation handle. The rotation handle is a light-blue circle at the top of a selected element.

#### To rotate elements freely with the mouse

1. Select one or more elements.
2. Grab the rotation handle of the primary element.
3. Drag the mouse across the screen. All selected elements are rotated around their own points of origin as you move the mouse.
4. Release the mouse button.

#### To rotate elements by multiple of 15 degrees with the mouse

1. Select one or more elements.
2. Grab the rotation handle of the primary element.
3. Press and hold the Shift key.

4. Drag the mouse across the screen. All selected elements are rotated in multiples of 15 degrees around their own points or origin as you move the mouse.
5. Release the mouse button and Shift key.

### To rotate elements by multiple of 45 degrees with the mouse

1. Select one or more elements.
2. Grab the rotation handle of the primary element.
3. Press and hold the Ctrl key.
4. Drag the mouse across the screen. All selected elements are rotated in multiples of 45 degrees around their own points or origin as you move the mouse.
5. Release the mouse button and Ctrl key.

## Related Topics

[Rotating Elements](#)

### Rotating Elements by Changing the Angle Property

You can change the angle property of one or more selected elements.

#### To rotate elements by changing the angle property

1. Select one or more elements.
2. In the Properties Editor, type a value in the **Angle** box.
3. Click Enter. The selected elements rotate to the specified angle.

## Related Topics

[Rotating Elements](#)

### Rotating Elements by 90 Degrees

You can rotate elements in 90 degrees clockwise or counter-clockwise increments.

To rotate elements by multiples of 15 and 45 degrees, see [Rotating Elements with the Mouse](#).

#### To rotate elements by 90 degrees clockwise

1. Select one or more elements.
2. On the **Arrange** menu, point to **Transform**, and then click **Rotate Clockwise**. The selected elements rotate by 90 degrees clockwise.

#### To rotate elements by 90 degrees counter-clockwise

1. Select one or more elements.
2. On the **Arrange** menu, point to **Transform**, and then click **Rotate Counter Clockwise**. The selected elements

rotate by 90 degrees counter-clockwise.

## Related Topics

[Rotating Elements](#)

### Moving the Origin of an Element

You can change the point of origin of any element. The point of origin specifies around which point the element rotates or flips. By default the point of origin is in the center of the element.

You can change the point of origin:

- With the mouse on the canvas.
- By specifying the absolute origin in the Properties Editor.
- By specifying the relative origin in the Properties Editor.

## Related Topics

[Working with Graphic Elements](#)

[Changing Points of Origin with the Mouse](#)

[Changing Points of Origin in the Properties Editor](#)

### Changing Points of Origin with the Mouse

You can change the point of origin for an element with the mouse.

#### To change the point of origin for an element with the mouse

1. Select an element on the canvas.
2. Move the mouse over the rotation handle of the element. The point of origin icon for the element appears.
3. Drag the Point of Origin icon to where you want to place the new point of origin for the element.
4. Release the mouse button.

## Related Topics

[Moving the Origin of an Element](#)

### Changing Points of Origin in the Properties Editor

You can change the absolute or relative point of origin in the Properties Editor.

The absolute point of origin shows the position of the point of origin in relation to the canvas. The absolute point of origin changes when the element moves.

The relative point of origin shows the position of the point of origin in relation to the center of the element. The relative point of origin does not change when the element moves.

## To change the point of origin in the Properties Editor

1. Select one or more elements on the canvas.
2. In the Properties Editor, do one of the following:
  - Type the absolute coordinates in the x, y format for the point of origin.
  - Type the relative coordinates in the x, y format for the point of origin.
3. Click Enter. The points of origin move to the specified absolute position or to the specified position in relation to the center points of the selected elements.

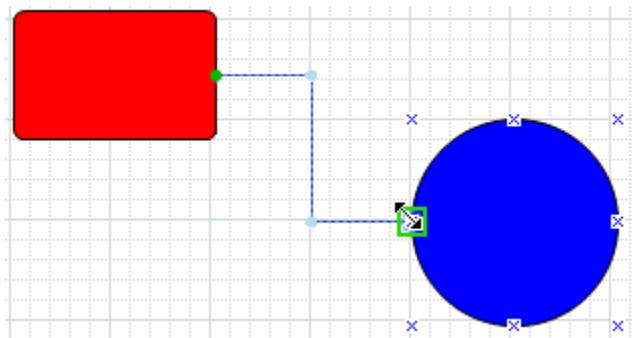
For example, if you have two elements, you can set the relative point of origin to 10, 10 to place the points of origin for both elements 10 pixels to the right and 10 pixels below the corresponding center points of each element.

## Related Topics

[Moving the Origin of an Element](#)

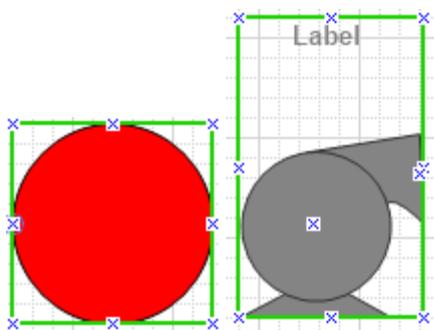
## Add Connectors Between Graphic Elements

A connector is a line drawn between graphic elements. A connector starts at a connection point on one graphic element and ends at a connection point on another element. Connectors are particularly useful for complex graphics like flow diagrams, industrial piping, or electrical wiring diagrams that incorporate many lines between graphic elements.

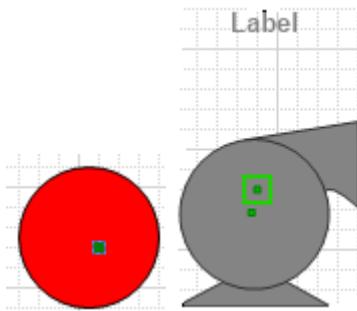


Connectors change length or orientation in response to changes to connected graphic elements during design time or run time. Graphic elements within a graphic can be moved, resized, or rotated and still maintain the connector between elements.

Connection points are the locations on a graphic element to attach a connector. A default set of eight connection points appear on the bounding rectangle around a graphic element or an embedded graphic.



You can also add custom connection points to graphic elements or embedded graphics if you want to place a connector at a different position than on a bounding rectangle.



One or more control points appear on an Angled connector based on the number of angles in the connector path. Using your mouse, you can move a control point horizontally or vertically to change the shape of a connector between the fixed connection points on both graphic elements. By default, a control point is placed at the intersection point of each right angle in a connector.

Connector lines do not maintain their horizontal and vertical orientation with 90 degree angles when placed in a graphic whose dimensions exceed 1280 by 1280 pixels. Instead, the connector will revert to a straight line between connection points.

You can also add control points to a connector if you want to change the shape of its path. For more information about adding control points to a connector, see [Change the Shape of a Connector](#).

## Related Topics

[Draw a Connector](#)

[Adding Connection Points](#)

[Change Connector Properties](#)

## Draw a Connector

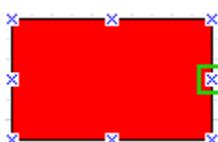
Use the Connector tool to draw a connector between graphic elements. The Connector tool initially attempts to draw a connector with a minimum number of angles. You can change the shape of the initial connector path using control points to redraw the path if necessary.

A connector supports Symbol Wizards like any other graphic element. You can associate a connector with a Symbol Wizard layer by dragging the connector element to the layer during design time. You can also remove the connector from a layer by removing the connector from the association list. If a connector is hidden based on the Symbol Wizard's Wizard Option configuration, the connector does not appear during run time.

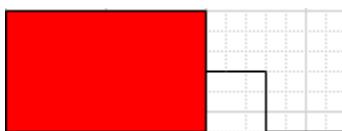
Press the Esc key to cancel drawing a connector. Also, clicking on the Industrial Graphic Editor's canvas takes you out of connector drawing mode.

### To draw a connector

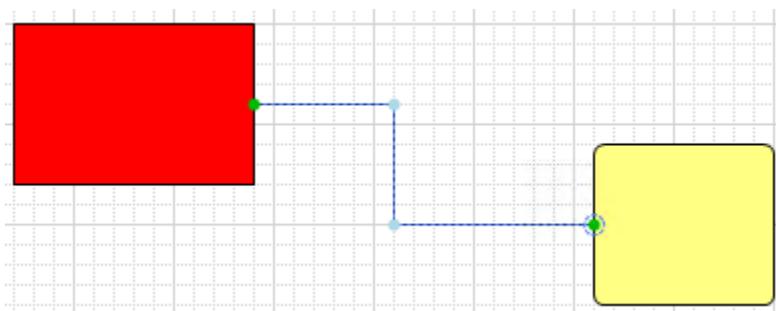
1. Open a graphic in Industrial Graphic Editor that you want to add a connector.
2. Click once on the **Connector** icon within the **Tools** pane to draw a single connector.  
If you want to draw multiple connectors, double click on the **Connector** icon.
3. Move your mouse over the first graphic element that you want to add a connector.  
The default connection points appear when you move your mouse over a graphic element.
4. Place the mouse over the connection point where you want to place the connector on the graphic element.  
A green rectangle appears around the connection point when it is selected.



A start connection point has an automatic offset, which is a perpendicular straight line segment from the start connection point to the first angle in the connector path. An automatic offset prevents the connector from following the border of the originating graphic element. No automatic offset is applied to the terminating connection point on the connected graphic element.



5. Press and hold your left mouse key and drag the mouse to the second graphic element.  
Connection points appear when you move the mouse over the second graphic element.



6. Release the mouse button when you are over a selected connection point on the second graphic element.  
The connector appears as a line between both graphic elements.
7. If necessary, use connector control points to change the shape of the connector between the connected graphic elements.

### Related Topics

[Add Connectors Between Graphic Elements](#)

## Adding Connection Points

Use the Connection Point tool to place additional connection points at other locations on a graphic element than the bounding rectangle. Also, you can place custom connection points on an embedded graphic and connect to them.

---

**Note:** Custom connection points are part of the parent graphic element and cannot be grouped. Also, custom connection points added to a graphic element that is part of a Symbol Wizard layer are shown when the graphic element is part of the Symbol Wizard's current configuration.

---

Press the Esc key to cancel adding a connection point. Also, clicking on the Industrial Graphic Editor's canvas takes you out of connection point addition mode.

### To add connection points

1. Open a graphic that you want to add one or more connection points.
2. Click once on the **Connection Point** icon from the **Tools** pane to draw a single connection point on a graphic element.

If you want to draw multiple connection points, double click on the **Connection Point** icon.

3. Move your mouse to a location within a graphic element that you want to place a new connection point.

---

**Note:** Connection points can be added within the bounding rectangle of a hosting element of an embedded graphic.

---

4. Click once.

The new connection point appears as a green rectangle at the location you selected.

### To change the position of a connection point

You can change the position of a connection point that you added to a graphic element or a graphic.

1. Click on the connection point you want to move to select it.
2. Keep the left mouse key pressed.
3. Drag the connection point to a new location and release the mouse key.

The **X** and **Y** properties show the coordinate position of the connection point.

You can also change the location of a connection point you added by changing the X and Y coordinate values assigned to the connection point's **X** and **Y** properties.

## Related Topics

[Add Connectors Between Graphic Elements](#)

### Change Connector Properties

A connector includes a set of **Appearance**, **Line Style**, and **Runtime Behavior** properties. These properties can be modified during design time.

During run time, you can use animation to change property values that affect the appearance or behavior of a connector. For Angled or Straight connectors, you can use Line Style or Element Style animation. A connection point does not support any type of animation.

	Property	Description
<b>Appearance Properties</b>		
	ConnectionType	Type of connector (Angled or Straight). Angled is the default, which contains a set of connector line segments with 90 degree angles between them.
	ElementStyle	Element style applied to a connector to change the line color, fill, and pattern. Line styles can be applied to Angled and Straight types of connectors. None is the default.
	X	Horizontal coordinate of the left most border of a selected graphic element or graphic that has an attached connector. The X coordinate value is the number of pixels between the left vertical border of the Industrial Graphic Editor's canvas area to the left most border position of the selected graphic element or graphic.
	Y	Vertical coordinate of the top border of a selected graphic element or graphic that has an attached connector. The Y coordinate value is the number of pixels between the top horizontal border of the Industrial Graphic Editor's canvas area to the top of the selected graphic element or graphic.
	Start	Read-only X and Y coordinates of a connector's start point with respect to the origin at the top left corner of the Industrial Graphic Editor's canvas.
	End	Read-only X and Y coordinates of a connector's end point with respect to the origin at the top left corner of the Industrial Graphic Editor's canvas.

	Property	Description
<b>Line Style Properties</b>		
	LineWeight	Line weight of an Angled or Straight type of connector. 1 is the default.
	LinePattern	Line pattern of an Angled or Straight type of connector. Solid is the default.
	StartCap	Shape of the line start point of an Angled or Straight type of connector. Flat is the default.
	EndCap	Shape of line end point of an Angled or Straight type of connector. Flat is the default.
	LineColor	Line color of an Angled or Straight type of connector. Black is the default.
<b>Runtime Behavior Properties</b>		
	Enabled	Connector animation is enabled or disabled during run time. Enabled is the default.
	Visible	Connector is visible or hidden during run time. Visible is the default.

## Related Topics

[Add Connectors Between Graphic Elements](#)

[Change the Type of Connector](#)

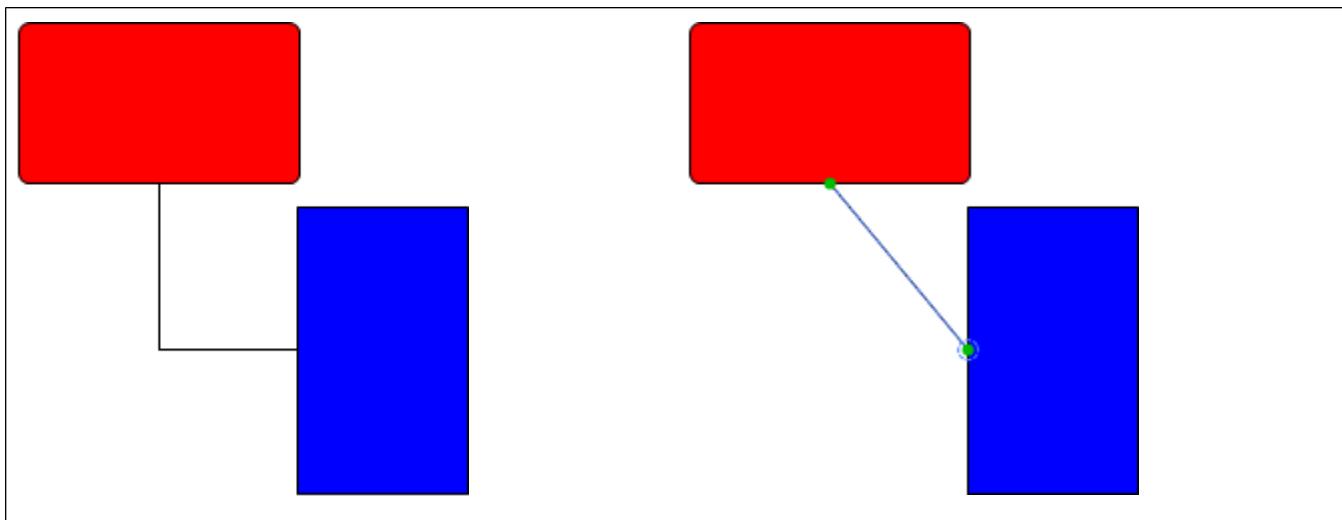
[Change the Length of a Connector](#)

[Change the Shape of a Connector](#)

# Change the Type of Connector

You can select the type of connector by setting an option for the **ConnectionType** property in the **Appearance** pane of the Industrial Graphic Editor.

The default connector type is Angled, which consists of horizontal and vertical lines with a 90 degree angle between them. A Straight connector is a straight line between the connection points on different graphic elements.



#### To change the type of connector

1. Click on a connector to select it.  
The **Connection Type** property appears in the **Appearance** pane of the Industrial Graphic Editor.
2. Select a connector type from the drop-down list of the **Connection** property.  
The appearance of the selected connector changes to the type you selected.

#### Related Topics

[Change Connector Properties](#)

## Change the Length of a Connector

You can change the length of a connector to move it to another connection point on a graphic element or detach it from a graphic element.

---

**Note:** A connector is not required to start or end at a connection point on a graphic element. Connectors can be drawn on the canvas detached from any graphic elements.

#### To change the length of a connector

1. Click on a connector to select it.  
The start point of a connector appears as a green circle. A halo appears around the end point.
  2. Select either the start or end point of the connector and keep your left mouse key pressed.
  3. Drag the start or end point of a connector to a new location and release the mouse key.  
The length of the connector changes until you release your mouse key. The **Start** or **End** properties show a new coordinate position based on whether you moved the connector's start or end point.
- You can also change the location of a connector's start or end points by changing the X and Y coordinate values assigned to the connector's **Start** or **End** properties.

## Related Topics

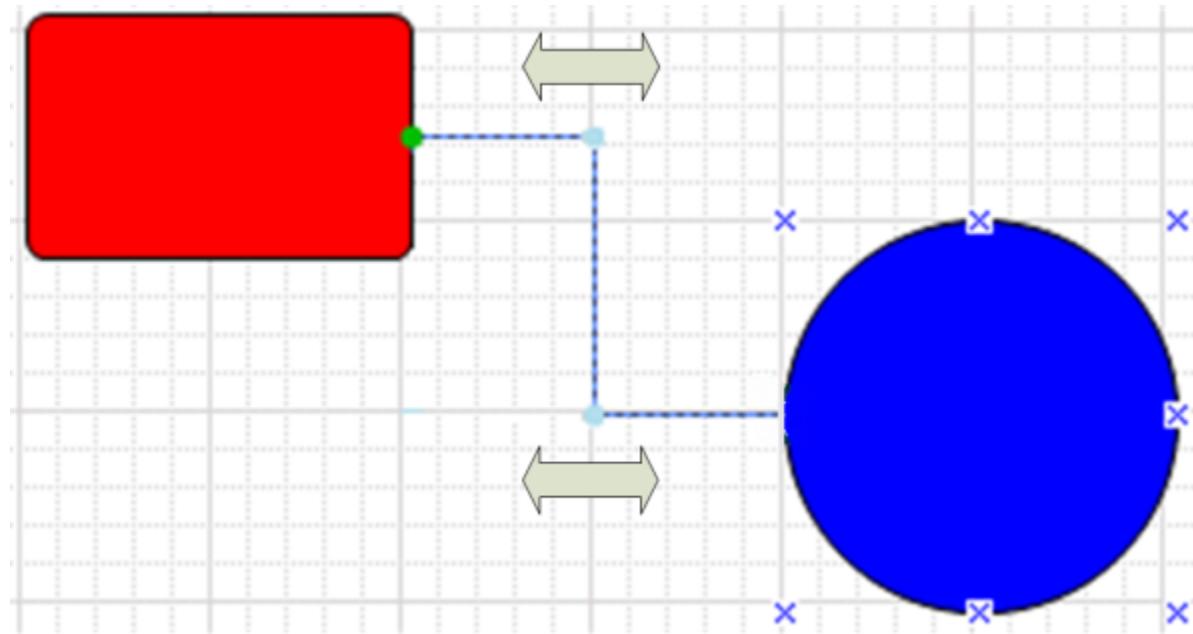
[Change Connector Properties](#)

# Change the Shape of a Connector

An Angled connector includes one or more control points that can be moved to change the shape of a connector.

The movement of control points is restricted to changing the shape of the connector without changing the fixed position of the connection points on graphic elements. For example, the two control points shown in the figure below are part of an Angled connector. Both control points can be moved horizontally to the same X coordinate position to change the position of the vertical line segment of the connector.

But, the line segments from the connection points of the graphic elements to their adjacent control points cannot be moved. In the following example, the control points cannot be moved vertically. To maintain the required right angles between line segments of an Angled connector would require the locations of the fixed connection points to be moved.



## To change the shape of a connector

1. Click on a connector to select it.  
A control point appears at the intersection point of each right angle in an Angled connector.
2. Click a control point and keep your mouse key pressed.  
The mouse cursor changes to a double arrow to indicate the control point can be moved.
3. Move the control point to change the shape of the connector and release the mouse key.

## Related Topics

[Change Connector Properties](#)

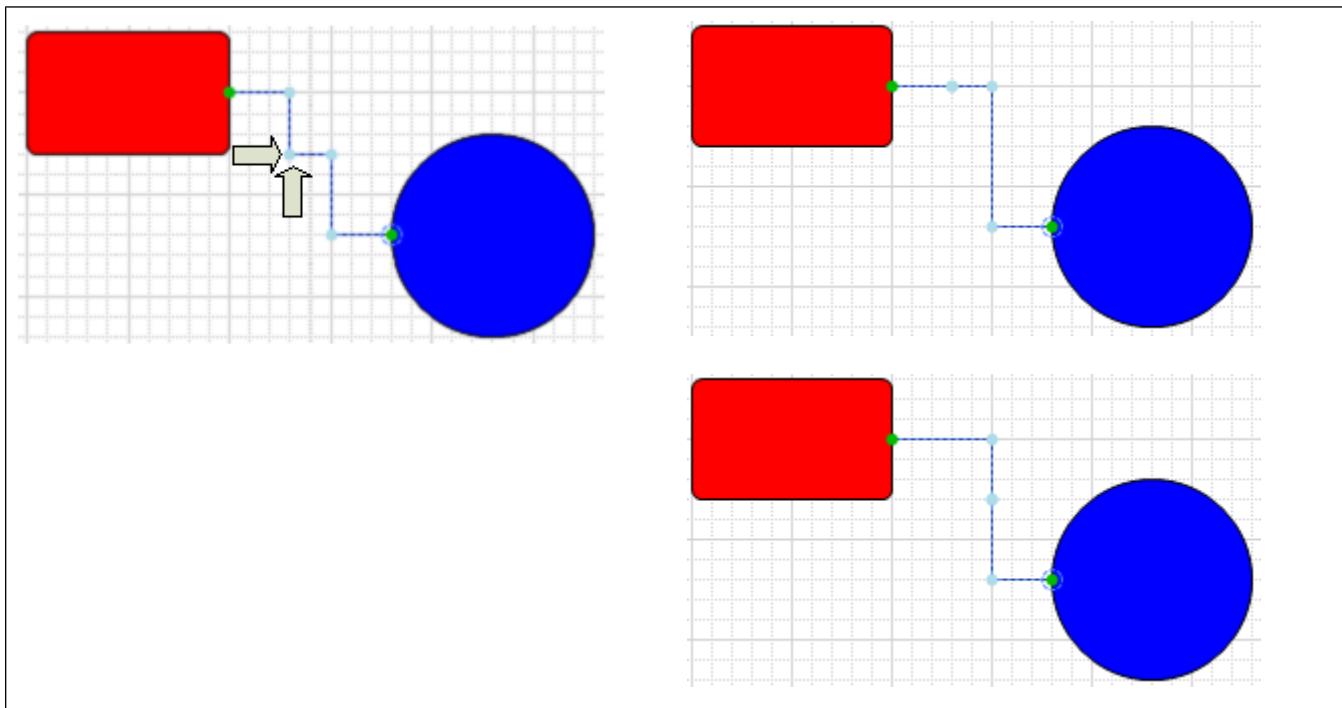
[Delete a Control Point](#)

## Delete a Control Point

### Delete a Control Point

If you want to remove a control point to change the shape of your connector, you must first reposition a line segment to ensure the resulting connector path contains only right angles before deleting a control point.

In the following example, a line segment needs to be repositioned vertically or horizontally to ensure the connector contains only right angles. After a line segment is repositioned, a control point can be deleted.



#### To delete a control point

1. Click on a connector to select it.
2. Move a line segment within the connector to ensure the connector path contains only right angles between its line segments.
3. With your Ctrl key pressed, place your cursor over the control point on the connector you want to delete.

The appearance of the cursor changes to a pen tip with a minus sign to indicate that a control point can be deleted from a connector.

---

**Note:** Control points at the right angles of a connector cannot be deleted. You can only remove control points on straight line segments.

---

4. Left click with your mouse to delete the control point.

The small blue circle on the connector disappears indicating the control point is deleted from the connector.

## Related Topics

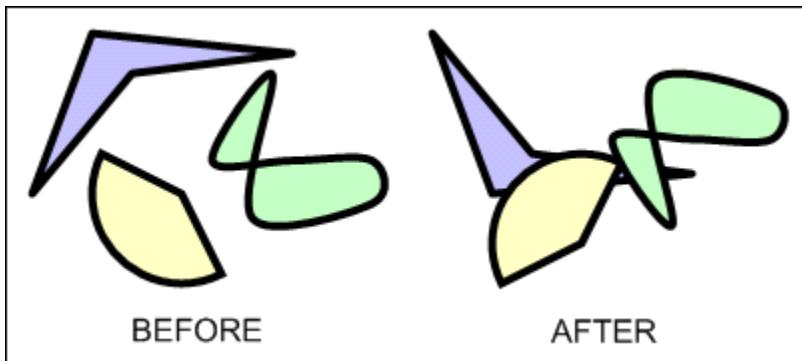
[Change the Shape of a Connector](#)

## Flipping Elements

You can flip elements on their horizontal or vertical axes. The axis for each element is determined by its point of origin. For more information on how to change the point of origin, see [Moving the Origin of an Element](#).

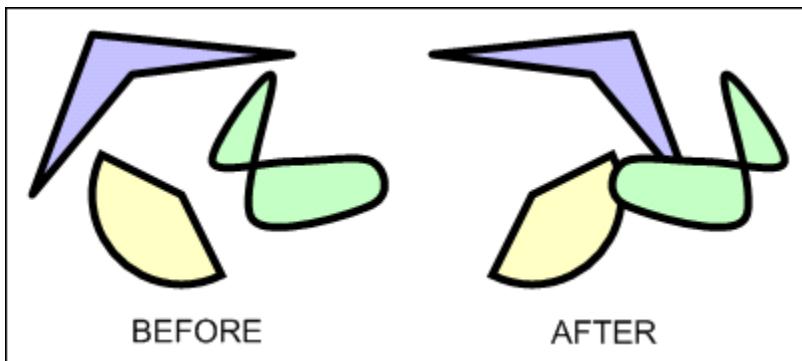
### To flip elements vertically

1. Select one or more elements.
2. On the **Arrange** menu, point to **Transform**, and then click **Flip Vertical**. The selected elements are flipped vertically on their horizontal axis.



### To flip elements horizontally

1. Select one or more elements.
2. On the **Arrange** menu, point to **Transform**, and then click **Flip Horizontal**. The selected elements are flipped horizontally on their vertical axis.



## Related Topics

[Working with Graphic Elements](#)

## Locking and Unlocking Elements

When you lock elements, they cannot be:

- Moved.
- Resized.
- Rotated.
- Aligned.
- Flipped.

You also cannot change the point of origin in locked elements. To enable these functions again, you need to unlock the elements.

### To lock elements

1. Select all elements that you want to lock.
2. Do one of the following:
  - On the **Arrange** menu, click **Lock**.
  - In the Properties Editor, set the Locked property to **True**.The selected elements appear with lock icons at their handles.

### To unlock elements

1. Select all elements that you want to unlock.
2. Do one of the following:
  - On the **Arrange** menu, click **Unlock**.
  - In the Properties Editor, set the Locked property to False.The lock icons disappear from the handles of the selected elements.

## Related Topics

[Working with Graphic Elements](#)

## Making Changes Using Undo and Redo

Use the Undo function to reverse an editing change you made to a graphic element in the Industrial Graphic Editor. After you undo a change, you can also restore the change by using the Redo function. The Undo and Redo functions appear as icons on the Industrial Graphic Editor's **Edit** menu.



You can undo one single change, or any number of changes that you have previously made. You can also redo any number of changes. These can be selected from a list.

### To undo a single change

- Do one of the following:

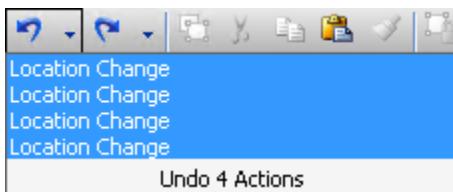
- Press Ctrl + Z.
- On the **Edit** menu, click **Undo**.

### To redo a single change

- Do one of the following:
  - Press Ctrl + Y.
  - On the **Edit** menu, click **Redo**.

### To undo a specified number of previous changes

1. On the toolbar, click the **Undo** icon. The **Undo** list appears with one or more descriptions of what changes were made.



2. Select a change from the list or click the bottom list entry to undo all changes. The changes up to and including the selected item are undone.

### To redo a specified number of previously undone changes

1. On the toolbar, click the **Redo** icon. The **Redo** list appears with a description of what the undone changes were.
2. Select a the change from the list or click the bottom list entry to redo all changes. The changes down to and including the selected item are redone.

## Related Topics

[Working with Graphic Elements](#)

## Working with Groups of Elements

You can group together multiple elements. This is useful to bind certain element together so that they are not inadvertently moved. The group is treated as a new element.

You can:

- Create a group from one or more elements.
- Ungroup the elements without losing their original configuration information.
- Add more elements to an existing group.
- Remove elements from a group.
- Edit the elements of a group without having to ungroup them.

## Related Topics

- [Working with Graphic Elements](#)
- [Creating a Group of Elements](#)
- [Ungrouping](#)
- [Adding Elements to Existing Groups](#)
- [Removing Elements from Groups](#)
- [Editing Components within a Group](#)

### Creating a Group of Elements

After you create elements, you can group them. Grouping elements lets you manage the elements as one unit. Groups are assigned default names when you create them, such as Group1, Group2, and so on. After you create a group, you can rename it.

Groups can have properties that are different than the properties of the elements.

#### To create a group

1. Select the elements you want as part of the new group.
2. On the **Arrange** menu, point to **Grouping**, and then click **Group**. The elements are combined into a group. The group is listed in the Elements List.
3. Rename the group as required. To do this:
  - a. In the Elements List, click the group name and click again. The group name is in edit mode.
  - b. Type a new name and click Enter. The group is renamed.
  - c. You can also rename a group or elements by changing the Name property in the Properties Editor.

## Related Topics

- [Working with Groups of Elements](#)

### Ungrouping

After you create a group, you can ungroup it if you no longer want it.

If the group included elements and other groups, when you ungroup, the original elements and groups again exist as independent items. To ungroup any subgroups, select each one and ungroup it separately.

If you ungroup a set of elements and elements already exist with the names of the grouped elements, then the newly ungrouped elements are renamed.

#### To ungroup

1. Select the groups you want to ungroup.
2. On the **Arrange** menu, point to **Grouping**, and then click **Ungroup**. The groups are converted to the original elements. The group name is removed from the Elements List and the element names appear.

## Related Topics

[Working with Groups of Elements](#)

### Adding Elements to Existing Groups

After you create a group, you can add elements or other groups to an existing group.

For example, you can combine a group that represents a valve with another group that represents a tank to create a new group that can be called a tank unit.

You can add:

- Elements to groups.
- Groups to the primary selected group.

#### To add elements to an existing group

- On the canvas, select the group and also elements and groups that you want to add.
- Right-click a selected element or on the group, point to **Grouping**, and then click **Add to Group**. The selected elements are added to the group.

---

**Note:** You can also add elements to existing groups by using the Elements List in similar way.

---

## Related Topics

[Working with Groups of Elements](#)

### Removing Elements from Groups

After you create a group, you can remove elements from the group. This lets you remove one or more elements you no longer want in that group.

Removing elements from the group removes them from the canvas. It also removes any scripts or animations you added to the element.

#### To remove an element from a group

1. On the canvas, select the group with the elements that you want to remove.
2. Click the group again to enter inline editing mode.
3. Select the elements that you want to remove from the group.
4. Right-click a selected elements, point to **Grouping**, and then click **Remove from Group**. The selected elements are removed from the group.

---

**Note:** You can also remove elements from existing groups by using the Elements List in similar way.

---

## Related Topics

[Working with Groups of Elements](#)

## Editing Components within a Group

You can edit components within a group without having to dissolve the group. Do this by:

- Selecting the element in Elements List.
- Using the **Edit Group** command on the shortcut menu.
- Slowly double-clicking to enter inline editing mode.

### To edit components within a group by using the Elements List

1. In the Elements List, expand the group that contains the element that you want to edit.
2. Select the element that you want to edit. The element appears selected in the group and the group is outlined with a diagonal pattern.
3. Edit the element with the Properties Editor, by mouse or by menu according to your requirements.
4. Click outside the group.

### To edit components within a group by using the Edit Group command

1. On the canvas, select the group that you want to edit.
2. On the menu **Edit**, click **Edit Group 'GroupName'**. The group is outlined with a diagonal pattern.
3. Select the element that you want to edit.
4. Edit the element with the Properties Editor, by mouse, by menu or pop-up menu according to your requirements.
5. Click outside the group.

---

**Note:** If you move the position of an element in a group outside the group, the group size is automatically changed to incorporate the new position of the element.

---

## Related Topics

[Working with Groups of Elements](#)

## Using Path Graphics

You can join a set of open elements, such as lines, to create a new closed element. The new closed element is called a path graphic.

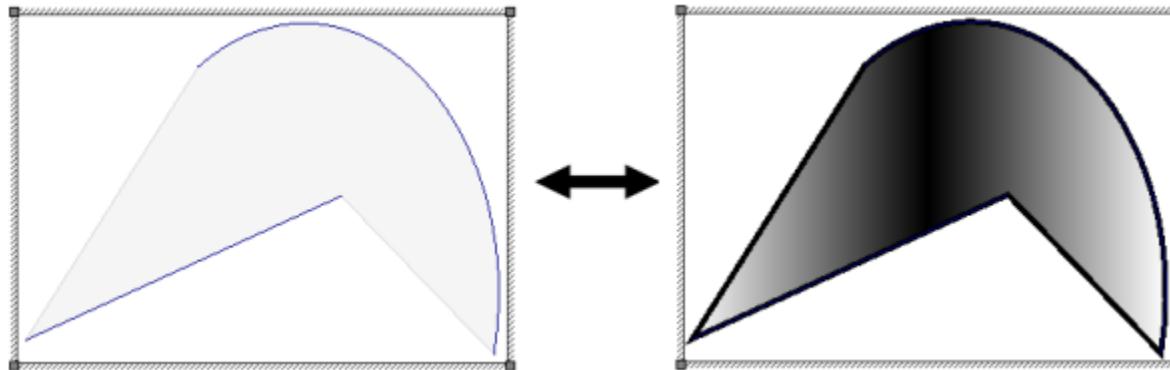
You can:

- Create a path graphic by joining open elements.
- Break the path graphic into its elements.
- Edit the path graphic in its entirety or by editing its elements.
- Add new elements to the path graphic.
- Remove elements from the path graphic.

You can view a path graphic in two modes:

- Element mode shows you the individual elements contained in the path graphic and determine its shape. Elements that make up the path graphic are shown as blue lines. The points where the elements are connected are shown as grey lines.
- Path mode shows you the path graphic in its final rendering, including fill styles and lines styles.

When you are in inline editing mode, you can switch between both modes by pressing the space bar. This lets you preview the path graphic without leaving the inline editing mode.



## Related Topics

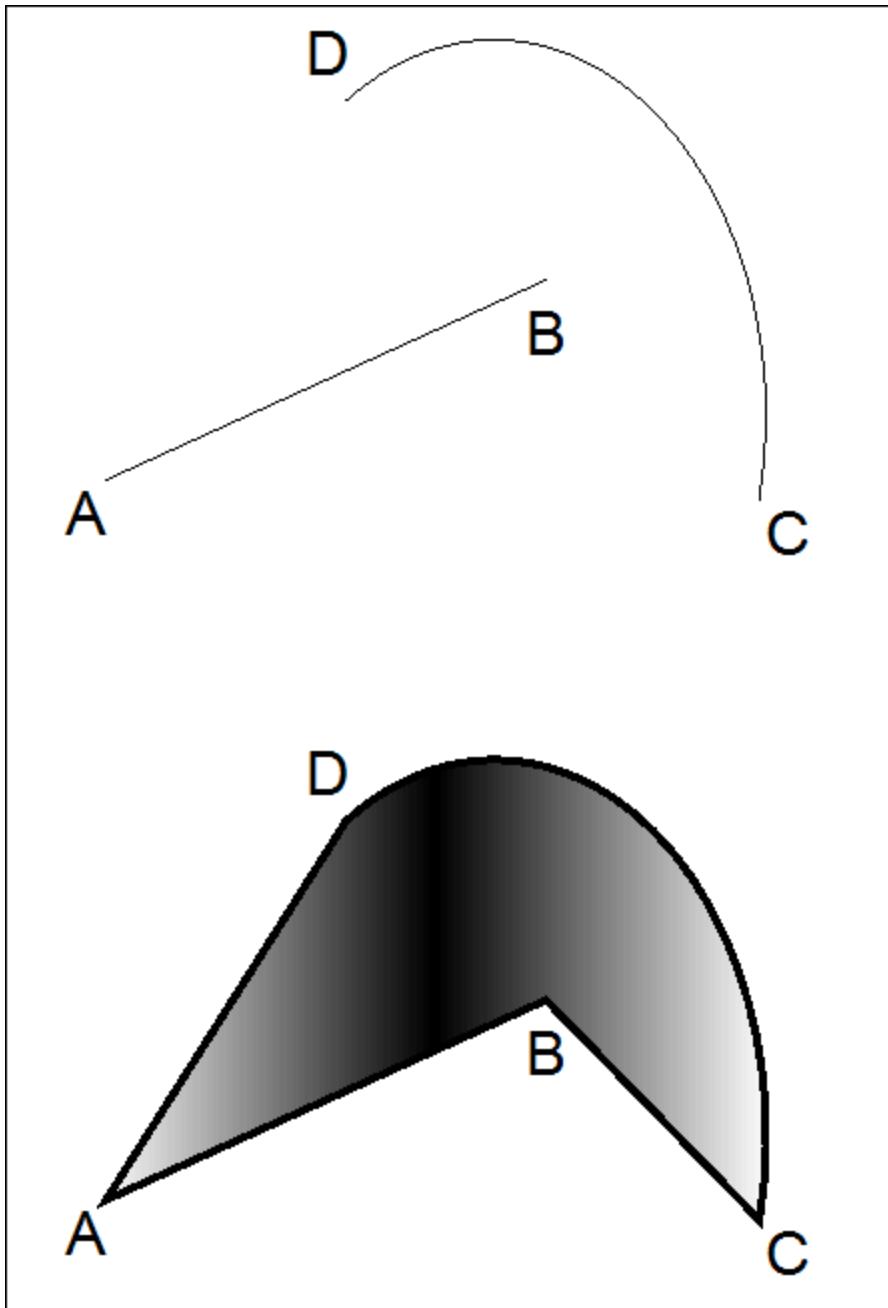
- [Working with Graphic Elements](#)
- [Creating a Path Graphic](#)
- [Breaking the Path of a Path Graphic](#)
- [Changing a Path Graphic](#)
- [Adding Elements to an Existing Path Graphic](#)

## Creating a Path Graphic

You can create a path graphic from one or more open elements such as lines, polylines, curves, and arcs.

The path graphic is created according to the start and end points and the z-order of the open elements that create it.

For example, if you draw a line from point A to point B, then an arc from point C to point D, and then join these elements in a path graphic, the path graphic is described by a straight edge from points A to B, a straight edge from points B to C, a curved edge from points C to D, and closed by a straight edge from points D to A.



**Note:** If the Path Graphic doesn't appear as you expected after you create it, then you can swap the end points or change the z-order of one or more elements. For more information, see and [Changing the Z-order of an Element in a Path Graphic](#).

#### To create a path graphic

1. Select one or more open elements.
2. On the **Arrange** menu, point to **Path**, and then click **Combine**. A new path graphic is created from the selected open elements.

## Related Topics

[Using Path Graphics](#)

### Breaking the Path of a Path Graphic

You can break the path of a path graphic so that it is broken into its individual open elements. When you do so, the path graphic loses its unique properties such as fill style and line style.

#### To break the path of a path graphic

1. Select one or more path graphics.
2. On the **Arrange** menu, point to **Path**, and then click **Break**.

## Related Topics

[Using Path Graphics](#)

### Changing a Path Graphic

You can edit an existing path graphic on the canvas by accessing the individual elements of which it consists. For each individual element, you can:

- Move.
- Rotate.
- Change size.
- Change start and sweep angles if the elements are arcs.
- Change control points if the elements are curves or polylines.
- Swap the end points of an element in a path graphic.
- Change the z-order or the elements in a path graphic.

The path graphic is updated while you edit the individual elements.

## Related Topics

[Using Path Graphics](#)

[Moving Elements in a Path Graphic](#)

[Resizing Elements in a Path Graphic](#)

[Editing Start and Sweep Angles of Elements in a Path Graphic](#)

[Editing Element Control Points in a Path Graphic](#)

[Swapping the End Points of an Element in a Path Graphic](#)

[Changing the Z-order of an Element in a Path Graphic](#)

# Moving Elements in a Path Graphic

You can move elements in a path graphic. If you move an element outside of the path graphic boundary, the boundary is redrawn to include the moved element.

## To move an element within a path graphic

1. Select the path graphic you want to edit.
2. Do one of the following:
  - a. On the **Edit** menu, click **Edit Path**.
  - b. Slowly double-click the path graphic.
- The path graphic appears in element mode.
3. Select the individual element within the path graphic you want to move. You can also do this by selecting the element in the Elements List.
4. Click a solid part of the element and drag it to the new position. The element is moved.
5. Click outside the path graphic on the canvas. The path graphic is shown in path mode.

## Related Topics

[Changing a Path Graphic](#)

# Resizing Elements in a Path Graphic

You can resize elements in a path graphics. If you resize an element outside of the path graphic boundary, the boundary is redrawn to include the resized element.

## To resize an element within a path graphic

1. Select the path graphic you want to edit.
2. Do one of the following:
  - a. On the **Edit** menu, click **Edit Path**.
  - b. Slowly double-click the path graphic.
- The path graphic appears in element mode.
3. Select the individual element within the path graphic you want to resize. You can also do this by selecting the element in the Elements List.
4. Click and drag any of the resize handles of the selected element. The element is resized.
5. Click outside the path graphic on the canvas. The path graphic is shown in path mode.

## Related Topics

[Changing a Path Graphic](#)

# Editing Start and Sweep Angles of Elements in a Path Graphic

If your path graphic contains arcs, you can edit the start and sweep angles of these elements. If changing the angle of an element causes it to overlap the path graphic boundary, the boundary is redrawn to include the changed element.

## To edit start or sweep angle of an element within a path graphic

1. Select the path graphic you want to edit.
2. Do one of the following:
  - a. On the **Edit** menu, click **Edit Path**.
  - b. Slowly double-click the path graphic.

The path graphic appears in element mode.
3. Select the individual element within the path graphic for which you want to change the start or sweep angle. You can also do this by selecting the element in the Elements List.
4. Click the element again. The element appears in edit mode with its start angle and sweep angle.
5. Click outside the path graphic on the canvas. The path graphic is shown in path mode.

## Related Topics

[Changing a Path Graphic](#)

# Editing Element Control Points in a Path Graphic

If your path graphic contains curves or polylines, you can edit the control points of these elements. If changing the control points of the element causes it to overlap the path graphic boundary, the boundary is redrawn to include the changed element.

## To edit control points of an element within a path graphic

1. Select the path graphic you want to edit.
2. Do one of the following:
  - a. On the **Edit** menu, click **Edit Path**.
  - b. Slowly double-click the path graphic.

The path graphic appears in element mode.
3. Select the curve or polyline element within the path graphic for which you want to change the control points. You can also do this by selecting the element in the Elements List.
4. Click the element again. The element appears in edit mode with its control points.
5. Drag any of the control points to shape the curve or polyline.
6. Click outside the path graphic on the canvas. The path graphic is shown in path mode.

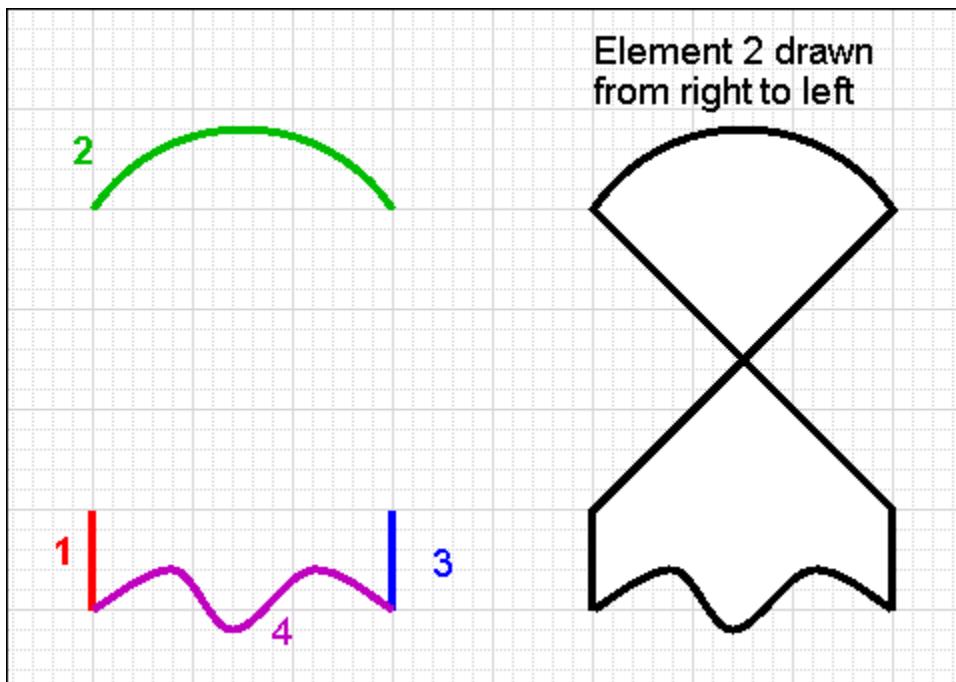
## Related Topics

[Changing a Path Graphic](#)

# Swapping the End Points of an Element in a Path Graphic

The path graphic is created by following the direction in which you draw its elements.

If a path graphic does not appear as expected, this can be caused by drawing an element in a different direction as intended. You can see this if one of the path graphic edges appears crossed over when connecting to the previous and next element.



You can fix this by swapping the end points of the element where this appears.

### To swap the end points of an element within a path graphic

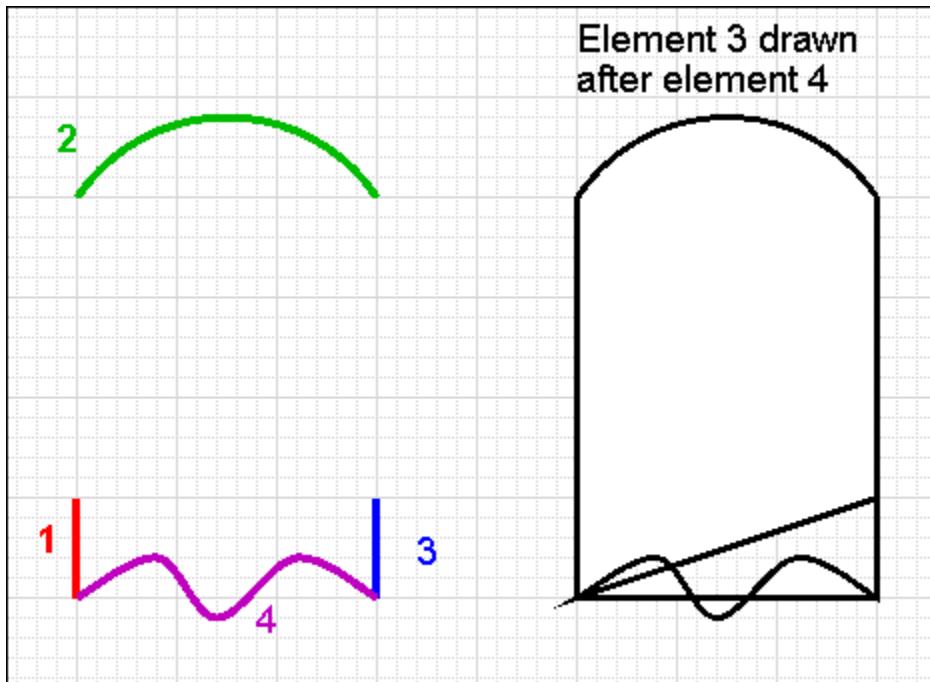
1. Select the path graphic you want to edit.
2. Do one of the following:
  - a. On the **Edit** menu, click **Edit Path**.
  - b. Slowly double-click the path graphic.  
The path graphic appears in element mode.
3. Select the individual element within the path graphic for which you want to swap the end points. You can also do this by selecting the element in the Elements List.
4. Right-click that element and select **Path, Swap End Points** on the context menu. The end points of the selected element are swapped and the path graphic is updated accordingly.
5. Click outside the path graphic on the canvas. The path graphic is shown in path mode.

## Related Topics

[Changing a Path Graphic](#)

# Changing the Z-order of an Element in a Path Graphic

If a path graphic does not appear as expected, this can be caused by drawing an element in a different z-order as intended. You can see this if one of the path graphic edges jumps across the path graphic area.



You can fix this by changing the z-order of the element where this appears.

**Note:** The z-order of elements in a path graphic is only applicable within the path graphic.

### To change the z-order of an element within a path graphic

1. Select the path graphic you want to edit.
2. Do one of the following:
  - a. On the **Edit** menu, click **Edit Path**.
  - b. Slowly double-click the path graphic.  
The path graphic appears in element mode.
3. Select the individual element within the path graphic for which you want to change the z-order. You can also do this by selecting the element in the Elements List.
- Note:** You can see the elements in their z-order in the Elements List. Alternatively, you can select one from the Elements List and change its z-order.
4. On the **Arrange** menu, point to **Order**, and then click:

- **Send To Back** to send the element to the back of the set of elements of the path graphic.
  - **Send Backward** to send the element one order backward.
  - **Sent To Front** to send the element to the front of the set of elements of the path graphic.
  - **Send Forward** to send the element one order forward.
5. Click outside the path graphic on the canvas. The path graphic is shown in path mode.

## Related Topics

[Changing a Path Graphic](#)

### Adding Elements to an Existing Path Graphic

You can easily add elements to an existing path graphic. You can add:

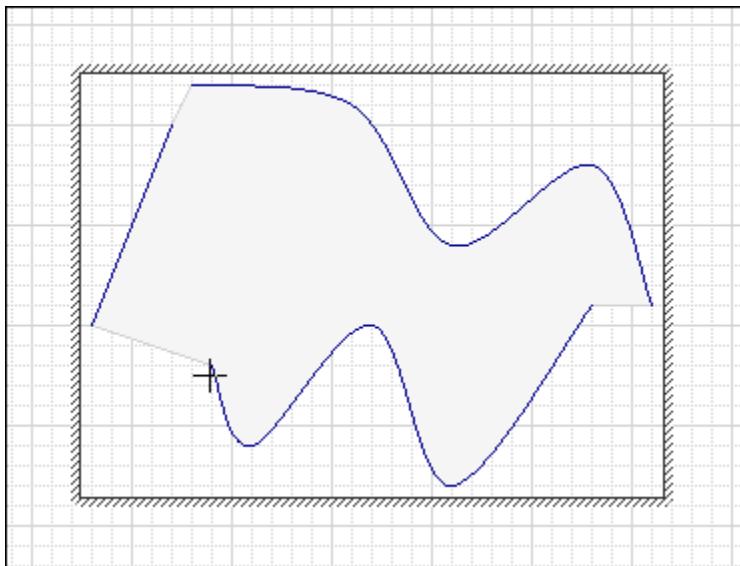
- New elements, which you draw while the path graphic is in edit mode.
- Existing elements, which are already on the canvas.

You can only add open elements such as lines, polylines, curves, and arcs to an existing path graphic.

You can only set the origin of a new element within the frame of the existing path graphic. If you click anywhere outside the path graphic, the edit mode is exited and the element you are drawing is a new element.

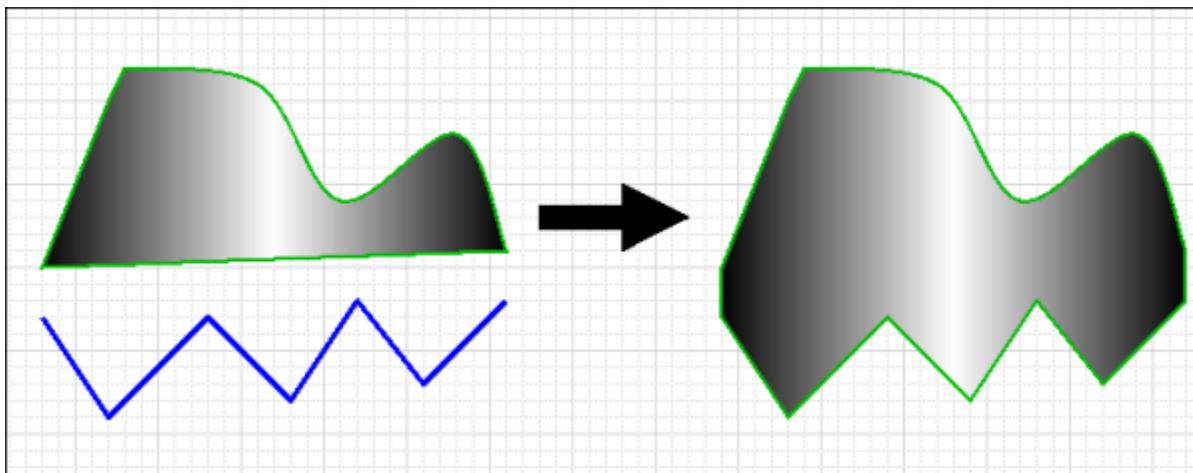
#### To add new elements to an existing path graphic

1. Select the path graphic to which you want to add a new element.
2. On the **Edit** menu, click **Edit Path**. The path graphic appears in element mode.
3. Select the new element you want to add from the Tools panel.
4. Draw the element as you would normally. While you are drawing the element, the path graphic is updated.



### To add existing elements to an existing path graphic

1. Select the path graphic and all elements that you want to add to the path graphic.
2. Right-click a solid part of a selected element, point to **Path**, and then click **Add To Path**. The selected elements are added to the selected path graphic.



### Related Topics

[Using Path Graphics](#)

### Removing Elements from a Path Graphic

You can remove individual elements from a path graphic. The elements are not deleted, but appear outside the path graphic.

You cannot remove the last element of a path graphic.

### To remove elements from a path graphic

1. Select the path graphic from which you want to delete individual elements.
2. On the **Edit** menu, click **Edit Path**. The path graphic appears in element mode.
3. Shift + click one or more elements to remove.

---

**Note:** You can also select the elements to remove from the Elements List by holding CTRL key during the selection.

---

4. Right-click any selected element, point to **Path**, and then click **Remove From Path**. The selected element is removed from the path graphic and the path graphic is updated accordingly.

### Related Topics

[Using Path Graphics](#)

## Editing Common Properties of Elements and Graphics

### In This Chapter

- [Editing the Name of an Element](#)
- [Editing the Fill Properties of an Element](#)
- [Editing the Line Properties of an Element](#)
- [Setting the Text Properties of an Element](#)
- [Setting Style](#)
- [Setting the Transparency Level of an Element](#)
- [Adjusting the Colors and Transparency of a Gradient](#)
- [Enabling and Disabling Elements for Run-Time Interaction](#)
- [Changing the Visibility of Elements](#)
- [Editing the Tab Order of an Element](#)
- [Using the Format Painter to Format Elements](#)
- [Editing the General Properties of a Graphic](#)

### Editing the Name of an Element

Some properties are common to most types of elements, such as fill, line styles, and visibility. You can:

- Edit the name of an element.
- Edit the fill properties of an element.
- Edit the line properties of an element.
- Edit the text properties of an element.
- Set the style.
- Set the transparency level of an element.
- Adjusting colors and style for an element's gradient style.
- Enable and disable elements for run-time interaction.
- Change the visibility of an element.
- Change the tab order of an element.
- Use the Format Painter to format elements.
- Edit the general properties of a graphic.

The name of an element uniquely identifies the element on the drawing surface.

When you draw a new element on the drawing surface, it is assigned a default name. You can then change its name in the Properties Editor or the Elements List.

Element names are case-insensitive and unique within the same element hierarchy. It is possible to have two elements with the same name if one is, for example, in a group and the other outside that group.

### To change an element's name in the Properties Editor

1. Select the element on the drawing surface.
2. In the Properties Editor, click the value for the **Name** box.
3. Type a new name and click Enter.

### To change an element's name in the Elements List

1. Select the element in the Elements List.
2. Click the element in the Elements List again.
3. Type a new name and click Enter.

## Related Topics

[Editing Common Properties of Elements and Graphics](#)

### Editing the Fill Properties of an Element

You can configure the following fill properties for an element:

- Fill style as solid color, gradient, pattern or texture
- Unfilled style
- Fill orientation, relative to the element or to the screen
- Fill behavior, which determines if the object is to be filled horizontally, vertically, or both
- Horizontal fill direction
- Vertical fill direction
- Percent of horizontal fill
- Percent of vertical fill

## Related Topics

[Editing Common Properties of Elements and Graphics](#)

[Setting Fill Style](#)

[Setting Unfilled Style](#)

[Setting Fill Orientation](#)

[Setting Fill Behavior](#)

[Setting Horizontal Fill Direction and Percentage](#)

[Setting Vertical Fill Direction and Percentage](#)

### Setting Fill Style

You can configure the fill style of one or more elements. You can do this to:

- Selected elements on the toolbar.
- Style properties in the Properties Editor.
- Nested style properties, such as just one color of a multi-colored gradient.

### To configure the fill style of an element with the toolbar

1. Select one or more elements you want to configure.
2. On the toolbar, click the down arrow to the right of the **Fill Color** icon. The fill style list appears.
3. Configure the fill color. Do any of the following:
  - Click **No Fill** to configure an empty element.
  - Click a predefined solid color in the display.
  - Click **More Solid Colors** to open the style selection dialog box and select a solid color.
  - Click **Color Picker** to select a color from the screen.
4. Configure the fill gradient, pattern, or texture. Do any of the following:
  - Click a predefined gradient.
  - Click **More Gradients** to open the style selection dialog box and configure a gradient.
  - Click **Patterns** to open the style selection dialog box and select a pattern.
  - Click **Textures** to open the style selection dialog box and select a texture.

For more information about the style selection dialog box, see [Setting Style](#).

### To configure the fill style by setting style properties

1. Select one or more elements.
2. In the Properties Editor, locate the **FillStyle** property.
3. Click the browse button to open the style selection dialog box. For more information about the style selection dialog box, see [Setting Style](#).

### To configure the fill style by setting gradient color style properties

1. Select one or more elements with gradient fill style.
2. In the Properties Editor, locate the **Color1**, **Color2**, and **Color3** properties.
3. Click the browse button for any of these to set the selected gradient color from the style selection dialog box. For more information, see [Setting Style](#).

## Related Topics

[Editing the Fill Properties of an Element](#)

### Setting Unfilled Style

You can configure an element's unfilled style. The unfilled style of an element determines the element's unfilled portion at design time and run time.

## To configure the unfilled style of an element

1. Select one or more elements.
2. In the Properties Editor, click **UnfilledStyle**.
3. Click the browse button in the **UnfilledStyle** line. The style selection dialog box appears.
4. Select a solid color, gradient, pattern, or texture. For more information about the style selection dialog box, see [Setting Style](#).
5. Click **OK**.

## Related Topics

[Editing the Fill Properties of an Element](#)

### Setting Fill Orientation

You can configure an element's fill orientation in the Properties Editor. The fill orientation property determines if the fill style is relative to the screen or element.

- If relative to the screen, the gradient, pattern, or texture does not rotate with the element.
- If relative to the element, the gradient, pattern, or texture rotates with the element.

## To configure an element's fill orientation

1. Select one or more elements you want to configure.
2. In the Properties Editor, click **FillOrientation**.
3. From the list in the same line, click **RelativeToScreen** or **RelativeToGraphic**.

## Related Topics

[Editing the Fill Properties of an Element](#)

### Setting Fill Behavior

You can set the fill behavior of an element. The fill can be:

- Horizontal.
- Vertical.
- Both horizontal and vertical.

## To set an element's fill behavior

1. Select one or more elements you want to configure.
2. In the Properties Editor, set the property **FillBehavior** to one of the following:
  - **Horizontal**

- Vertical
- Both

## Related Topics

[Editing the Fill Properties of an Element](#)

### Setting Horizontal Fill Direction and Percentage

An element can fill:

- From left to right.
- From right to left.

You can also set the amount you want the element to be horizontally filled by as a percentage.

#### To set an element's horizontal fill direction and percentage

1. Select one or more elements you want to configure.
2. In the Properties Editor, set the **HorizontalDirection** property to:
  - **Right** to fill from left to right.
  - **Left** to fill from right to left.
3. For the **HorizontalPercentFill** property, type a percentage (0 - 100) in the value box.

## Related Topics

[Editing the Fill Properties of an Element](#)

### Setting Vertical Fill Direction and Percentage

An element can fill:

- From bottom to top.
- From top to bottom.

You can also set the amount you want the element to be vertically filled by as a percentage.

#### To set an element's vertical fill direction and percentage

1. Select one or more elements you want to configure.
2. In the Properties Editor, set the **VerticalDirection** property to:
  - **Top** to fill from bottom to top.
  - **Bottom** to fill from top to bottom.
3. For the **VerticalPercentFill** property, type a percentage (0 - 100) in the value box.

## Related Topics

[Editing the Fill Properties of an Element](#)

### Editing the Line Properties of an Element

You can set the line properties for any element that contains lines, such as:

- Lines and polylines.
- Rectangles, rounded rectangles, and ellipses.
- Curves, closed curves, and polygons.
- Arcs, pies, and chords.
- Text boxes.

You can set the:

- Start and end points for lines, arcs, and H/V lines.
- Line weight, which is the thickness of a line.
- Line pattern, which is the continuity of a line. For example, a continuous line, a dotted line, a dashed line, or a combination.
- Line style, which is the fill style of a line.
- Shape and size of the end points of a line. For more information, see [Setting Line End Shape and Size](#).

---

**Note:** You can also set the element's line properties in the **Line Format** properties group in the Properties Editor.

---

## Related Topics

[Editing Common Properties of Elements and Graphics](#)

[Setting Start or End Points of a Line](#)

[Setting the Line Weight](#)

[Setting the Line Pattern](#)

[Setting the Line Style](#)

### Setting Start or End Points of a Line

After you draw a line or H/V line, you can change its start or end points in the Properties Editor.

#### To set the line or H/V line start or end point

1. Select a line or H/V line.
2. In the Properties Editor, type coordinate values X, Y for the **Start** or **End** properties.

## Related Topics

[Editing the Line Properties of an Element](#)

### Setting the Line Weight

You can set a line weight from 0 pixels to 255 pixels for any element that contains lines. You can set the line weight using the **Format** menu, the toolbar, or the **LineWeight** property in the Properties Editor.

---

**Note:** Large line weight settings can cause unexpected behavior, especially with curves and line end styles.

#### To set the line weight using the Format menu

1. Select one or more elements.
2. On the **Format** menu, click **Line Weight**.
3. To use a predefined line weight, select it from the list.
4. To use another line weight, click **More Line Options**. The **Select Line Options** dialog box appears. In the **Weight** box, type a new line weight from 0 to 255 and then click **OK**.

## Related Topics

[Editing the Line Properties of an Element](#)

### Setting the Line Pattern

You can set the line pattern for any element that contains lines. The line pattern specifies the continuity of a line (continuous, dotted, dashed) and not its fill properties.

#### To set the line pattern

1. Select one or more elements.
2. On the **Format** menu, click **Line Pattern**.
3. To use a predefined line pattern, select it from the list.
4. To use another line pattern, click **More Line Options**. The **Select Line Options** dialog box appears. In the **Pattern** list, select a pattern, and then click **OK**.

---

**Note:** You can also set the line pattern by changing the **LinePattern** property in the Properties Editor.

## Related Topics

[Editing the Line Properties of an Element](#)

### Setting the Line Style

You can set the line style for any element that contains lines. Setting the line style is similar to setting the fill style. You can also set the solid color, gradient, pattern, and texture for a line.

## To set the line style

1. Select one or more elements.
2. On the toolbar, click the **Line Color** icon. The line style list appears.
3. Configure the line color. Do any of the following:
  - Click a predefined solid color in the display.
  - Click **More Solid Colors** to open the style selection dialog box and select a solid color.
  - Click **Color Picker** to select a color from the screen.
4. Configure the line gradient, pattern, or texture. Do any of the following:
  - Click a predefined gradient.
  - Click **More Gradients** to open the style selection dialog box and configure a gradient.
  - Click **Patterns** to open the style selection dialog box and select a pattern.
  - Click **Textures** to open the style selection dialog box and select a texture.

For more information about the style selection dialog box, see [Setting Style](#).

---

**Note:** You can also set the element's line style in the Properties Editor. If you do this, you can configure the solid color, gradient, pattern, or texture in the style selection dialog box. For more information, see [Setting Style](#).

---

## Related Topics

[Editing the Line Properties of an Element](#)

## Setting the Text Properties of an Element

You can set the following for text, text box, and button elements:

- The text that appears
- The format in which the text appears
- The font of the text
- The alignment of the text
- The text style

You can also substitute strings in text, text box, and button elements.

## Related Topics

[Editing Common Properties of Elements and Graphics](#)

[Setting the Displayed Text](#)

[Setting the Text Display Format](#)

[Setting the Text Font](#)

[Setting the Text Color](#)

[Setting the Text Alignment](#)

## Substituting Strings

### Setting the Displayed Text

You can set the text of a text element, text box, or button in the canvas or by changing the Text property in the Properties Editor.

#### To set the text to display

1. Select the text element, text box or button on the canvas.
2. On the **Edit** menu, click **Edit Text**. The selected element appears in edit mode.
3. Type a text string and press **Enter**.

### Related Topics

[Setting the Text Properties of an Element](#)

### Setting the Text Display Format

You can configure how values are shown for the text in a text box or button. For example, as a rounded float with the format #.###.

You can format the text display for the:

- Text element and the button element in the same way as in the HMI or with the TextFormat property in the Properties Editor.
- Text box element only with the **TextFormat** property.

#### To set the text display format

1. Select a text element, text box, or button.
2. In the **Properties Editor**, type a format for the **TextFormat** property.

### Related Topics

[Setting the Text Properties of an Element](#)

### Setting the Text Font

You can change the font style and font size of a text using:

- The **Format** menu.
- The **Font** property in the **Properties Editor**.
- Lists on the toolbar.

## To set the text font, font style, and size

1. Select a text element, a text box, or a button element on the canvas.
2. On the **Format** menu, click **Fonts**. The **Font** dialog box appears.
3. Set the font, font style, size, and effects.
4. Click **OK**.

## Related Topics

[Setting the Text Properties of an Element](#)

### Setting the Text Color

You can set the text color as a solid color, a gradient, a pattern, or a texture.

**Note:** You can also change the text color in the **Properties Editor** with the **TextColor** property.

#### To set the text color

1. Select a text element, a text box, or a button element on the canvas.
2. Click the **Text Color** icon.
3. Configure the text color. Do any of the following:
  - Click a predefined solid color in the display.
  - Click **More Solid Colors** to open the style selection dialog box and select a solid color.
  - Click **Color Picker** to select a color from the screen.
4. Configure the text gradient, pattern, or texture. Do any of the following:
  - Click a predefined gradient.
  - Click **More Gradients** to open the style selection dialog box and configure a gradient.
  - Click **Patterns** to open the style selection dialog box and select a pattern.
  - Click **Textures** to open the style selection dialog box and select a texture.

For more information about the style selection dialog box, see [Setting Style](#).

## Related Topics

[Setting the Text Properties of an Element](#)

### Setting the Text Alignment

You can change the horizontal and vertical positioning of text within a text box element or button element.

You can also change the positioning for a text element. If the text is modified at design time or run time, the alignment sets how the element boundary changes to fit around the modified text.

**Note:** You can also set the text alignment in the **Properties Editor** by setting the **Alignment** property.

If the element is a text box or a button, then the text is aligned accordingly.

If the element is a text element and you then modify the text at design time or run time, the text is anchored to

the point of alignment.

- Text right alignments move additional text further over to the left.
- Text left alignments move additional text to the right.
- Changes in font size leave the point of alignment unchanged and modify the frame accordingly.

### To set the text alignment

1. Select a text element, text box element or button element on the canvas.
2. On the **Format** menu, point to **Text Alignment**, and then click the appropriate command:

Click this command	To
<b>Top Left</b>	Align the text at the top left frame handle.
<b>Top Center</b>	Align the text at the top middle frame handle.
<b>Top Right</b>	Align the text at the top right frame handle.
<b>Middle Left</b>	Align the text at the middle left frame handle.
<b>Middle Center</b>	Align the text in the middle of the element.
<b>Middle Right</b>	Align the text at the middle right frame handle.
<b>Bottom Left</b>	Align the text at the bottom left frame handle.
<b>Bottom Center</b>	Align the text at the bottom center frame handle.
<b>Bottom Right</b>	Align the text at the bottom right frame handle.

## Related Topics

[Setting the Text Properties of an Element](#)

## Substituting Strings

You can search and replace strings of any element that have the **Text** property on your canvas. You can use the basic mode to replace strings in a list.

You can also use advanced functions, such as find and replace, ignore, case-sensitivity, and wildcards.

You cannot substitute static strings that are used in an Radio Button Group, List Box or Combo Box.

If you substitute strings for a text element in an embedded graphic, that text element is not updated if you change the source graphic's text. For example, an embedded graphic contains a text graphic with the string "SomeTextHere". You substitute "SomeTextHere" with "MyText", and then change the source graphic text from "SomeTextHere" to "NewText". The text in the embedded graphic will still show "MyText".

## To substitute strings in a graphic by using the list

1. Select one or more elements.
2. Do one of the following:
  - Press Ctrl + L.
  - On the **Special** menu, click **Substitute Strings**.  
The **Substitute Strings** dialog box appears.
3. In the **New** column, type the text to be replaced.
4. Click **OK**.

## To substitute strings in a graphic by using advanced functions

1. Select one or more elements.
2. Do one of the following:
  - Press Ctrl + E.
  - On the **Special** menu, click **Substitute Strings**.  
The **Substitute Strings** dialog box appears.
3. Click **Find & Replace**. The dialog box expands and shows advanced options.
4. Configure the search strings. Do any of the following:
  - To find specific strings in the list, type a string in the **Find What** box and click **Find Next** to find the next string.
  - To replace a selected found string with another string, type a string in the **Replace with** box and click **Replace**.
  - To replace multiple strings, type values in the **Find What** and **Replace with** boxes and click **Replace all**.
5. Configure the search options. Do any of the following:
  - If you want the search to be case-sensitive, click **Match Case**.
  - To find only entire words that match your search string, click **Match Whole Word Only**.
  - To use wildcards, click **Use Wildcards**. Use an asterisk (\*) to search for any sequence of characters. Use a question mark (?) to search for strings with one variable character.
6. Click **OK**.

## Related Topics

[Setting the Text Properties of an Element](#)

## Setting Style

You can set the fill, line, and text style from various places in the Industrial Graphic Editor using the style selection dialog box. The style selection dialog box is common to any element for which you can set a solid color, gradient, pattern, or texture. You can also set the transparency of the style.

Because you can open the style selection dialog box from different places in the Industrial Graphic Editor, the dialog box header can be different.

Also, not all tabs may be available. For example, for setting one color of a gradient in the Properties Editor, you can only select a solid color from the style selection dialog box.

## Related Topics

- [Editing Common Properties of Elements and Graphics](#)
- [Setting a Solid Color](#)
- [Setting a Gradient](#)
- [Setting a Pattern](#)
- [Setting a Texture](#)
- [Setting the Style to No Fill](#)
- [Setting the Transparency of a Style](#)

### Setting a Solid Color

You can set a solid color using the **Solid Color** tab in the style selection dialog box. You can set a solid color from the:

- Standard palette.
- Color disc and bar.
- Value input boxes.
- Color picker.
- Custom palette.

You can also:

- Add the new color to the custom palette.
- Remove a color from the custom palette.
- Save the custom palette.
- Load a custom palette.

## Related Topic

- [Setting Style](#)
- [Setting a Solid Color from the Standard Palette](#)
- [Setting a Solid Color with the Value Input Boxes](#)
- [Setting a Solid Color with the Color Picker](#)
- [Setting a Solid Color from the Custom Palette](#)
- [Adding and Removing Colors in the Custom Palette](#)
- [Saving and Loading the Custom Palette](#)

# Setting a Solid Color from the Standard Palette

You can set a solid color from the standard palette using the **Solid Color** tab in the style selection dialog box. The standard palette is a set of 48 predefined colors you can use to quickly select a solid color.

## To set a solid color from the Standard Palette

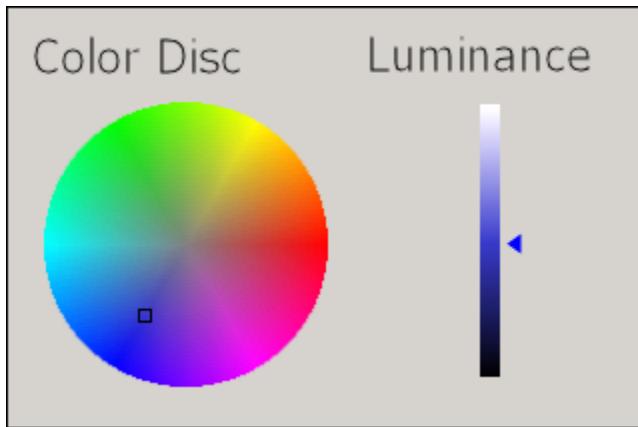
1. In the style selection dialog box, click the **Solid Color** tab.
2. In the **Standard Palette** area, click a color. The new color appears in the **New** color box on the right of the dialog box.
3. Click **OK**.

## Related Topics

[Setting a Solid Color](#)

# Setting a Solid Color from the Color Disc and Bar

You can set a solid color using the color disc and bar on the **Solid Color** tab in the style selection dialog box. The color disc and bar let you graphically select the color and the luminance (brightness).



## To set a solid color from the color disc and bar

1. In the style selection dialog box, click the **Solid Color** tab.
2. Click on the color disk to select a color. The bar is updated and shows the selected color in varying degrees of luminance (brightness).
3. Click on the bar to select a luminance (brightness). The new color appears in the **New** color box on the right of the dialog box.
4. Click **OK**.

## Related Topics

[Setting a Solid Color](#)

# Setting a Solid Color with the Value Input Boxes

You can set a solid color by typing values that define the color, such as:

- Red component (0-255).
- Green component (0-255).
- Blue component (0-255).
- Hue (0-255).
- Saturation (0-255).
- Luminance (0-255).

## To set a solid color with the value input boxes

1. In the style selection dialog box, click the **Solid Color** tab.
2. In the **Red**, **Green**, **Blue**, **Hue**, **Sat.** and **Lum.** boxes, type respective values. The resulting color appears in the **New** color box on the right of the dialog box and also on the color wheel and bar.
3. Click **OK**.

## Related Topics

[Setting a Solid Color](#)

# Setting a Solid Color with the Color Picker

You can set a solid color by using the color picker on the **Solid Color** tab in the style selection dialog box. The color picker lets you select a color from anywhere on the screen.

## To set a solid color with the color picker

1. In the style selection dialog box, click the **Solid Color** tab.
2. Click the **Color Picker** button. The color picker pointer appears.
3. Select a color from anywhere on the screen by moving the mouse. As you move the mouse, the new color appears in the **New** color box on the right of the dialog box.
4. Click the mouse to complete the color selection.
5. Click **OK**.

## Related Topics

[Setting a Solid Color](#)

# Setting a Solid Color from the Custom Palette

You can set a solid color from the custom palette on the **Solid Color** tab in the style selection dialog box. The custom palette is a set of colors that you want to frequently use. You can save the custom palette to a .pal file or load a custom palette from a .pal file.

To use colors from the custom palette, you first need to add them. For more information, see [Adding and Removing Colors in the Custom Palette](#).

## To set a solid color from the custom palette

1. In the style selection dialog box, click the **Solid Color** tab.
2. In the **Custom Palette** area, select a color. The new color appears in the **New** color box on the right of the dialog box.
3. Click **OK**.

## Related Topics

[Setting a Solid Color](#)

# Adding and Removing Colors in the Custom Palette

You can add up to 36 solid colors to the custom palette. You can also remove any colors from the custom palette. You cannot add a color that is already in the custom palette.

## To add a solid color to the custom palette

1. In the style selection dialog box, click the **Solid Color** tab.
2. Add the color. Do any of the following:
  - Select a solid color from the custom palette.
  - Select a solid color from the color disc and bar.
  - Type values for red, green, blue, hue, saturation, and luminance.
  - Select a solid color with the color picker.The new solid color appears in the **New** color box on the right of the dialog box.
3. Click the add button above **Custom Palette**. The solid color is added to the **Custom Palette** area.

## To remove a solid color from the custom palette

1. In the style selection dialog box, click the **Solid Color** tab.
2. In the **Custom Palette** area, select the solid color you want to remove.
3. Click the delete button above **Custom Palette**. The solid color is removed from the custom palette.

## Related Topics

[Setting a Solid Color](#)

# Saving and Loading the Custom Palette

You can save the current custom palette or load a previously saved custom palette. The custom palette is loaded from or saved to a Windows Palette file (.pal).

After you save or load a custom palette, the .pal file is not connected to the graphic in any way.

### To save a custom palette

1. In the style selection dialog box, click the **Solid Color**.
2. Click the **Save Palette** button. The **Save Palette** dialog box appears.
3. Browse to the location where you want to save the custom palette, type a name, and then click **Save**. The custom palette is saved as a palette file.

### To load a custom palette

1. In the style selection dialog box, click the **Solid Color** tab.
2. Click the **Load Palette** button.
3. If you currently have colors in the custom palette, a message appears. Click **Yes** to continue and overwrite the current colors in the custom palette.
4. In the **Load Palette** dialog box, browse to the location of the palette file, select it, and then click **Open**. The custom palette is loaded from the selected file.

## Related Topics

[Setting a Solid Color](#)

## Setting a Gradient

You can configure gradients by the:

- Number of colors - 1, 2 or 3.
- Direction - horizontal, vertical, radial, or customized.
- Variant - depending on your selection for the number of colors and direction.
- Color distribution shape - triangular with options to configure the center and falloff.
- Focus scales - width and height.

You set a gradient on the **Gradient** tab in the style selection dialog box.

## Related Topics

[Setting Style](#)

[Setting the Number of Colors for a Gradient](#)

[Setting the Direction of the Gradient](#)

[Changing the Variant of a Gradient](#)

[Setting the Color Distribution Shape](#)

[Setting the Focus Scales of a Gradient](#)

# Setting the Number of Colors for a Gradient

You can set the number of colors you want to use in a gradient.

- If you use one color, the gradient is between this solid color and a specified shade of black to white.
- If you use two colors, the gradient is between these two colors.
- If you use three colors, the gradient is between these three colors in sequence.

### To set a gradient using one color

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Colors** area, click **One**. A color selection box and a slider for the dark to light selection appears.
3. Click the color selection box to open the **Select Solid Color 1** dialog box. Select a solid color and click **OK**. For more information about this dialog box, see [Setting a Solid Color](#).
4. Move the slider between **Dark** and **Light**. The new gradient appears in the **New** color box on the right of the dialog box.
5. Click **OK**.

### To set a gradient using two colors

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Colors** area, click **Two**. Two color selection boxes appear.
3. Click the **Color 1** or **Color 2** color field to select a color from the style selection dialog box. For more information about this dialog box, see [Setting a Solid Color](#).  
The new gradient appears in the **New** color box on the right of the dialog box.
4. Click **OK**.

### To set a gradient for three colors

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Colors** area, select **Three**. Three color selection boxes appear.
3. Click the **Color 1**, **Color 2** or **Color 3** color field to select a color from the style selection dialog box. For more information about this dialog box, see [Setting a Solid Color](#).  
The new gradient appears in the **New** color box on the right of the dialog box.

4. Click **OK**.

## Related Topics

[Setting a Gradient](#)

# Setting the Direction of the Gradient

You can configure the direction of the gradient to be one of the following:

- Horizontal - from side to side
- Vertical - up and down
- Radial - circular from the center outwards
- Custom angle - across the element at a specified angle

### To set a horizontal gradient

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Direction** area, click **Horizontal**. The new gradient appears in the **New** color box on the right of the dialog box.
3. Click **OK**.

### To set a vertical gradient

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Direction** area, click **Vertical**. The new gradient appears in the **New** color box on the right of the dialog box.
3. Click **OK**.

### To set a radial gradient

1. In the style selection dialog box, click the **Gradient** tab.
  2. In the **Direction** area, click **Radial**.
  3. Set the center location. Do any of the following:
    - In the **Horizontal** and **Vertical** boxes, type values for the center location.
    - Click and drag the center point in the adjacent box.
- The new gradient appears in the **New** color box on the right of the dialog box.

4. Click **OK**.

### To set the custom angle of a gradient

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Direction** area, click **Custom**.

3. Set the angle. Do any of the following:
  - In the **Angle** text box, type a value for the angle.
  - Click and drag the angle bar in the adjacent box.

The new gradient appears in the **New** color box on the right of the dialog box.

4. Click **OK**.

## Related Topics

[Setting a Gradient](#)

# Changing the Variant of a Gradient

You can change the variant of a gradient. The variants are alternate gradients with the same colors you can quickly select.

### To change the variant of a gradient

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Variants** area, click on a variant gradient.  
The new gradient appears in the **New** color box on the right of the dialog box.
3. Click **OK**.

## Related Topics

[Setting a Gradient](#)

# Setting the Color Distribution Shape

You can configure the distribution shape of a triangle gradient with one or two colors.

- In a triangular distribution, the gradient from one color to the next rises and falls at the same rate.

You can also configure the peak and the falloff.

- The peak specifies the offset of the gradient if it has one or two colors.
- The falloff specifies the amplitude of the gradient if it has one or two colors.

Additionally, you can configure the center point of a radial gradient if it is defined by three colors.

### To use a triangular gradient

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Color Distribution Shape** area, click **Triangular**. The new gradient appears in the **New** color box on the right of the dialog box.

3. Click **OK**.

#### To set the peak of a gradient with one or two colors

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Color Distribution Shape** area, do one of the following:
  - Use the **Peak** slider to specify the peak.
  - In the **Peak** box, type a value from 0 to 100.

The new gradient appears in the **New** color box on the right of the dialog box.

3. Click **OK**.

#### To set the falloff of a gradient with one or two colors

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Color Distribution Shape** area, do one of the following:
  - Use the **Falloff** slider to specify the peak.
  - In the **Falloff** box, type a value from 0 to 100.

The new gradient appears in the **New** color box on the right of the dialog box.

3. Click **OK**.

#### To set the center point of a radial gradient with three colors

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Color Distribution Shape** area, do one of the following:
  - Use the **Center** slider to specify the peak.
  - In the **Center** box, type a value from 0 to 100.

The new gradient appears in the **New** color box on the right of the dialog box.

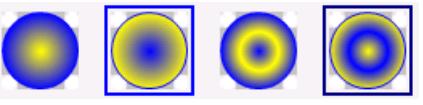
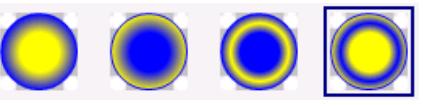
3. Click **OK**.

### Related Topics

[Setting a Gradient](#)

## Setting the Focus Scales of a Gradient

You can set the focus scales of a radial gradient. The focus scales acts as a magnification of the gradient. You can set the height and width of the focus scales.

Height	Width	Appearance
0	0	
50	50	

### To set the height and width of the focus scales for a gradient

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Focus Scales** area, do one of the following:
  - Move the **Height & Width** slider to specify the height and width.
  - In the text box, type the value for the height and width.
 The new gradient appears in the **New** color box on the right of the dialog box.
3. Click **OK**.

## Related Topics

[Setting a Gradient](#)

## Setting a Pattern

You can set a pattern for an element. The following table describes the pattern options:

Pattern	Options
Horizontal	Simple, Light, Narrow, Dark, Dashed
Vertical	Simple, Light, Narrow, Dark, Dashed
Percent	05, 10, 20, 25, 30, 40, 50, 60, 70, 75, 80, 90
Grid	Small, Large, Dotted
Checker Board	Small, Large
Diagonals	Forward, Backward, Dashed Upward/Downward, Light/Dark/Wide Upward/Downward
Diamond	Dotted, Outlined, Solid
Cross	Diagonal
Brick	Horizontal, Diagonal
Confetti	Small, Large

Pattern	Options
Others	Zig Zag, Wave, Weave, Plaid, Divot, Shingle, Trellis, and Sphere

Patterns consist of the foreground color and the background color that you can change.

### To set a pattern

1. In the style selection dialog box, click the **Pattern** tab.
2. Select a pattern. The new pattern appears in the **New** color box on the right of the dialog box.
3. If you want to change the foreground color of the pattern, click the **Foreground** color selection box. The style selection dialog box appears. Select a solid color and click **OK**.
4. If you want to change the background color of the pattern, click the **Background** color selection box. The style selection dialog box appears. Select a solid color and click **OK**.  
For more information about setting a solid color, see [Setting a Solid Color](#).
5. Click **OK**.

## Related Topics

[Setting Style](#)

## Setting a Texture

Textures are images you can use as styles for lines, fills and text. You can stretch the image or tile the image across the entire element to be filled.

### To set a texture

1. In the style selection dialog box, click the **Textures** tab.
2. Click **Select Image**. The **Open** dialog box appears. You can import the following image formats: .BMP, .GIF, .JPG, .JPEG, .TIF, .TIFF, .PNG, .ICO, .EMF. Animated GIF images are not supported.
3. Browse to and select an image file and click **Open**. The new pattern appears in the **New** color box on the right of the dialog box.
4. Configure the size mode. Do one of the following:
  - Click **Tile** to create a pattern that repeats itself.
  - Click **Stretch** to enlarge (or shrink) the pattern across the selected element.
5. Click **OK**.

## Related Topics

[Setting Style](#)

## Setting the Style to No Fill

You can set the style to "No Fill". For example if you set the fill style of a rectangle element to No Fill, the

background of the rectangle appears transparent.

### To set the No Fill style

1. In the style selection dialog box, click the **No Fill** tab.

The No Fill style appears as a red cross-through line in the **New** color box on the right of the dialog box.

2. Click **OK**.

## Related Topics

[Setting Style](#)

### Setting the Transparency of a Style

You can set the transparency of a solid color, gradient, pattern, or texture.

#### To set the transparency of a style

1. Open the style selection dialog box.



2. At the bottom of the dialog box, do one of the following:

- Drag the **Transparency** slider handle left or right to change the transparency percentage.
- In the **Transparency** text box, type a percentage value.

The new style appears in the **New** color box.

3. Click **OK**.

## Related Topics

[Setting Style](#)

### Setting the Transparency Level of an Element

You can set the transparency level of an element. Levels range from 0 percent (opaque) to 100 percent (transparent).

Transparency of a group of elements behaves in a special way.

#### To set the transparency level of an element

1. Select one or more elements.
2. On the **Format** menu, click **Transparency**.
3. To use a predefined level, select it from the list.
4. To use a different level, click **More Transparency Levels**. The **Select Transparency Level** dialog box appears. Type a transparency level in the **Transparency** text box or use the slider to select a transparency level.
5. Click **OK**.

**Note:** You can also set the transparency level by changing the **Transparency** property in the Properties Editor.

## Related Topics

[Editing Common Properties of Elements and Graphics](#)

### Adjusting the Colors and Transparency of a Gradient

You can easily change the colors and transparency of an element with a gradient style.

For example, you can create pipes with a gradient style of different colors. You can change the pipe color, but still keep the 3-D appearance.

You do this in the Properties Editor using the Color1, Color2, Color3, and Transparency sub-properties.

#### To tweak the colors and transparency of a gradient

1. Select the element for which you want to change colors or transparency.
2. In the Properties Editor, locate the appropriate style setting. This can be:
  - FillColor
  - LineColor
  - TextColor
  - UnFillColor
3. Click the + icon to expand the property. The Color1, Color2, Color3, and Transparency sub-properties are shown.
4. Do one of the following:
  - Click the color box of one of the color sub-properties.
  - Type a new value for the transparency and click **Enter**.
5. Click the browse button. The style selection dialog box appears.
6. Select a color from the style selection dialog box and click **OK**. The solid color is applied to the selected element.

## Related Topics

[Editing Common Properties of Elements and Graphics](#)

### Enabling and Disabling Elements for Run-Time Interaction

You can enable or disable elements so that the run time user cannot use any interaction animations, such as:

- User input.
- Horizontal and vertical sliders.
- Pushbuttons.
- Action scripts.
- Showing and hiding graphics.

Other animations such as horizontal fills and tooltips continue to work as expected.

### To enable an element for run-time interaction

1. Select one or more elements you want to enable.
2. In the Properties Editor **Runtime Behavior** group, set the Enabled property to True.

### To disable an element for run-time interaction

1. Select one or more elements you want to disable.
2. In the Properties Editor **Runtime Behavior** group, set the Enabled property to False.

## Related Topics

[Editing Common Properties of Elements and Graphics](#)

## Changing the Visibility of Elements

You can configure elements to be hidden or shown at run time.

The visibility of an element does not affect its animations. Even when an element is invisible, its animations continue to be evaluated.

### To configure an element to be shown at run time

1. Select one or more elements you want to have shown at run time.
2. In the Properties Editor **Runtime Behavior** group, set the Visible property to True.

### To configure an element to be hidden at run time

1. Select one or more elements you want to have hidden at run time.
2. In the Properties Editor **Runtime Behavior** group, set the Visible property to False.

## Related Topics

[Editing Common Properties of Elements and Graphics](#)

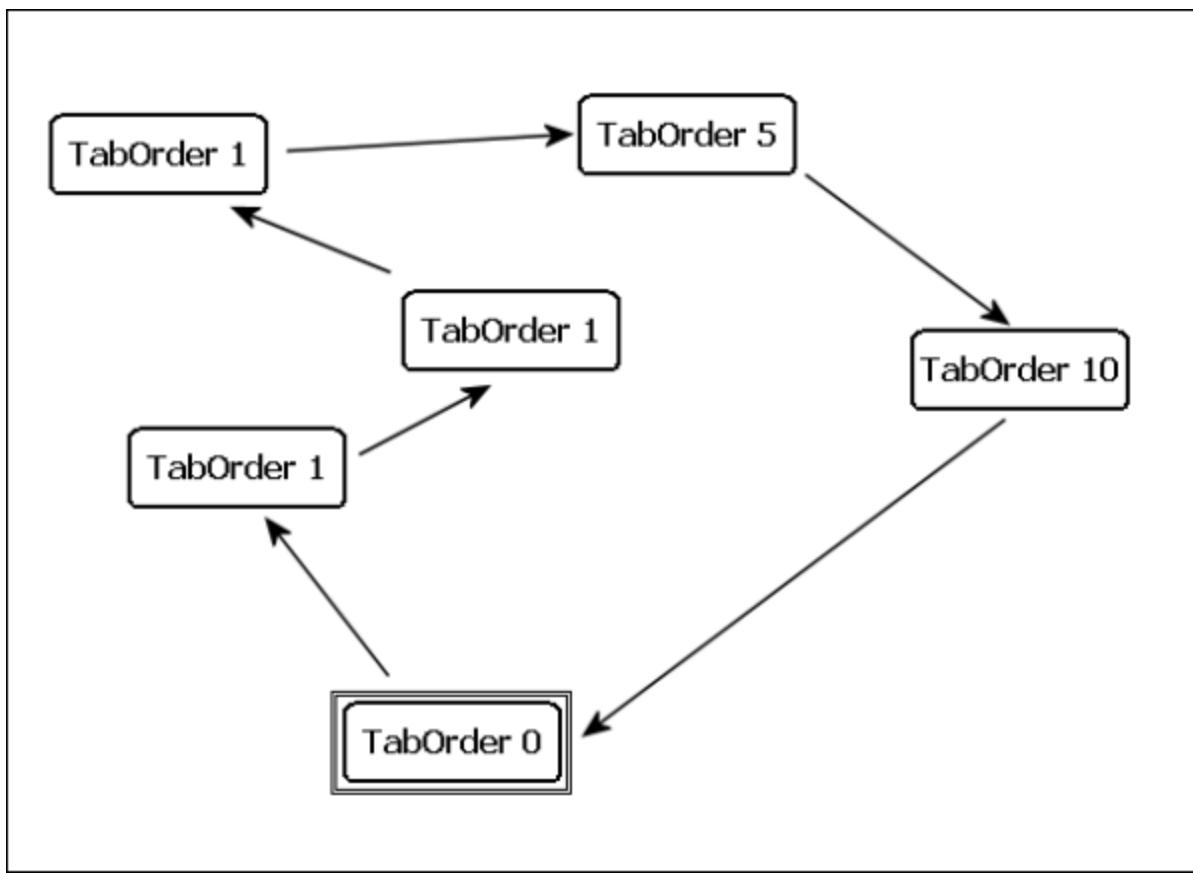
## Editing the Tab Order of an Element

You can configure the elements on the canvas so that at run time you can use the Tab key to put each element in focus in a specified sequence. This sequence is called the tab order.

By default, when you place elements on the canvas, they have a tab order number of 0. Elements with the same tab order number are placed into focus by tabbing at run time according to their z-order. This means they are tabbed through at run time according to their position in the Elements List.

You can override the tab order by assigning a unique index number to the TabOrder property of each element.

Lower tab order numbers take precedence over higher tab order numbers. You must change this value to determine the tab order sequence.



Verify that the TabStop property of each element is set to true. When the TabStop property is set to true, you can use the Tab key at run time to switch to the selected element.

#### To edit the element's tab order

1. Select the element for which you want to set the tab order.
2. In the Properties Editor, set the **TabStop** property is set to True.
3. Type a unique value for the **TabOrder** property.

#### Related Topics

[Editing Common Properties of Elements and Graphics](#)

#### Using the Format Painter to Format Elements

You can apply formatting of one element to other elements quickly by using the format painter. You can apply the format of one element:

- One time to other elements.
- In repetitive mode to other elements.

When you use the format painter, it copies the following formats of the element if applicable to the target elements:

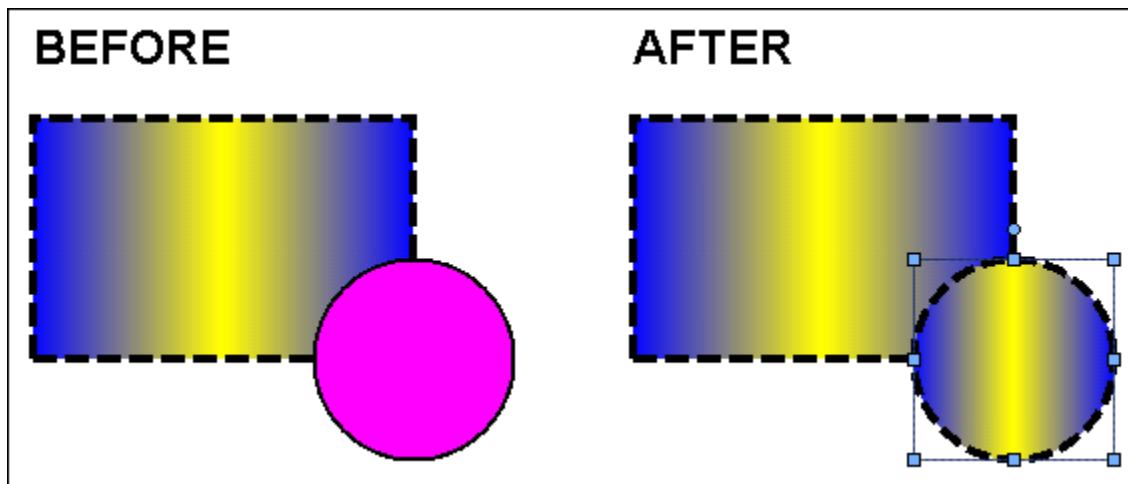
- Font family, size, and style
- Text style, alignment, and word wrap settings
- Line style, weight, pattern, and ends
- Transparency
- Fill style, orientation, behavior, horizontal percent fill, and vertical percent fill
- Unfilled style
- Horizontal and vertical direction properties

You cannot use the format painter for:

- The status element
- An element that is part of a path
- Groups of elements
- Elements in different hierarchy groups

### To copy the format of an element one time

1. Select the element with the format you want to copy.
2. On the **Edit** menu, click **Format Painter**. The pointer appears as the format painter cursor.
3. Select the element you want to apply the format to. The format is applied to the clicked element.



### To copy the format of an element in repetitive mode

1. Select the element with the format you want to copy.
2. On the toolbar, double-click the Format Painter icon. The pointer appears as the format painter cursor.
3. Click each element you want to apply the format to. The format is applied to the clicked element.
4. Repeat Step 3 for any other elements you want to apply the format to.
5. When you are done, press the Esc key.

## Related Topics

[Editing Common Properties of Elements and Graphics](#)

### Editing the General Properties of a Graphic

You can configure the general properties of a graphic. The general properties determine the overall appearance and behavior of the graphic. You can:

- Add a meaningful description to your graphic.
- Enable anti-aliasing, or smoothing, for your graphic to improve its appearance. The anti-aliasing filter essentially blurs the elements slightly at the edges.
- Allow or prevent the opening of more than one graphic or display from a graphic. One example is a graphic with multiple Show Symbol animations. If this option is enabled, you can open more than one pop-up and each pop-up is modeless.

#### To edit the description of a graphic

1. Click on the canvas so that no elements are selected.
2. In the Properties Editor, type a meaningful description for the **Description** property.

#### To use smoothing (anti-aliasing) for a graphic

1. Click on the canvas so that no elements are selected.
2. In the Properties Editor, select True for the **Smoothing** property.

#### To enable multiple pop-ups for a graphic

1. Click on the canvas so that no elements are selected.
2. In the Properties Editor, select True for the **MultiplePopupsAllowed** property.

## Related Topics

[Editing Common Properties of Elements and Graphics](#)

## Editing Graphic-Specific and Element-Specific Properties

### In This Chapter

[About Graphic- and Element-Specific Properties](#)

[Setting the Radius of Rounded Rectangles](#)

[Setting Line End Shape and Size](#)

[Setting Auto Scaling and Word Wrapping for a Text Box](#)

[Using Images](#)

[Using Buttons](#)[Editing Control Points](#)[Changing the Tension of Curves and Closed Curves](#)[Changing Angles of Arcs, Pies and Chords](#)[Monitoring and Showing Quality and Status](#)[Using Windows Common Controls](#)

## About Graphic- and Element-Specific Properties

You can configure graphic-specific and element-specific properties. For properties that are common to all or most elements, see [Editing Common Properties of Elements and Graphics](#).

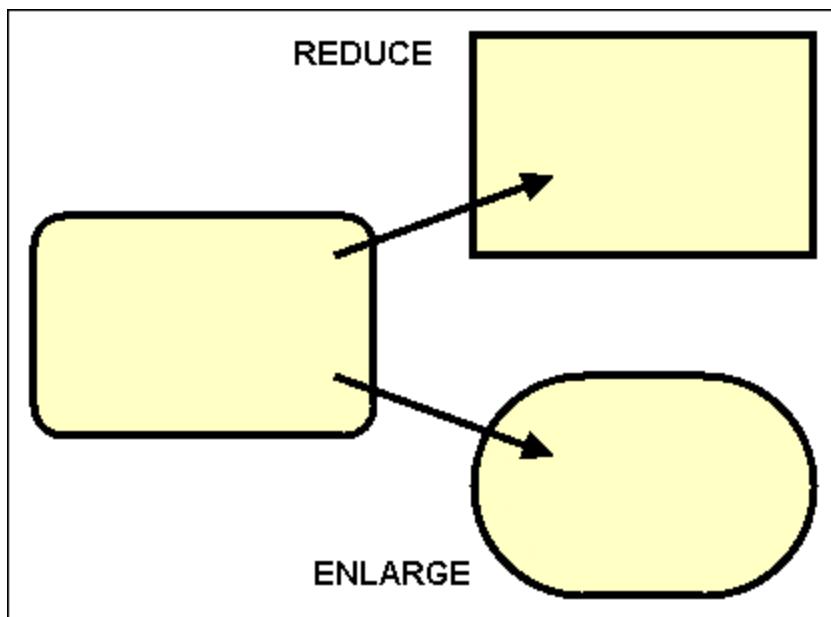
You can configure:

- General properties of a graphic.
- Radius of rounded rectangles.
- Shape and end appearance of lines and H/V lines.
- Auto-sizing and word-wrapping in text boxes.
- Image-specific properties.
- Button-specific properties.
- Control points and tension in curves.
- Angles in pies, chords, and arcs.
- Status elements.
- Windows common controls.

## Setting the Radius of Rounded Rectangles

You can specify the radius, in pixels, of the corners of rounded rectangles. The radius determines their "roundness". You can:

- Enlarge or reduce the radius of the rounded rectangle on the fly. The easiest way to do this is with the keyboard.
- Set the radius of the rounded rectangle to a specific value using the Properties Editor.



Rounded rectangles maintain their radius when their size is changed. If the graphic containing rounded rectangles is embedded into a window and resized, the radius is not affected. This can have adverse affects on the graphic representation of your graphic.

#### To enlarge the radius of a rounded rectangle

1. Select one or more rounded rectangles on the canvas.
2. Press and hold Shift and the + key on the number pad. The radius is enlarged, and the rounded rectangle becomes more round.

#### To reduce the radius of a rounded rectangle

1. Select one or more rounded rectangles on the canvas.
2. Press and hold Shift and the minus (-) key on the number pad. The radius is reduced, and the rounded rectangle becomes more rectangular.

#### To set the radius of a rounded rectangle exactly

1. Select one or more rounded rectangles on the canvas.
2. In the Properties Editor, change the value for Radius property and click **Enter**. The selected rounded rectangles are updated accordingly.

### Related Topics

[Editing Graphic-Specific and Element-Specific Properties](#)

### Setting Line End Shape and Size

You can set the line end shape and size for any element that contains open lines such as lines, H/V lines, polylines, curves, and arcs.

For a line end, you can set the shape to be an arrowhead, diamond, circle, or square. You can set the size if the line end shape is an arrowhead.

### To set the line end shape

1. Select one or more elements.
2. On the **Format** menu, click **Line Ends**.
3. To use a predefined line end shape, select it from the list.
4. To use another line shape, click **More Line Options**. The **Select Line Options** dialog box appears.
5. Do the following:
  - a. In the **Line Start** list, click a shape for the start of the line.
  - b. In the **Line End** list, click a shape for the end of the line.
  - c. Click **OK**.

### To set the size of the line arrowheads

1. Select one or more open line elements.
2. On the **Format** menu, click **More Line Options**. The **Select Line Options** dialog box appears.
3. Select a size on the **Line Start Size** list if the line starts with an arrowhead. Valid sizes are: XX Small, X Small, Small, Medium Small, Medium, Medium Large, Large, X Large, XX Large.
4. Select a size on the **Line End Size** list if the line ends with a shape.
5. Click **OK**.

---

**Note:** You can also set the line end shapes by changing the **StartCap** and **EndCap** properties in the Properties Editor.

---

## Related Topics

[Editing Graphic-Specific and Element-Specific Properties](#)

## Setting Auto Scaling and Word Wrapping for a Text Box

You can configure a text box to auto scale the text or to word wrap the text within the text box.

- For auto scaling, the text is resized to fit the text box.
- For word wrapping, the text in a text box continues on the next line.

### To auto scale the text in a text box

1. Select one or more text boxes.
2. In the Properties Editor, set the **AutoScale** property to true.

### To word wrap the text in a text box

1. Select one or more text boxes.

2. In the Properties Editor, set the WordWrap property to true.

## Related Topics

[Editing Graphic-Specific and Element-Specific Properties](#)

## Using Images

You can place images on the canvas. This is a two step process:

1. Draw a frame which specifies the target size of the image.
2. Import the image from an image file.

After you place an image on the canvas, you can:

- Set the display mode (ImageStyle).
- Set the image alignment (ImageAlignment).
- Set the transparency color (HasTransparentColor, TransparentColor properties).
- Open the image in an image editing application.
- Select a different image for the image element.

## Related Topics

[Editing Graphic-Specific and Element-Specific Properties](#)

[Placing an Image on the Canvas](#)

[Setting the Image Display Mode](#)

[Setting the Image Alignment](#)

[Setting the Image Color Transparency](#)

[Editing the Image](#)

[Setting the Image Editing Application](#)

[Selecting a Different Image](#)

## Placing an Image on the Canvas

You can place an image on the canvas. The image data is imported from an image file. You can import the following image formats: .BMP, .GIF, .JPG, .JPEG, .TIF, .TIFF, .PNG, .ICO, .EMF.

You cannot use animated GIF images.

### To place an image on the canvas

1. In the **Tools** panel, select the image icon.
2. Click the canvas where you want to place the image and drag the mouse to draw a rectangle that will contain your image.
3. Release the mouse button. The **Open** dialog box appears.

4. Browse to and select an image file, and then click **Open**. The image is loaded into the image frame.  
If the image frame is smaller than the image, the image is cropped to fit into the frame. If the image frame is larger than the image, the image appears in its original size.

## Related Topics

[Using Images](#)

### Setting the Image Display Mode

You can set the way the image appears on the canvas.

- In normal mode, the image is not stretched or tiled. You can resize the image frame with the resizing handles.
- In stretch mode, the image is stretched so that it fills its frame.
- In tile mode, the image is repeated so that a tiled pattern that fills its frame is created.
- In auto mode, the image frame is enlarged or reduced to the image size. The resizing handles are locked. When the image style of an image element is Auto, you cannot change its size.

#### To stretch an image to the image frame

1. Select the image element you want to stretch.
2. In the Properties Editor, select **ImageStyle**.
3. In the list, click **Stretch**. The image is stretched to the image frame.

#### To tile an image in an image frame

1. Select the image element you want to tile.
2. In the Properties Editor, select **ImageStyle**.
3. In the list, click **Tile**. The image is tiled to fill the image frame.

#### To set an image frame size to its image size

1. Select the image element you want to adjust.
2. In the Properties Editor, select **ImageStyle**.
3. In the list, click **Auto**. The image frame is enlarged or reduced to the image size.

## Related Topics

[Using Images](#)

### Setting the Image Alignment

The image alignment specifies where the image appears in an image frame. By default, images appear in the center of the image frame. You can change this setting to one of the following:

- Top left, top center, or top right
- Middle left, center, or middle right
- Bottom left, bottom center, or bottom right

---

**Note:** You can also set the image alignment in the **ImageAlignment** property in the Properties Editor.

---

### To set the image alignment

1. Select the image element with the image you want to align.
2. In the Properties Editor, select **ImageAlignment**.
3. In the list, click one of the following options: TopLeft, TopCenter, TopRight, MiddleLeft, Centers, MiddleRight, BottomLeft, BottomCenter or BottomRight. The image is aligned accordingly in the image frame.

## Related Topics

[Using Images](#)

### Setting the Image Color Transparency

You can use image color transparency to specify that a color within an image is partially or entirely transparent. When you configure image transparency, you must:

- Enable color transparency for images.
- Specify the color that is to be transparent.

Setting the image color transparency is different than setting the transparency of the image element, as it only applies to one color. Image transparency applies to the entire image.

### To enable image color transparency

1. Select the image element.
2. In the Properties Editor, select **HasTransparentColor**.
3. In the list, click **True**.

### To set the transparency color for an image

1. Select the image element.
2. On the **Edit** menu, click **Select Image Transparent Color**. The pointer becomes a color picker.
3. Click the color you want to use as the transparency color. The image is updated with the new transparency color.

---

**Note:** You can also select a transparency color with the **TransparentColor** property in the Properties Editor. For more information about setting the color, see [Setting a Solid Color](#).

---

## Related Topics

[Using Images](#)

## Editing the Image

You can edit the image in an image element by opening it in an image editing application.

You can specify the image editor by changing the designer preferences. For more information, see [Setting the Image Editing Application](#).

### To edit an image

1. Select the image element with the image you want to edit.
2. On the **Edit** menu, click **Edit Image**. The image is opened with the associated image editing application.
3. Make changes to the image as needed, save the image and close the image editing application. The image is updated on the canvas.

## Related Topics

[Using Images](#)

## Setting the Image Editing Application

You can specify the image editor that opens when you select an image for editing. You can select a currently registered image editing application or add one.

### To set the image editing application

1. On the **Special** menu, click **Preferences**. The **Designer Preferences** dialog box appears.
2. Select an image editor from the Image Editor list.

### To add an image editing application

1. On the **Special** menu, click **Preferences**. The **Designer Preferences** dialog box appears.
2. In the **Image Editor** list, click **Choose Custom Editor**. The **Select Image Editing Application** dialog box appears.
3. Browse to and select the executable file of the image editing application and click **Open**. The image editor is added to the list.

## Related Topics

[Using Images](#)

## Selecting a Different Image

You can change the current image of an image element by selecting a new image.

### To select a different image

1. Select the image element with the image you want to change.
2. On the **Edit** menu, click **Select Image**. The **Open** dialog box appears.

3. Browse to and select an image file, and then click **Open**. The image is loaded into the image frame.

**Note:** You can also select a different image by clicking the browse button in the **Image** property in the Properties Editor .

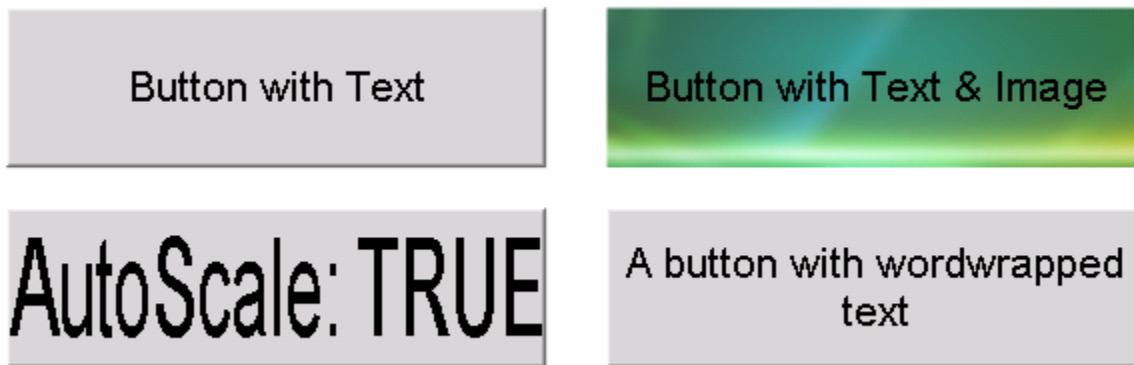
## Related Topics

[Using Images](#)

## Using Buttons

You can add a text caption or an image to buttons that belong to Industrial Graphics. If a button includes a text caption, you can:

- Automatically scale the font size
- Configure the text to wrap within the button



## Related Topics

[Editing Graphic-Specific and Element-Specific Properties](#)

[Automatically Scaling Text in Buttons](#)

[Wrapping Text in Buttons](#)

[Configuring Buttons with Images](#)

## Automatically Scaling Text in Buttons

You can automatically scale text so that the font size is adapted to the button size.

### To automatically scale text in buttons

1. Select the button element on the canvas.
2. In the Properties Editor, set the **AutoScale** property to True.

## Related Topics

[Using Buttons](#)

### Wrapping Text in Buttons

You can wrap text in buttons.

#### To wrap text in buttons

1. Select the button element on the canvas.
2. In the Properties Editor, set the **WordWrap** property to True.

## Related Topics

[Using Buttons](#)

### Configuring Buttons with Images

You can use buttons with an image/SVG in Industrial Graphics.

- The "up" image appears after a button is released and returns to the up position during run time
- The "down" image appears after a button is pressed and locks in the down position during run time

You can edit an up image or a down image after you assign it to a button.

#### To use a down image or up image on a button

1. Select the button element on the canvas.
2. In the Properties Editor, select **Image** in the property **ButtonStyle** list.
3. Click the browse button of the **UpImage** property and select an image/SVG in the **Open** dialog box. This is the image that appears on the button by default and also when the button is released.
4. Click the browse button of the **DownImage** property and select an image/SVG in the **Open** dialog box. This image appears after the button is clicked.

#### To edit an up image or a down image of a button

1. Right-click the button element on the canvas. The context menu appears.
2. Click **Edit Button Image**, then click one of the following:
  - Edit Up Image
  - Edit Down ImageThe up image or down image is opened in the default image editor.
3. Edit the image.
4. Save the image and close the image editor. The up image or down image is updated.

## Related Topics

[Using Buttons](#)

## Editing Control Points

Control points determine the shapes of polylines, polygons, curves, and closed curves. To change the shape of these elements after they have been placed on the canvas, you can:

- Move individual control points.
- Add or remove control points.

## Related Topics

[Editing Graphic-Specific and Element-Specific Properties](#)

[Moving Control Points](#)

[Adding and Removing Control Points](#)

## Moving Control Points

After you place a polyline, polygon, curve, or closed curve on the canvas, you can change its shape by editing its control points.

### To move the control points of a polyline, polygon, curve, or closed curve

1. Select the polyline, polygon, curve, or closed curve.
2. On the **Edit** menu, click **Edit Control Points**. The control points of the element are shown.
3. Click a control point you want to change and drag it to the new location. The element is updated accordingly.
4. Repeat the previous step for all control points you want to change.

## Related Topics

[Editing Control Points](#)

## Adding and Removing Control Points

You can add or remove control points from polylines, polygons, curves, and closed curves.

### To add control points to a curve or closed curve

1. Select the curve or closed curve.
2. On the **Edit** menu, click **Edit Control Points**. The control points of the element are shown.
3. Press and hold the Shift key.
4. Move the mouse over the curve or closed curve at the point you want to add a control point. The pointer appears as a pen with a plus graphic.

5. Click the curve or closed curve. The control point is added to the curve or closed curve.
6. Repeat the last step for any other control points you want to add.
7. When you are done, release the Shift key.

### To delete control points from a curve or closed curve

1. Select the curve or closed curve.
2. On the **Edit** menu, click **Edit Control Points**. The control points of the element are shown.
3. Press and hold the Ctrl key.
4. Move the mouse over the control point you want to remove. The pointer appears as a pen with a minus graphic.
5. Click the control point. The control point is removed from the curve or closed curve.
6. Repeat the last step for any other control points you want to remove. We recommend you have at least two control points.
7. When you are done, release the Ctrl key.

## Related Topics

[Editing Control Points](#)

### Changing the Tension of Curves and Closed Curves

After you place a curve or a closed curve, you can change its tension. The tension specifies how tightly the curve bends through the control points. Valid range are float values from 0 (tightly) to 2 (loosely).

---

**Note:** You can also change the tension of a curve or closed curve by changing the value for the **Tension** property in the Properties Editor.

### To edit the tension of a curve or closed curve

1. Select the curve or closed curve.
2. In the Properties Editor, type a float value from 0 to 2 for the **Tension** property.

## Related Topics

[Editing Graphic-Specific and Element-Specific Properties](#)

### Changing Angles of Arcs, Pies and Chords

After you place an arc, pie, or chord, you can change the start and sweep angles of elements. You can change the angles to any integer degree from 0 to 359. When you change the angles, you can press the Shift and Ctrl keys to make the angle snap to multiples of 15 or 45 degrees.

You can also move the start angle and sweep angle at the same time. The object appears to be rotated around its arc/pie/chord center point while keeping the same center point angle.

---

**Note:** You can also change the start or sweep angle of an arc, pie or chord in the **StartAngle** or **SweepAngle** properties in the Properties Editor. For more information, see [Utilizing Sweep Angle Run-Time Properties](#).

### To change the start or sweep angle of an arc, pie, or chord

1. Select the arc, pie, or chord.
2. On the **Edit** menu, click **Edit Start and Sweep Angles**. The start and sweep angle handles appear on the selected element.
3. If you want the angle to be multiples of 15 degrees, press and hold the SHIFT key.
4. If you want the angle to be multiples of 45 degrees, press and hold the CTRL key.
5. Grab the start angle or the sweep angle handle and drag it to the new location. The element is updated accordingly.

### To change the start and sweep angles of an arc, pie, or chord together

1. Select the arc, pie, or chord.
2. On the **Edit** menu, click **Edit Start and Sweep Angles**. The start and sweep angle handles appear on the selected element.
3. Select the start angle or the sweep angle handle and keep the mouse button down.
4. Press and hold the Alt key.
5. If you want additionally either angles to be multiples of 15 degrees, press and hold the Shift key.
6. If you want additionally either angles to be multiples of 45 degrees, press and hold the Ctrl key.
7. Drag the mouse. The start angle and sweep angle are changed accordingly.
8. When you are done, release the mouse button and then any keys.

## Related Topics

[Editing Graphic-Specific and Element-Specific Properties](#)

### Utilizing Sweep Angle Run-Time Properties

The 2 and 3 point arc, pie, and chord graphic elements contain **StartAngle** and **SweepAngle Appearance** properties. These properties can be assigned values in a client script that change during run time to show moving sweep angle or start angle lines as part of arc, pie, and chord graphic elements.

Sweep angle run-time properties are well suited for showing a current value within a range of possible values. For example, the movement of a chord sweep angle can show a pie chart with fill that indicates the current time within a repetitive period. Or, the movement of an arc sweep angle can represent a pointer to the current value within a range of possible values like a tachometer.

### To configure sweep angle run-time properties

1. Place an arc, pie, or chord graphic object on the Industrial Graphic Editor canvas.
2. Select the graphic element to show its **Properties** equipment.items.
3. Assign **StartAngle** and **SweepAngle** properties as values of a client script that change based on run-time events.

## Related Topics

[Changing Angles of Arcs, Pies and Chords](#)

## Monitoring and Showing Quality and Status

You can configure your graphic to show non-good status and quality of equipment.items or tags in different ways:

- A status element shows a specific icon depending on the quality and status of configured equipment.items, tags or elements.
- The text, fill, or line appearance of elements is overridden depending on the quality and status of the equipment.items and tags they reference.
- Elements are drawn with an outline depending on the quality and status of the equipment.items they reference.

---

**Note:** Quality and status elements might not be supported by all HMIs. Refer to your HMI help for more information.

---

## Related Topics

[Editing Graphic-Specific and Element-Specific Properties](#)

[Using Status Elements](#)

## Using Status Elements

Status elements show a specified graphic depending on the quality and status of:

- Equipment.items and tags configured for specific animated elements.
- One or more specified equipment.items or tags.

You can assign status elements to an animation in three steps:

1. Draw the status element on the canvas.
2. Associate the status element with animated elements on the canvas and/or equipment.items that provide the quality and status data to be monitored.
3. If needed, configure the appearance of the status element.

## Related Topics

[Monitoring and Showing Quality and Status](#)

## Using Windows Common Controls

You can add the following Windows common controls to your graphic:

- Radio button group
- Check box
- Edit box
- Combo box
- Calendar control
- DateTime picker
- List box

You can place these Windows common controls as you would any other element by selecting them from the **Tools** panel. You click on the canvas to position a common control and, with exception of the calendar control, drag the rectangle to set of the control.

After placing the control on the canvas, you can then configure:

- Background color and text color (with exception of the DateTime Picker control).
- Other control-specific properties in the Properties Editor.
- Control-specific animations.
- The common Value property in scripting to read from and write to the Windows common control at run time.

## Related Topics

[Editing Graphic-Specific and Element-Specific Properties](#)

[Changing Background Color and Text Color of Windows Common Controls](#)

[Reading and Writing the Selected Value at Run Time](#)

[Configuring Radio Button Group Controls](#)

[Configuring Check Box Controls](#)

[Configuring Edit Box Controls](#)

[Configuring Combo Box Controls](#)

[Configuring Calendar Controls](#)

[Configuring DateTime Picker Controls](#)

[Configuring List Box Controls](#)

## Changing Background Color and Text Color of Windows Common Controls

You can change the background color and text color of all Windows common controls with exception of the DateTime Picker control.

The background color and text color of the Windows common controls can be only solid colors, not gradients, patterns, nor textures.

### To set the background color of a Windows common control

1. Select the Windows common control.
2. In the Properties Editor, click the browse button of the Fill Color property. The **Select Fill Color** dialog box

appears.

3. Select a solid color and click **OK**. For more information, see [Setting a Solid Color](#). The Windows common control background color changes accordingly.

### To set the text color of a Windows common control

1. Select the Windows common control.
2. In the Properties Editor, click the browse button of the TextColor property. The **Select Text Color** dialog box appears.
3. Select a solid color and click **OK**. For more information, see [Setting a Solid Color](#). The Windows common control text color changes accordingly.

## Related Topics

[Using Windows Common Controls](#)

### Reading and Writing the Selected Value at Run Time

You can use the Value property that is common to all Windows common controls. It is not visible in the Properties Editor. You can use the value property in a script or other animation links.

The following table shows you the data type, a description on how the value property is used, and an example for each Windows common control.

Control	Data Type	Description	Example
Radio Button Group	Boolean, Integer, Real or String	Reads the value of the selected item, or selects the item with that value if it exists.	RadioButtonGroup1.Value = "Mixing";
Check Box	Boolean	Sets or reads the checked status.	CheckBox1.Value = 1;
Edit Box	String	Sets or reads the text contents.	EditBox1.Value = "Hello World";
Combo Box	Integer	Reads the value of the selected item, or selects the item with that value if it exists.	ComboBox1.Value = 5;
Calendar	Time	Sets or reads the selected date.	Calendar1.Value = "11/15/2006 11:12:34 AM";
DateTime Picker	Time	Sets or reads the selected date and time.	DateTimePicker1.Value = "11/15/2006 2:55:12 PM";

Control	Data Type	Description	Example
List Box	Integer	Reads the value of the selected item, or selects the item with that value if it exists.	ListBox1.Value = "John Smith";

## Related Topics

[Using Windows Common Controls](#)

### Configuring Radio Button Group Controls

You can use a Radio Button Group control element to exclusively select an option from a group of options at run time.

You can set the:

- 3D appearance of buttons.
- Layout of the radio button group options.

You can also use properties that are specific to the Radio Button Group control in scripting. At run time you can access the script to view and modify the Radio Button Group control.

## Related Topics

[Using Windows Common Controls](#)

[Setting the 3D appearance of a Radio Button Group Control](#)

[Setting the Layout of the Radio Button Group Options](#)

[Using Radio Button Group-Specific Properties at Run Time](#)

# Setting the 3D appearance of a Radio Button Group Control

You can set the 3D appearance of a radio button group control. This affects how the option circles appear.

	Three-dimensional appearance
	Flat appearance in same color as option text

### To set the 3D appearance of a radio button group control

1. Select the radio button group control.
2. In the Properties Editor, select from the list for the ControlStyle property:

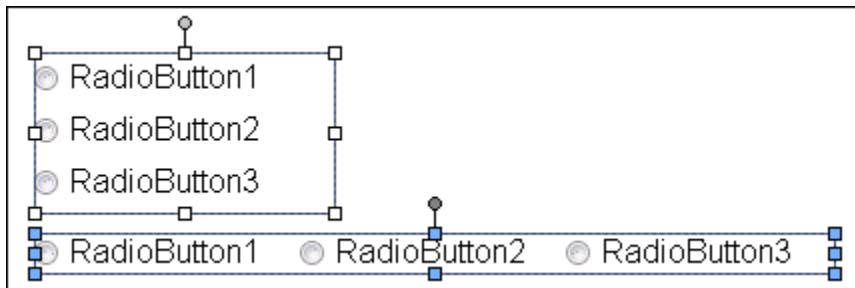
- Click **ThreeD** for a three-dimensional appearance.
- Click **Flat** for a flat two-dimensional appearance in the same color as the option text.

## Related Topics

[Configuring Radio Button Group Controls](#)

# Setting the Layout of the Radio Button Group Options

You can set the layout of the radio button group options in a vertical or horizontal direction.



## To set the layout of the radio button group options

1. Select the radio button group control.
2. In the Properties Editor, select from the list for the Layout property:
  - Click **Vertical** to arrange the options under each other.
  - Click **Horizontal** to arrange the options next to each other.

**Note:** You can set this option also in the radio button group animation dialog box.

## Related Topics

[Configuring Radio Button Group Controls](#)

# Using Radio Button Group-Specific Properties at Run Time

You can use properties that are specific to the Radio Button Group control at run-time. These properties are:

- **Count** - returns the number of radio buttons in the Radio Button Group control.
- **SelectedValue** - reads the value of the selected item, or selects the item with that value if it exists.

These properties are available when you browse for a Radio Button Group control in your element browser.

## Related Topics

[Configuring Radio Button Group Controls](#)

### Configuring Check Box Controls

You can use a Check Box control for users to reset a Boolean equipment.item during run time.

You can set the following properties of the Check Box control:

- Default state, checked or unchecked.
- Caption text of the Check Box control button.
- 3D appearance of the Check Box control button.

## Related Topics

[Using Windows Common Controls](#)

[Setting the Default State of a Check Box Control](#)

[Setting the Caption Text of a Check Box Control](#)

[Setting the 3D appearance of a Check Box Control](#)

# Setting the Default State of a Check Box Control

You can set the default state of a check box control to be checked or unchecked.

### To set the default state of a check box control

1. Select the Check Box control.
2. In the Properties Editor, select from the list for the **Checked** property:
  - Click **False** to use an unchecked check box by default.
  - Click **True** to use a checked check box by default.

## Related Topics

[Configuring Check Box Controls](#)

# Setting the Caption Text of a Check Box Control

You can set the caption text of a Check Box control.

### To set the caption text of a Check Box control

1. Select the Check Box control.
2. In the Properties Editor, type a text string in the Caption property value box.

## Related Topics

[Configuring Check Box Controls](#)

# Setting the 3D appearance of a Check Box Control

You can set the appearance of the check box within the control to be either flat or three-dimensional.

	Three-dimensional appearance
	Flat appearance in same color as caption text

### To set the 3D appearance of a Check Box control

1. Select the check box control.
2. In the Properties Editor, select from the list for the ControlStyle property:
  - Click **ThreeD** for a three-dimensional check box.
  - Click **Flat** for a flat two-dimensional check box in the same color as the caption text.

## Related Topics

[Configuring Check Box Controls](#)

## Configuring Edit Box Controls

You can use an Edit Box control to create a box during run time in which users can enter text or view text.

You can configure the following properties of an Edit Box control:

- Set the default text.
- Wrap text to the next line in the edit box at design time and run time.
- Configure it so that the run-time text is read-only.

## Related Topics

[Using Windows Common Controls](#)

[Setting the Default Text in an Edit Box Control](#)

[Configuring the Text to Wrap in an Edit Box Control](#)

[Configuring the Text to be Read-Only in an Edit Box Control](#)

# Setting the Default Text in an Edit Box Control

You can set the default text that appears in an edit box control during run time.

## To set the default text in an Edit Box control

1. Select the edit box control.
2. In the Properties Editor, type a text in the Text property. The text appears in the edit box control at design time. At run time, it can be overwritten with the value of a configured equipment.item.

## Related Topics

[Configuring Edit Box Controls](#)

# Configuring the Text to Wrap in an Edit Box Control

You can configure the edit box control to wrap text at design time and run time. This lets you view and type strings in a more compact way.

## To configure text-wrapping in an edit box control

1. Select the edit box control.
2. In the Properties Editor, select from the list for the Multiline property:
  - Click **True** to enable text-wrapping at run time.
  - Click **False** to disable text-wrapping at run time.

## Related Topics

[Configuring Edit Box Controls](#)

# Configuring the Text to be Read-Only in an Edit Box Control

You can configure the Edit Box control to only show text at run time and prevent the run-time user from writing back to the associated equipment.item.

## To configure the text to be read-only in an Edit Box control

1. Select the edit box control.
2. In the Properties Editor, set the ReadOnly property to **True**.

---

**Note:** To enable writing back to the associated equipment.item at run time, you can set the **ReadOnly**

---

property to **False**.

---

## Related Topics

[Configuring Edit Box Controls](#)

### Configuring Combo Box Controls

You can use Combo Box controls to select an option from a foldable list.



You can configure:

- Drop-down type of combo box control.
- Width of the drop-down list.
- Integral height flag of the drop-down list to avoid clipping of the items in simple combo box controls.
- Maximum number of items to appear in the drop-down list.

You can also use properties that are specific to the Combo Box control in scripting. At run time, you can access the script to view and modify the items in the Combo Box control.

## Related Topics

[Using Windows Common Controls](#)

[Setting the Type of Combo Box Control](#)

[Setting the Width of the Drop-Down List](#)

[Avoiding Clipping of Items in the Simple Combo Box Control](#)

[Setting the Maximum Number of Items to Appear in the Combo Box Drop-Down List](#)

[Using Combo Box-Specific Properties at Run Time](#)

# Setting the Type of Combo Box Control

You can use one of the following combo box control types:

- Simple - no drop-down list, values can be entered
- DropDown - has a drop-down list, values can be entered
- DropDownList - has a drop-down list, values cannot be entered

### To set the type of Combo Box control

1. Select the combo box control.
2. In the Properties Editor, select from the list for the DropDownType property:

- Simple
- DropDown
- DropDownList

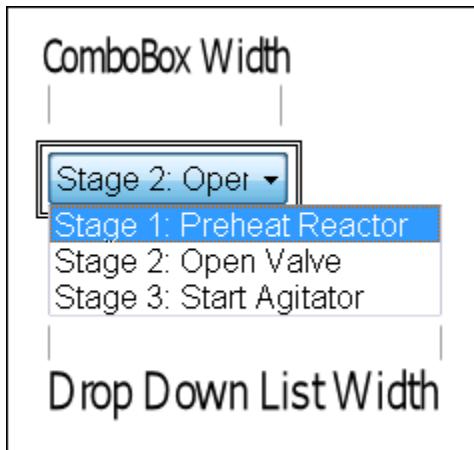
## Related Topics

[Configuring Combo Box Controls](#)

# Setting the Width of the Drop-Down List

You can set the width of the expanded drop-down list when the user clicks on it. This setting can be used to save space of the folded combo box control at run time.

Typically you set the drop-down list width greater than the width of the combo box on the canvas.



If you set the drop-down list width smaller than the combo box control width on the canvas, the drop-down list is the same width as the combo box control.

### To set the width of the combo box drop-down list

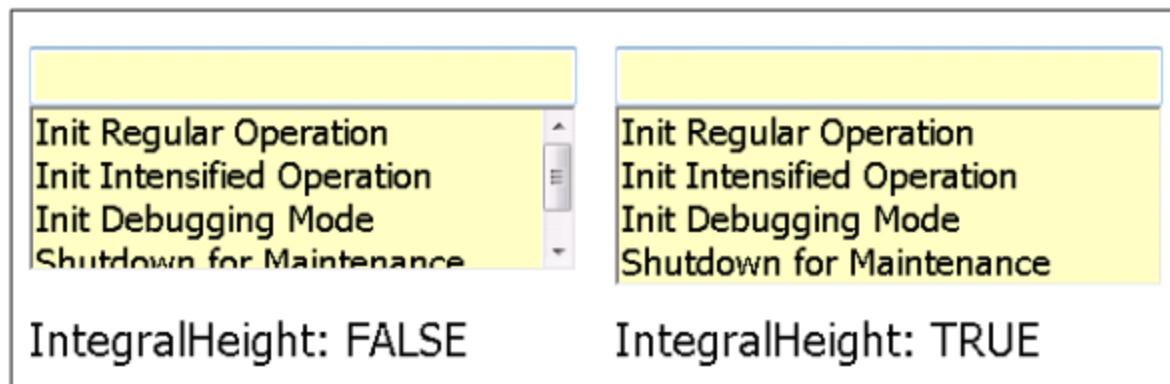
1. Select the combo box control.
2. In the Properties Editor, type a width value in the DropDownWidth property value box.

## Related Topics

[Configuring Combo Box Controls](#)

# Avoiding Clipping of Items in the Simple Combo Box Control

You can avoid clipping of items in the simple combo box control list by setting the IntegralHeight property to true. The combo box list height is then changed so that no items appear clipped.



**To avoid clipping of items in the drop-down list**

1. Select the combo box control.
2. In the Properties Editor, select **True** as the value for the **IntegralHeight** property.

## Related Topics

[Configuring Combo Box Controls](#)

# Setting the Maximum Number of Items to Appear in the Combo Box Drop-Down List

You can limit the number of items that appear at any given time in the combo box drop-down list.

**To set the maximum number of items to appear in the drop-down list**

1. Select the combo box control.
2. In the Properties Editor, type the maximum number as a value for the **MaxDropDownItems** property.

## Related Topics

[Configuring Combo Box Controls](#)

# Using Combo Box-Specific Properties at Run Time

You can use properties that are specific to the Combo Box control at run time.

- The **Count** property returns the number of items in a Combo Box control.
- The **NewIndex** property returns the index of the last item added to the Combo Box list.

These properties are available when you browse for a Combo Box control in your element browser.

## Related Topics

[Configuring Combo Box Controls](#)

### Configuring Calendar Controls

You can use the Calendar control to select a date from one or more monthly calendar sheets.

You can:

- Set the number of calendar month sheets to be shown.
- Set the first day of the week.
- Show or hide today's date on the bottom of the control.
- Set the fill and text colors of the calendar title.
- Set the text color for trailing dates.
- Set the date value of the Calendar Control that is used as default at run time.

## Related Topics

[Using Windows Common Controls](#)

[Setting the Number of Calendar Month Sheets](#)

[Setting the First Day of the Week](#)

[Showing or Hiding Today's Date on a Calendar Control](#)

[Setting Title Fill Color and Text Color on a Calendar Control](#)

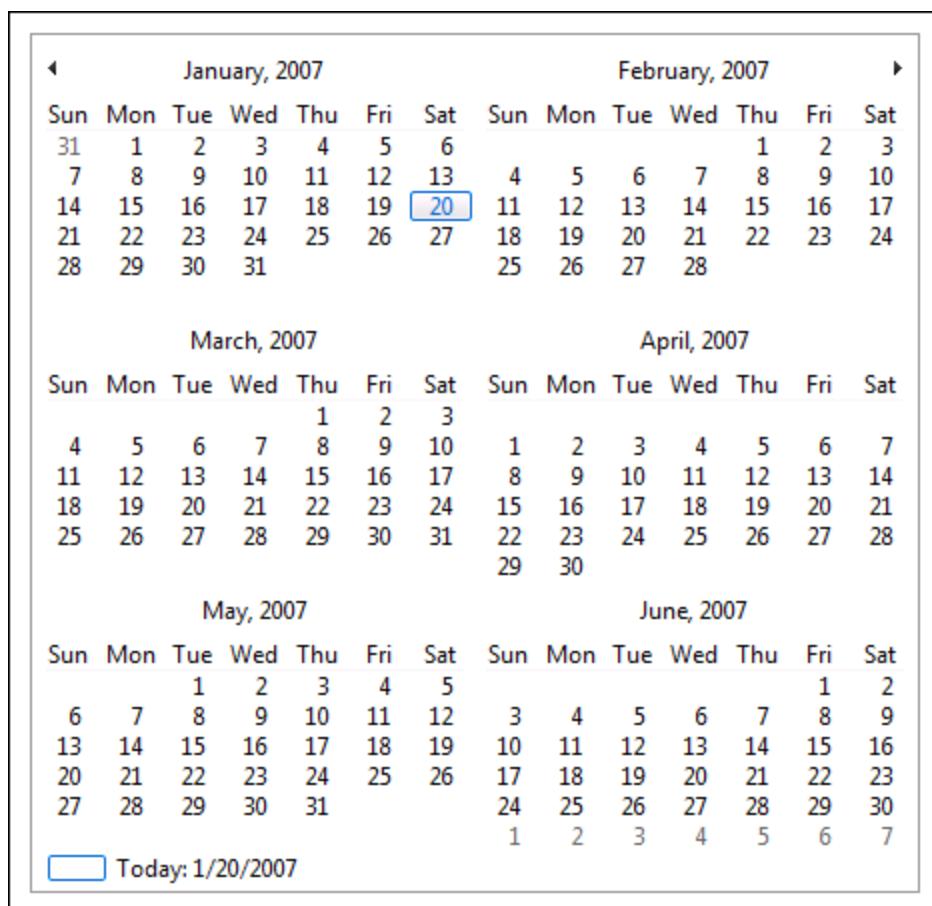
[Setting the Text Color for Trailing Dates in a Calendar Control](#)

[Setting the Default Value of the Calendar Control](#)

# Setting the Number of Calendar Month Sheets

You can set the number of calendar month sheets to be shown by specifying the number of month columns and month rows. The number of columns and rows in the calendar control depends on the font size and the width of the calendar control.

For example, you can show six months in a calendar control by specifying two columns and three rows.



### To set the number of calendar month sheets

1. Select the Calendar control.
2. In the **Properties Editor**, configure the calendar properties:
  - For the **CalendarColumns** property, specify the number of columns in the calendar control.
  - For the **CalendarRows** property, specify the number of rows in the calendar control.

### Related Topics

[Configuring Calendar Controls](#)

## Setting the First Day of the Week

You can set the first day of the week for the calendar control. This is the day that appears on the most left column of each calendar month sheet.

You can set it to:

- The default as defined by the operating system.
- Any day of the week.

**To set the first day of the week**

1. Select the Calendar control.
2. In the **Properties Editor**, select from the list for the **FirstDayOfWeek** property:
  - Click **Default** to use the operating system setting.
  - Click the day of the week.

**Related Topics**

[Configuring Calendar Controls](#)

## Showing or Hiding Today's Date on a Calendar Control

You can show or hide today's date on the bottom of a calendar control

**To show or hide today's date on the bottom of a calendar control**

1. Select the Calendar control.
2. In the Properties Editor, set the **ShowToday** property to one of the following:
  - **True** to show today's date
  - **False** to hide today's date

**Related Topics**

[Configuring Calendar Controls](#)

## Setting Title Fill Color and Text Color on a Calendar Control

You can set the title fill color and title text color on a calendar control.

Changing the title fill color also affects the:

- Color of the week days.
- Fill color of the indication box of today's date.

When you change the title text color, this also affects the text color of the indication box of today's date.



#### To change the title fill color of a Calendar control

1. Select the Calendar control.
2. In the **Properties Editor**, click the browse button for the **TitleFillColor** property. The **Select FillColor** dialog box appears. For more information, see [Setting Style](#).
3. Select a color and click **OK**. The title fill color is changed accordingly.

#### To change the title fill color of a Calendar control

1. Select the Calendar control.
2. In the **Properties Editor**, click the browse button for the **TitleTextColor** property. The **Select FillColor** dialog box appears. For more information, see [Setting Style](#).
3. Select a color and click **OK**. The title text color is changed accordingly.

### Related Topics

[Configuring Calendar Controls](#)

## Setting the Text Color for Trailing Dates in a Calendar Control

You can set the text color for dates outside the month for any month sheet in a calendar control.

#### To set the text color for trailing dates

1. Select the Calendar control.
2. In the **Properties Editor**, click the browse button for the **TrailingTextColor** property. The **Select Text Color** dialog box appears. For more information, see [Setting Style](#).
3. Select a color and click **OK**. The text color of the trailing dates is changed accordingly.

## Related Topics

[Configuring Calendar Controls](#)

# Setting the Default Value of the Calendar Control

You can set the default value of the Calendar Control. The default value is a date that the control uses when it is shown the first time.

### To set the default value of the calendar control

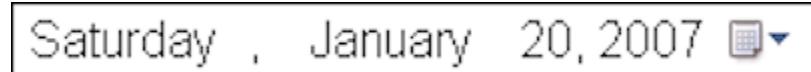
1. Select the Calendar control.
2. In the **Properties Editor**, set the **DefaultValue** property to the date value you want to use as default at run time.

## Related Topics

[Configuring Calendar Controls](#)

## Configuring DateTime Picker Controls

Use the DateTime Picker control to select a date or time.



You can configure the DateTime Picker control to show:

- A long format, such as Friday, August 11, 2008.
- A short format, such as 8/11/2008.
- Just the time, such as 9:16:36 PM.
- A custom time format, such as 8/11/2008 9:16:36 PM.

You can also set the default value of the DateTime Picker control.

### To set the long date format

1. Select the DateTime Picker control.
2. In the Properties Editor, set the Format property to **Long**.

### To set the short date format

1. Select the DateTime Picker control.
2. In the Properties Editor, set the Format property to **Short**.

### To set only time display

1. Select the DateTime Picker control.

2. In the Properties Editor, set the Format property to **Time**.

### To set a custom date/time format

1. Select the DateTime Picker control.
2. In the Properties Editor, set the Format property to Custom.
3. Type the time format in the value box for the CustomFormat property. Use the following letters as placeholders

h	The one or two-digit hour in 12-hour format.
hh	The two-digit hour in 12-hour format. Single digit values are preceded by a zero.
H	The one or two-digit hour in 24-hour format.
HH	The two-digit hour in 24-hour format. Single digit values are preceded by a zero.
t	The one-letter AM/PM abbreviation ("AM" is shown as "A").
tt	The two-letter AM/PM abbreviation ("AM" is shown as "AM").
m	The one or two-digit minute.
mm	The two-digit minute. Single digit values are preceded by a zero.
s	The one or two-digit seconds.
ss	The two-digit seconds. Single digit values are preceded by a zero.
d	The one or two-digit day.
dd	The two-digit day. Single digit day values are preceded by a zero.
ddd	The three-character day-of-week abbreviation.
dddd	The full day-of-week name.
M	The one or two-digit month number.
MM	The two-digit month number. Single digit values are preceded by a zero.
MMM	The three-character month abbreviation.
MMMM	The full month name.

y	The one-digit year (2001 is shown as "1").
yy	The last two digits of the year (2001 is shown as "01").
yyyy	The full year (2001 is shown as "2001").

You can use any other characters, except "g" in the property. These characters then appear at design time and run time in the control.

#### To set the default value in a DateTime Picker control

1. Select the DateTime Picker control.
2. In the Properties Editor, set the DefaultValue property to the date and time value you want to use as default at run time.

### Related Topics

[Using Windows Common Controls](#)

### Configuring List Box Controls

You can create a list box for users to select an option from a scrollable list during run time.

You can:

- Configure the list box to avoid clipping of its contained items. When you set the Integral Height flag, the list box control is resized so that no items are clipped.
- Specify if you want the control to be scrollable in horizontal direction at run time. This enables the user to see the full text if the item captions are wider than the control itself.
- Use properties that are specific to the List Box control in scripting. At run time you can access the script to view and modify the items in the List Box control.

### Related Topics

[Using Windows Common Controls](#)

[Avoiding Clipping of Items in the List Box Control List](#)

[Using a Horizontal Scroll Bar in a List Box Control](#)

[Using List Box-Specific Properties at Run Time](#)

## Avoiding Clipping of Items in the List Box Control List

In the list of a List Box control, some items may appear vertically clipped. You can configure the List Box control to avoid this clipping by setting the IntegralHeight property.

**To avoid clipping of items in the List Box control**

1. Select the list box control.
2. In the **Properties Editor**, select **True** as value for the **IntegralHeight** property.

**Related Topics**[Configuring List Box Controls](#)

## Using a Horizontal Scroll Bar in a List Box Control

You can configure a horizontal scroll bar in a List Box Control so that at run time the user can scroll the list horizontally to see items that are wider than the control.

**To configure a horizontal scroll bar**

1. Select the List Box control.
2. In the **Properties Editor**, select **True** as value for the **HorizontalScrollbar** property.

**Related Topics**[Configuring List Box Controls](#)

## Using List Box-Specific Properties at Run Time

You can use properties that are specific to the List Box control at run time.

- The **Count** property returns the number of items in a List Box control.
- The **NewIndex** property returns the index of the last item added to the List Box list.
- The **SelectedValue** property reads the value of the selected item, or selects the item with that value if it exists.
- The **TopIndex** property returns the index of the top most item in the list.

These properties are available when you browse for a List Box control in your element browser.

**Related Topics**[Configuring List Box Controls](#)

## Using Custom Properties

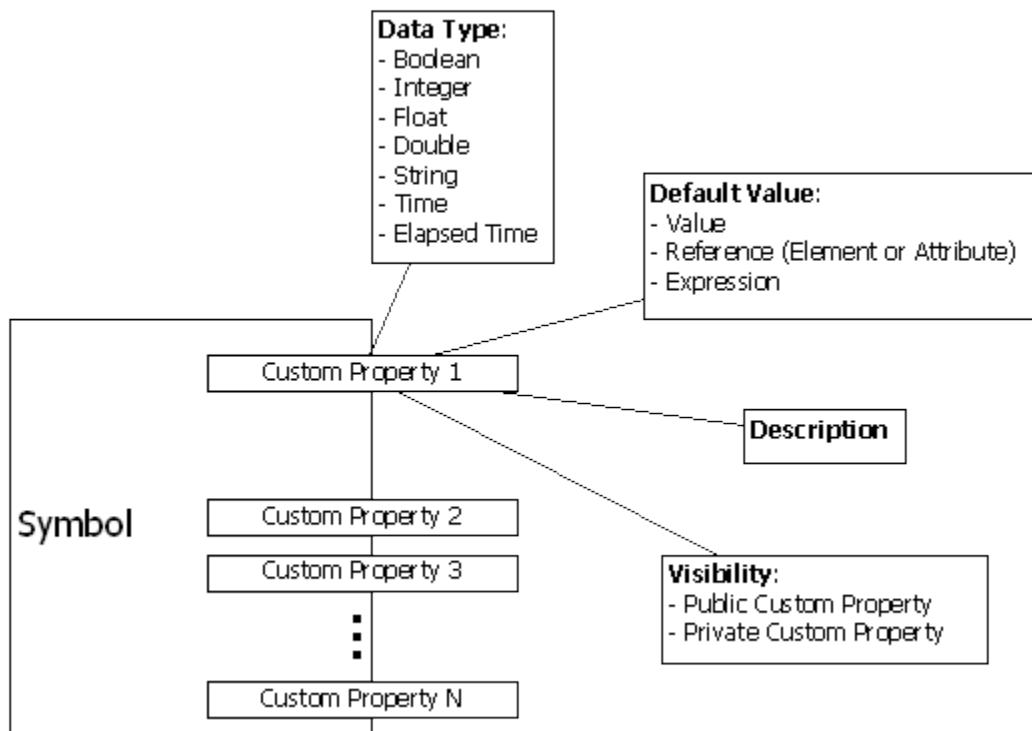
### In This Chapter

[About Custom Properties](#)

[Managing Custom Properties](#)

### About Custom Properties

You can configure and use custom properties to extend the functionality of symbols and use them in combination with the HMI/SCADA software equipment.items or tags. You can use binding with custom properties to dynamically change the reference of a custom property.



You can associate custom properties with functionality you want exposed and that you want to be reusable. You can also use custom properties to connect an embedded graphic to equipment.items and tags in your HMI/SCADA software.

### Related Topics

[Using Custom Properties](#)

### Managing Custom Properties

You manage all custom properties of a graphic using the **Edit Custom Properties** dialog box.

From the **Custom Properties** dialog box, you can:

- Add and delete custom properties.
- Set the types and data types of custom properties.
- Set the default values of custom properties.
- Determine the visibility of each custom property.
- Add a description for each custom property.
- Validate and clear custom properties.

You can also:

- Rename custom properties.
- Link custom properties to external sources.
- Override custom properties with new values.
- Revert custom property values to their default values.

## Related Topics

[Using Custom Properties](#)

[Adding and Deleting Custom Properties](#)

[Configuring Custom Properties](#)

[Validating Custom Properties](#)

[Clearing the Configuration of Custom Properties](#)

[Renaming Custom Properties](#)

[Linking Custom Properties to External Sources](#)

[Overriding Custom Properties](#)

[Reverting to Original Custom Property Values](#)

## Adding and Deleting Custom Properties

You can add and delete custom properties from a graphic.

### To add a custom property

1. Click the canvas to cancel any selected elements.
2. On the **Special** menu, click **Custom Properties**. The **Edit Custom Properties** dialog box appears.
3. Click the **Add** icon. A new line is added in the custom properties list.
4. Type a name for the new custom property and click **Enter**.

You can see the name of the graphic and the custom property in the header of the dialog box.

- If the graphic includes an embedded graphic, the name of the custom property cannot be the same as the name of the embedded graphic or of an element of the embedded graphic.
  - If the graphic includes a script, the name of the custom property and a nested class property in the script cannot be the same.
5. Configure the custom property on the right side of the **Edit Custom Properties** dialog box. For more information, see [Configuring Custom Properties](#).

6. Click **OK**.

### To delete a custom property

1. Click the canvas to cancel any selected elements.
2. On the **Special** menu, click **Custom Properties**. The **Edit Custom Properties** dialog box appears.
3. Select the custom property you want to delete and click the **Remove** icon. When a message appears requesting confirmation to delete the custom property, click **Yes**. The custom property is removed from the custom properties list.
4. Click **OK**.

## Related Topics

[Managing Custom Properties](#)

### Configuring Custom Properties

You can configure custom properties when you create them or at a later point of time.

#### To configure a custom property

1. Click the canvas of the graphic.
2. On the **Special** menu, click **Custom Properties**. The **Edit Custom Properties** dialog box appears.
3. Select the custom property you want to edit. The configuration for the selected custom property appears at the right of the dialog box.

**Note:** The header of the configuration area shows the graphic name on the right, for example "Symbol\_001," and it shows the custom property name on the left, for example "MyCustomProperty." Its reference in a script would be "Symbol\_001.MyCustomProperty."

4. In the **Data Type** list, click the data type of the custom property. You can select one of the following:

Data Type	Graphic
Boolean	
Double	
Elapsed Time	
Float	



5. If you want to:
  - Make the property read-only at design time and prevent further changes to it when the graphic is embedded into another graphic, click the Lock icon.
  - Make the property read-only at run time and prevent its value being changed, click the Lock icon.
6. In the **Default Value** box, type a literal value, reference, or expression or browse for a reference using the **Browse** icon.
7. If the selected data type is String, Time or Elapsed Time, you can click the **T** icon or tag icon.
  - Select the **T** icon to indicate that the default value is a static value.
  - Select the tag icon to indicate that the default value is a reference to a value.
8. In the **Visibility** box, configure how the graphic is visible. Do one of the following:
  - Click **Public** if you want the custom property to be visible and can be used in a source graphic if the graphic is embedded.
  - Click **Private** if you want the custom property to be hidden and no reference be made to it outside of the defining graphic.
9. In the **Description** box, type a meaningful description for the custom property.

## Related Topics

[Managing Custom Properties](#)

## Validating Custom Properties

You can validate custom properties to track down and avoid configuration errors.

### To validate a custom property

1. Click on the canvas to cancel any selected elements.
2. On the **Special** menu, click **Custom Properties**. The **Edit Custom Properties** dialog box appears.
3. Select the custom property you want to validate and click the **Validate** icon. Required boxes are highlighted by a red box, possible errors appear in the status area under the custom properties list.

## Related Topics

[Managing Custom Properties](#)

## Clearing the Configuration of Custom Properties

You can clear the configuration of custom properties. This resets the properties to their default values.

### To clear the configuration of a custom property

1. In the **Edit Custom Properties** dialog box, select the custom property.
2. Click the **Clear** icon. The configured values are reset to their default values.

## Related Topics

[Managing Custom Properties](#)

## Renaming Custom Properties

You can rename custom properties.

### To rename a custom property

1. In the **Edit Custom Properties** dialog box, select the custom property.
2. Click the custom property again. The custom property is in edit mode.
3. Type the new custom property name and click **Enter**. The custom property is renamed.

---

**Note:** If the graphic includes an embedded graphic, the name of the custom property cannot be the same as the name of the embedded graphic or of an element of the embedded graphic. If the graphic includes a script, the name of the custom property and a nested class property in the script cannot be the same.

## Related Topics

[Managing Custom Properties](#)

## Linking Custom Properties to External Sources

You can link custom properties of a graphic directly to external sources by:

- Configuring objects that point to external sources and then point the custom property at the corresponding equipment.item reference.

## Related Topics

[Managing Custom Properties](#)

## Overriding Custom Properties

You can override the custom property default values of embedded graphics within graphics in the Industrial Graphic Editor.

---

**Note:** When you override the custom property, it appears bold in the custom property list.

You can override the following custom property values:

- Default value
- Visibility, but only from public to private, not private to public
- Description
- Locked state
- String mode setting

You cannot override the data type of a custom property.

## Related Topics

[Managing Custom Properties](#)

### Reverting to Original Custom Property Values

After you override a custom property value, you can revert to the original custom property value. This can be done for overridden custom properties of embedded graphics in other graphics.

#### To revert to the original custom property value

- In the **Edit Custom Properties** dialog box, click the Revert icon. The custom property value reverts to its original value.

## Related Topics

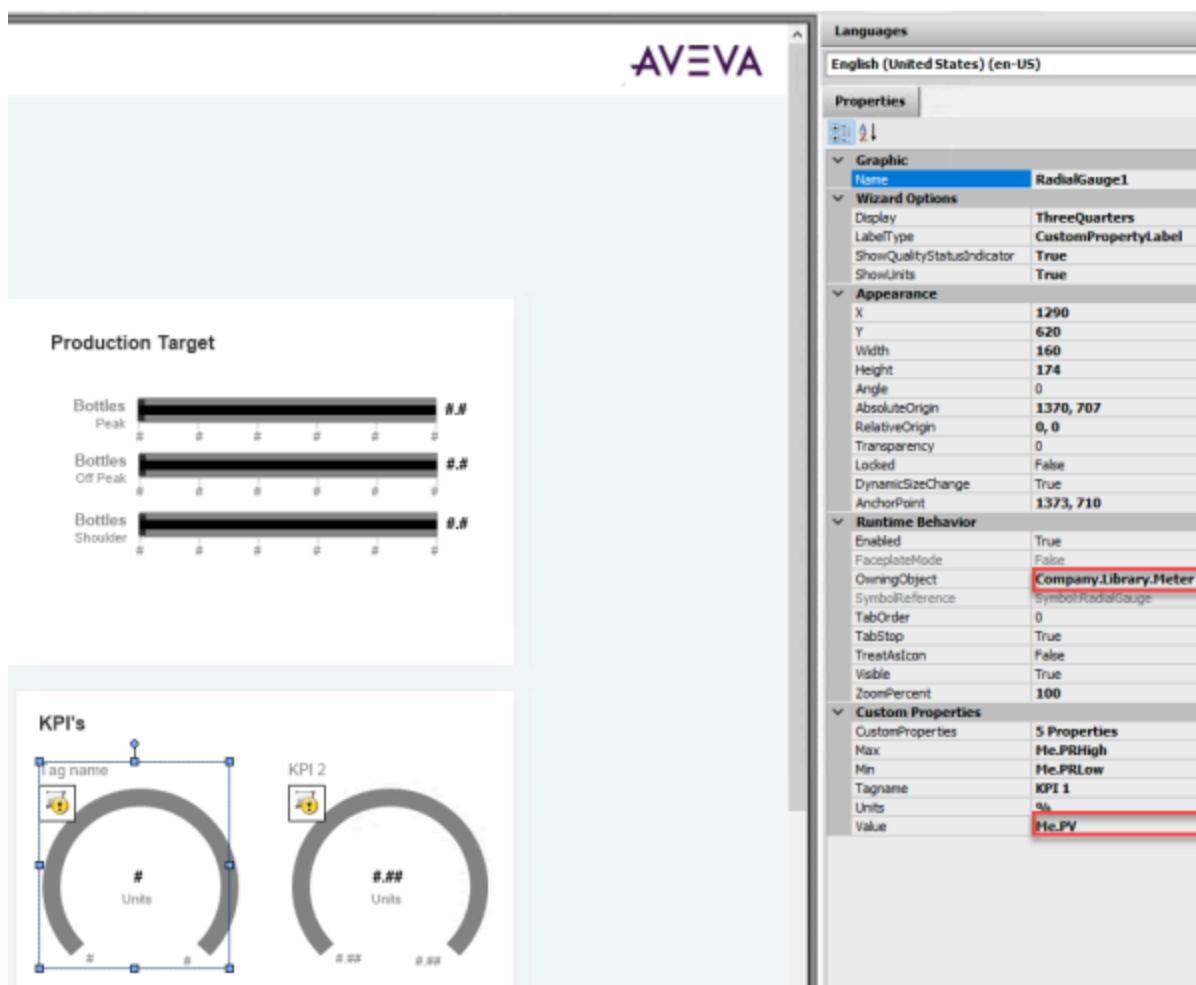
[Managing Custom Properties](#)

## Owning Object and Relative Referencing

For an embedded symbol, you can specify an owning object. This allows you to associate the symbol with a cluster, tag or equipment reference. When you configure a symbol with relative references to an owning object, the embedded symbol will reflect values in the context of the owning object.

## Example

A radial gauge symbol appears on the Industrial Graphics page named **Overview** in the Plant SCADA's **ExampleSA** project. If you open this page from the **Visualization** activity, it will appear in the Industrial Graphics Editor. Select the radio gauge object to view its **Properties** in the pane to the right of the editor.



- **Owning Object:**

The **OwningObject** property in the **Runtime Behaviour** section has the value "Company.Library.Meter". This is the Meter Equipment as defined in the project's equipment model. Any relative references will refer to this equipment.

- **Relative Reference:**

In the example, the Max, Min and Value Custom properties are using relative references to equipment items "PRHigh", "PRLow" and "PV" respectively. This is indicated by the use of the "Me" reserved keyword.

At runtime, "Me" will be substituted with the Owning Object reference. Using the "Value" property, with the value of "Me.PV" as an example, at runtime a subscription will be made not to Me.PV, but to Company.Library.Meter.PV.

## Configure an Owning Object

Follow the steps to define a cluster, tag or equipment reference as the owning object for an embedded symbol.

1. Open a page in AIG Editor from the **Visualization** activity, and embed a symbol on it.
2. Select the embedded symbol to view the **Properties** section in the right-side pane.
3. Enter a cluster, tag or equipment reference in the **OwningObject** property of the **Runtime Behaviour** section.

4. Save and close the page.

## Configure a Relative Reference

Relative referencing is adding references to the embedded symbol, so that in runtime the symbol will reflect the value of the owned object. Configure references using the "Me" keyword followed by the Tag, or Item, or partial Equipment.Item, as per the requirement. For example: Me.Tag1, Me.PV or Me.Equipment.PV.

Following are two methods to configure references to an embedded symbol:

- Hard-coding relative references of custom properties, animation expression, or script within the symbol
- Relative referencing in custom properties of an embedded symbol.

### Hard coding Relative References of custom properties, animation expression, or script within the symbol

To hard code relative references:

1. Open a symbol in AIG Editor from the **Visualization** activity.
2. Open a [Configuring a Value Display Animation](#) for an element in the symbol.
3. Specify a relative reference "Me.<tag name/Item/Equipment.Item>" in **Expression Or Reference**. This should exist for the intended owning object expect to be set when the symbol is embedded. Click **OK**.
4. Save and close the symbol.
5. Open a page in AIG Editor from the **Visualization** activity.
6. Embed the symbol in the page, and specify an owning object for the symbol.
7. Save and close the page.

At runtime, the configured element will reflect the value of the owned object. If the owned object does not have the specified tag (in relative reference), you cannot reconfigure it in the page. Hence, in runtime, it will not reflect the <tag/item/equipment.item> value.

### Relative Referencing in Custom Properties of an Embedded Symbol

To add relative reference for an embedded symbol:

1. Open a symbol in the Industrial Graphics Editor from the **Visualization** activity.
2. [Configuring Custom Properties](#) for the symbol.
3. Open a [Configuring a Value Display Animation](#) for an element in the symbol.
4. Specify the configured custom property in **Expression Or Reference**, and click **OK**.
5. Save and close the symbol.
6. Open a page in AIG Editor from the **Visualization** activity.
7. Embed the symbol in the page, and specify an owing object for the symbol.
8. Under **Custom Properties** specify the configured custom property value as "Me.<tag/item/equipment.item>". This should exist for the specified owning object.
9. Save and close the page.

At runtime, the configured element will reflect the value of the resolved reference belonging to the owning object.

## Working with Element Styles

Styles can be used to manage the appearance of elements, status animations and numeric values in AVEVA™ Industrial Graphics applications.

A style defines a set of visual properties that determine the appearance of elements such text, lines, graphic outlines, and interior fill (see [Visual Properties Defined by Element Styles](#)).

Styles are defined in Plant SCADA Studio's **Standards** activity. The **Styles** view hosts the **Configure Styles** dialog, which allows you to configure an overriding style for a selected visual element.

For more information, see the topic [Styles](#) in the *Standards* section of the Plant SCADA documentation.

Style can then be applied to selected objects in the Industrial Graphics Editor.

## See Also

[Applying Element Styles to Elements](#)

[Applying Element Styles to Groups of Elements](#)

[Configuring an Animation Using Element Styles](#)

## Visual Properties Defined by Element Styles

The following table lists the visual properties of graphic elements defined in an Element Style.

Graphic Element	Element Properties
Text	<ul style="list-style-type: none"><li>• Font family</li><li>• Font size</li><li>• Font style</li><li>• Font color</li><li>• Blink On/Off</li></ul>
Fill	<ul style="list-style-type: none"><li>• Fill color</li><li>• Fill gradient</li><li>• Fill pattern</li><li>• Fill texture</li><li>• Blink On/Off</li></ul>
Line	<ul style="list-style-type: none"><li>• Line pattern</li><li>• Line weight</li></ul>

Graphic Element	Element Properties
	<ul style="list-style-type: none"><li>• Line color</li><li>• Blink On/Off</li></ul>
Outline	<ul style="list-style-type: none"><li>• Outline Show/Hide</li><li>• Outline Pattern</li><li>• Outline Weight</li><li>• Outline Color</li><li>• Blink On/Off</li></ul>

An Element Style may not define every visual property. If a property value is not defined in an applied Element Style, the element's native style is used and can be changed. However, if an element's property value is defined in an applied Element Style, the element's native properties are disabled and cannot be changed.

## Related Topics

[Working with Element Styles](#)

## Applying Element Styles to Elements

You can apply Element Styles to one or more graphic elements. Unlike setting Element Style overrides that change the appearance of an Element Style's properties, applying an Element Style to a graphic element overrides the element's native properties.

Applying Element Styles to graphic elements can help standardize the appearance of those elements and show the current state of an object represented by a symbol or graphic.

For more information, see the topic [Styles](#) in the *Standards* section of the Plant SCADA documentation.

## Related Topics

[Using the Element Style List](#)  
[Using the Properties Grid](#)  
[Using Format Painter](#)  
[Clearing an Element Style](#)

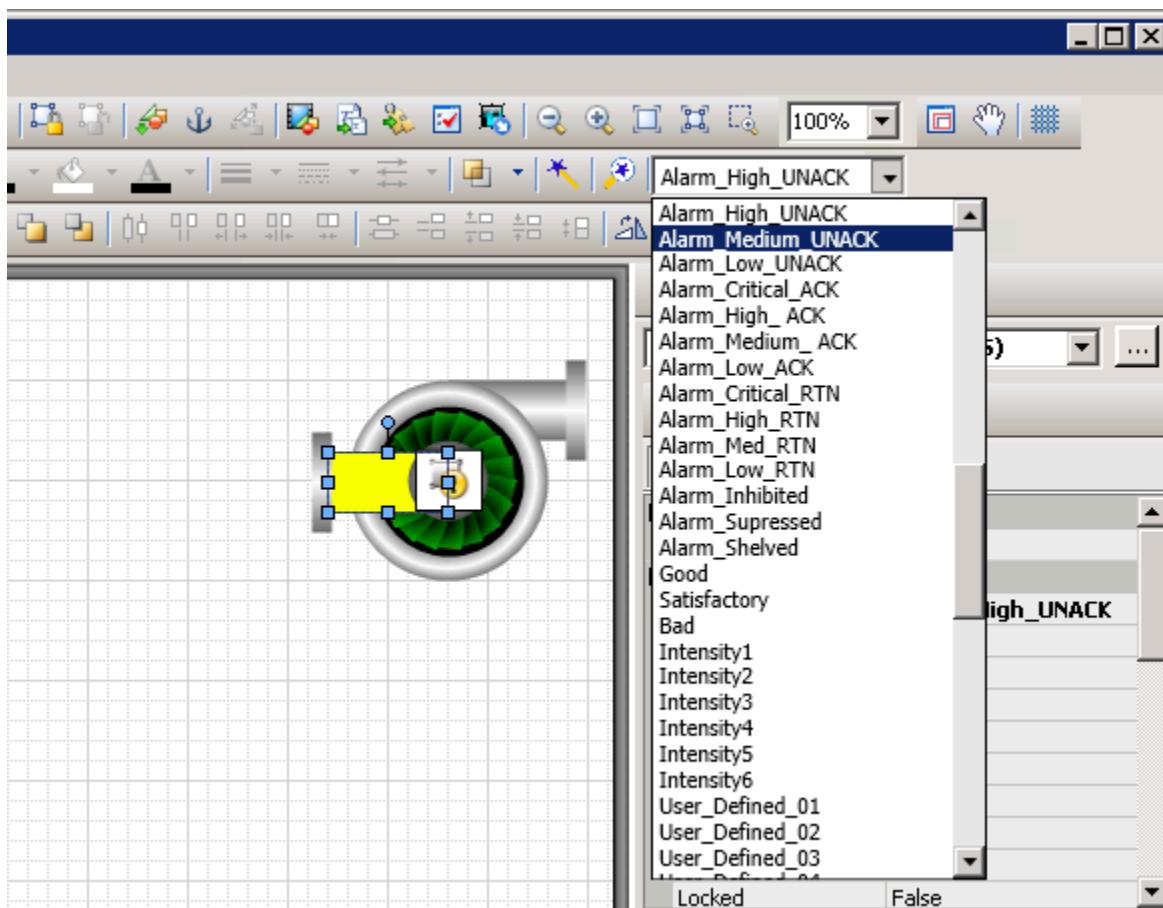
## Using the Element Style List

The Industrial Graphic Editor menu bar contains an **Element Style** list to select an Element Style and apply it to a selected element of a graphic.

### To apply an Element Style to a graphic element

1. Open the graphic in the Industrial Graphic Editor.

2. Select one or more elements from the graphic.
3. Select an Element Style from the **Element Styles** list to apply to the selected elements.



## Related Topics

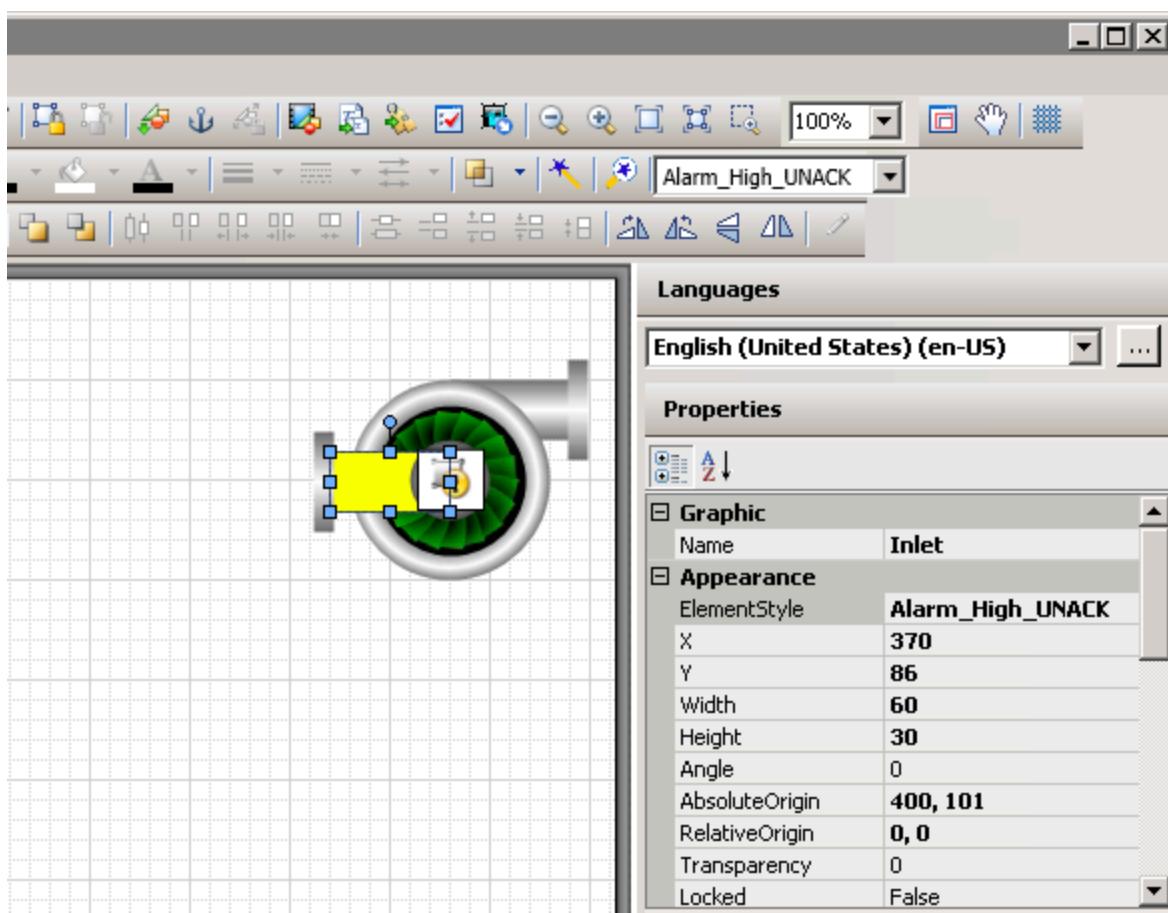
[Applying Element Styles to Elements](#)

## Using the Properties Grid

The Industrial Graphic Editor **Properties** view contains an **Element Style** Appearance item to select an Element Style and apply it to a selected element of a graphic.

### To apply an Element Style from the Properties Editor

1. Open the symbol or graphic in the Industrial Graphic Editor.
2. Select one or more elements from the graphic or symbol.
3. In the **Appearance** category of the **Properties** Editor, select an Element Style from the **Element Style** list.



## Related Topics

[Applying Element Styles to Elements](#)

### Using Format Painter

You can use the Industrial Graphic Editor's Format Painter to copy an Element Style from one graphic element to another.

#### To apply an Element Style using Format Painter

1. Open a graphic in the Industrial Graphic Editor.
2. Select the element with the Element Style you want to copy.
3. On the **Edit** menu, click **Format Painter**. The pointer appears as the Format Painter cursor.
4. Select the elements you want to apply the Element Style to. The Element Style is applied to the selected elements.

## Related Topics

[Applying Element Styles to Elements](#)

## Clearing an Element Style

When an Element Style is applied to an element, you cannot edit the element's styles that are controlled by the applied Element Style. However, you can clear the application of the Element Style so that all of the styles can be edited.

### To clear an Element Style

1. Select the element.
2. Select **None** in the Element Style list.

## Related Topics

[Applying Element Styles to Elements](#)

## Selecting an Element Style as a Default for a Canvas

You can select an Element Style at the canvas level of the Industrial Graphic Editor. The selected Element Style is applied to any graphic element or groups that you create on the canvas.

## Related Topics

[Applying Element Styles to Elements](#)

## Applying Element Styles to Groups of Elements

You can apply an Element Style on a group of elements in the same way that you apply an Element Style to a single graphic element. To enable group support for Element Styles, set the group's run-time behavior to **TreatAsIcon**.

## Related Topics

[Working with Element Styles](#)

[Setting a Group's Run-time Behavior to TreatAsIcon](#)

[Understanding Element Style Behavior with a Group of Elements](#)

## Setting a Group's Run-time Behavior to TreatAsIcon

To apply an Element Style to a graphic element group, set the group's **TreatAsIcon** property 'to True. Otherwise, the Element Style lists are disabled when an element group is selected.

### To set a group's TreatAsIcon property to true

1. Select the element group to which the Element Style will be applied.
2. On the **Properties** menu, click **Run-time Behavior** and click **TreatAsIcon**.
3. Select **True** from the drop-down list.

## Related Topics

[Applying Element Styles to Groups of Elements](#)

### Understanding Element Style Behavior with a Group of Elements

- The Element Style applied to a group has higher precedence than the property styles applied to individual graphic elements in the group.
- If the Element Style applied to a group of elements has undefined property styles, then the element continues to use its Element Style or element-level settings for undefined property styles.
- If the Element Style that is applied to a group of elements has defined property styles, then those property styles override the property styles defined at the element level for elements in the group.
- An Element Style cannot be applied to a nested element group.
- If you add an element to a group that has a group-level Element Style applied, the group Element Style is applied to it.

## Related Topics

[Applying Element Styles to Groups of Elements](#)

### Configuring an Animation Using Element Styles

You can configure an element or a group of elements with a:

- Boolean animation that applies Element Styles based on a binary True/False condition.
- Truth table animation that applies Element Styles based on a range of possible values.

The truth table animation that applies Element Styles:

- Associates expressions of any supported data type to an Element Style.
- Defines as many conditions as required and applies a separate Element Style for each condition
- Defines the conditions to apply an Element Style by specifying a comparison operator (=, >, >=, <, <=) and a breakpoint, which itself can be a value, an equipment.item reference, or an expression.
- Arranges conditions in the order that Element Styles are processed.

---

**Note:** Bit state animations are not supported by Plant SCADA.

---

## Related Topics

[Working with Element Styles](#)

[Configuring a Boolean Animation Using Element Styles](#)

[Configuring a Truth Table Animation with Element Styles](#)

## Configuring a Boolean Animation Using Element Styles

You can configure an element or a group of elements with a Boolean animation that uses only two Element Styles.

### To configure an element or a group of elements with an Element Style that uses Boolean animation

1. Open the symbol or graphic in the Industrial Graphic Editor.
2. Select the element or element group.
3. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
4. Click the **Add** icon and select **Element Style**. The Element Style animation is added to the Animation list and the **Element Style** state selection panel appears.
5. Click the **Boolean** button. The **Boolean Element Style** configuration panel appears.
6. In the **Boolean** text box, enter a Boolean numeric value, equipment.item reference, or an expression.
7. Clear **ElementStyle** in the **True, 1, On** area or **False, 0, Off** area if you do not want a different Element Style for the true or false condition than the default Element Style that is shown in the **Element Style** list.
8. In the **True, 1, On** area, select the Element Style in the list to use when the expression is true.
9. In the **False, 0, Off** area, select the Element Style in the list to use when the expression is false.
10. Click **OK**.

## Related Topics

[Configuring an Animation Using Element Styles](#)

## Configuring a Truth Table Animation with Element Styles

You can configure an element or a group of elements with a Truth Table animation that can select multiple Element Styles based on a set of evaluated values or expressions.

### To configure an element or a group of elements with an Element Style that uses Truth Table animation

1. Open the symbol or graphic.
2. Select the element or group.
3. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
4. Click the **Add** icon and select **Element Style**. The Element Style animation is added to the Animation list and the **Element Style** state selection panel appears.
5. Click the **Truth Table** button. The **Truth Table Element Style** configuration panel appears. The Element Style that is applied to the element is shown in the **Element Style** list at the bottom of the panel.
6. In the **Expression Or Reference** area:
  - Select the data type of the expression from the list.
  - Type a value, equipment.item reference or expression in the text box.
7. If the data type of the expression is string or internationalized string, you can specify to ignore the case by selecting **Ignore Case**.

8. In the **Truth Table**, select the **Element Style** check box and select the Element Style for one of the conditions to be defined in the truth table.
9. In the **Operator** column, select a comparison operator.
10. In the **Value or Expression** column, type a value, equipment.item reference, or expression.
11. To add other conditions:
  - a. Click the **Add** icon. An additional condition is added to the truth table.
  - b. Select the **Element Style** check box, select the Element Style for the condition, select an operator, and enter the condition value or expression.
12. After adding all truth table conditions, click **OK**.

Truth Table animation is typically used to set Element Styles to the different states of an object. For example, you can set Truth Table conditions to show different Element Styles that represent the following alarm conditions:

- When the equipment.item TankLevel\_001.PV is 0 then no Element Style is applied.
- When the equipment.item TankLevel\_001.PV is less than 20, then the Element Style is Alarm\_Minor\_Dev.
- When the equipment.item TankLevel\_001.PV is greater than the equipment.item Standards.TankMax then the Element Style is Alarm\_Major\_Dev.

## Related Topics

[Configuring an Animation Using Element Styles](#)

[Deleting a Condition from an Animation Truth Table](#)

[Changing the Processing Order of Element Styles in a Truth Table Animation](#)

# Deleting a Condition from an Animation Truth Table

You can delete a condition from an animation Truth Table to remove the associated Element Style from the animation.

### To delete a condition from a Truth Table animation that uses Element Styles

1. Open the **Edit Animations** dialog box, **Truth Table Element Style** panel.
2. Select the condition you want to delete.
3. Click the **Remove** icon. The condition is removed.

## Related Topics

[Configuring a Truth Table Animation with Element Styles](#)

# Changing the Processing Order of Element Styles in a Truth Table Animation

You can change the processing order of Element Styles by moving the conditions up or down in the Truth Table list. The Element Style at the top of the Truth Table list is processed first. The remaining Element Styles are processed in order based on their position from the top of the list.

## To change the processing order of Element Style conditions

1. Open the **Edit Animations** dialog box, **Truth Table Element Style** panel.
2. Select the condition you want to move up or down the condition list in order for it to be processed sooner or later.
3. Click the:
  - **Arrow up** icon to move the condition up in the truth table.
  - **Arrow down** icon to move the condition down in the truth table.

## Related Topics

[Configuring a Truth Table Animation with Element Styles](#)

## Animating Graphic Elements

### In This Chapter

- [About Animations](#)
- [Adding an Animation to an Element](#)
- [Reviewing which Animations are Assigned to an Element](#)
- [Showing and Hiding the Animation List](#)
- [Removing Animations from an Element](#)
- [Enabling and Disabling Animations](#)
- [Validating the Configuration of an Animation](#)
- [Clearing the Configuration of Custom Properties](#)
- [Managing Animations](#)
- [Configuring Common Types of Animations](#)
- [Configuring Element-Specific Animations](#)
- [Cutting, Copying and Pasting Animations](#)
- [Substituting References in Elements](#)

### About Animations

You can use animations to change the appearance of graphic elements at run time. Animations are driven by

data that comes from equipment.item or tag values and expressions as well as element properties.

You can use:

- **Visualization animations** such as visibility, fill style, line style, text style, blinking, percent fill horizontal, percent fill vertical, horizontal location, vertical location, width, height, orientation, value display or tooltip.
- **Interaction animations** such as disable, user input, horizontal slider, vertical slider, pushbutton, action script, show graphic or hide graphic.
- **Element-specific animations** for the Status element and Windows common control elements.

Each element in your Industrial graphic can have one or more animations. You can disable and enable individual animations. You can also cut, copy and paste animations between elements. Only animations supported by the target element are pasted.

You can also substitute references and strings in animations.

---

**Note:** Not all animations are available for all element types. Some animations do not make logical sense, such as line style with a text element. You cannot select or copy these invalid combinations.

---

## Related Topics

[Animating Graphic Elements](#)

### Adding an Animation to an Element

You can add one or more animations to a single element in your Industrial graphic.

#### To add an animation to an element

1. Select the element to which you want to add an animation.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.  
You can also add an animation from the Animation Summary in the lower right corner of the Industrial Graphic Editor.
3. Click the **Add** icon. The list of animations appears.
4. Select an animation from the list. The animation is added to the Animation list. You can configure the selected animation from the Edit Animations dialog box.

---

**Note:** Depending on the animation type, you may get an animation state selection panel instead. For more information, see [Reviewing which Animations are Assigned to an Element](#).

---

## Related Topics

[Animating Graphic Elements](#)

### Reviewing which Animations are Assigned to an Element

You can review which animations are assigned to an element and change the number of animations or their configuration at the same time.

## To review which animations are assigned to an element

1. Select the element. The assigned animations appear in the Animation Summary in the lower right of the Industrial Graphic Editor.  
You can also review which animations are assigned to an element by double-clicking it.
2. Select an animation to view further information on how the element is configured with that animation.

## Related Topics

[Animating Graphic Elements](#)

## Showing and Hiding the Animation List

You can show or hide the Animation list. If you hide the Animation list, the configuration space expands, giving you more space to configure the animations.

### To hide the Animation list

- In the **Edit Animations** dialog box, click the **Hide** icon. The Animation list hides and the configuration space expands.

### To show the Animation list

- In the **Edit Animations** dialog box, click the **Show** icon. The Animation list appears and the configuration space reduces to its default width.

## Related Topics

[Animating Graphic Elements](#)

## Removing Animations from an Element

You can remove an animation from an element by using the **Edit Animations** dialog box. You can remove animations from an element for:

- Individual animations
- All animations at the same time

### To remove an animation from an element

1. Select the element in which you want to remove an animation.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.  
You can also remove an animation from the Animation Summary in the lower right of the Industrial Graphic Editor.
3. Select the animation you want to remove from the Animation list.
4. Click the **Remove** icon. A message appears.

5. Click **Yes**. The animation is removed from the list and no longer associated with the element.

### To remove all animations from an element

1. Select one or more elements from which you want to remove all animations.
2. Do one of the following:
  - Right-click, point to **Animations** and then click **Clear**.
  - On the **Edit** menu, point to **Animations**, and then click **Clear**.

All animations are removed from the selected elements.

## Related Topics

[Animating Graphic Elements](#)

## Enabling and Disabling Animations

You can enable or disable animations for an element. When you disable an animation, its configuration is not lost. This lets you see, for example, each animation independently from each other.

### To disable an animation

1. Select the element with the animation you want to disable.
2. On the **Special** menu, click **Edit Animations**. The Edit Animations dialog box appears.  
You can also disable animations from the Animation Summary in the lower right corner of the Industrial Graphic Editor.
3. Locate the animation you want to disable from the Animation list on the left of the dialog box.
4. Select **Disabled** from the list of that row.
5. Repeat for any other animations you want to disable and click **OK** when you are done.

### To enable an animation

1. Select the element with the animation you want to enable.
2. On the **Special** menu, click **Edit Animations**. The Edit Animations dialog box appears.  
You can also enable animations from the Animation Summary of the Industrial Graphic Editor.
3. Locate the animation you want to enable from the Animation list.
4. Select **Enabled** from the list of that row.
5. Repeat for any other animations you want to enable and click **OK** when you are done.

## Related Topics

[Animating Graphic Elements](#)

## Validating the Configuration of an Animation

You can validate the configuration of an animation. If the configuration contains an error, an exclamation mark appears next to the **Animation** icon.

Examples of animation configuration errors include:

- Animation is disabled
- Syntax errors such as data mismatches
- Required values not specified
- Specified values out of valid range

### To validate the configuration of an animation

1. Select the element that contains the animations you want to validate.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Select the animation you want to validate.
4. Click the **Validate** icon. The currently selected animation is validated. Possible errors are highlighted.

## Related Topics

[Animating Graphic Elements](#)

## Clearing the Configuration from an Animation

You can clear all data from the configuration boxes of an animation and reset the settings to their defaults.

### To clear all data from the configuration boxes of an animation

1. In the **Edit Animations** dialog box, select the animation.
2. In the configuration panel, click the **Clear** icon. All data from the configuration boxes is cleared and the settings are reset to their defaults.

## Related Topics

[Animating Graphic Elements](#)

## Managing Animations

You can easily manage animations in the **Edit Animations** dialog box. You can:

- Change the way the list of animations appears.
- Switch easily between multiple animations of an element.

You can also do this for the Animation Summary in the lower right corner of the Industrial Graphic Editor.

## Related Topics

- [Animating Graphic Elements](#)
- [Organizing the Animation List](#)
- [Switching between Animations](#)

### Organizing the Animation List

You can organize the list of animations alphabetically or by category.

#### To organize the Animation list

- In the **Edit Animations** dialog box, click the:
  - **Alphabetic sort** icon to sort alphabetically.
  - **Category** icon to sort by category.

## Related Topics

- [Managing Animations](#)

### Switching between Animations

If you configure more than one animation for an element, you can easily switch between their configuration panels without having to use the Animation list. This is particularly useful when the Animation list is hidden.

#### To switch between animations

- In the **Edit Animations** dialog box, on the configuration panel click the left or right arrow icon.  
The configuration panel changes to the configuration panel of the previous or next animation.

## Related Topics

- [Managing Animations](#)

### Configuring Common Types of Animations

Every animation type has its own set of configuration parameters. This section shows you how to configure each type of animation and what references it can use.

You can configure:

- Visualization animations such as:
  - Visibility animations
  - Fill style, line style or text style animations
  - Blink animations
  - Alarm Border animations

- Horizontal or vertical percent fill animations
- Horizontal or vertical location animations
- Width or height animations
- Point animations
- Orientation animations
- Value display animations
- Tooltip animations
- Interaction animations such as:
  - Disable animation
  - User input animation
  - Horizontal and vertical slider animations
  - Pushbutton animations
  - Action script animations
  - Show or hide animations

## Related Topics

[Animating Graphic Elements](#)  
[Configuring a Visibility Animation](#)  
[Configuring a Fill Style Animation](#)  
[Configuring a Line Style Animation](#)  
[Configuring a Text Style Animation](#)  
[Configuring a Blink Animation](#)  
[Configuring an Alarm Border Animation](#)  
[Configuring a Percent Fill Horizontal Animation](#)  
[Configuring a Percent Fill Vertical Animation](#)  
[Configuring a Horizontal Location Animation](#)  
[Configuring a Vertical Location Animation](#)  
[Configuring a Width Animation](#)  
[Configuring a Height Animation](#)  
[Configuring a Point Animation](#)  
[Configuring an Orientation Animation](#)  
[Configuring a Value Display Animation](#)  
[Configuring a Tooltip Animation](#)  
[Configuring a Disable Animation](#)  
[Configuring a User Input Animation](#)  
[Configuring a Horizontal Slider Animation](#)  
[Configuring a Vertical Slider Animation](#)  
[Configuring a Pushbutton Animation](#)

- [Configuring an Action Script Animation](#)
- [Configuring a Show Symbol Animation](#)
- [Configuring a Hide Symbol Animation](#)
- [Configuring a Hyperlink Animation](#)

## Configuring a Visibility Animation

You can configure an element with a visibility animation.

---

**Note:** Truth table and bit state animations are not supported for visibility animations.

### To configure an element with a visibility animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Visibility**. The visibility animation is added to the Animation list and the **Visibility** configuration panel appears.
4. In the **Boolean** box, type a Boolean numeric value, equipment.item reference or expression.
5. Select **True, 1, On** if you want the element to show, when the expression is true, otherwise select **False, 0, Off**.

## Related Topics

- [Configuring Common Types of Animations](#)

## Configuring a Fill Style Animation

You can configure an element with a:

- Boolean fill style animation.
- Truth table fill style animation.

---

**Note:** Bit state animations are currently not supported for fill style animations.

The truth table fill style animation lets you:

- Associate expressions of any data type supported by the HMI with a fill style.
- Define as many fill styles as you require and associate each one with a condition.

You can define the conditions by specifying an comparison operator (=, >, >=, <, <=) and a breakpoint, which itself can be a value, an equipment.item reference, or an expression.

You can add conditions, delete conditions, and also change the order in which the conditions are processed.

## Related Topics

- [Configuring Common Types of Animations](#)
- [Configuring a Boolean Fill Style Animation](#)

[Configuring a Truth Table Fill Style Animation](#)

# Configuring a Boolean Fill Style Animation

You can configure an element with a discrete fill style animation.

## To configure an element with a Boolean fill style animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Fill Style**. The fill style animation is added to the Animation list and the **Fill Style** state selection panel appears.
4. Click the **Boolean** button. The **Boolean Fill Style** configuration panel appears.
5. In the **Boolean** box, type a Boolean numeric value, equipment.item reference or expression.
6. Clear **Color** in the **True, 1, On** area or **False, 0, Off** area if you do not want a different fill style for the true or false condition than the default fill style.
7. In the **True, 1, On** area, click the color box to configure the fill color when the expression is true. The **Select FillColor** dialog box appears. For more information, see [Setting Style](#).
8. In the **False, 0, Off** area, click the color box to configure the fill color when the expression is false. The **Select FillColor** dialog box appears. For more information, see [Setting Style](#).
9. Click **OK**.

## To set default fill style in a Boolean fill style animation

1. Open the **Edit Animations** dialog box, **Boolean Fill Style** panel.
2. In the **Element Fill Style** area, click the color box to select a style from the **Select FillColor** dialog box.

## To use default fill style in a Boolean fill style animation

1. Open the **Edit Animations** dialog box, **Boolean Fill Style** panel.
2. Clear **Color** to use the corresponding default fill style.

## Related Topics

[Configuring a Fill Style Animation](#)

# Configuring a Truth Table Fill Style Animation

You can configure an element with a fill style animation based on a truth table.

## To configure an element with a truth table fill style animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.

3. Click the **Add** icon and select **Fill Style**. The fill style animation is added to the Animation list and the **Fill Style** state selection panel appears.
4. Click the **Truth Table** button. The **Truth Table Fill Style** configuration panel appears.
5. In the **Expression Or Reference** area:
  - Select the data type of the expression from the list.
  - Type a value, equipment.item reference or expression in the text box.
6. If the data type of the expression is string or internationalized string, you can specify to ignore the case by selecting **Ignore Case**.
7. In the **Truth Table**, click the color box in the **Color** column. The **Select FillColor** dialog box appears. For more information, see [Setting Style](#).
8. In the **Operator** column, select the comparison operator.
9. In the **Value or Expression** column, type a value, equipment.item reference, or expression.
10. To add further conditions, see .
11. Click **OK**.

#### To set the default fill style for a truth table fill style animation

1. Open the **Edit Animations** dialog box, **Truth Table Fill Style** panel.
2. In the **Element Fill Style** area, click the color box. The **Select FillColor** dialog box appears. For more information, see [Setting Style](#).

#### To use the default fill style in a truth table fill style animation

1. Open the **Edit Animations** dialog box, **Truth Table Fill Style** panel.
2. Locate the condition for which you want to set the style to default style.
3. Clear the mark for that condition in the **Color** column of the truth table. The associated style is the same as the style for the **Element Fill Style**.

#### To add a condition to a truth table fill style animation

1. Open the **Edit Animations** dialog box, **Truth Table Fill Style** panel.
2. Click the **Add** icon. An additional condition is added to the truth table.
3. Configure color, operator and breakpoint value according to your requirements.

#### To delete a condition from an analog fill style animation

1. Open the **Edit Animations** dialog box, **Truth Table Fill Style** panel.
2. Select the condition you want to delete.
3. Click the **Remove** icon. The condition is removed.

#### To change the processing order of fill style conditions

1. Open the **Edit Animations** dialog box, **Truth Table Fill Style** panel.
2. Select the condition you want to move up or down the condition list in order for it to be processed sooner or

later.

3. Click the:

- **Arrow up** icon to move the condition up in the truth table.
- **Arrow down** icon to move the condition down in the truth table.

For example, you want to model an analog fill color animation that describes the following conditions:

- When the equipment.item TankLevel\_001.PV is 0 then the fill style is solid black.
- When the equipment.item TankLevel\_001.PV is smaller than 20, then the fill style is solid red.
- When the equipment.item TankLevel\_001.PV is greater than the equipment.item Standards.TankMax then the fill style is red with a diagonal pattern.
- In all other cases, the fill style is solid blue.

## Related Topics

[Configuring a Fill Style Animation](#)

### Configuring a Line Style Animation

You can configure an element with a:

- Boolean line style animation.
- Truth table line style animation.

---

**Note:** Bit state animations are currently not supported for line style animations.

---

The truth table line style animation lets you:

- Associate expressions of any data type supported by your HMI/SCADA software with a line style.
- Define as many line styles as you want and associate each one with a condition.

You can define the conditions by specifying an comparison operator (=, >, >=, <, <=) and a breakpoint, which itself can be a value, an equipment.item reference or an expression.

You can add conditions, delete conditions and also change the order in which the conditions are processed.

## Related Topics

[Configuring Common Types of Animations](#)

[Configuring a Boolean Line Style Animation](#)

[Configuring a Truth Table Line Style Animation](#)

# Configuring a Boolean Line Style Animation

You can configure an element with a Boolean line style animation. You can use a new style or use all or parts of the default appearance of a line for:

- Line style.
- Line thickness.
- Line pattern.

### To configure an element with a Boolean line style animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Line Style**. The line style animation is added to the Animation list and the **Line Style** state selection panel appears.
4. Click the **Boolean** button. The **Boolean Line Style** configuration panel appears.
5. In the **Boolean** box, type a Boolean numeric value, equipment.item reference or expression.
6. In the **True, 1, On** area, click the **Color** box to configure the line style when the expression is true. The **Select FillColor** dialog box appears. For more information, see [Setting Style](#).
7. In the **Weight** box, type a value for the line thickness when the expression is true.
8. From the **Pattern** list, select a line pattern for the line when the expression is true.
9. Repeat the above steps for the false condition in the **False, 0, Off** area.
10. Click **OK**.

### To set default line style, thickness and/or pattern in a Boolean line style animation

1. Open the **Edit Animations** dialog box, **Boolean Line Style** panel.
2. In the **Element Line Style** area, select a style, type a value for the width and select a pattern for the default Boolean line style.

### To use default line style, thickness and/or pattern in a Boolean line style animation

1. Open the **Edit Animations** dialog box, **Boolean Line Style** panel.
2. In the **True, 1, On** or **False, 0, Off** areas, clear **Color**, **Weight** and/or **Pattern** to use the corresponding default style, weight and/or pattern.

## Related Topics

[Configuring a Line Style Animation](#)

# Configuring a Truth Table Line Style Animation

You can configure an element with a truth table line style animation.

### To configure an element with a truth table line style animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Line Style**. The line style animation is added to the Animation list and the **Line**

**Style** state selection panel appears.

4. Click the **Truth Table** button. The **Truth Table Line Style** configuration panel appears.
5. In the **Expression or Reference** box:
  - Select the data type of the expression from the list.
  - Type a value, equipment.item reference or expression in the text box.
6. If the data type of the expression is string or internationalized string, you can specify to ignore the case by selecting **Ignore Case**.
7. In the **Truth Table**, click the color box in the **Color** column. The **Select FillColor** dialog box appears. For more information, see [Setting Style](#).
8. Select the truth options. Do one of more of the following:
  - In the **Weight** column, type a value for the line weight.
  - In the **Pattern** column, select a line pattern.
  - In the **Operator** column, select the comparison operator.
  - In the **Value or Expression** column, type a value, equipment.item reference or expression.
  - To add further conditions, see .
9. Click **OK**.

#### To set the default line style, width or pattern for a truth table line style animation

1. Open the **Edit Animations** dialog box, **Truth Table Line Style** panel.
2. In the **Element Line Style** area, select a style, type a value for the width and select a pattern for the default truth table line style.

#### To use the default line style, width or pattern in a truth table line style animation

1. Open the **Edit Animations** dialog box, **Truth Table Line Style** panel.
2. Locate the condition for which you want to change the line style, width or pattern.
3. To use the default line style for the condition, clear the mark in the **Color** column of the truth table.
4. To use the default line width for the condition, clear the mark in the **Width** column of the truth table.
5. To use the default line pattern for the condition, clear the mark in the **Pattern** column of the truth table.

#### To add a condition to a truth table line style animation

1. In the **Edit Animations** dialog box, **Truth Table Line Style** panel, click the **Add** icon. An additional condition is added to the truth table.
2. Configure color, weight, pattern, operator and breakpoint value according to your requirements.

#### To delete a condition from an analog line color animation

1. In the **Edit Animations** dialog box, **Truth Table Line Style** panel, select the condition you want to delete.
2. Click the **Remove** button. The condition is removed.

### To change the processing order of line style conditions

1. Open the **Edit Animations** dialog box, **Truth Table Line Style** panel
2. Select the condition you want to move up or down the condition list in order for it to be processed sooner or later.
3. Click the:
  - **Arrow up** icon to move the condition up in the truth table.
  - **Arrow down** icon to move the condition down in the truth table.

## Related Topics

[Configuring a Line Style Animation](#)

### Configuring a Text Style Animation

You can configure an element with a:

- Boolean text style animation.
- Truth table text style animation.

---

**Note:** Bit state animations are currently not supported for text style animations.

---

The truth table text style animation lets you:

- Associate expressions of any data type supported by your HMI/SCADA software with a text style.
- Define as many text styles as you want and associate each one with a condition.

You can define the conditions by specifying an comparison operator (=, >, >=, <, <=) and a breakpoint, which itself can be a value, an equipment.item reference or an expression.

You can add conditions, delete conditions and also change the order in which the conditions are processed.

## Related Topics

[Configuring Common Types of Animations](#)

[Configuring a Boolean Text Style Animation](#)

[Configuring a Truth Table Animation with Element Styles](#)

# Configuring a Boolean Text Style Animation

You can configure an element with a Boolean text style animation.

### To configure an element with a Boolean text style animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.

3. Click the **Add** icon and select **Text Style**. The text style animation is added to the Animation list and the **Text Style** state selection panel appears.
4. Click the **Boolean** button. The **Boolean Text Style** configuration panel appears.
5. In the **Boolean** box, type a Boolean numeric value, equipment.item reference or expression.
6. In the **True, 1, On** area, click the **Color** box to configure the text style when the expression is true. The **Select FillColor** dialog box appears. For more information, see [Setting Style](#).
7. Click the browse button for the **Font** box, to select a font, font style and size for the text when the expression is true.
8. Repeat the above steps for the false condition in the **False, 0, Off** area.
9. Click **OK**.

#### To set default text style and/or font in a Boolean text style animation

1. Open the **Edit Animations** dialog box, **Boolean Text Style** panel.
2. In the **Element Text Style** area, select a style and/or a font for the default Boolean text style.

#### To use default text style and/or font in a Boolean text style animation

1. Open the **Edit Animations** dialog box, **Boolean Text Style** panel.
2. In the **True, 1, On** or **False, 0, Off** areas, clear **Color** and/or **Font** to use the corresponding default style and/or font.

## Related Topics

[Configuring a Text Style Animation](#)

# Configuring a Truth Table Text Style Animation

You can configure an element with a truth table text style animation.

#### To configure an element with a truth table text style animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Text Style**. The text style animation is added to the Animation list. The **Text Style** information page appears.
4. Click the **Truth Table** button. The **Truth Table Text Style** configuration panel appears.
  - Select the data type of the expression from the list.
  - Type a value, equipment.item reference or expression in the text box.
5. If the data type of the expression is string or internationalized string, you can specify to ignore the case by selecting **Ignore Case**.
6. In the **Truth Table**, click the color box in the **Color** column. The **Select FillColor** dialog box appears. For more information, see [Setting Style](#).
7. Select the truth options. Do one of more of the following:

- Click on the cell in the **Font** column to select a font.
  - In the **Operator** column, select the comparison operator.
  - In the **Value or Expression** column, type a value, equipment.item reference or expression.
  - To add further conditions, see .
8. Click **OK**.

### To set the default text style or font for a truth table text style animation

1. Open the **Edit Animations** dialog box, **Truth Table Text Style** panel.
2. In the **Element Text Style** area, select a style and a font for the default truth table text style.

### To use the default text style or font in a truth table text style animation

1. Open the **Edit Animations** dialog box, **Truth Table Text Style** panel.
2. Locate the condition for which you want to change the text style or font.
3. To use the default text style for the condition, clear the mark in the **Color** column of the truth table.
4. To use the default font for the condition, clear the mark in the **Font** column of the truth table.

### To add a condition to a truth table text style animation

1. In the **Edit Animations** dialog box, **Truth Table Text Style** panel, click the **Add** icon. An additional condition is added to the truth table.
2. Configure style, font, operator and breakpoint value according to your requirements.

### To delete a condition from a truth table text style animation

1. In the **Edit Animations** dialog box, **Truth Table Text Style** panel, select the condition you want to delete.
2. Click the **Remove** button.

### To change the processing order of text style conditions

1. Open the **Edit Animations** dialog box, **Truth Table Text Style** panel
2. Select the condition you want to move up or down the condition list in order for it to be processed sooner or later.
3. Click the:
  - **Arrow up** icon to move the condition up in the truth table.
  - **Arrow down** icon to move the condition down in the truth table.

## Related Topics

[Configuring a Text Style Animation](#)

[Configuring a Blink Animation](#)

You can configure an element with a blink animation. You can specify:

- The blinking speed: slow, medium or fast.
- If the element should blink invisibly or if it should blink with specified colors.

**Note:** Truth table and bit state animations are currently not supported for blink animations.

## To configure an element with a blink animation

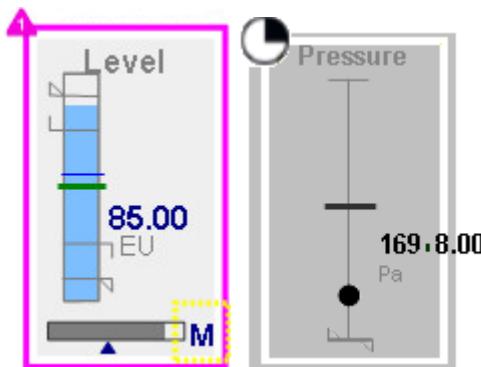
1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Blink**. The blink animation is added to the Animation list and the **Blink** configuration panel appears.
4. In the **Boolean** box, type a Boolean numeric value, equipment.item reference or expression.
5. In the **Blink When Expression Is** area, select:
  - **True, 1, On** to enable blinking when the expression is true.
  - **False, 0, Off** to enable blinking when the expression is false.
6. In the **Blink Speed** area, select **Slow,Medium** or **Fast** for the blinking speed.
7. In the **Blink Attributes** area, select **Blink Visible With These Attributes** or **Blink Invisible**.
8. If you select **Blink Visible With These Attributes**, you can configure the styles used at run time for the text, line and fill component of the element. Click on the corresponding color box, and the **Select FillColor** dialog box appears. For more information, see [Setting Style](#).
9. Click **OK**.

## Related Topics

[Configuring Common Types of Animations](#)

## Configuring an Alarm Border Animation

Alarm Border animation shows a highly visible border around a graphic or graphic element when an alarm occurs. The color and fill pattern of the border indicates the severity and current state of the alarm, if supported by your HMI/SCADA software. Plant operators can quickly recognize alarm conditions when Alarm Border animation is used.



Alarm Border animation also shows an indicator icon at the top left corner of the border around a closed graphic element. For open pie or arc graphic elements, the indicator icon is placed at the top-left most location of the start and end points.

Alarm severity (1-4) or the current alarm mode (Shelved, Silenced, Disabled) appear as part of the indicator icon. The indicator icon can be shown or hidden as a configurable option of Alarm Border animation.

Alarm Border animation adheres to the following precedence rules with other functions that can change the appearance of a graphic:

1. Quality status
2. Alarm Border animation
3. Element Style animation
4. Style animations
5. Element Style on canvas

## Related Topics

[Configuring Common Types of Animations](#)

[Understanding Requirements of Alarm Border Animations](#)

[Understanding the Behavior of Alarm Border Animations](#)

[Configuring Alarm Border Animation](#)

[Configuring Optional Alarm Border Animation Characteristics](#)

# Understanding Requirements of Alarm Border Animations

Alarm border animation can be applied to all types of graphics except embedded graphics and nested groups. Alarm border animation can also be applied to graphic elements and group elements.

## Related Topics

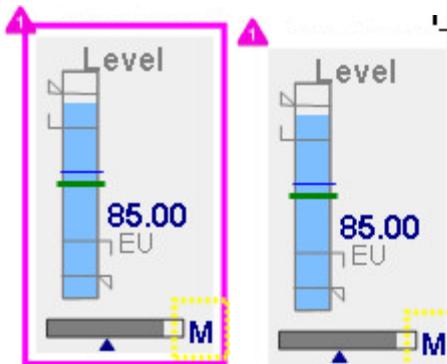
[Configuring an Alarm Border Animation](#)

# Understanding the Behavior of Alarm Border Animations

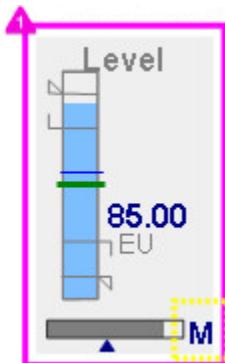
Alarm Border animation appears around a graphic element based on the current state of the object's aggregated alarm equipment.items, or other configured alarm inputs supported by your HMI/SCADA software. The appearance of the alarm border itself reflects the current alarm state and the user's interaction with the alarm.

- A graphic's process value transition into an alarm state.

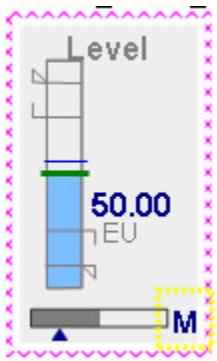
Alarm Border animation appears around the graphic based on alarm severity with blinking.



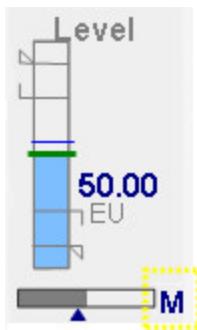
- The user acknowledges an alarm with the process value still in an alarm state.  
Alarm Border animation appears around the graphic without blinking.



- The alarm value returns to normal without the user acknowledging the alarm.  
Alarm Border animation remains around the graphic in a defined Return to Normal visual style without blinking.

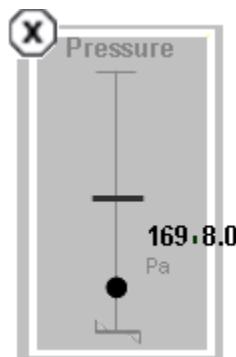


- The alarm value returns to normal and the user acknowledges the alarm.  
Alarm Border animation no longer appears around a graphic.



- The user suppresses an alarm.

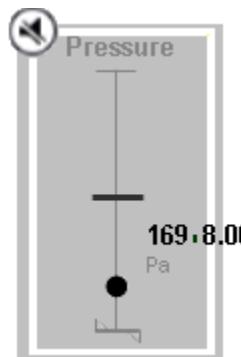
Alarm Border animation remains around the graphic in a defined suppressed/disabled visual style without blinking. The indicator shows the suppressed/disabled alarm mode icon.



- The user silences an alarm

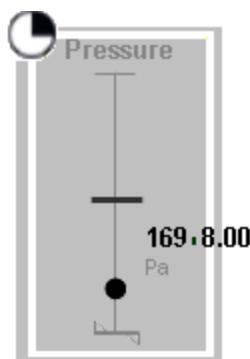
Alarm Border animation remains around the graphic in a defined silenced visual style without blinking. The indicator shows the silenced alarm mode icon.

A silenced alarm mode takes precedence over an alarm in shelved mode.



- The user shelves an alarm

Alarm Border animation remains around the graphic in a defined shelved visual style without blinking. The indicator shows the shelved alarm mode icon.



If a new alarm condition occurs when Alarm Border animation appears around a graphic, the animation updates to show the new alarm state. In the case of aggregation alarms, Alarm Border animation shows the highest current alarm state.

## Related Topics

[Configuring an Alarm Border Animation](#)

# Configuring Alarm Border Animation

Alarm Border animation can be configured by selecting **Alarm Border** from the list of **Visualization** animations. For Situational Awareness Library symbols, Alarm Border animation can be selected as a Wizard Option of the graphic.

The **Alarm Border** animation dialog box contains mutually exclusive fields to set the referenced equipment.items for aggregate or individual alarms.

If your HMI/SCADA software supports aggregating alarms from different plant or functional areas, you can specify Alarm Border animation by entering an equipment.item, object, or tag name in the **Use Standard Alarm-Urgency References** field of the **Alarm Border** dialog box.

The selected object equipment.items or tags typically map to the following aggregation alarm equipment.items:

- AlarmMostUrgentAcked
- AlarmMostUrgentInAlarm
- AlarmMostUrgentMode
- AlarmMostUrgentSeverity
- AlarmMostUrgentShelved

For individual alarms, you can specify Alarm Border animation by entering equipment.item or object names in the **Use Custom Alarm-Urgency References** fields of the **Alarm Border** dialog box.

- **InAlarm Source**

Indicates the InAlarm status (True/False) of the most urgent alarm that is in the InAlarm state or waiting to be Acked. If no alarms are in the InAlarm state or waiting to be Acked, the value is False.

- **Acked Source**

Indicates the acknowledgement status (True/False) of the most urgent alarm that is in the InAlarm state or

waiting to be Acked. If no alarms are in an InAlarm state or waiting to be Acked, the value is True, which means no acknowledgement is needed.

- **Mode Source**

Indicates the alarm mode (Enable/Silence/Disable/Shelved) of the most urgent alarm that is in the InAlarm state or waiting to be Acked. If alarms are configured for an equipment.item, but no alarms are in the InAlarm state or waiting to be Acked, the value is the same as the AlarmMode of the object.

- **Severity Source**

Indicates the severity as an integer (1-4) of the most urgent alarm current in an InAlarm state. If no alarms are in an InAlarm state or waiting to be acknowledged, the value is 0.

- **Shelved Source**

Indicates the current Shelved status (True/False) of the most urgent alarm that is in the InAlarm state or waiting to be Acked. If no alarms are in the InAlarm state or waiting to be Acked, the value is False.

To set Alarm Border animation for individual alarms, specify references to the following alarm equipment.items or tags:

- InAlarm equipment.item
- Acked equipment.item
- Mode equipment.item
- Severity equipment.item
- Shelved equipment.item

Alarm Border animation subscribes to these equipment.items or tags. Based on the alarm state of these equipment.items or tags, Alarm Border animation is applied to the graphic element in run time.

### To configure Alarm Border animation

1. Open a graphic in the Industrial Graphic Editor.
2. Select the graphic to show the graphic elements listed in the **Elements** pane of the Industrial Graphic Editor.
3. Select a graphic element from the **Elements** list to apply Alarm Border animation.
4. Click **Add Animation** to show the list of animation types.
5. Select **Alarm Border** from the list of **Visualization** animations.

The **Alarm Border** dialog box appears with a set of configuration options.

6. Select either **Use Standard Alarm-Urgency References** or **Use Customized Alarm-Urgency References**.

- **To use Standard Alarm-Urgency References**

Click **Browse** and select an equipment.item, tag, or object name, then Click **OK**.

Both direct and relative references to an object or tag are supported. An expression cannot be used to reference the object.

- **To use Customized Alarm-Urgency References**

Click **Browse** and select an equipment.item, a graphic element, or a tag name for all Source fields shown beneath **Use Customized Alarm Urgency References**, then click **OK**.

All fields must contain values and cannot be left blank. Expressions, external references, and custom properties can be entered in all fields.

7. Enter a custom property, a constant (True/False), an external reference, or an expression in the **Show Alarm Indicator** field to set the condition when an alarm indicator icon is shown or hidden.

## Related Topics

[Configuring an Alarm Border Animation](#)

# Configuring Optional Alarm Border Animation Characteristics

You can complete a set of optional tasks to customize the appearance of Alarm Border animation.

## Related Topics

[Configuring an Alarm Border Animation](#)

[Changing Alarm Border Indicator Icons](#)

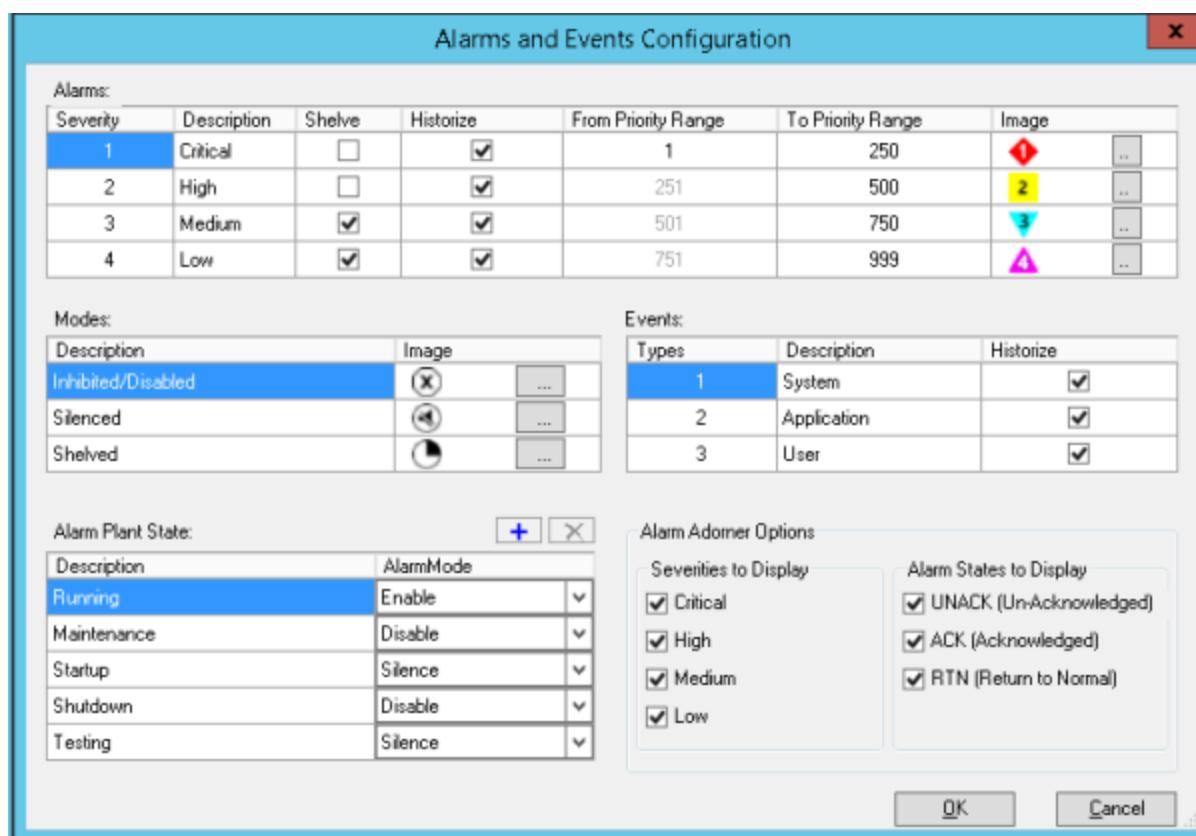
[Modify Alarm Border Animation Element Styles](#)

# Changing Alarm Border Indicator Icons

## Changing Alarm Border Indicator Icons

Alarm border animation shows an indicator icon at the top left corner of the border with alarm severity as a number from 1 to 4 if your HMI/SCADA software supports it. Other indicator images represent alarm suppressed, silenced and shelved modes.

A default Alarm Border indicator image is assigned to each alarm mode and severity level. The default images appear in the **Image** fields of the **Alarm and Event Priority Mapping and Historization** dialog box. The images are saved in an XML file. For details about the location of the file, refer to your HMI/SCADA software documentation.



The default Alarm Border indicator images can be replaced by custom images. Supported image file types include .bmp, .gif, .jpg, .jpeg, .tif, .tiff, .png, .ico and .emf.

### To replace Alarm Border indicator images

1. Create Alarm Border indicator images that will replace the default indicator images.
2. Navigate to the Alarm Priority Mapping configuration dialog. The **Alarm and Event Priority Mapping and Historization** dialog box appears. The navigation path will differ according to your HMI/SCADA software. Typically, the path will be through menu items to configure your project or application components.
3. Click the **Search** button next to the Alarm Border indicator or mode image to be replaced. The **Open** dialog box appears to locate the replacement Alarm Border indicator images.
4. Go to the folder containing the replacement images.
5. Select an image file and click **Open**.
6. Verify the new Alarm Border indicator image replaced the original image in the **Alarm and Event Priority Mapping and Historization** dialog box.

At run time, Alarm Border animation reads the XML file from the location where it is stored. If images are not available from the XML file, an Alarm Indicator does not appear during Alarm Border animation.

## Related Topics

[Configuring Optional Alarm Border Animation Characteristics](#)

# Modify Alarm Border Animation Element Styles

## Modify Alarm Border Animation Element Styles

The color and fill pattern of alarm borders are set by the Outline properties of a set of AlarmBorder Element Styles. The following table shows the Element Styles applied to Alarm Border animations by alarm severity and alarm state.

The assignment of these Element Styles to alarm conditions cannot be changed. Only the assigned Element Style's Outline properties can be changed to modify the line pattern, line weight, and line color of alarm borders.

Alarm Severity	Alarm State	Element Style
1	UnAcknowledged	AlarmBorder_Critical_UNACK
1	Acknowledged	AlarmBorder_Critical_ACK
1	Return To Normal	AlarmBorder_Critical_RTN
2	UnAcknowledged	AlarmBorder_High_UNACK
2	Acknowledged	AlarmBorder_High_ACK
2	Return To Normal	AlarmBorder_High_RTN
3	UnAcknowledged	AlarmBorder_Medium_UNACK
3	Acknowledged	AlarmBorder_Medium_ACK
3	Return To Normal	AlarmBorder_Medium_RTN
4	UnAcknowledged	AlarmBorder_Low_UNACK
4	Acknowledged	AlarmBorder_Low_ACK
4	Return To Normal	AlarmBorder_Low_RTN
All	Suppressed	AlarmBorder_Suppressed
All	Shelved	AlarmBorder_Shelved
All	Silenced	AlarmBorder_Silenced

### Related Topics

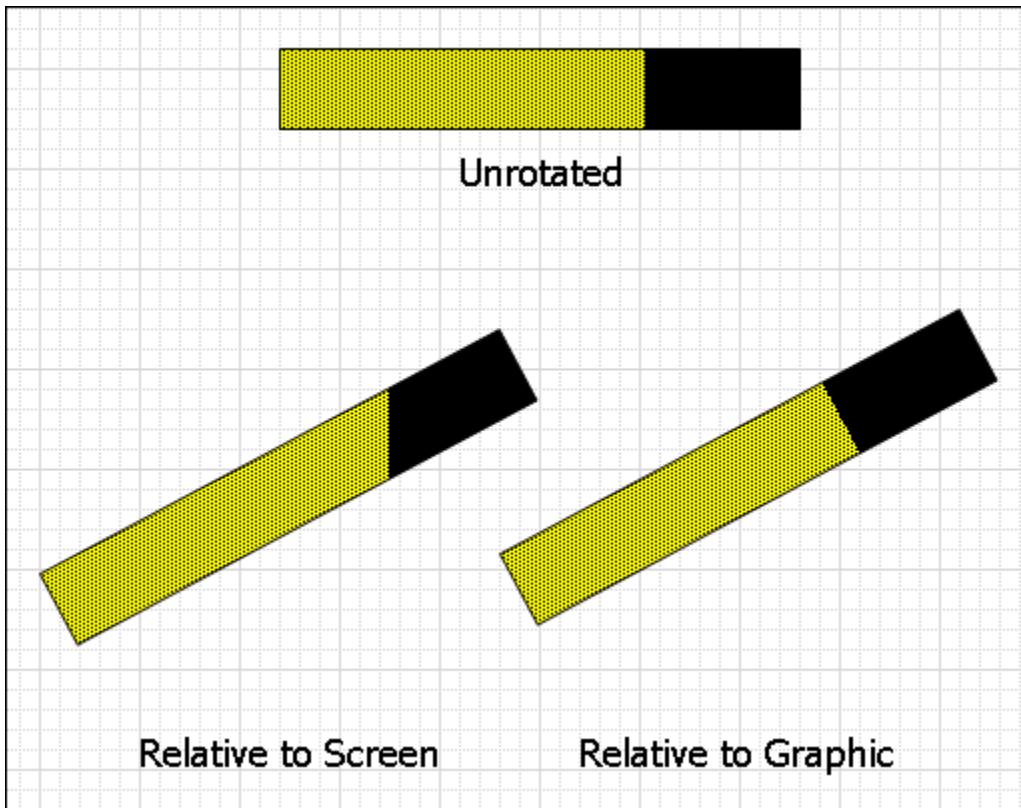
[Configuring Optional Alarm Border Animation Characteristics](#)

## Configuring a Percent Fill Horizontal Animation

You can configure an element with a percent fill horizontal animation.

Besides specifying the expressions that determine how much of the element is filled at run time, you can also specify:

- **Fill direction:** from left to right or right to left.
- **Unfill color:** the style of the background when the element has 0 percent filling.
- **Fill orientation:** if the filling is in relation to the element or to the screen. This affects how the fill appears if the orientation of the element changes. If the fill is in relation to the screen and the element or graphic are rotated, the fill remains in relation to the screen.



---

**Note:** The fill orientation is a common setting to the percent fill horizontal and percent fill vertical animations.

---

You can also preview how the percent fill horizontal animation appears at run time.

### To configure an element with a percent fill horizontal animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **% Fill Horizontal**. The percent fill horizontal animation is added to the Animation list and the **% Fill Horizontal** configuration panel appears.
4. Specify the settings. Do one or more of the following:
  - In the **Analog** box, type an analog value, equipment.item reference or expression.

- In the **Value - At Min Fill** box, type an analog value, equipment.item reference or expression that causes the minimum percent of filling at run time.
  - In the **Value - At Max Fill** box, type an analog value, equipment.item reference or expression that causes the maximum percent of filling at run time.
  - In the **Fill - Min%** box, type an analog value, equipment.item reference or expression to specify the minimum percent of filling.
  - In the **Fill - Max%** box, type an analog value, equipment.item reference or expression to specify the maximum percent of filling.
  - In the **Colors** area, click the:
    - Fill Color** box to select a style from the **Select FillColor** dialog box. This is the fill style of the element.
    - Unfilled Color** box to select a style from the **Select FillColor** dialog box. This is the unfilled fill style of the element.
- For more information, see [Setting Style](#).

5. In the **Direction** area, select:
  - **Right** - to fill from left to right.
  - **Left** - to fill from right to left.
6. In the **Orientation** area, select:
  - **Relative to Graphic** - so that the filling is in relation to the element and the filling rotates with the element.
  - **Relative to Screen** - so that the filling is in relation to the screen and the filling does not rotate with the element.
7. You can preview your configuration by using the slider in the **Preview** area. Drag the slider to see how different values affect the appearance at run time.
8. Click **OK**.

## Related Topics

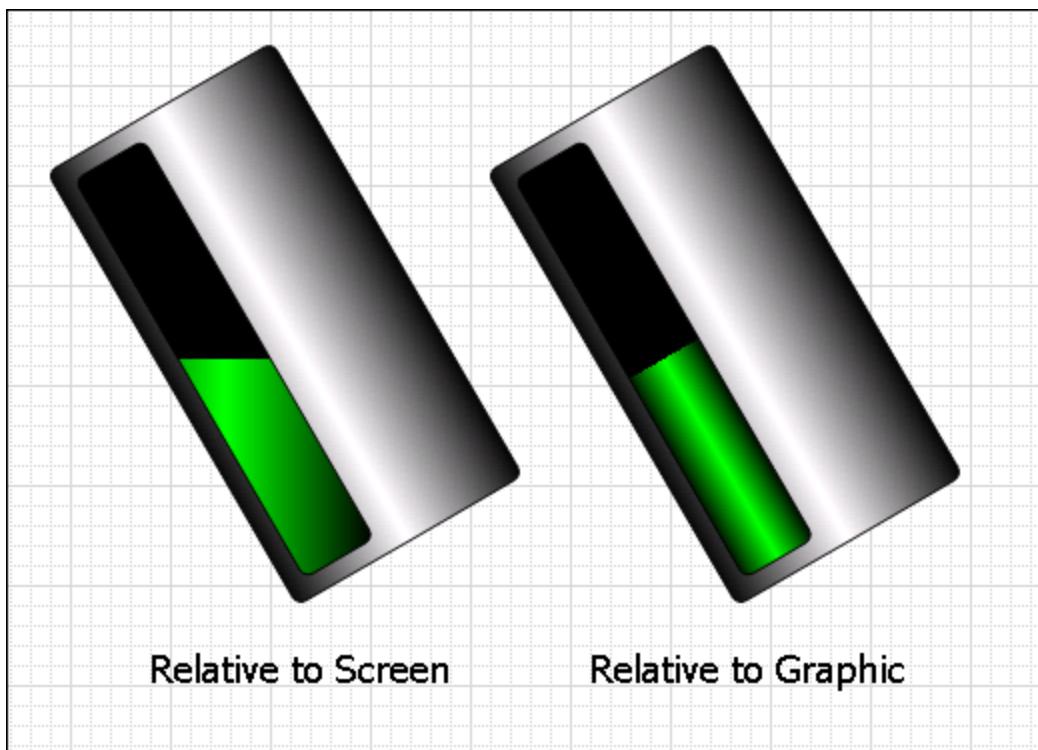
[Configuring Common Types of Animations](#)

### Configuring a Percent Fill Vertical Animation

You can configure an element with a percent fill vertical animation.

Besides specifying the expressions that determine how much of the element is filled at run time, you can also specify:

- **Fill direction:** from lower to top or top to lower.
- **Unfill color:** the style of the background when the element has 0 percent filling.
- **Fill orientation:** if the filling is in relation to the element or to the screen. This affects how the fill appears if the orientation of the element changes. If the fill is in relation to the screen and the element or graphic are rotated, the fill remains in relation to the screen.



**Note:** The fill orientation is a common setting to the percent fill horizontal and percent fill vertical animations.

#### To configure an element with a percent fill vertical animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **% Fill Vertical**. The percent fill vertical animation is added to the Animation list and the **% Fill Vertical** configuration panel appears.
4. In the **Analog** box, type an analog value, equipment.item reference or expression.
5. In the **Value-At Min Fill** box, type an analog value, equipment.item reference or expression that causes the minimum percent of filling at run time.
6. In the **Value-At Max Fill** box, type an analog value, equipment.item reference or expression that causes the maximum percent of filling at run time.
7. In the **Fill-Min%** box, type an analog value, equipment.item reference or expression to specify the minimum percent of filling.
8. In the **Fill-Max%** box, type an analog value, equipment.item reference or expression to specify the maximum percent of filling.
9. In the **Colors** area, click the:
  - **Fill Color** box to select a style from the **Select FillColor** dialog box. This is the fill style of the element.
  - **Unfilled Color** box to select a style from the **Select FillColor** dialog box. This is the unfilled fill style of the element.For more information, see [Setting Style](#).
10. In the **Direction** area, select:
  - **Up** - to fill from lower to top.

- **Down** - to fill from top to lower.
11. In the **Orientation** area, select:
    - **Relative to Graphic** - so that the filling is in relation to the element and the filling rotates with the element.
    - **Relative to Screen** - so that the filling is in relation to the screen and the filling does not rotate with the element.
  12. You can preview your configuration by using the slider in the **Preview** area. Drag the slider to see how different values affect the appearance at run time.
  13. Click **OK**.

## Related Topics

[Configuring Common Types of Animations](#)

### Configuring a Horizontal Location Animation

You can configure an element with a horizontal location animation.

#### To configure an element with a horizontal location animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The Edit Animations dialog box appears.
3. Click the **Add** icon and select **Location Horizontal**. The horizontal location animation is added to the Animation list and the Location Horizontal configuration panel appears.
4. In the **Analog** box, type an analog value, equipment.item reference or expression.
5. In the **Value-At Left End** box, type an analog value, equipment.item reference or expression that corresponds to the offset specified by the **Movement-To Left** value.
6. In the **Value-At Right End** box, type an analog value, equipment.item reference or expression that corresponds to the offset specified by the **Movement-To Right** value.
7. In the **Movement-To Left** box, type an analog value, equipment.item reference or expression for the maximum offset to the left.
8. In the **Movement-To Right** box, type an analog value, equipment.item reference or expression for the maximum offset to the right.
9. Click **OK**.

## Related Topics

[Configuring Common Types of Animations](#)

### Configuring a Vertical Location Animation

You can configure an element with a vertical location animation.

## To configure an element with a vertical location animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Location Vertical**. The vertical location animation is added to the Animation list and the **Location Vertical** configuration panel appears.
4. In the **Analog** box, type an analog value, equipment.item reference or expression.
5. In the **Value - At Top** box, type an analog value, equipment.item reference or expression that corresponds to the offset specified by the **Movement - Up** value.
6. In the **Value - At lower** box, type an analog value, equipment.item reference or expression that corresponds to the offset specified by the **Movement - Down** value.
7. In the **Movement - Up** box, type an analog value, equipment.item reference or expression for the maximum offset upwards.
8. In the **Movement - Down** box, type an analog value, equipment.item reference or expression for the maximum offset downwards.
9. Click **OK**.

## Related Topics

[Configuring Common Types of Animations](#)

## Configuring a Width Animation

You can configure an element with a width animation. You can also specify if the element is to be anchored to its left, center, right side or origin.

### To configure an element with a width animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Width**. The width animation is added to the Animation list and the **Width** configuration panel appears.
4. In the **Analog** box, type an analog value, equipment.item reference or expression.
5. In the **Value-At Min Size** box, type an analog value, equipment.item reference or expression that corresponds to the minimum width specified by the **Width-Min%** value.
6. In the **Value-At Max Size** box, type an analog value, equipment.item reference or expression that corresponds to the maximum width specified by the **Width-Max%** value.
7. In the **Width-Min%** box, type an analog value, equipment.item reference or expression for the minimum width in percent of the original element.
8. In the **Width-Max%** box, type an analog value, equipment.item reference or expression for the maximum width in percent of the original element.
9. In the **Anchor** area, select:
  - **Left** - to specify that the left of the element is anchored.
  - **Center** - to specify that the horizontal center of the element is anchored.

- **Right** - to specify that the right side of the element is anchored.
  - **Origin** - to specify that the origin of the element is anchored.
10. Click **OK**.

## Related Topics

[Configuring Common Types of Animations](#)

### Configuring a Height Animation

You can configure an element with a height animation. You can also specify if the element is to be anchored to its top side, middle, lower side or origin.

#### To configure an element with a height animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Height**. The height animation is added to the Animation list and the **Height** configuration panel appears.
4. In the **Analog** box, type an analog value, equipment.item reference or expression.
5. In the **Value-At Min Size** box, type an analog value, equipment.item reference or expression that corresponds to the minimum height specified by the **Height-Min%** value.
6. In the **Value-At Max Size** box, type an analog value, equipment.item reference or expression that corresponds to the maximum height specified by the **Height-Max%** value.
7. In the **Height-Min%** box, type an analog value, equipment.item reference or expression for the minimum height in percent of the original element.
8. In the **Height-Max%** box, type an analog value, equipment.item reference or expression for the maximum height in percent of the original element.
9. In the **Anchor** area, select:
  - **Top** - to specify that the top side of the element is anchored.
  - **Middle** - to specify that the vertical center of the element is anchored.
  - **Lower** - to specify that the lower side of the element is anchored.
  - **Origin** - to specify that the origin of the element is anchored.
10. Click **OK**.

## Related Topics

[Configuring Common Types of Animations](#)

### Configuring a Point Animation

Point animation changes the X and Y coordinate values of one or more selected points on a graphic or graphic element. During run time, the X and the Y coordinates of a selected point are set to an expression or reference that evaluates to a calculated real floating point value.

Point animation supports negative floating point values, which may cause the animation to go out of scope of the visualization window. In the case when a point's expression evaluates to a null value or causes an exception, animation stops and the point retains its original value.

The X and Y coordinates of a point can be configured as a pair or individually. If only the X coordinate of a point is configured, then the Y coordinate value is kept constant. The animation shows the point of the graphic element traversing the X axis. Likewise, if only the Y coordinate of a point is configured, then the animation shows the point traversing the Y axis with the point's X axis value held constant.

After selecting point animation, a list of configurable points is retrieved from the graphic element based on the following conditions.

- If the graphic element is a multi-point graphic type (Line, HV/Line, Polyline, Curve, Polygon, Closed curve), animation control points appear on the graphic element in preview mode.
- If the graphic element is not a supported multi-point graphic, then the top left X and Y coordinate of the graphic element is selected as the animation point.

In the case of an element group consisting of several graphic elements, the animation point is the top left corner of the rectangle around all grouped elements.

## To configure point animation

1. Open the graphic in Industrial Graphic Editor.
2. Select a graphic element.
3. On the **Special** menu, click **Edit Animations**.  
The **Edit Animations** dialog box appears.
4. Click the **Add Animation** button to show a list of Visualization and Interaction animations.
5. Select **Point** from the Visualization animation list.  
The **Point** dialog box appears with a list of points and a preview of the points on the graphic. The list shows each point as a pair of X and Y fields to enter an expression or a reference that evaluates to a floating point value.
6. Select a point from the list of points.  
The selected point changes to orange in the preview of the graphic.
7. Enter an expression, constant, or reference in the Point field.
8. Repeat steps 6-7 to animate other points in the graphic.
9. Save your changes.

## Related Topics

[Configuring Common Types of Animations](#)

## Configuring an Orientation Animation

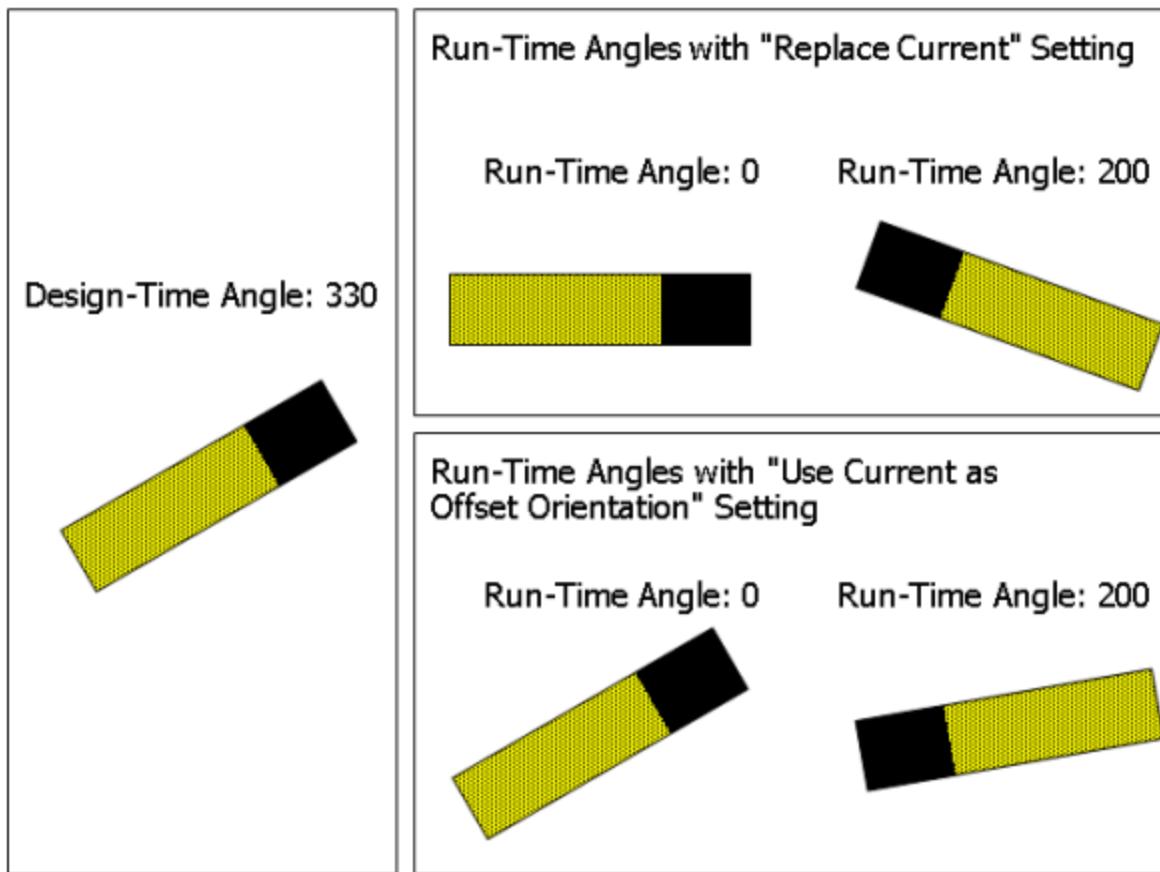
You can configure an element with an orientation animation. You can also:

- Specify a different orientation origin.
- Ignore or accept the design-time orientation of the element on the canvas.

- Preview the orientation at run time with a slider.

### To configure an element with an orientation animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Orientation**. The orientation animation is added to the Animation list and the **Orientation** configuration panel appears.
4. In the **Analog** box, type an analog value, equipment.item reference or expression.
5. In the **Value-At CCW End** box, type an analog value, equipment.item reference or expression that corresponds to the maximum angle in degrees for the counter-clockwise orientation as specified by the **Orientation-CCW** value.
6. In the **Value-At CW End** box, type an analog value, equipment.item reference or expression that corresponds to the maximum angle in degrees for the counter-clockwise orientation as specified by the **Orientation-CW** value.
7. In the **Orientation-CCW** box, type an analog value, equipment.item reference or expression for the maximum orientation in counter-clockwise direction in degrees.
8. In the **Orientation-CW** box, type an analog value, equipment.item reference or expression for the maximum orientation in clockwise direction in degrees.
9. In the **Orientation Offset** area, select:
  - **Replace Current** to ignore the design-time orientation of the element as it appears on the canvas and to use absolute orientation.
  - **Use Current as Offset Orientation** to orientate the element at run time in relation to its design-time orientation on the canvas.



10. If you use current as offset orientation, you can type an offset value in the text box next to **Use Current as Offset Orientation**. This affects the orientation of the element on the canvas.
11. In the **Current Relative Origin** area, type values in the **dX** and **dY** boxes to specify the rotation origin as offset from the element center point. This affects the point of origin of the element on the canvas.
12. You can preview the orientation and how run-time values affect the appearance of the element, by dragging the slider in the **Preview** area.
13. Click **OK**.

## Related Topics

[Configuring Common Types of Animations](#)

### Configuring a Value Display Animation

You can configure an element with a value display animation. You can show:

- A Boolean value as a Message.
- An Analog value.
- A string value.
- A time or date value.

**Note:** Truth table and bit state animations are currently not supported for value display animations.

## Related Topics

- [Configuring Common Types of Animations](#)
- [Configuring a Boolean Value Display Animation](#)
- [Configuring an Analog Value Display Animation](#)
- [Configuring a String Value Display Animation](#)
- [Configuring a Time Value Display Animation](#)

# Configuring a Boolean Value Display Animation

You can configure an element to show a Boolean value as a message.

### To configure an element with a Boolean value display animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Value Display**. The value display animation is added to the Animation list and the **Value Display** state selection panel appears.
4. Click the **Boolean** button. The **Boolean Value Display** configuration panel appears.
5. In the **Boolean** box, type a Boolean value, equipment.item reference or expression.
6. In the **True Message** box, type a value, equipment.item reference or expression for the text display when the expression is true.
7. In the **False Message** box, type a value, equipment.item reference or expression for the text display when the expression is false.
8. Click **OK**.

## Related Topics

- [Configuring a Value Display Animation](#)

# Configuring an Analog Value Display Animation

You can configure a text element, TextBox, or button to show an analog value. You can also configure an analog value display animation for a grouped element when the TreatAsIcon property is True.

### To configure an element with an analog value display animation

1. Select the text element, TextBox, Button, or grouped element that you want to configure Analog Value Display animation.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Value Display**. The value display animation is added to the Animation list and the **Value Display** state selection panel appears.
4. Click the **Analog** button. The **Analog Value Display** configuration panel appears.

5. In the **Analog** box, type an analog value, equipment.item reference or expression.
6. Click the **Text Format** drop-down list and select a global number style.
  - **Format String** is the default numeric format, which includes a text field to enter characters that specify a number format.
  - Selecting custom from the **Text Format** list shows another drop-down list to select a number format.
    - **Fixed Width**: Appears on the **Analog Value** Display dialog box for every Custom number format. When selected, the run-time number value will not exceed the text length specified for Value Display animation.
    - **Precision**: Appears on the **Analog Value Display** dialog box for the **Fixed Decimal** and **Exponential Custom** number formats. **Precision** sets the precision of the fractional part of a number to the right of the decimal point.
    - **Bits From** and **To**: Appear on the **Analog Value Display** dialog box for the **Hex** and **Binary** number formats. **Bits From** sets the starting bit position of a hex or binary number shown during run time. **To** sets the ending bit position of a hex or binary number shown during run time.
7. Click **OK**.

Except for the **Format String** and **Custom** text styles, all other text styles are global number styles that do not need further configuration.

## Related Topics

[Configuring a Value Display Animation](#)

[Configuring Value Display Animation with the FormatStyle Property](#)

# Configuring Value Display Animation with the FormatStyle Property

## Configuring Value Display Animation with the FormatStyle Property

When Value Display or User Input animation have a text format configured with a number style from your HMI/SCADA software, numeric data shown during run time is formatted in accordance to the number style selected for the animation.

Changes to a number style are not propagated during run time. Any changes to global number styles become effective only after the HMI/SCADA software is restarted.

A number style can be changed during run time using the FormatStyle property. FormatStyle is a text element run-time property that displays the name of the current applied global number style. The value of the FormatStyle property can be set as the active number style of Value Display and User Input animation during run time .

- If the name applied to the FormatStyle property during run time is the name of a global number style, the

text element is reformatted using the new applied number style for Value Display or User Input animation.

- If the FormatStyle property is assigned a value that does not match any global number style, the value of FormatStyle remains unchanged and a warning message is logged to SMC logger.
- If FormatStyle is set to an empty string, the text format of User Input and Value Display animation reverts to the value specified for **Text Format** during design time.
- If a text element is inside a group in which the **TreatAsIcon** property is set to True, then the group's text format overrides the first text child element for Value Display or User Input animation.
- If a text element's FormatStyle property changes in run time, the new number style is used to format the text element. Because a graphic group does not have TextFormat and FormatStyle properties, using the FormatStyle property is the only way to change the format of text in run time.

## Related Topics

[Configuring an Analog Value Display Animation](#)

# Configuring a String Value Display Animation

You can configure an element to show a string value.

### To configure an element with a string value display animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Value Display**. The value display animation is added to the Animation list and the **Value Display** state selection panel appears.
4. Click the **String** button. The **String Value Display** configuration panel appears.
5. In the **String** box, type a string value, equipment.item reference or expression.
6. Click **OK**.

## Related Topics

[Configuring a Value Display Animation](#)

# Configuring a Time Value Display Animation

You can configure an element to show a time value.

Use the following letters to set the time format:

h	The one or two-digit hour in 12-hour format.
hh	The two-digit hour in 12-hour format. Single digit values are preceded by a zero.

H	The one or two-digit hour in 24-hour format.
HH	The two-digit hour in 24-hour format. Single digit values are preceded by a zero.
t	The one-letter AM/PM abbreviation ("AM" appears as "A").
tt	The two-letter AM/PM abbreviation ("AM" appears as "AM").
m	The one or two-digit minute.
mm	The two-digit minute. Single digit values are preceded by a zero.
s	The one or two-digit seconds.
ss	The two-digit seconds. Single digit values are preceded by a zero.
d	The one or two-digit day.
dd	The two-digit day. Single digit day values are preceded by a zero.
ddd	The three-character day-of-week abbreviation.
dddd	The full day-of-week name.
M	The one or two-digit month number.
MM	The two-digit month number. Single digit values are preceded by a zero.
MMM	The three-character month abbreviation.
MMMM	The full month name.
y	The one-digit year (2001 appears as "1").
yy	The last two digits of the year (2001 appears as "01").
yyy	The full year (2001 appears as "2001").

The format for elapsed time is:

[–][DDDDD] [HH:MM:]SS[.fffffff]

Use the following letters to set the elapsed time format:

DDDDD	The number of days. Valid values are 0 to 999999.
HH	The two-digit hour in 24-hour format. Single digit

	values are preceded by a zero. Valid values are 00 to 23.
MIM	The two-digit month number. Single digit values are preceded by a zero. Valid values are 00 to 59.
SS	The two-digit seconds. Single digit values are preceded by a zero. Valid values are 00 to 59.
fffffff	Optional fractional seconds to right of decimal, and can be one through seven digits.

**Note:** You can use any other characters, except "g" in the property. These characters then appear at design time and run time in the control.

### To configure an element with a time value display animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Value Display**. The value display animation is added to the Animation list and the **Value Display** state selection panel appears.
4. Click the **Time** button. The **Time Value Display** configuration panel appears.
5. In the **Time or Elapsed Time** box, type a time or elapsed time value, equipment.item reference or expression.
6. In the **Text Format** box, type a format for the value output. If you change this value, the **TextFormat** property of the element also changes.
7. Click **OK**.

## Related Topics

[Configuring a Value Display Animation](#)

## Configuring a Tooltip Animation

You can configure an element with a tooltip animation.

**Note:** Bit state animations are currently not supported for tooltip animations.

### To configure an element with a tooltip animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Tooltip**. The tooltip animation is added to the Animation list and the **Tooltip** configuration panel appears.
4. Click the **String** button. The String Tooltip configuration panel appears.
5. In the **Expression** box, type:
  - A static value and make sure the **Input Mode** icon is set to static.

- An equipment.item reference or expression and make sure the **Input Mode** icon is set to equipment.item or reference.
6. Click **OK**.

## Related Topics

[Configuring Common Types of Animations](#)

### Configuring a Disable Animation

You can configure an element with a disable animation. This lets you disable user interaction with an element depending on a run-time value or expression.

---

**Note:** Truth table and bit state animations are currently not supported for disable animations.

---

#### To configure an element with a disable animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The Edit Animations dialog box appears.
3. Click the **Add** icon and select **Disable**. The disable animation is added to the Animation list and the Disable configuration panel appears.
4. Click the **Boolean** button.
5. In the Boolean box, type a Boolean numeric value, equipment.item reference or expression.
6. In the **Disabled When Expression is** area, select:
  - **True, 1, On** in which case the element is disabled at run time whenever the expression is true, and enabled whenever the expression is false.
  - **False, 0, Off** in which case the element is disabled at run time whenever the expression is false, and enabled whenever the expression is true.
7. Click **OK**.

## Related Topics

[Configuring Common Types of Animations](#)

### Configuring a User Input Animation

You can configure an element with a user input animation for the following data types:

- Boolean
- Analog (integer, float, double)
- String
- Time
- Elapsed time

## Related Topics

- [Configuring Common Types of Animations](#)
- [Configuring a User Input Animation for a Discrete Value](#)
- [Configuring a User Input Animation for an Analog Value](#)
- [Configuring a User Input Animation for a String Value](#)
- [Configuring a User Input Animation for a Time Value](#)
- [Configuring a User Input Animation for an Elapsed Time Value](#)

# Configuring a User Input Animation for a Discrete Value

You can configure an element with a user input animation for a Boolean value.

### To configure an element with a user input animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **User Input**. The user input animation is added to the Animation list and the **User Input** state selection panel appears.
4. Click the **Boolean** button. The **Boolean Value User Input** configuration panel appears.
5. Specify the options. Do one or more of the following:
  - In the **Boolean** box, type an equipment.item reference or browse for one by using the browse button.
  - In the **Message to User** box, type a value, equipment.item reference or expression. This is the text that appears as prompt on the Boolean value input dialog box at run time.
  - In the **Prompt - True Message** box, type a value, equipment.item reference or expression. This is the text that appears on the button that causes the equipment.item to be set to true.
  - In the **Prompt - False Message** box, type a value, equipment.item reference or expression. This is the text that appears on the button that causes the equipment.item to be set to false.
  - Specify that the input dialog box appears by pressing a key or key combination. In the **Shortcut** area. Select a shortcut key in the **Key** list. Select **CTRL** and/or **SHIFT** to combine the shortcut key with the **CTRL** key and/or **SHIFT** key.
  - If you don't want the discrete value display element to show the True Message and False Message, select **Input Only**.
  - In the **Display Value - True Message** box, type a value, equipment.item reference or expression. This is the text that appears on the canvas when the associated equipment.item is true.
  - In the **Display Value - False Message** box, type a value, equipment.item reference or expression. This is the text that appears on the canvas when the associated equipment.item is false.
  - Make sure that the input modes of all boxes are set correctly. Click the **Input Mode** icon to set a static value or an equipment.item reference or expression.
6. Click **OK**.

## Related Topics

[Configuring a User Input Animation](#)

# Configuring a User Input Animation for an Analog Value

You can configure an element with a user input animation for an analog value.

### To configure an element with a user input animation for an analog value

1. Select the text element, TextBox, Button, or grouped element that you want to configure for User Input animation.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon from the **Animations** pane and select **User Input**.  
User Input is added to the **Interaction** list and **User Input** state selection panel appears.
4. Click the **Analog** button. The **Analog Value User Input** configuration panel appears.
5. In the **Analog** box, type an equipment.item reference or browse for one by using the browse button.
6. In the **Message to User** box, type a value, equipment.item reference, or expression. This text appears to prompt for the analog value input dialog box at run time.
7. Make sure that the input mode of the **Message to User** box is set correctly. Click the **Input Mode** icon to set a static value or an equipment.item reference or expression.
8. If you want to restrict the range of input values, you can do so in the **Value Limits** area by:
  - First selecting **Restrict Values**.
  - Enter values, equipment.item references, or expressions in the **Minimum** and **Maximum** boxes.
9. In the **Shortcut** area, specify that an **Input** dialog box appears by pressing a key or key combination. Select a shortcut key from the **Key** list. Select **CTRL** and/or **SHIFT** to combine the shortcut key with the Ctrl key and/or Shift key.
10. Select **Input Only** to restrict the text of a graphic from showing the current value of the reference equipment.item.  
If unchecked, the text of a graphic shows the current value of the reference equipment.item.
11. Select **Use Keypad** to show a keypad during run time for the user to type a data value.
12. Click the Text Format drop-down list and select a global number format.  
For more information about the listed global number styles, see [Setting Global Number Styles](#).

**Format String** is the default numeric format, which includes a text entry field to assign a number format using four characters:

Numeric Format Character	Description
Zero, (0)	Represents a digit at each specified position of a real number. Forces leading zeros to the integer part of a number and trailing zeros to the fractional part of a

Pound sign, (#)	Represents a digit at that position within a number.
Comma, (,)	Inserts a comma at the specified position of a real number.
Decimal point, (.)	Inserts a decimal point at the specified position of a real number.

Except for the Format String and Custom text styles, all other text styles are global number styles that do not need further configuration.

13. Click **OK**.

## Related Topics

[Configuring a User Input Animation](#)

# Configuring a User Input Animation for a String Value

You can configure an element with a user input animation for a string value.

### To configure an element with a user input animation for a string value

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **User Input**. The user input animation is added to the Animation list and the **User Input** state selection panel appears.
4. Click the **String** button. The **String Value User Input** configuration panel appears.
5. In the **String** box, type an equipment.item reference or browse for one by using the browse button.
6. In the **Message to User** box, type a value, equipment.item reference or expression. This is the text that appears as prompt on the string value input dialog box at run time.
7. Make sure that the input mode of the **Message to User** box is set correctly. Click the **Input Mode** icon to set a static value or an equipment.item reference or expression.
8. You can specify that the **Input** dialog box appears by pressing a key or key combination. In the **Shortcut** area. Select a shortcut key in the **Key** list. Select **CTRL** and/or **SHIFT** to combine the shortcut key with the CTRL key and/or SHIFT key.
9. If you don't want the string value display element to show the string input result on the canvas, select **Input Only**.
10. If you want to use the keypad to type the string value, select **Use Keypad**.
11. If you select **Input Only** and want to see placeholders during the input at run time, select **Echo Characters**.
12. If you are configuring a password input:
  - Select **Password**.

- Type in the replacement character in the adjacent box.
- Select **Encrypt** if you want to encrypt the string that holds the password.

**Important:** Password encryption only works within the context of HMI/SCADA software applications that support password encryption. Do not encrypt the string if you want to pass it to an external security system, such as the operating system or a SQL Server database. The external security system cannot read the encrypted password string and access will fail.

- 
13. Click **OK**.

## Related Topics

[Configuring a User Input Animation](#)

# Configuring a User Input Animation for a Time Value

You can configure an element with a user input animation for a time value.

### To configure an element with a user input animation for a time value

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **User Input**. The user input animation is added to the Animation list and the **User Input** state selection panel appears.
4. Click the **Time** button. The **Time Value User Input** configuration panel appears.
5. In the **Time** box, type an equipment.item reference or browse for one by using the browse button.
6. In the **Message to User** box, type a value, equipment.item reference or expression. This is the text that appears as prompt on the time value input dialog box at run time.
7. Verify that the input mode of the **Message to User** box is set correctly. Click the **Input Mode** icon to set a static value or an equipment.item reference or expression.
8. Specify that the **Input** dialog box appears by pressing a key or key combination. In the **Shortcut** area. Select a shortcut key in the **Key** list. Select **CTRL** and/or **SHIFT** to combine the shortcut key with the Ctrl key and/or Shift key.
9. If you don't want the time value display element to show the time input result on the canvas, select **Input Only**.
10. To use the current date and time as default, select **Use Current Date/Time as Default**.
11. Select:
  - **Use Input Dialog** to use the **Time User Input** dialog box at run time to type date and time values in individual boxes.
  - **Use Calendar** to use the **Time User Input** dialog box at run time to type a date with the calendar control.
12. If you select **Use Input Dialog** to type the time value, you can select:
  - **Date and Time** to type date and time.
  - **Date** to only type a date.

- **Time** to only type a time.  
Select **Show Seconds** if you also want to input seconds.
13. If you want to format your text after input, type a valid text format in the **Text Format** box. Use the following letters to set the time format:

h	The one or two-digit hour in 12-hour format.
hh	The two-digit hour in 12-hour format. Single digit values are preceded by a zero.
H	The one or two-digit hour in 24-hour format.
HH	The two-digit hour in 24-hour format. Single digit values are preceded by a zero.
t	The one-letter AM/PM abbreviation ("AM" appears as "A").
tt	The two-letter AM/PM abbreviation ("AM" appears as "AM").
m	The one or two-digit minute.
mm	The two-digit minute. Single digit values are preceded by a zero.
s	The one or two-digit seconds.
ss	The two-digit seconds. Single digit values are preceded by a zero.
d	The one or two-digit day.
dd	The two-digit day. Single digit day values are preceded by a zero.
ddd	The three-character day-of-week abbreviation.
dddd	The full day-of-week name.
M	The one or two-digit month number.
MM	The two-digit month number. Single digit values are preceded by a zero.
MMM	The three-character month abbreviation.
MMMM	The full month name.
y	The one-digit year (2001 appears as "1").
yy	The last two digits of the year (2001 appears as

	"01").
yyyy	The full year (2001 appears as "2001").

**Note:** You can use any other characters, except "g" in the property. These characters then appear at design time and run time in the control.

14. Click **OK**.

## Related Topics

[Configuring a User Input Animation](#)

# Configuring a User Input Animation for an Elapsed Time Value

You can configure an element with a user input animation for an elapsed time value.

### To configure an element with a user input animation for an elapsed time value

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **User Input**. The user input animation is added to the Animation list and the **User Input** state selection panel appears.
4. Click the **Elapsed Time** button. The **Elapsed Time Value User Input** configuration panel appears.
5. In the **Elapsed Time** box, type an equipment.item reference or browse for one by using the browse button.
6. In the **Message to User** box, type a value, equipment.item reference or expression. This is the text that appears as prompt on the elapsed time value input dialog box at run time.
7. Make sure that the input mode of the **Message to User** box is set correctly. Click the **Input Mode** icon to set a static value or an equipment.item reference or expression.
8. Specify that the **Input** dialog box appears by pressing a key or key combination. In the **Shortcut** area. Select a shortcut key in the **Key** list. Select **CTRL** and/or **SHIFT** to combine the shortcut key with the CTRL key and/or SHIFT key.
9. If you don't want the elapsed time value display element to show the time elapsed input result on the canvas, select **Input Only**.
10. Select **Use Dialog** to use the **Elapsed Time User Input** dialog box to type the elapsed time value at run time.
11. If you select **Use Dialog** to type the elapsed time value, you can optionally select:
  - Show Days** if you also want to input days.
  - Show Milliseconds** if you also want to input milliseconds.
12. Click **OK**.

## Related Topics

[Configuring a User Input Animation](#)

### Configuring a Horizontal Slider Animation

You can configure an element with a horizontal slider animation. This lets you drag an element at run time in horizontal direction and write a corresponding value back to an equipment.item.

#### To configure an element with a horizontal slider animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Slider Horizontal**. The horizontal slider animation is added to the Animation list and the **Slider Horizontal** configuration panel appears.
4. In the **Analog** box, type an equipment.item reference or browse for one by using the browse button.
5. In the **Value - Left Position** box, type an analog value, equipment.item reference or expression that corresponds to the offset specified by the **Movement - To Left** value.
6. In the **Value - Right Position** box, type an analog value, equipment.item reference or expression that corresponds to the offset specified by the **Movement - To Right** value.
7. In the **Movement - To Left** box, type an analog value, equipment.item reference or expression for the maximum offset to the left in pixels.
8. In the **Movement - To Right** box, type an analog value, equipment.item reference or expression for the maximum offset to the right in pixels.
9. You can select where the cursor is anchored to the element when it is dragged at run time. In the **Cursor Anchor** area, select:
  - **Left** to anchor the element at its left.
  - **Center** to anchor the element at its center point.
  - **Right** to anchor the element at its right side.
  - **Origin** to anchor the element at its point of origin.
10. You can select if position data from the slider is written continuously to the equipment.item, or only one time when the mouse button is released. In the **Write Data** area, select **Continuously** or **On mouse release**.
11. If you want a tooltip to appear on the element showing the current value during dragging, select **Show Tooltip**.
12. Preview the movement as it appears in run time by dragging the slider in the **Preview** area.
13. Click **OK**.

## Related Topics

[Configuring Common Types of Animations](#)

### Configuring a Vertical Slider Animation

You can configure an element with a vertical slider animation.

## To configure an element with a vertical slider animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Slider Vertical**. The vertical slider animation is added to the Animation list and the **Slider Vertical** configuration panel appears.
4. In the **Analog** box, type an equipment.item reference or browse for one by using the browse button.
5. In the **Value - Top Position** box, type an analog value, equipment.item reference or expression that corresponds to the offset specified by the **Movement - Up** value.
6. In the **Value - lower Position** box, type an analog value, equipment.item reference or expression that corresponds to the offset specified by the **Movement - Down** value.
7. In the **Movement - Up** box, type an analog value, equipment.item reference or expression for the maximum offset upwards in pixels.
8. In the **Movement - Down** box, type an analog value, equipment.item reference or expression for the maximum offset downwards in pixels.
9. You can select where the cursor is anchored to the element when it is dragged at run time. In the **Cursor Anchor** area, select:
  - **Top** to anchor the element at its top side.
  - **Middle** to anchor the element at its middle point.
  - **Lower** to anchor the element at its lower side.
  - **Origin** to anchor the element at its point of origin.
10. You can select if position data from the slider is written continuously to the equipment.item, or only one time when the mouse button is released. In the **Write Data** area, select **Continuously** or **On mouse release**.
11. If you want a tooltip to appear on the element showing the current value during dragging, select **Show Tooltip**.
12. Preview the movement as it appears in run time by dragging the slider in the **Preview** area.
13. Click **OK**.

## Related Topics

[Configuring Common Types of Animations](#)

## Configuring a Pushbutton Animation

You can configure an element with a pushbutton animation to change Boolean, analog or string references.

## Related Topics

[Configuring Common Types of Animations](#)

[Configuring a Pushbutton Animation for a Boolean Value](#)

[Configuring a PushButton Animation for an Analog Value](#)

[Configuring a PushButton Animation for a String Value](#)

# Configuring a Pushbutton Animation for a Boolean Value

You can configure an element with a pushbutton to change a Boolean value.

## To configure an element with a pushbutton animation to change a Boolean value

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Pushbutton**. The pushbutton animation is added to the Animation list and the **Pushbutton** state selection panel appears.
4. Click the **Boolean** button. The **Boolean Pushbutton** configuration panel appears.
5. In the **Boolean** box, type a Boolean equipment.item reference or browse for one by using the browse button.
6. In the **Action** list, select:
  - **Direct** so the value becomes true when the element is clicked and the mouse button held. The value returns to false when the mouse button is released.
  - **Reverse** so the value becomes false when the element is clicked and the mouse button held. The value returns to true when the mouse button is released.
  - **Toggle** so the value becomes true if it is false and false if it is true when the element is clicked.
  - **Set** so the value is set to true when the element is clicked.
  - **Reset** so the value is set to false when the element is clicked.
7. If you select **Toggle, Set** or **Reset** as action, you can configure the action to be performed when the mouse button is released instead of pressed down. To do this, select **On button release**.
8. If you select Direct, Reverse, Reset or Set as action, you can configure the value to be written:
  - Continuously by selecting **Continuously while button is pressed**. Also specify the frequency the value is to be sent, by typing a value in the **Delay between value send** box.
  - One time by clearing **Continuously while button is pressed**.
9. Specify that the pushbutton action is executed by pressing a key or key combination. In the **Shortcut** area. Select a shortcut key in the **Key** list. Select **CTRL** and/or **SHIFT** to combine the shortcut key with the Ctrl key and/or Shift key.
10. Preview the pushbutton run-time behavior by clicking **Button** in the **Preview** area.
11. Click **OK**.

## Related Topics

[Configuring a Pushbutton Animation](#)

# Configuring a PushButton Animation for an Analog Value

You can configure an element with a pushbutton to set an analog value.

## To configure an element with a pushbutton animation to set an analog value

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The Edit Animations dialog box appears.
3. Click the **Add** icon and select **Pushbutton**. The pushbutton animation is added to the Animation list and the Pushbutton state selection panel appears.
4. Click the **Analog** button. The Analog Pushbutton configuration panel appears.
5. In the **Analog** box, type an equipment.item reference or browse for one by using the browse button.
6. From the **Action** list, select:
  - **Direct** so the value becomes Value1 when the element is clicked and the mouse button held. The value returns to Value2 when the mouse button is released.
  - **Toggle** so the value becomes Value1 if it is Value2 and Value2 if it is Value1 when the element is clicked.
  - **Set** so the value is set to Value1 when the element is clicked
  - **Increment** so the value is increased by Value1.
  - **Decrement** so the value is decreased by Value1.
  - **Multiply** so the value is multiplied with Value1.
  - **Divide** so the value is divided by Value1.
7. In the boxes **Value1** and, if applicable, **Value2**, type analog values, equipment.item references or references.
8. You can configure the value to be written when the mouse button is released instead. Select **On button release**. This does not apply if you select Direct as action.
9. You can configure the value to be written:
  - Continuously by selecting the **Continuously while button is pressed**. Also specify the frequency the value is to be sent, by typing a value in the **Delay between value send** box.
  - One time by clearing the **Continuously while button is pressed**.  
This does not apply if you select Toggle as action.
10. Specify that the pushbutton action is executed by pressing a key or key combination. In the **Shortcut** area. Select a shortcut key in the **Key** list. Select **CTRL** and/or **SHIFT** to combine the shortcut key with the **CTRL** key and/or **SHIFT** key.
11. Preview the pushbutton run-time behavior by clicking **Button** in the **Preview** area. Click the button multiple times to preview the value changes over a period of time.
12. Click **OK**.

## Related Topics

[Configuring a Pushbutton Animation](#)

# Configuring a PushButton Animation for a String Value

You can configure an element with a pushbutton to set a string value.

## To configure an element with a pushbutton animation to set a string value

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The Edit Animations dialog box appears.
3. Click the **Add** icon and select **Pushbutton**. The pushbutton animation is added to the Animation list and the Pushbutton state selection panel appears.
4. Click the **Analog** button. The String Pushbutton configuration panel appears.
5. In the **String** box, type an equipment.item reference or browse for one by using the browse button.
6. From the **Action** list, select:
  - **Direct** so the value becomes Value1 when the element is clicked and the mouse button held. The value returns to Value2 when the mouse button is released.
  - **Toggle** so the value becomes Value1 if it is Value2 and Value2 if it is Value1 when the element is clicked
  - **Set** so the value is set to Value1 when the element is clicked.
7. In the boxes **Value1** and, if applicable, **Value2**, type string values, equipment.item references or references.
8. Verify that the input modes of the **Value1** and **Value2** boxes are set correctly. Click the **Input mode** icons to set a static values or an equipment.item references or expressions.
9. You can configure the value to be written when the mouse button is released instead. Select **On button release**. This does not apply if you select Direct as action.
10. You can configure the value to be written:
  - Continuously by selecting the **Continuously while button is pressed**. Also specify the frequency the value is to be sent, by typing a value in the **Delay between value send** box.
  - One time by clearing the **Continuously while button is pressed**.  
This does not apply if you select Toggle as action.
11. Specify that the pushbutton action is executed by pressing a key or key combination. In the **Shortcut** area. Select a shortcut key in the **Key** list. Select CTRL and/or SHIFT to combine the shortcut key with the Ctrl key and/or Shift key.
12. Preview the pushbutton run-time behavior by clicking **Button** in the **Preview** area.
13. Click **OK**.

## Related Topics

[Configuring a Pushbutton Animation](#)

[Configuring an Action Script Animation](#)

You can configure an element with an action script animation.

You can assign multiple action scripts to one element that are activated in different ways such as:

Use this...	To activate the action script when the...
<b>On Left Click/Key/Touch Down</b>	left mouse button or a specific key is pressed or a tap on a touch display.
<b>While Left Click/Key/Touch Down</b>	left mouse button or a specific key is pressed and held or the touch display is tapped and held.
<b>On Left/Key/Touch Up</b>	left mouse button or a specific key is released or the tap is released from a touch display.
<b>On Left Double Click/Double Tap</b>	left mouse button is double-clicked or the touch display is tapped twice in quick succession.
<b>On Right Click/Long Press</b>	right mouse button is pressed or a long press on the touch display.
<b>While Right Down</b>	right mouse button is pressed and held.
<b>On Right Up</b>	right button is released.
<b>On Right Double Click</b>	right mouse button is double-clicked.
<b>On Center Click</b>	center mouse button is pressed.
<b>While Center Click</b>	center mouse button is pressed and held.
<b>On Center Up</b>	center mouse button is released.
<b>On Center Double Click</b>	center mouse button is double-clicked.
<b>On Mouse Over</b>	pointer is moved over the element.
<b>On Mouse Leave</b>	pointer is moved out of the element.
<b>On Startup</b>	element is first shown in the HMI/SCADA run time display.
<b>While Mouse Over</b>	pointer is over the element.

**Note:** To expand the available space for your script, you can use the **Expansion** buttons to hide the script header and/or the Animation list.

### To configure an element with an action script animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Action Scripts**. The action scripts animation is added to the Animation list and the **Action Scripts** configuration panel appears.
4. From the **Trigger type** list, select the trigger that activates the action script at run time.

5. If you select a trigger type that starts with "While", type how frequently the action script is executed at run time in the **Every** box.
6. If you select the trigger types **On Mouse Over or On Mouse Leave**, the **Every** box label shows **After** instead. Type a value in the **After** box. This value specifies after what delay the action script is executed at run time.
7. Specify a trigger type that involves pressing a key is run by typing a key or key combination. In the **Shortcut** area. Select a shortcut key in the **Key** list. Select **Ctrl** and/or **Shift** to combine the shortcut key with the Ctrl key and/or Shift key.
8. Create your script in the action script window.
9. Click **OK**.

## Related Topics

[Configuring Common Types of Animations](#)

[Configuring an Action Script Animation with a "Mouse-Down" Event Trigger](#)

# Configuring an Action Script Animation with a "Mouse-Down" Event Trigger

Action scripts that are activated with a "mouse-down" event (for example, On Left Click/Key/Touch Down) trigger two separate events:

- A mouse-down event, when the button is pressed.
- A mouse-up event, when the button is released.

As a result, timing issues can occur in if the user holds the button, even momentarily, before releasing it. These timing issues can affect how a pushbutton graphic is displayed. The pushbutton image may continue to show as depressed (active state), even after the user has released the button.

You can avoid this potential timing issue by using a "mouse-up" trigger instead of the "mouse down." A "mouse-up" trigger only sends one event (when the button is released), thus eliminating any timing issues that could result from a delay in the user releasing the button.

## Related Topics

[Configuring an Action Script Animation](#)

## Configuring a Show Symbol Animation

You can configure an element with a show symbol animation. A Show symbol animation shows a specified graphic in a new dialog box, when the element is clicked on.

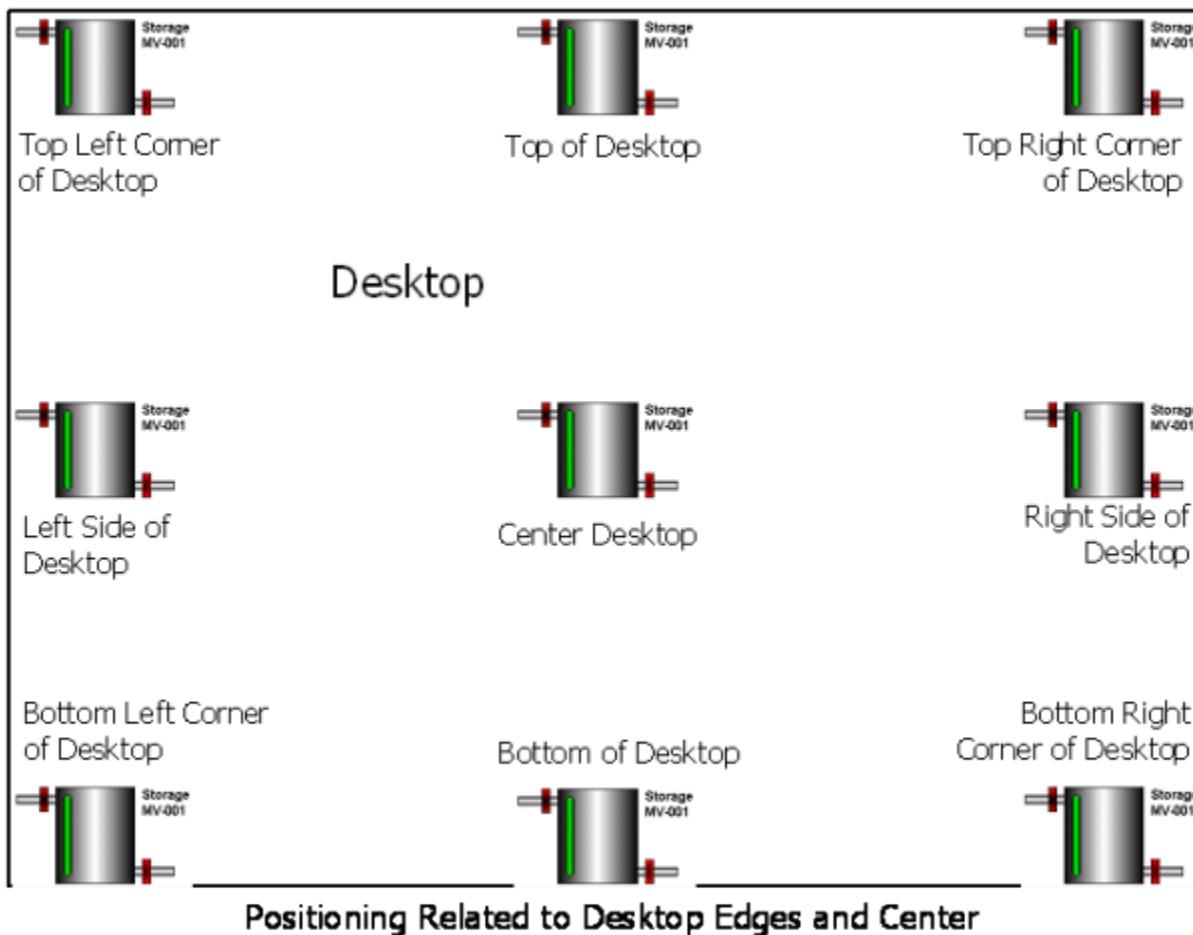
You can configure:

- Which graphic appears in the new window.
- If the window has a title bar, and if so if it has a caption.

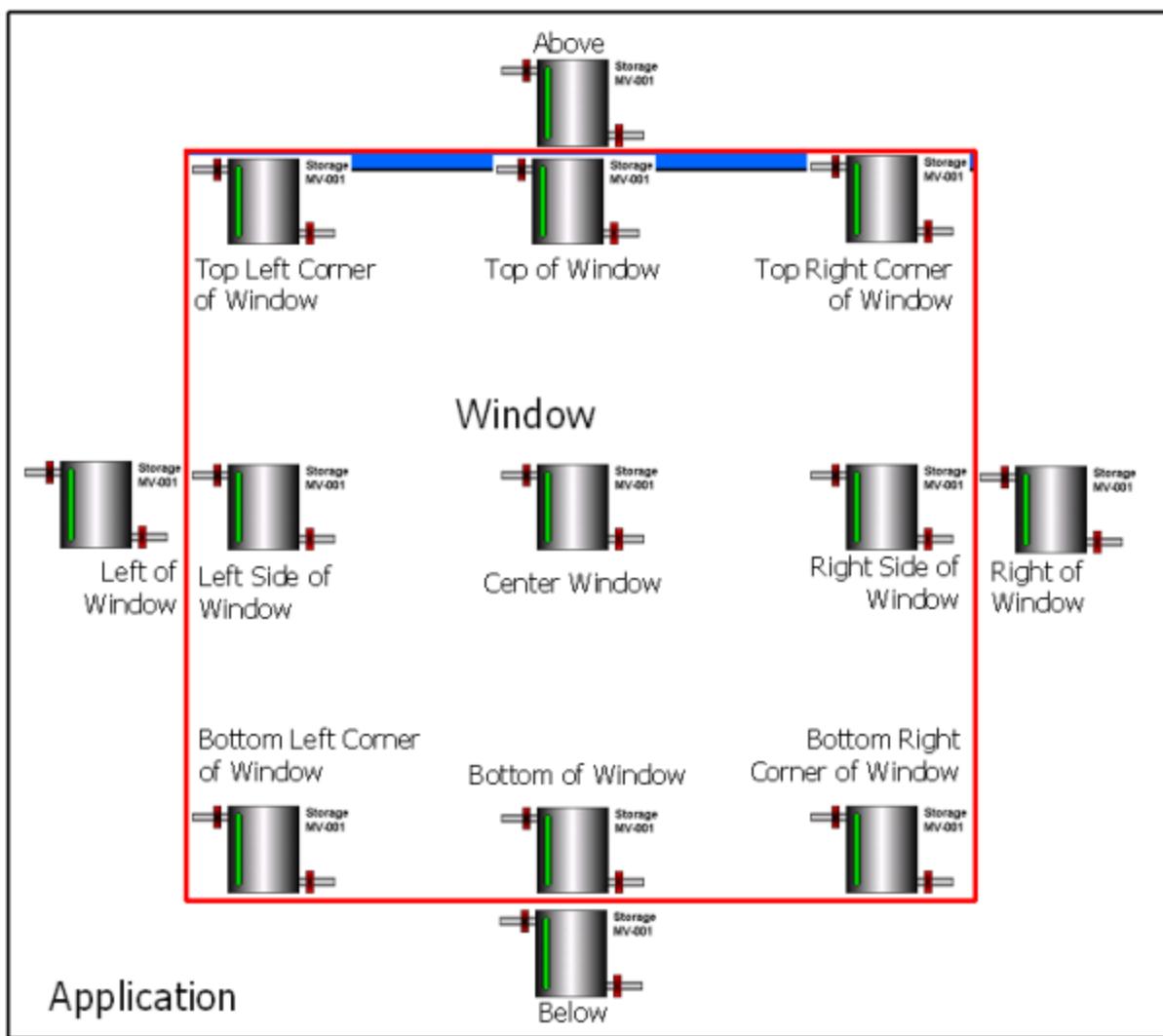
- If the window is modal or modeless.
- If the window has a close button.
- If the window can be resized.
- The initial window position.
- The size of the window.

You can configure the position to be in relation of the:

- Desktop, such as at edges, corners, or at center.

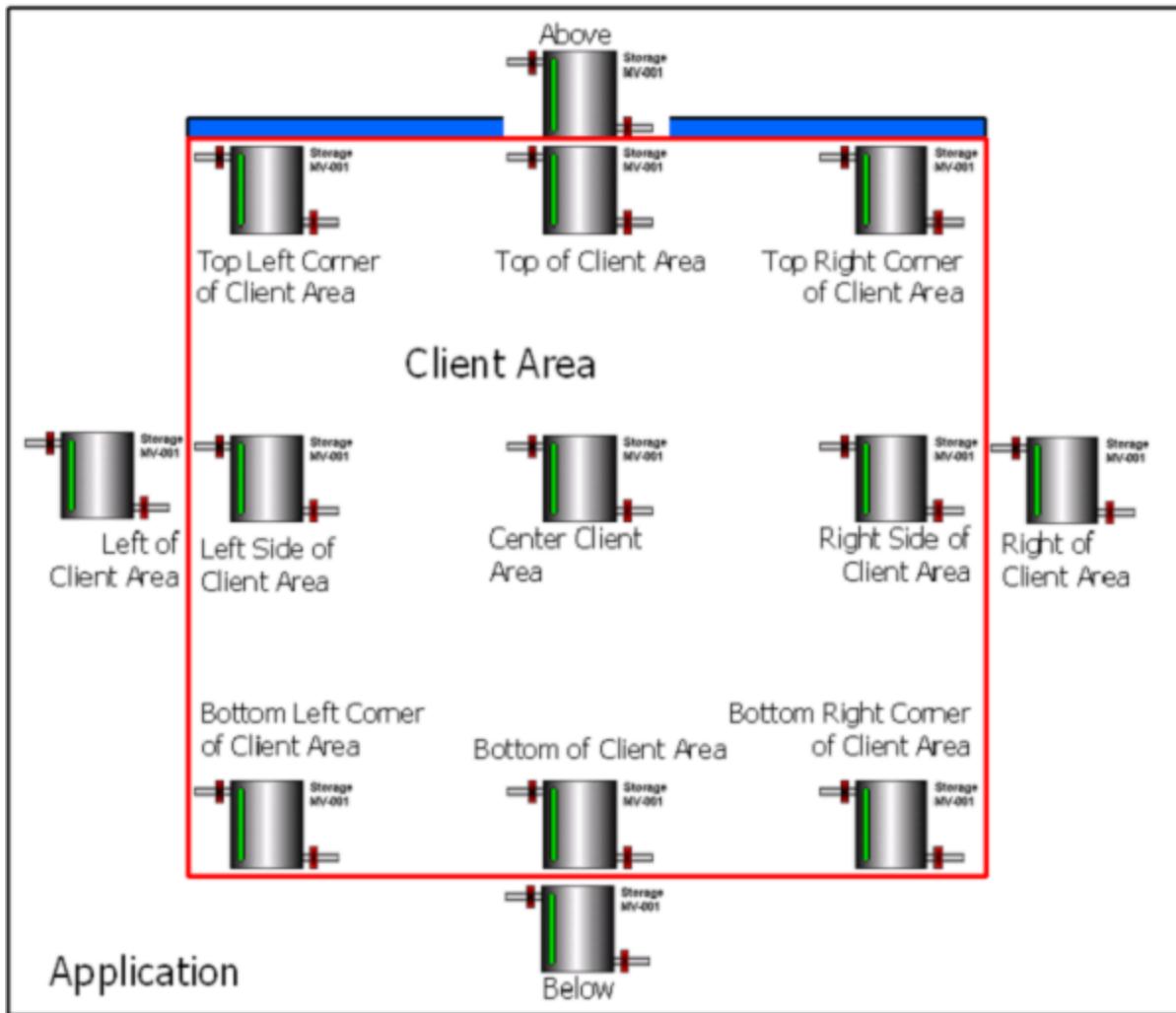


- Window, such as at one of its edges, its corners, its center or above, below, to the left or right. The window area includes the title bar if it appears.



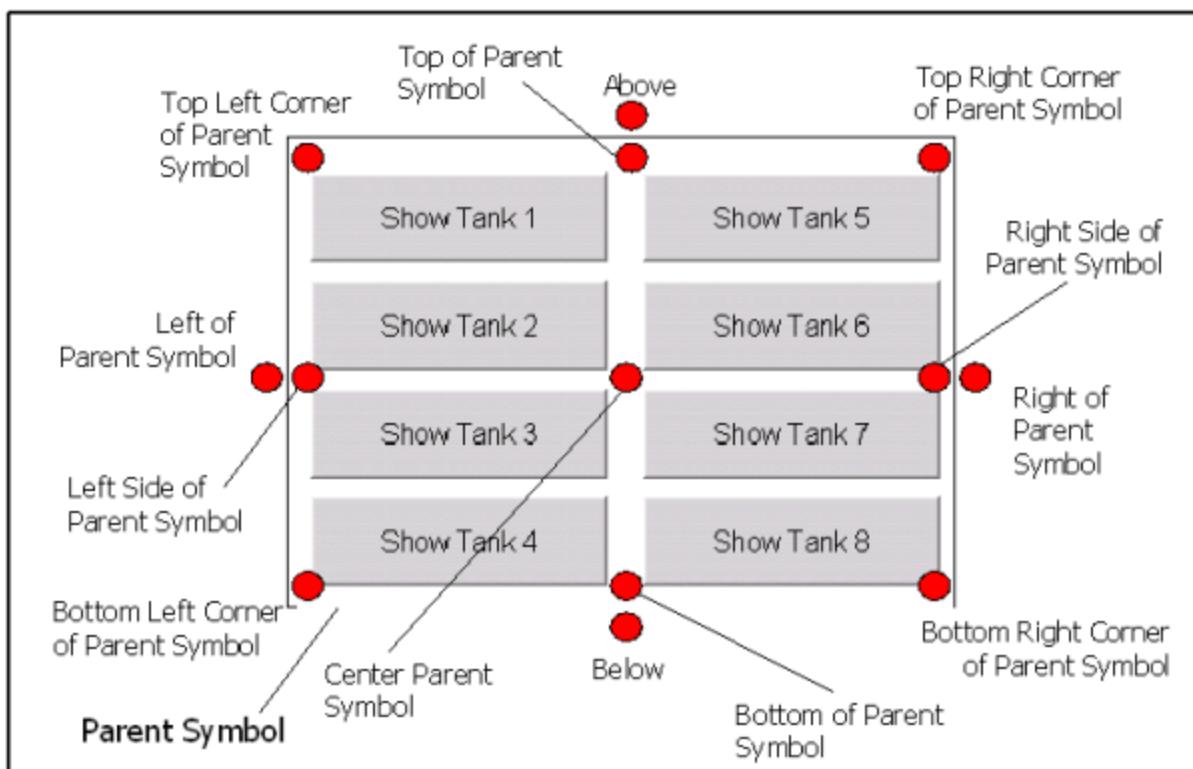
### Positioning Related to Window

- Client Area of your HMI/SCADA software.



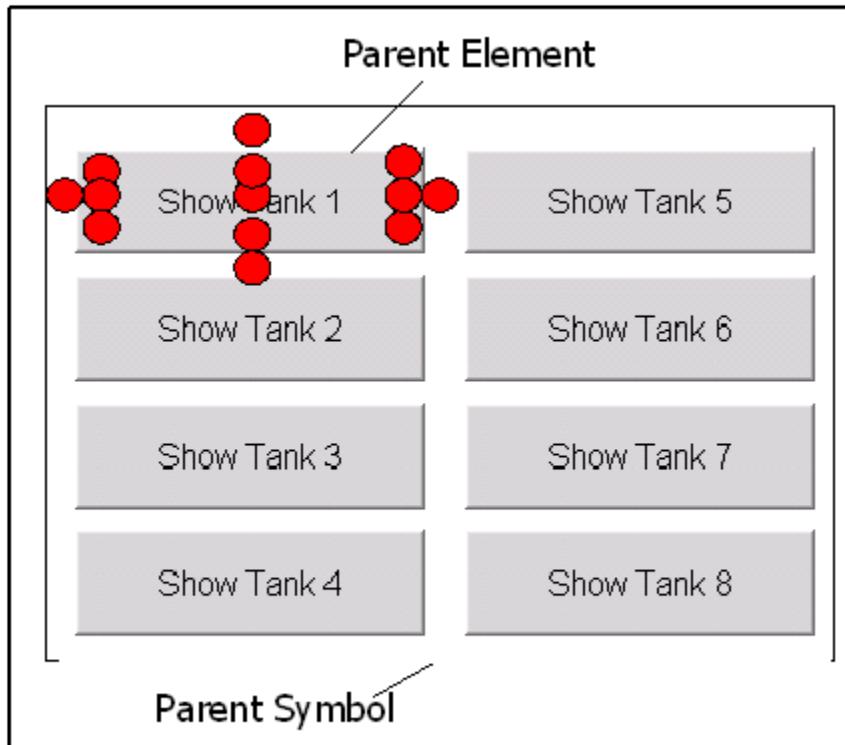
### Positioning Related to Client Area

- Source Symbol, in which case the Show Symbol window is positioned in relation to the entire source symbol that contains the element that called the window.



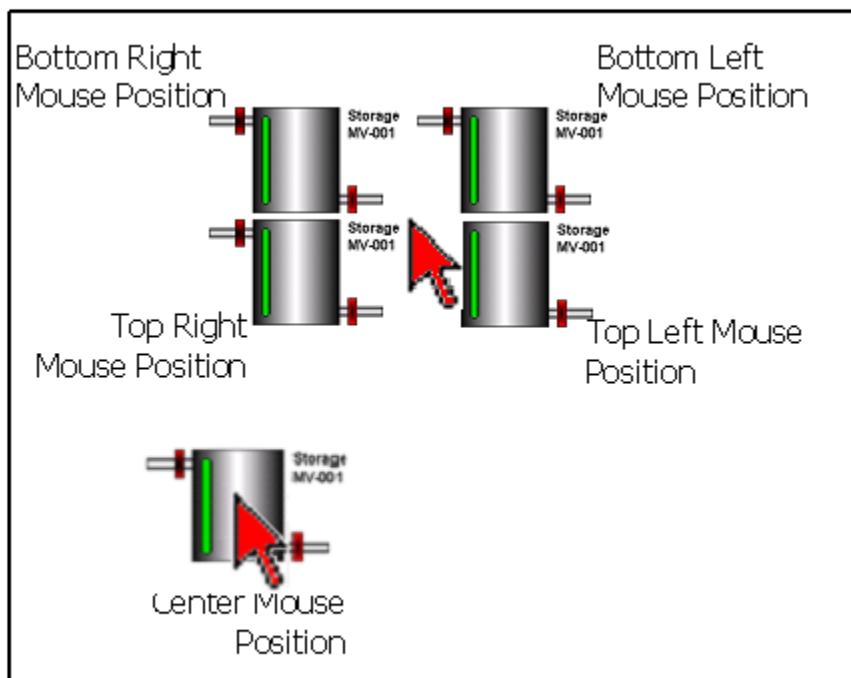
### Positioning Related to Parent Symbol

- Parent Element, in which case the Show Symbol window is positioned in relation just to the element that called the Show Symbol window.



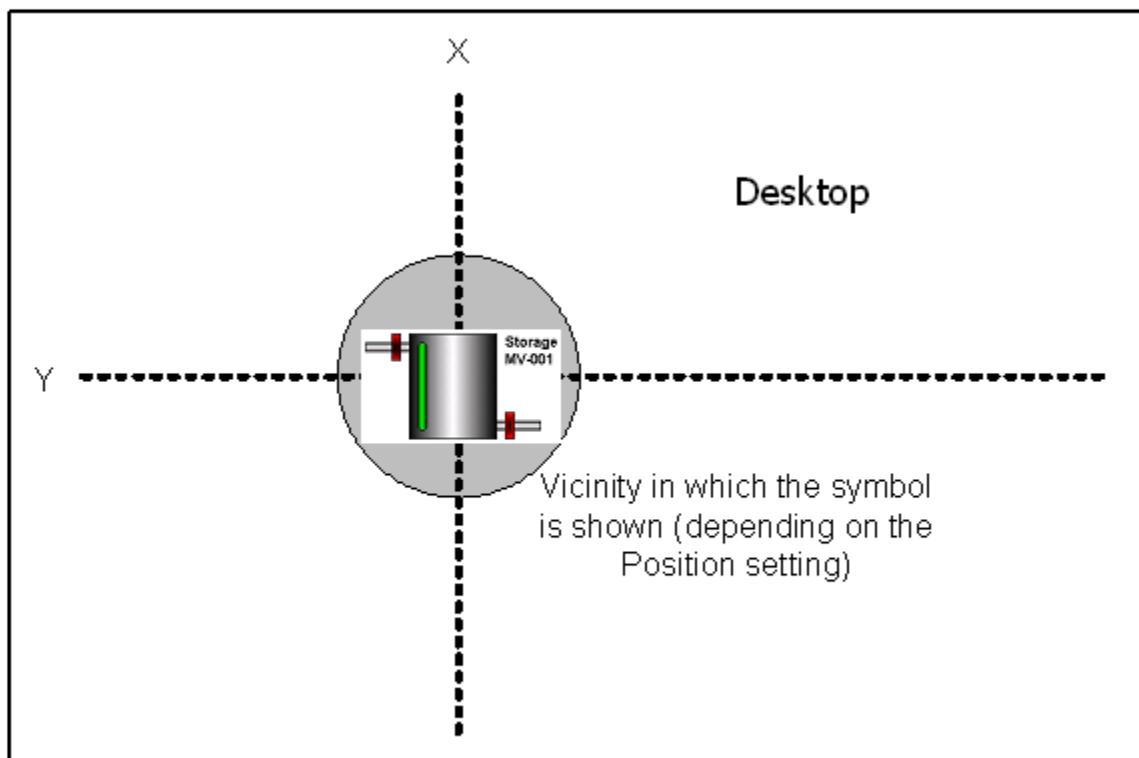
### Positioning Related to Parent Element

- Mouse, in which case the Show Symbol window is positioned in relation to the pointer coordinates.



### Selected Positionings Related to Mouse Pointer

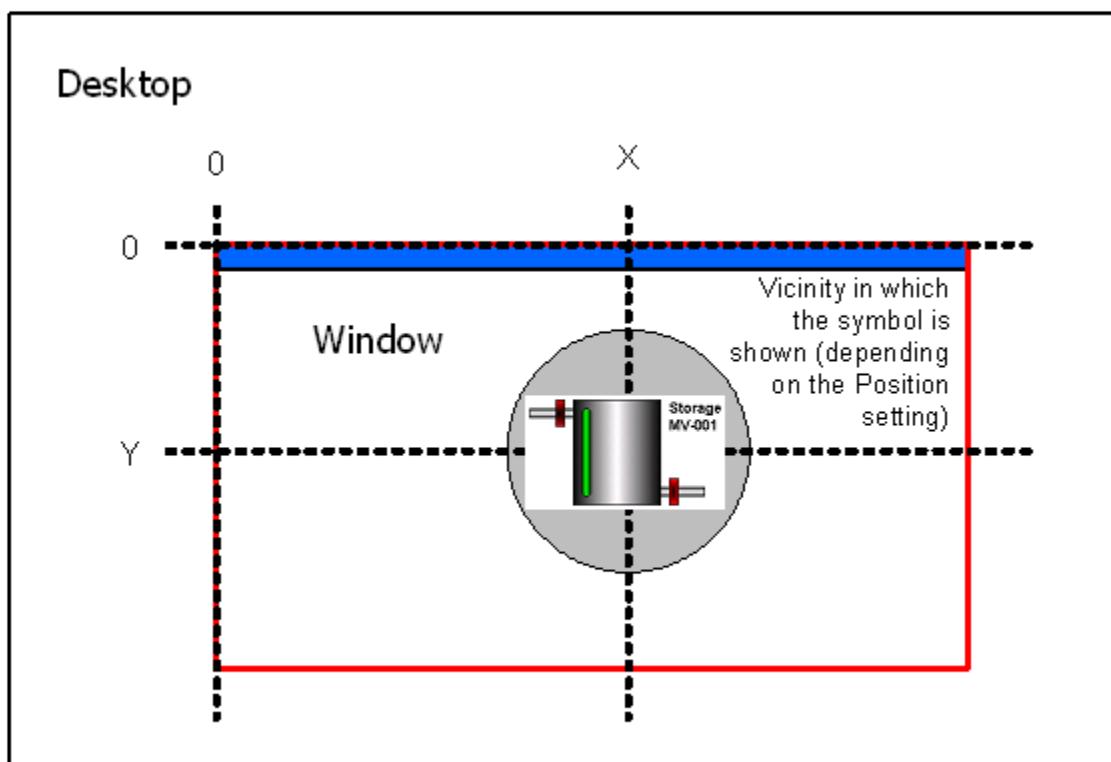
- Desktop coordinates. The graphic is placed in the vicinity of coordinates that relate to the desktop.



### X, Y Positioning Related to Desktop

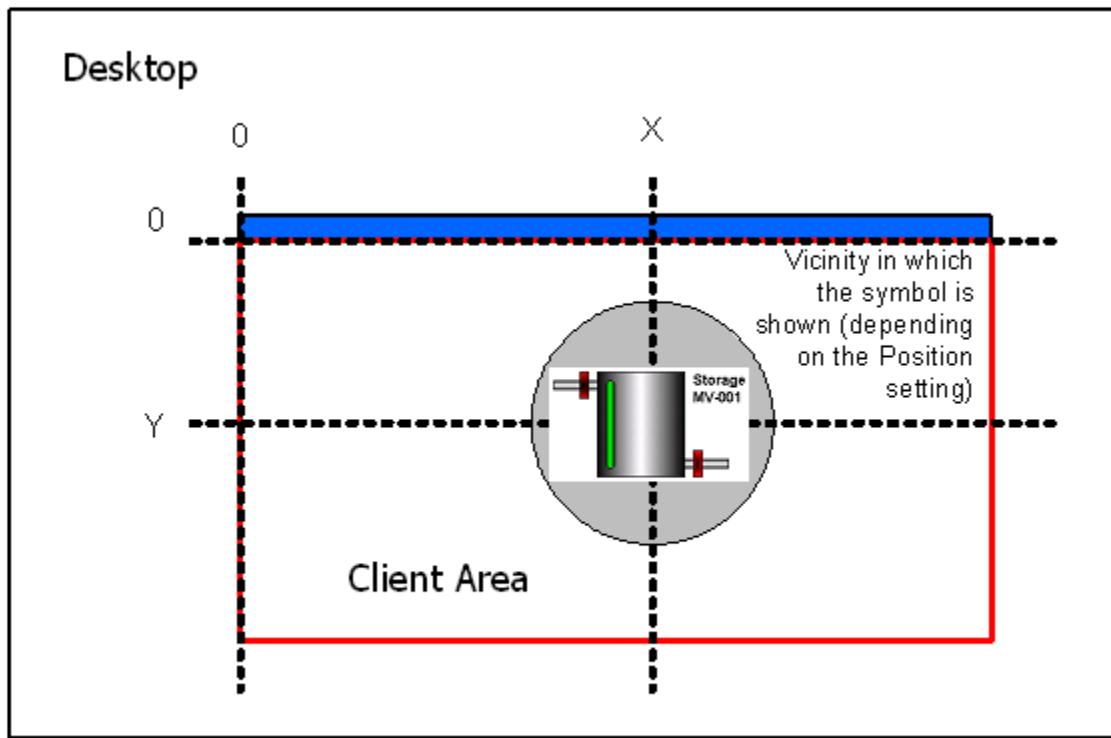
- Window coordinates. The graphic is placed in the vicinity of coordinates that relate to the window, including

the title bar if shown.



### X, Y Positioning Related to Window

- Client Area coordinates. The graphic is placed in the vicinity of coordinates that relate to the client area.



### X, Y Positioning Related to Client Area

## To configure an element with a show symbol animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Show Symbol**. The Show Symbol animation is added to the Animation list.
4. Configure the graphic. Do one or more of the following:
  - In the **Reference** box, type a graphic name or browse for one by using the browse button.
  - To add a title bar to the graphic, select **Has Title Bar**.
  - To use the graphic name as window title, select **Use Symbol Name for Window Title**.
  - Select the window type, **Modal** or **Modeless**.
  - To add a Close button, select **Has Close Button**.
  - To add resize controls, select **Resizable**.
5. Select where you want the window to appear by selecting a position in the **Position** lists. The first list contains positions that are in relation to the item of the second list. Select one of the following:
  - **Center**
  - **Top Left Corner**
  - **Top Right Corner**
  - **Left**
  - **Right of**
  - **Lower**
  - **Below**
  - **Above**
  - **Top**
  - **Left of**
  - **Right Side**
  - **Lower Left Corner**
  - **Lower Right Corner**

From the second list, select the item the position is referring to:

  - **Desktop**: relative to the entire desktop
  - **Window**: relative to the window.
  - **Client Area**: relative to the client area.
  - **Parent Symbol**: relative to the entire graphic that calls it.
  - **Parent Element**: relative to the element or element group that calls it.
  - **Mouse**: relative to the pointer.
  - **Desktop X,Y**: relative to a specified coordinate on the desktop.
  - **Window X,Y**: relative to a specified coordinate of the window.
  - **Client Area X,Y**: relative to a specified coordinate of the client area.
6. If you select **Desktop X,Y** or **Window X,Y** or **Client Area X,Y** as position, type the new coordinates in the **X** and **Y** value boxes.

7. Select how large you want the window to be in the **Size** list. You can select:
  - **Relative to Symbol** to make the window size the same as the size of the graphic.
  - **Custom Width and Height** to specify a width and height.  
Depending on your selection of the item the graphic is referring to, you can select:
    - **Relative to Desktop** to adjust the window size relative to the size of the desktop.
    - **Relative to Window** to adjust the window size relative to window that contains the graphic that calls it.
    - **Relative to Client Area** to adjust the window size relative to the client area.
    - **Relative to Parent Symbol** to adjust the window size relative to the size of the graphic that calls it.
    - **Relative to Parent Element** to adjust the window size relative to the size of the element that calls it.
8. Continue specifying position information.
  - If you select **Relative...** as size, enter a scaling percentage in the **Scale Symbol** box.
  - If you select **Custom Width and Height** as size, type the new width and height in the **W** and **H** boxes.
  - If you select **Desktop, Window, Client Area, Parent Symbol or Parent Element** as referred item, you can configure the object to be stretched horizontally or vertically. Do one or both of the following:
    - Select **Stretch symbol to fit ... width** and enter a height in the **H** box.
    - Select **Stretch symbol to fit ... height** and enter a width in the **W** box.
9. You can specify that the graphic window appears by pressing a key or key combination. In the **Shortcut** area:
  - Select a shortcut key in the **Key** list.
  - Select **Ctrl** and/or **Shift** to combine the shortcut key with the CTRL key and/or SHIFT key.
10. Click **OK**.

## Related Topics

[Configuring Common Types of Animations](#)

### Configuring a Hide Symbol Animation

You can configure an element with a Hide Symbol animation. The Hide Symbol animation lets you close:

- The current graphic
- A graphic that is shown by a specified element.

### To configure an element with a Hide Symbol animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Hide Symbol**. The Hide Symbol animation is added to the Animation list and the **Symbol Hide** configuration panel appears.
4. Select:
  - **Current Symbol** if you want to close the currently shown graphic.
  - **Symbol shown by an element** if you want to close a graphic that appears by that element. Type the element name in the adjacent box.

5. You can specify that the graphic window closes by pressing a key or key combination. In the **Shortcut** area:
  - Select a shortcut key in the **Key** list.
  - Select **Ctrl** and/or **Shift** to combine the shortcut key with the Ctrl key and/or Shift key.
6. Click **OK**.

## Related Topics

[Configuring Common Types of Animations](#)

### Configuring a Hyperlink Animation

You can configure an element with a hyperlink animation, i.e. a link when clicked will launch the default browser with a customizable URL. The animation will allow you to construct a URL animation using static text, reference or a compound expression.

#### To configure an element with a hyperlink animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animation** dialog box appears.
3. Click the **Add** icon and select **Hyperlink**. The hyperlink animation is added to the Animation list and the **Hyperlink** configuration panel appears.
4. The hyperlink can be specified as static text, an expression, or a reference.
  - Select the T icon to indicate that it is a static value.
  - Select the tag icon to indicate that it is a reference to a value.
5. For a static value type the complete URL. For example: www.google.com.
6. For a reference or expression value, select a data source.
7. You can also form expression that will use tag values in runtime to generate a URL. For example:  
"www.google.com/search?q=" + CPSearchText
8. Select the **Protocol**; http or https.
9. You can specify a shortcut by pressing a key or key combination. In the **Shortcut** area:
  - Select a shortcut key in the **Key** list.
  - Select **Ctrl** and/or **Shift** to combine the shortcut key with the Ctrl key and/or Shift key.
10. If the value is set as static text, then you can click the **Preview** button and test the link.
11. Click **OK**.

## Related Topics

[Configuring Common Types of Animations](#)

### Configuring Element-Specific Animations

Some elements have their own unique animation type that can only be used for that element type. You cannot remove their unique animation, but depending on the element you can add and remove other common

animations.

The elements with specific animations are:

- Status element
- Windows common controls

**Note:** Multi pen trends are currently not supported by Plant SCADA.

---

## Related Topics

- [Animating Graphic Elements](#)
- [Configuring Animation for a Status Element](#)
- [Configuring a Radio Button Group Animation](#)
- [Configuring a Check Box Animation](#)
- [Configuring an Edit Box Animation](#)
- [Configuring a Combo Box Animation](#)
- [Configuring a Calendar Control Animation](#)
- [Configuring a DateTime Picker Animation](#)
- [Configuring a List Box Animation](#)
- [Configuring a Trend Pen](#)
- [Submitting the Value Changes](#)
- [Format Strings in Element-Specific Animations](#)

### Configuring Animation for a Status Element

You can configure the Status element with a DataStatus animation to indicate quality and status from:

- equipment.items and tags used in elements with animation.
- equipment.items and tags directly.

The appearance of the Status element depends on the settings in the **Configure Quality and Status Display** dialog box. For more information, see [Configuring Animation for a Status Element](#).

The DataStatus animation is only used by the Status element and cannot be removed from the Status element.

#### To configure a DataStatus animation

1. Select the Status element.
2. On the **Special** menu, click **Edit Animations**. The Edit Animations dialog box appears, showing the **DataStatus** configuration panel.
3. In the **Available Graphic Elements** list, select all elements for which you want to monitor their equipment.item quality and status.
4. Click the **>>** button to add them to the **Selected Graphic Elements** list.
5. Click the **Expression** tab. The Expression panel appears.
6. In the **Value or Expression** list, type a value or expression that can be a literal, or a reference or element

property.

**Tip:** You can also browse for the reference by clicking the browse button.

7. To add more values or expressions, click the **Add** button. An additional row is added for data input.
8. Click **OK**.

## Related Topics

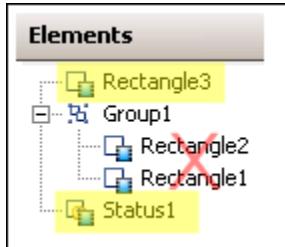
[Configuring Element-Specific Animations](#)

[Restrictions of the Status Element](#)

# Restrictions of the Status Element

Keep the Status element in the same hierarchical level as the animated elements with the equipment.items you want to monitor.

If you move elements out of their hierarchical level after you associate them with a Status element, for example, by grouping them, their equipment.items are no longer monitored.



To avoid this problem, move a new Status element in the hierarchical level you want to monitor, or associate it directly with the equipment.items you want to monitor.

## Related Topics

[Configuring Animation for a Status Element](#)

## Configuring a Radio Button Group Animation

The Radio Button Group animation is only used by the Radio Button Group element.

You can create a:

- **Static** radio button group - uses static captions and values that you define in the configuration panel.
- **Enum** radio button group - uses captions and values contained in an enum data type.

**Note:** Array radio button groups are currently not supported in Plant SCADA.

## Related Topics

[Configuring Element-Specific Animations](#)

[Configuring a Static Radio Button Group Animation](#)

[Configuring an Enum Radio Button Group Animation](#)

# Configuring a Static Radio Button Group Animation

You can configure a radio button group with static values and captions.

## To configure a static radio button group animation

1. Select the radio button group element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears and the **Static Radio Button Group** configuration panel appears on the right side.
3. In the **Reference** box, type an equipment.item reference that is to be tied to the selected value at run time. You can select when to submit the value changes. For more information, see [Submitting the Value Changes](#).
4. In the **Static Values and Captions** list, configure the captions of the radio button group and also the values that correspond to them to:
  - **Add an option** - click the **Add** icon.
  - **Delete an option** - select it in the list and click the **Remove** icon.
  - **Move an option up** the list - select it in the list and click the **Arrow up** icon.
  - **Move an option down** the list - select it in the list and click the **Arrow down** icon.
5. To use the values themselves as captions, select **Use Values as Captions**.
6. Orientate the radio button group in vertical or horizontal direction. Select **Vertical** or **Horizontal**.
7. Click **OK**.

## Related Topics

[Configuring a Radio Button Group Animation](#)

# Configuring an Enum Radio Button Group Animation

You can configure a radio button group with values from an enum equipment.item and captions.

## To configure an enum radio button group animation

1. Select the radio button group element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Enum** button. The **Enum Radio Button Group** configuration panel appears on the right side.
4. In the **Enum Reference** box, type an enum equipment.item reference. The **Enum Values and Captions** list shows the values from the enum reference.

You can select when to submit the value changes. For more information, see [Submitting the Value Changes](#).

5. To define your own captions, clear **Use Values as Captions** and type them in the list.
6. Set **Items Sorting** to:
  - **None** to show the items in the order they are in the enum equipment.item.
  - **Ascending** to show the items sorted in ascending order.
  - **Descending** to show the items sorted in descending order.
7. Orientate the radio button group in vertical or horizontal direction. Select **Vertical** or **Horizontal**.
8. Click **OK**.

## Related Topics

[Configuring a Radio Button Group Animation](#)

## Configuring a Check Box Animation

The Check Box animation is only used by the Check Box element.

### To configure a Check Box animation

1. Select the Check Box element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears and the **Check Box** configuration panel appears on the right side.
3. In the **Checked value - Boolean** box, type an equipment.item reference. The equipment.item reference is tied to the selected state of the check box control at run time.  
You can select when to submit the value changes. For more information, see [Submitting the Value Changes](#).
4. To set the caption of the check box at run-time, select **Override caption at Runtime with the following expression** and type a string value or equipment.item reference or expression in the **String Expression** box.
5. Click **OK**.

## Related Topics

[Configuring Element-Specific Animations](#)

## Configuring an Edit Box Animation

The Edit Box animation is only used by the Edit Box element. You cannot remove this animation from the Edit Box element, but you can add certain common animations.

You can also use Edit Box-specific methods in scripting to get and set the text at run time. You can browse these methods in your element browser with the Edit Box selected. For more information on these methods, see [Configuring Edit Box Methods](#).

### To configure an Edit Box animation

1. Select the Edit Box element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears and the **Edit Box**

configuration panel appears on the right side.

3. In the **String Reference** box, type a string equipment.item reference. The string equipment.item reference is tied to the text in the edit box at run time.

You can select when to submit the value changes. For more information, see [Submitting the Value Changes](#).

**Tip:** Use the **On Trigger Condition** option to set when the Edit Box element writes the run-time value to the reference. This avoids a conflict between the run-time value of the Edit Box and run-time value of the reference.

4. In the **Configuration** area, select:
  - **Multiline** to wrap the text into multiple lines in the edit box.
  - **Read-Only** to use the edit box to only show text and not allow text input.
  - **Maximum Length** to limit the maximum numbers of characters you can type in the edit box control. You can specify the maximum number in the **Characters** box.
5. Enter a default text in the **Text** box.

## Related Topics

[Configuring Element-Specific Animations](#)

### Configuring a Combo Box Animation

The Combo Box animation is only used by the Combo Box element.

You can create a:

- **Static** combo box - uses static captions and values that you define in the configuration panel.
- **Array** combo box - uses captions and values contained in an array if supported by your HMI/SCADA software.
- **Enum** combo box - uses captions and values contained in an enum data type if supported by your HMI/SCADA software.

You can also use Combo Box-specific methods in scripting to perform various functions at run time. You can browse these methods in your element browser with the Combo Box selected.

For more information on these methods, see [Configuring Combo Box and List Box Methods](#).

## Related Topics

[Configuring Element-Specific Animations](#)

[Configuring a Static Combo Box Animation](#)

[Configuring an Array Combo Box Animation](#)

[Configuring an Enum Combo Box Animation](#)

# Configuring a Static Combo Box Animation

You can configure a combo box with static values and captions.

### To configure a static combo box animation

1. Select the combo box element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears and the **Static Combo Box** configuration panel appears on the right side.
3. In the **Reference** box, type an equipment.item reference that is to be tied to the selected value at run time. You can select when to submit the value changes. For more information, see [Submitting the Value Changes](#).
4. In the **Static Values and Captions** list, configure the captions of the combo box and also the values that correspond to them:
  - **Add an option** - click the **Add** icon.
  - **Delete an option** - select it in the list and click the **Remove** icon.
  - **Move an option up** the list - select it in the list and click the **Arrow up** icon.
  - **Move an option down** the list - select it in the list and click the **Arrow down** icon.
5. Specify how you want to use captions. Do one of more of the following:
  - To use the values themselves as captions, select **Use Values as Captions**.
  - To alphabetically sort the captions, select **Sorted**.
  - To enable duplicate captions, select **Allow Duplicates**.

**Note:** If you clear **Allow Duplicates** and click **OK**, all duplicate captions are removed from combo box on the canvas. The captions are case-insensitive, so that for example "item1" is considered a duplicate of "Item1". The removal of the duplicate items is reflected when you re-open the **Edit Animations** dialog box.

6. Select the type of combo box from the **Type** list. Select:
  - **Simple** - at run time you can type a value, or select one by using arrow up and arrow down buttons. However, you cannot see the list of values.
  - **DropDown** - at run time you can type a value, or select one from the list.
  - **DropDownList** - at run time you can only select a value from the list, but not type one.
7. Click **OK**.

### Related Topics

[Configuring a Combo Box Animation](#)

## Configuring an Array Combo Box Animation

You can configure a combo box with values from an array and captions.

### To configure an array combo box animation

1. Select the combo box element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Array** button. The **Array Combo Box** configuration panel appears on the right side.
4. In the **Reference** box, type an equipment.item reference that is to be tied to the selected value at run time.

The **Array Values and Captions** list shows the values from the array reference.

You can select when to submit the value changes. For more information, see [Submitting the Value Changes](#).

5. To define your own captions, clear **Use Values as Captions** and type them in the list.
6. If you want to format the value before it appears as a caption, type a text format string in the **Format** box, for example `#.####`. Preceding zeroes are ignored if the array data type is numeric.
7. Set **Items Sorting** to:
  - **None** to show the items in the order they are in the array equipment.item.
  - **Ascending** to show the items sorted in ascending order.
  - **Descending** to show the items sorted in descending order.
8. Click **OK**.

## Related Topics

[Configuring a Combo Box Animation](#)

# Configuring an Enum Combo Box Animation

You can configure a combo box with values from an enum equipment.item and captions.

### To configure an enum combo box animation

1. Select the combo box element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Enum** button. The **Enum Combo Box** configuration panel appears on the right side.
4. In the **Enum Reference** box, type an enum equipment.item reference. The **Enum Values and Captions** list shows the values from the enum reference.

You can select when to submit the value changes. For more information, see [Submitting the Value Changes](#).
5. To define your own captions, clear **Use Values as Captions** and type them in the list.
6. Set **Items Sorting** to:
  - **None** to show the items in the order they are in the enum equipment.item.
  - **Ascending** to show the items sorted in ascending order.
  - **Descending** to show the items sorted in descending order.
7. Click **OK**.

## Related Topics

[Configuring a Combo Box Animation](#)

### Configuring a Calendar Control Animation

The Calendar Control animation is only used by the Calendar Control element. The Calendar Control date format depends on the regional settings of the operating system.

## To configure a Calendar control animation

1. Select the Calendar control element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears and the **Calendar** configuration panel appears on the right side.
3. In the **Date Reference** box, type a Time equipment.item reference that is to be tied to the selected value at run time.  
You can select when to submit the value changes. For more information, see [Submitting the Value Changes](#).
4. To restrict the date the user can select at run time, specify limits as follows:
  - In the **MinDate** box, type a lower limit for the date.
  - In the **MaxDate** box, type an upper limit for date.
5. To show some dates as bold, in the **Bolded Dates** box, type a reference that points to an equipment.item array with time data type.
6. To show today's date on the calendar control, select **Show Today**.
7. To change the colors of the calendar control, click in the **Calendar Colors** area the following color boxes:
  - **Month Background**
  - **Month Trailing Forecolor**
  - **Title Background**
  - **Title Foreground**The **Select FillColor** dialog box appears and you can select a solid color.
8. Click **OK**.

## Related Topics

[Configuring Element-Specific Animations](#)

## Configuring a DateTime Picker Animation

The DateTime Picker animation is only used by the DateTime Picker element.

### To configure a DateTime Picker animation

1. Select the DateTime Picker control element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears and the **DateTime Picker** configuration panel appears.
3. In the **Time Reference** box, type a Time equipment.item reference that is to be tied to the selected value at run time.  
You can select when to submit the value changes. For more information, see [Submitting the Value Changes](#).
4. To set the datetime format, select one of the following from the **Format** list:
  - **Long** to show the date and time in the long format of the operating system, for example: Thursday, August 03 2006.
  - **Short** to show the date and time in the short format of the operating system, for example: 8/3/2006.
  - **Time** to show just the time in the time format of the operating system, for example: 3:46:09 PM.

- **Custom** to specify your own time format. Use the following letters to set the time format:

h	The one or two-digit hour in 12-hour format.
hh	The two-digit hour in 12-hour format. Single digit values are preceded by a zero.
H	The one or two-digit hour in 24-hour format.
HH	The two-digit hour in 24-hour format. Single digit values are preceded by a zero.
t	The one-letter AM/PM abbreviation ("AM" appears as "A").
tt	The two-letter AM/PM abbreviation ("AM" appears as "AM").
m	The one or two-digit minute.
mm	The two-digit minute. Single digit values are preceded by a zero.
s	The one or two-digit seconds.
ss	The two-digit seconds. Single digit values are preceded by a zero.
d	The one or two-digit day.
dd	The two-digit day. Single digit day values are preceded by a zero.
ddd	The three-character day-of-week abbreviation.
dddd	The full day-of-week name.
M	The one or two-digit month number.
MM	The two-digit month number. Single digit values are preceded by a zero.
MMM	The three-character month abbreviation.
MMMM	The full month name.
y	The one-digit year (2001 appears as "1").
yy	The last two digits of the year (2001 appears as "01").
yyyy	The full year (2001 appears as "2001").

**Note:** You can use any other characters, except "g" in the property. These characters then appear at

---

design time and run time in the control.

5. To restrict the date the user can select at run time, you can specify limits in the:
  - **MinDate** box - type a lower limit for the date.
  - **MaxDate** box - type an upper limit for date.
6. To change the colors of the calendar control that drops down, click in the **Calendar Colors** area the following color boxes:
  - Month Background
  - Month Trailing Forecolor
  - Title Background
  - Title ForegroundThe **Select FillColor** dialog box appears and you can select a solid color.

## Related Topics

[Configuring Element-Specific Animations](#)

### Configuring a List Box Animation

The List Box animation is only used by the List Box element.

You can create a:

- **Static** list box - uses static captions and values that you define in the configuration panel.
- **Array** list box - uses captions and values contained in an array if supported by your HMI/SCADA software.
- **Enum** list box - uses captions and values contained in an enum data type if supported by your HMI/SCADA software.

You can also use List Box-specific methods in scripting to perform various functions at run time. You can browse these methods in your element browser with the List Box selected.

For more information on these methods, see [Configuring Combo Box and List Box Methods](#).

## Related Topics

[Configuring Element-Specific Animations](#)

[Configuring a Static List Box Animation](#)

[Configuring an Array List Box Animation](#)

[Configuring an Enum List Box Animation](#)

# Configuring a Static List Box Animation

You can configure a list box with static values and captions.

### To configure a static list box animation

1. Select the list box element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears and the **Static List Box** configuration panel appears on the right side.
3. In the **Reference** box, type an equipment.item reference that is to be tied to the selected value at run time. You can select when to submit the value changes. For more information, see [Submitting the Value Changes](#).
4. In the **Static Values and Captions** list, configure the captions of the list box and also the values that correspond to them. To:
  - **Add an option** - click the **Add** icon.
  - **Delete an option** - select it in the list and click the **Remove** icon.
  - **Move an option up** the list - select it in the list and click the **Arrow up** icon.
  - **Move an option down** the list - select it in the list and click the **Arrow down** icon.
5. Specify how you want to use captions. Do one or more of the following:
  - If you want to use the values themselves as captions, select **Use Values as Captions**.
  - If you want to alphabetically sort the captions, select **Sorted**.
  - If you want to allow duplicate captions, select **Allow Duplicates**.
6. Click **OK**.

### Related Topics

[Configuring a List Box Animation](#)

## Configuring an Array List Box Animation

You can configure a list box with values from an array and captions.

### To configure an array list box animation

1. Select the list box element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Array** button. The **ArrayList Box** configuration panel appears on the right side.
4. In the **Reference** box, type an equipment.item reference that is to be tied to the selected value at run time. You can select when to submit the value changes. For more information, see [Submitting the Value Changes](#).
5. In the **Array Reference** box, type or browse for an array equipment.item. The **Array Values and Captions** list shows the values from the array reference.
6. To define your own captions, clear **Use Values as Captions** and type them in the list.
7. To format the value before it appears as a caption, type a text format string in the **Format** box, for example `#.###`. Preceding zeroes are ignored if the array data type is numeric.
8. Set **Items Sorting** to:
  - **None** to show the items in the order they are in the array equipment.item.

- **Ascending** to show the items sorted in ascending order.
  - **Descending** to show the items sorted in descending order.
9. Click **OK**.

## Related Topics

[Configuring a List Box Animation](#)

# Configuring an Enum List Box Animation

You can configure a list box with values from an enum equipment.item and captions.

### To configure an enum list box animation

1. Select the radio button group element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Enum** button. The **Enum List Box** configuration panel appears on the right side.
4. In the **Enum Reference** box, type an enum equipment.item reference. The **Enum Values and Captions** list shows the values from the enum reference.
5. You can select when to submit the value changes. For more information, see [Submitting the Value Changes](#).
6. To define your own captions, clear **Use Values as Captions** and type them in the list.
7. Set **Items Sorting** to:
  - **None** to show the items in the order they are in the enum equipment.item.
  - **Ascending** to show the items sorted in ascending order.
  - **Descending** to show the items sorted in descending order.
8. Click **OK**.

## Related Topics

[Configuring a List Box Animation](#)

## Configuring a Trend Pen

A Trend Pen shows a succession of process values as a trend line consisting of current and historical data updated at a minimum of one second intervals. A trend line gives operators a quick visual snapshot of a process value over a defined period.

## Related Topics

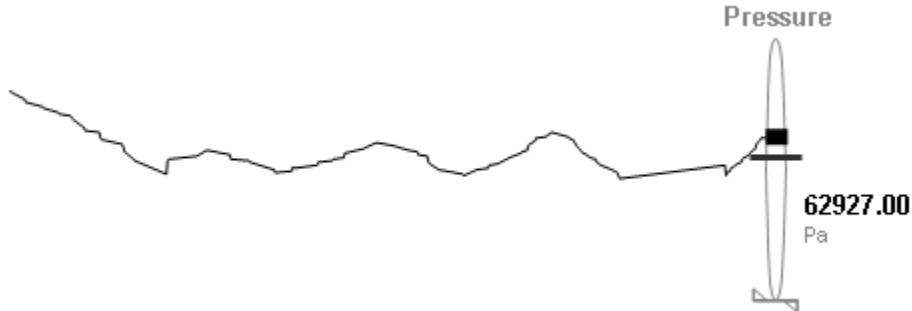
[Configuring Element-Specific Animations](#)

[Understanding the Types of Trend Plots](#)

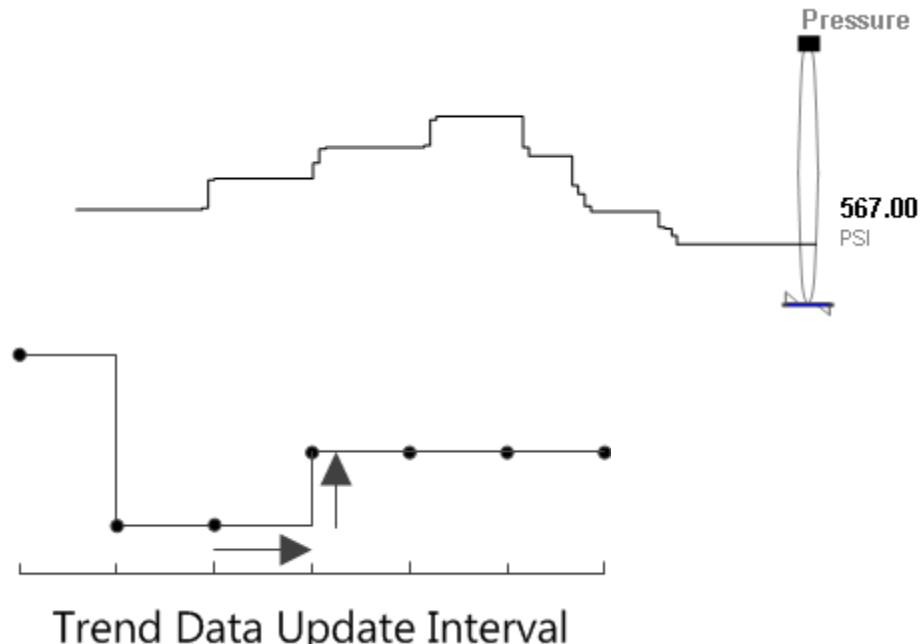
[Understanding the Types of Trend Pen Periods](#)

# Understanding the Types of Trend Plots

You can configure two types of Trend Pen plots. A Line plot draws a line between each successive data point during the trend period.



A Step Line plot draws a horizontal line from a trend data point to the time of the next point on the trend's X-axis, and then draws a vertical line to the data point. A Step Line plot is the default for a Trend Pen.



## Related Topics

[Configuring a Trend Pen](#)

# Understanding the Types of Trend Pen Periods

A trend time period is the interval of process values shown on the X-axis of the trend during run time, which consists of a start time, end time, and a duration.

- The StartTime and duration properties are read/write.

- The EndTime property is a read-only value that the system calculates by adding the duration to the start time.

You can configure two types of Trend Pen periods.

- Moving time period

In a Moving trend period, the start time of a trend period is the current time, and the end time is the duration of the time period from the start time. When the period ends, the next period begins. The start time for the next period is set to the end time of the previous trend period.

- Fixed time period

In a Fixed trend period, the start time is initially the current time. The start time of a trend period does not change automatically, but can be specified by a script using the StartTime property.

The end time of a Fixed trend period is set by the duration of the trend from the specified start time of the period.

$$\text{End Time} = \text{Start Time} + \text{Duration}$$

## Related Topics

[Configuring a Trend Pen](#)

## Submitting the Value Changes

You can configure a Windows common control to write the data:

- Immediately when it is selected in the control at run time.
- When a specified Boolean expression becomes true.

---

**Note:** The Boolean expression is a trigger that determines when the value is written from the control to the tag or equipment.item. If the value changes in the tag or equipment.item, then the value is written to the control, regardless of the trigger setting or condition.

### To submit value changes immediately

1. Open the Windows common control in the **Edit Animations** dialog box.
2. In the **Submit Value Changes** area, select **Immediately**.

### To submit value changes after a Boolean expression becomes true

1. Open the Windows common control in the Edit Animations dialog box.
2. In the **Submit Value Changes** area, select **On Trigger Condition**.
3. In the **Boolean Expression** box, type a Boolean expression or browse for a Boolean equipment.item.

## Related Topics

[Configuring Element-Specific Animations](#)

## Format Strings in Element-Specific Animations

Some element-specific animations support format strings that specify the format of data during run time when a graphic is displayed. Element specific-animations enable you to change the style of the displayed data without changing the graphic in the editor, either interactively through the use of static text or by referencing strings within data items, thereby making the format dynamic. A format string consists of one or more identifiers that define the output.

## Related Topics

[Configuring Element-Specific Animations](#)

[Numbers](#)

[Dates](#)

[Enumerations](#)

[Format String Examples](#)

# Numbers

The following table lists the basic number formatting.

Identifier	Type	Format	Example Output for "1.42"	Example Output for "-12400"
c	Currency	{0:c}	\$1.42	-\$12,400
d	Decimal (whole number)	{0:d}	Error	-12400
e	Scientific	{0:e}	1.420000e+000	-1.240000e+004
f	Fixed point	{0:f}	1.42	-12400.00
g	General	{0:g}	1.42	-12400
n	Number with commas for thousands	{0:n}	1.42	-12,400
r	Round trip	{0:r}	1.42	Error
x	Hexadecimal	{0:x4}	Error	cf90

The following table lists the custom number formatting.

Identifier	Type	Format	Example Output for "1500.42"	Notes
0	Zero placeholder	{0:00.0000}	1500.4200	Pads with zeroes.
#	Digit placeholder	{0:(#).##}	(1500).42	
.	Decimal point	{0:0.0}	1500.4	
,	Thousand separator	{0:0,0}	1,500	Must be between two zeroes.
,.	Number scaling	{0:0,..}	2	Comma adjacent to Period scales by 1000.
%	Percent	{0:0%}	150042%	Multiplies by 100, adds % sign.
e	Exponent placeholder	{0:00e+0}	15e+2	Many exponent formats available.
;	Group separator			Used to separate multiple formats in one string format (for example, including parentheses around a string if the value is negative; see <a href="#">Format String Examples</a> ).

## Related Topics

[Format Strings in Element-Specific Animations](#)

## Dates

Date formatting is dependent on the system's regional settings; the example strings here are for the U.S.

The following table lists the basic date formatting.

Identifier	Type	Example
d	Short date	10/12/2002
D	Long date	December 10, 2002

Identifier	Type	Example
t	Short time	10:11 PM
T	Long time	10:11:29 PM
f	Full date and time	December 10, 2002 10:11 PM
F	Full date and time (long)	December 10, 2002 10:11:29 PM
g	Default date and time	10/12/2002 10:11 PM
G	Default date and time (long)	10/12/2002 10:11:29 PM
M	Month day pattern	December 10
r	RFC1123 date string	Tue, 10 Dec 2002 22:11:29 GMT
s	Sortable date string	2002-12-10T22:11:29
u	Universal sortable, local time	2002-12-10 22:13:50Z
U	Universal sortable, GMT	December 11, 2002 3:13:50 AM
Y	Year month pattern	December, 2002

The following table lists the custom date formatting.

Identifier	Type	Format	Example Output
dd	Day	{0:dd}	10
ddd	Day name	{0:ddd}	Tue
dddd	Full day name	{0:dddd}	Tuesday
f, ff, ...	Second fractions	{0:fff}	932
gg, ...	Era	{0:gg}	A.D.
hh	2-digit hour	{0:hh}	10
HH	2-digit hour, 24-hr format	{0:HH}	22
mm	Minute 00-59	{0:mm}	38
MM	Month 01-12	{0:MM}	12
MMM	Month abbreviation	{0:MMM}	Dec
MMMM	Full month name	{0:MMMM}	December
ss	Seconds 00-59	{0:ss}	46

Identifier	Type	Format	Example Output
tt	AM or PM	{0:tt}	PM
yy	Year, 2 digits	{0:yy}	02
yyyy	Year	{0:yyyy}	2002
zz	Time zone offset, 2 digits	{0:zz}	-05
zzz	Full time zone offset	{0:zzz}	-05:00
:	Separator	{0:hh:mm:ss}	10:43:20
/	Separator	{0:dd/MM/yyyy}	10/12/2002

## Related Topics

[Format Strings in Element-Specific Animations](#)

# Enumerations

Identifier	Type
g	Default (flag names if available, otherwise decimal)
f	Flags always
d	Integer always
x	Eight-digit hex

## Related Topics

[Format Strings in Element-Specific Animations](#)

# Format String Examples

The following string is an example of a currency string:

```
String.Format("{0:$#,##0.00;($#,##0.00);Zero}", value);
```

This string example will output values as follows:

- **\$1,240.00** if passed **1243.50**.
- **(\$1,240.00)** if passed **-1243.50**.

- The string **Zero** if passed the number zero.

The following string is an example of a phone number string:

```
String.Format("{0:(###) ###-####}", 8005551212);
```

This string example will output **(800) 555-1212**.

## Related Topics

[Format Strings in Element-Specific Animations](#)

## Cutting, Copying and Pasting Animations

You can cut, copy and paste animations and their configuration between different elements. This is useful when you want to duplicate the animations of one element such as a line, to a different type of element such as a polyline.

If you try to paste an animation to an element that is already configured with that animation, or does not support this animation, a message appears informing you why you cannot paste the animation.

### To copy and paste animations between elements

1. Select the element from which you want to copy the animations.
2. On the **Edit** menu, point to **Animations**, and then click **Copy**.
3. Select one or more elements to which you want to paste the animations.
4. On the **Edit** menu, point to **Animations**, and then click **Paste**. The animation links are copied from the source element to the target elements.

### To cut and paste animations between elements

1. Select the element from which you want to cut the animations.
2. On the **Edit** menu, point to **Animations**, and then click **Cut**.
3. Select one or more elements to which you want to paste the animations.
4. On the **Edit** menu, point to **Animations**, and then click **Paste**. The animation links are removed from the source element and copied to the target elements.

## Related Topics

[Animating Graphic Elements](#)

## Substituting References in Elements

You can search and replace references used by any element on your canvas. You can use:

- basic mode by replacing strings in a list.
- advanced functions such as find and replace, ignore or respect case-sensitivity and wildcards.

### To substitute references in a graphic by using the list

1. Select one or more elements.
2. Do one of the following:
  - Press Ctrl + E.
3. On the **Special** menu, click **Substitute References**.  
The **Substitute References** dialog box appears.
4. In the **New** column, type the reference to be replaced.
5. Click **OK**. All references are substituted accordingly in the elements.

### To substitute references in a graphic by using find and replace functions

1. Select one or more elements.
2. Do one of the following:
  - Press Ctrl + E.
  - On the **Special** menu, click **Substitute References**.  
The **Substitute References** dialog box appears.
3. Click **Find & Replace**. The dialog box expands and shows find and replace parameters.
4. Specify your find and replace options. Do one of more of the following:
  - To find specific references in the list, type a string in the **Find What** box and click **Find Next** to find the next string.
  - To replace a selected found string with another string, type a wstring in the **Replace with** box and click **Replace**.
  - To replace multiple references, type values in the **Find What** and **Replace with** boxes and click **Replace all**.
  - To specify the search is case-sensitive, select **Match Case**.
  - To find only entire words that match your search string, select **Match Whole Word Only**.
  - To use wildcards, select **Use Wildcards**. Valid wildcards are "\*" (asterisk) and "?" (question mark).  
"\*" indicates any number of variable characters. For example. "s\*" to search for all strings starting with "s".  
"?" indicates one single variable character. For example, "M\_7?t" to search for all strings that start with "M\_7" and end with "t" and have exactly 5 characters.
5. Click **OK**. All text strings are substituted accordingly in the elements.

## Related Topics

[Animating Graphic Elements](#)

## Adding and Maintaining Graphic Scripts

### In This Chapter

- [About Graphic Scripts](#)
- [Configuring the Predefined Scripts of a Graphic](#)
- [Adding Named Scripts to a Graphic](#)
- [Editing Graphic Scripts](#)
- [Renaming Scripts in a Graphic](#)
- [Removing Scripts from a Graphic](#)
- [Substituting Equipment.item References in Scripts](#)
- [Using Methods in Scripting](#)

### About Graphic Scripts

You can associate scripts to your graphics. Scripts can add animation to a graphic or its elements that can be executed in run time.

---

**Caution:** If you configure scripts that affect more than element and graphic animation, the script processing may affect performance.

---

You can:

- Configure the predefined scripts of a graphic.
- Add named scripts to a graphic.
- Edit existing named or predefined scripts in a graphic.
- Rename named scripts in a graphic.
- Remove named scripts from a graphic.
- Substitute references in named or predefined scripts.
- Use element methods in named or predefined scripts.

The autocomplete feature is available in the Graphic Editor script editor.

### Related Topics

- [Adding and Maintaining Graphic Scripts](#)
- [Predefined and Named Scripts](#)
- [Execution Order of Graphic Scripts](#)
- [Graphic Script Time outs](#)
- [Error Handling](#)

### Predefined and Named Scripts

Predefined graphic scripts run:

- One time when the graphic is shown or opened: **On Show**
- Periodically while the graphic is showing: **While Showing**
- One time when the graphic is hidden or closed: **On Hide**
- Any combination of the above

Named graphic scripts enable any number of scripts to run that are triggered by values or expressions during runtime :

- Being true: **While True**
- Being false: **While False**
- Transitioning from false to true: **On True**
- Transitioning from true to false: **On False**
- Change in value and/or quality: **DataChange**

The name of named scripts can be up to 32 characters in length, contain at least one letter, and contain special characters, such as #, \$, and \_.

## Related Topics

[About Graphic Scripts](#)

### Execution Order of Graphic Scripts

When the graphic is showing, the scripts run in the following order:

1. On Show script.
2. Named scripts, not necessarily in the order that they appear in the list.

Any named script that is triggered by the DataChange trigger type runs the first time when the reference is subscribed to. This behavior is different than the DataChange trigger behavior of Application Server scripts and can take considerable time in intermittent networks.

**Note:** A named script will not run if the script is triggered by the DataChange trigger type and is bound to an HMI tag whose quality is Initializing, or whose quality is Bad and category is not OK.

## Related Topics

[About Graphic Scripts](#)

### Graphic Script Time outs

To avoid infinite loops in a graphic script, a time-out limit can be set in which FOR loops must complete execution. If a script loop does not complete execution within the time-out limit, WindowViewer automatically terminates the loop and writes a message to the Logger.

The time-out limit is checked only at the NEXT statement of the loop. Therefore, the first iteration of the loop is always executed, even if it exceeds the time-out limit.

## To change the time out for a graphic script

1. In WindowMaker, on the **Special** menu, point to **Configure** and click **WindowViewer**. The WindowViewer Properties dialog box appears.
2. Click the **Managed Application** tab.
3. In the **Script timeout (msec)** box, type a time-out value in milliseconds. Valid values are from 1 to 360,000.
4. Click **OK**.

## Related Topics

[About Graphic Scripts](#)

## Error Handling

A graphic script does not run if it contains a syntax error. When the graphic is loaded, a message is written to the Logger.

## Related Topics

[About Graphic Scripts](#)

## Configuring the Predefined Scripts of a Graphic

You can configure the predefined scripts of a graphic. The predefined scripts can consist of:

- A script that runs one time when the graphic opens (**On Show**).
- A script that runs periodically as long as the graphic is open (**While Showing**).
- A script that runs one time when the graphic closes (**On Hide**).

---

**Note:** The **Predefined Scripts** animation cannot be deleted. It can contain scripts for each trigger type **On Show**, **While Showing** and **On Hide**.

---

## To configure the predefined scripts for a graphic

1. Open the graphic in the Industrial Graphic Editor.
2. On the **Special** menu, click **Scripts**. The **Edit Scripts** dialog box appears.
3. In the **Trigger Type** list, click:

- **On Show** to configure a script that runs one time when the graphic opens.

If you create an **OnShow** script that uses a custom property bound to a tag, there is no guarantee that the tag data is valid when the script runs. This could occur because of the asynchronous nature of data subscriptions in HMI/SCADA software. Your script should first test the quality and status of the tag value before it is used in the rest of the script.

When the **On Show** trigger type is selected, the field that appears to the right of the **Type** list becomes a **Data Timeout** field. For more information about using this field, see [Ensuring Proper OnShow Script Execution](#).

---

**Note:** If you create an OnShow script that uses a custom property bound to a tag, there is no guarantee that the tag data is valid when the script runs. This is because of the asynchronous nature of the data subscriptions. Your script should first test the quality and status of the tag value before it is used in the rest of the script.

- **While Showing** to configure a script that runs periodically while the graphic is open.
  - **On Hide** to configure a script that runs one time when the graphic closes.
4. If you configured a **While Showing** script, type a time period in milliseconds in the **Period** box. This specifies after how many milliseconds the action script is executed.
- Note:** If you set the **While Showing** period too low, system performance may decrease.
5. Type your script in the main edit box.
- Note:** If the graphic includes a custom property, the name of the custom property and a nested class property in the script cannot be the same.
6. Select external data using your HMI's browser or explorer.
7. When you are done, click **OK**. The script editor checks the syntax of the script and may inform you of invalid syntax. Click:
  - **Yes** to save changes even if the script contains errors.
  - **No** to cancel the changes and close the script dialog box.

## Related Topics

[Adding and Maintaining Graphic Scripts](#)

[Ensuring Proper OnShow Script Execution](#)

### Ensuring Proper OnShow Script Execution

When an OnShow script includes external references to tags, it is possible that the data from these tags is not yet available when the OnShow script runs. As a result, the script might not work properly.

To avoid this situation, you can enter a value in the **Data Timeout** field. For the duration of the data time-out period, the system checks for the presence of the external reference data. After all external reference data is present, the system executes the OnShow script.

If the data time-out period expires before all external data is present, the OnShow script is executed. However, the script might not work properly.

The default value in the **Data Timeout** field is:

- For new Industrial Graphics, 1,000 ms.
- For some HMI symbols, 0 ms; that is, the presence of external reference data is not checked. Verify the behavior of your HMI/SCADA software.

The maximum data time-out value is 30,000 ms.

Note the following issues regarding OnShow scripts and the Data Timeout function:

- The Data Timeout function is not available for the other trigger script types. It would be rare for external reference data to not be available in time for those scripts.
- The execution of the OnShow script is not delayed if there is an invalid reference (that is, the reference's

quality is Bad).

- Named scripts are blocked until the OnShow script has completed, so some could be missed. For example, the named script OnDataChange might not run for the first few updates.
- Delayed OnShow scripts within nested embedded graphics might run out of order for the different nested levels. If the outer-most level is delayed but the inner levels are not delayed and are executed immediately, the order of execution will be changed.

## Related Topics

[Configuring the Predefined Scripts of a Graphic](#)

### Adding Named Scripts to a Graphic

You can add named scripts to a graphic. A named script runs:

- One time when the specified values, data or expressions change.
- Periodically if the values or expressions meet a certain criterion, such as being true.

Every named script can contain only one trigger type.

To add a named script to a graphic

1. Open the graphic in the Industrial Graphic Editor.
2. On the **Special** menu, click **Scripts**. The **Edit Scripts** dialog box appears.
3. Click the Add icon. A new entry is created in the **Named Scripts** list.
4. Type a name for the named script. The name appears on the right panel as header.
5. In the **Expression** box, do one of the following:
  - Type an expression, value or reference.
  - Browse for a reference.The expression acts as data source for the script trigger.
6. In the **Trigger** list, click:
  - **WhileTrue** to trigger the script periodically when the expression is true.
  - **WhileFalse** to trigger the script periodically when the expression is false.
  - **OnTrue** to trigger the script one time when the expression becomes true from false.
  - **OnFalse** to trigger the script one time when the expression becomes false from true.
  - **DataChange** to trigger the script one time when the expression or its quality changes. Select the **Quality Changes** check box to trigger the script when the quality of the specified expression changes.
7. If you want to specify how often the script is run when the trigger condition is fulfilled, type a time delay in milliseconds in the **Trigger Period** box.
8. If you want to specify by how much the evaluated value is expected to change before the script runs, type a deadband value in the **Deadband** box.
9. Type your script in the main edit box.
10. Use the Script Function Browser and Attribute Browser to select external data.

11. Click **OK**. The script editor validates the syntax of the script and identifies any errors. Click:
  - **Yes** to save changes even if the script contains errors.
  - **No** to cancel the changes and close the script dialog box.

## Related Topics

[Adding and Maintaining Graphic Scripts](#)

## Editing Graphic Scripts

You can edit predefined and named graphic scripts.

### To edit graphic scripts

1. Open the graphic in the Industrial Graphic Editor.
2. On the **Special** menu, click **Scripts**. The **Edit Scripts** dialog box appears.
3. Select the script from the list. The right pane shows the script configuration.
4. If you are editing a predefined script, select the script trigger from the **TriggerType** list:
  - **On Show** if the action script you want to edit runs one time after the graphic opens.
  - **While Showing** if the action script you want to edit runs periodically while the graphic is open.
  - **On Hide** if the action script you want to edit runs one time when the graphic closes.
5. Edit the action script in the script box.
6. Click **OK**.

## Related Topics

[Adding and Maintaining Graphic Scripts](#)

## Renaming Scripts in a Graphic

You can rename named scripts in a graphic. When you rename the named script, the functionality of the script does not change.

### To rename a named script

1. Open the graphic in the Industrial Graphic Editor.
2. On the **Special** menu, click **Scripts**. The **Edit Scripts** dialog box appears.
3. In the **Named Scripts** list, click the script you want to rename.
4. Click the script again. It appears in edit mode.
5. Enter a new name for the script and click **Enter**. The script is renamed.

## Related Topics

[Adding and Maintaining Graphic Scripts](#)

## Removing Scripts from a Graphic

You can remove predefined or named scripts from a graphic.

### To remove predefined scripts from a graphic

1. Open the graphic in the Industrial Graphic Editor.
2. On the **Special** menu, click **Scripts**. The **Edit Scripts** dialog box appears.
3. Select **Predefined Scripts** from the list.
4. In the **Trigger type** list, click:
  - **On Show** if the action script you want to remove runs one time after the graphic opens.
  - **While Showing** if the action script you want to remove runs periodically while the graphic is open.
  - **On Hide** if the action script you want to remove runs one time after the graphic closes.
5. Delete all the script content in the script box.
6. Click **OK**.

### To remove named scripts from a graphic

1. Open the graphic in the Industrial Graphic Editor.
2. On the **Special** menu, click **Scripts**. The **Edit Scripts** dialog box appears.
3. Select the named script from the list.
4. Click the Remove icon. A message appears.
5. Click **Yes**. The script is removed.

## Related Topics

[Adding and Maintaining Graphic Scripts](#)

## Substituting Equipment.item References in Scripts

You can substitute equipment.item references in scripts in the same way as you would with equipment.item references in elements. For more information, see [Substituting References in Elements](#).

## Related Topics

[Adding and Maintaining Graphic Scripts](#)

## Using Methods in Scripting

Some elements, such as the Edit Box, Combo Box and List Box controls, support methods in scripting. These methods can be used to perform various functions on the elements themselves at run time.

You can see the properties and methods supported by any given element by opening the browser and selecting the element.

You can use the methods of the:

- Edit Box control to save and load the text at run time to and from a file.
- Combo Box and List Box controls to access and change the contents of their lists at run time.

## Related Topics

[Adding and Maintaining Graphic Scripts](#)  
[Configuring Edit Box Methods](#)  
[Configuring Combo Box and List Box Methods](#)

### Configuring Edit Box Methods

You can use the methods of an Edit Box control to:

- Save the contained text at run time to a file.
- Load text into the control from a file at run time.

#### To save the contained text in an Edit Box control

- In an action script, use the following method:

```
ControlName.SaveText(FileName);
```

where ControlName is the name of the Edit Box control and FileName is the name of the file in which to save the contents of the control.

The text contained in the control at run time is saved to the specified file.

If you only specify a file name, the file is saved by default in the user folder of the operating system. For example: c:\documents and settings\username.

#### To load text into an Edit Box control from a file

- In an action script, use the following method:

```
ControlName.LoadText(FileName);
```

where ControlName is the name of the Edit Box control and FileName is the name of the file you want to load the text from.

The text contained in the file is loaded into the run time contents of the Edit Box control.

If you only specify a file name, by default, the file is expected to be in the user folder of the operating system. For example: c:\documents and settings\username.

## Related Topics

[Using Methods in Scripting](#)

### Configuring Combo Box and List Box Methods

The Combo Box and List Box controls have methods that you can use to access and change the items in the list at run time. Typically, you configure an action script to access these methods.

You can:

- Add and insert items into the list.
- Delete individual or all items from the list.
- Find an item in the list.
- Get the item caption based on a specified index.
- Associate the items with values.
- Load items from and save items to a file.

## Related Topics

[Using Methods in Scripting](#)

[Adding and Inserting Items into a List](#)

[Deleting Items from a List](#)

[Finding an Item in a List](#)

[Reading the Caption of a Selected Item in a List](#)

[Associating Items with Values in a List](#)

[Loading and Saving Item Lists](#)

# Adding and Inserting Items into a List

You can add an individual item:

- To the end of the list.
- Above the currently selected item.

### To add an item to a Combo Box or List Box list

- In an action script, use the following method:

```
ControlName.AddItem("ItemCaption");
```

where ControlName is the name of the Combo Box or List Box control and ItemCaption is the new item you want to add.

The item is added to the end of the list.

---

**Note:** You can specify an additional parameter `writeToSelectedItem` in the `.AddItem` function for Combo Box controls. If `writeToSelectedItem` is false, the newly added item is not written to the reference configured in the Combo box's *Selected Item Value*. For example: `Controlname.AddItem("ItemCaption", bool writeToSelectedItem);`

---

### To insert an item in a Combo Box or List Box list

- In an action script, use the following method:

```
Controlname.InsertItem("ItemCaption");
```

where ControlName is the name of the Combo Box or List Box control and ItemCaption is the new item you want to insert.

The item is inserted above the currently selected item in the list.

## Related Topics

[Configuring Combo Box and List Box Methods](#)

# Deleting Items from a List

You can delete:

- An individual item from a list.
- The selected item from a list.
- All items from a list.

If items cannot be deleted from a list at run time, no warning message is shown. Such items include Combo Box and List Box controls configured with enums or arrays.

### To delete an individual item from a Combo Box or List Box list

- In an action script, use the following method:

```
ControlName.DeleteItem(Index);
```

where ControlName is the name of the Combo Box or List Box control and Index is the index of the item you want to delete. The first item of the list has an index of 0.

The item at the specified index is deleted, subsequent items are moved up the list.

### To delete the selected item from a Combo Box or List Box list

- In an action script, use the following method:

```
ControlName.DeleteSelection();
```

where ControlName is the name of the Combo Box or List Box control.

The selected item is deleted, subsequent items are moved up the list.

### To delete all items from a Combo Box or List Box list

- In an action script, use the following method:

```
ControlName.Clear();
```

where ControlName is the name of the Combo Box or List Box control.

All items of the control are deleted.

## Related Topics

[Configuring Combo Box and List Box Methods](#)

# Finding an Item in a List

You can find an item in a Combo Box or List Box list. You specify the item caption, and the method returns the index number of the first item found. Otherwise, the method returns -1.

### Finding an item in a Combo Box or List Box list

- In an action script, use the following method:

```
Index = ControlName.FindItem("ItemCaption");
```

where ControlName is the name of the Combo Box or List Box control and ItemCaption is the caption of the item you are looking for.

The index is set to -1 if the item is not found, otherwise it contains the index of the first found item. The first item of the list has an index of 0.

## Related Topics

[Configuring Combo Box and List Box Methods](#)

# Reading the Caption of a Selected Item in a List

You can read the caption of a selected item in a Combo Box or List Box list.

### Reading the caption of a selected item in a Combo Box or List Box list

- In an action script, use the following method:

```
Caption = ControlName.GetItem(Index);
```

where ControlName is the name of the Combo Box or List Box control. Index is the index of the item for which you want to read the caption. The first item of the list has an index of 0.

Caption contains the item caption of the specified index.

## Related Topics

[Configuring Combo Box and List Box Methods](#)

# Associating Items with Values in a List

You can associate items with values in a Combo Box or List Box control. This is the same as using a secondary index system to identify items in the list.

You can:

- Set item data, which associates an item with a value
- Get item data, which returns the value that is associated with an item

### To set item data in a Combo Box or List Box list

- In an action script, use the following method:

```
ControlName.SetItemData(Index,Value);
```

where ControlName is the name of the Combo Box or List Box control, Index is the index of the item that you want to set and Value is the value you want to assign to the item. The first item of the list has an index of 0.

### To get item data in a Combo Box or List Box list

- In an action script, use the following method:

```
Value = ControlName.GetItemData(Index);
```

where ControlName is the name of the Combo Box or List Box control and Index is the index of the item for which you want to get the value. The first item of the list has an index of 0.

Value contains the value that is assigned to the item.

## Related Topics

[Configuring Combo Box and List Box Methods](#)

# Loading and Saving Item Lists

You can load and save all items in a list from and to a file.

### To load the item list for a Combo Box or List Box control from a file

- In an action script, use the following method:

```
ControlName.LoadList(FileName);
```

where ControlName is the name of the Combo Box or List Box control and FileName is the name of a file on the local harddrive or on the network.

If you only specify a file name, the file is expected to be in the users folder. For example: c:\documents and settings\username.

The list contained in the file is loaded and, if valid, the current list is overwritten.

### To save the item list for a Combo Box or List Box control to a file

- In an action script, use the following method:

```
Controlname.SaveList(FileName);
```

where ControlName is the name of the Combo Box or List Box control and FileName is the name of a file on the local harddrive or on the network.

If you only specify a file name, the file is saved to the users folder. For example, c:\documents and settings\username.

The list is saved to the specified file.

## Related Topics

[Configuring Combo Box and List Box Methods](#)

## Embedding Graphics within Graphics

### In This Chapter

[Embedding Graphics](#)

[Editing the Embedded Graphic](#)

[Overriding Custom Properties of the Source Graphic](#)

[Restoring an Embedded Graphic to the Original Size of its Source Graphic](#)

[Converting an Embedded Graphic to a Group](#)

[Detecting the Source Graphic of an Embedded Graphic](#)

[Editing the Source of an Embedded Graphic](#)

[Controlling Size Propagation of Embedded Graphics](#)

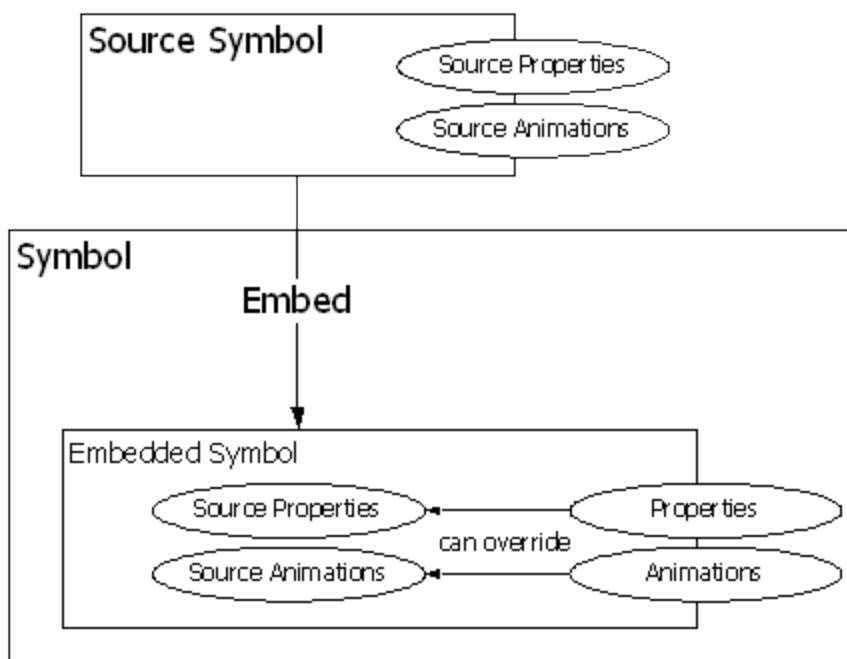
## Embedding Graphics

You can embed graphics into other graphics. This lets you split your visualization into modules and re-use already defined graphics. For example, you can create a valve graphic and embed it multiple times into a tank graphic.

When you embed a graphic into another graphic, you are creating a link to the original graphic. Any changes to the original graphic are propagated to all embedded instances.

You can:

- Embed a graphic within another graphic.
- Edit an embedded graphic.
- Restore an embedded graphic to the original size of its source graphic.
- Convert the embedded graphic to a graphic.
- Detect the source of an embedded graphic.
- Edit the source of an embedded graphic.
- Override the custom properties of the source graphic.
- Control the size propagation of an embedded graphic.
- Select an alternate or same graphic of an alternate object instance as source.
- Edit the object that contains the source graphic.
- Create a new instance of the object that contains the source graphic.



You can embed graphics from the Industrial Graphic Editor into other graphics.

When you embed a graphic, the animation links and scripts are inherited from the source graphic. You can only change the animations and scripts in the source graphic and all changes are propagated to the embedded graphic.

The embedded graphic appears with its original name appended by a number. The number is increased by one if you embed the same graphic again.

---

**Note:** If you embed graphics that have elements outside of the coordinates (0,0) and (2000,2000), the embedded graphic clips these elements. The name of the embedded graphic cannot be the same as a custom property of the graphic in which it is being embedded. The embedded graphic and the graphic in which it is being embedded cannot include elements that have the same name.

---

### To embed source graphics from the Industrial Graphic Editor

1. On the Industrial Graphic Editor **Edit** menu, click **Embed Graphic Symbol**. Your HMI's graphics browser appears.
2. Select a source graphic.
3. Click **OK**. The pointer appears in paste mode.
4. Click on the canvas to place the graphic.

### Related Topics

[Embedding Graphics within Graphics](#)

### Editing the Embedded Graphic

After you embed a source graphic into another graphic, its animations are inherited from the source graphic. The animation of the embedded graphic is controlled by the source graphic.

The embedded graphic itself has certain animations you can configure. The animations override the animations of the source graphic for the embedded graphic. These are:

- Visibility
- Blink
- Location Horizontal
- Location Vertical
- Width
- Height
- Orientation
- Disable

Furthermore you can override the following animations if you change the TreatAsIcon property of the embedded graphic to True:

- Tooltip
- User Input
- Slider Horizontal
- Slider Vertical
- Pushbutton
- Action Scripts
- Show Symbol
- Hide Symbol

### To override the configured animations of an embedded graphic

1. Select the embedded graphic.
2. In the Properties Editor, change the value for the TreatAsIcon property to True.

## Related Topics

[Embedding Graphics within Graphics](#)

## Overriding Custom Properties of the Source Graphic

You can override the value and description of a custom property of the embedded graphic if the custom property's visibility is set to Public in the source graphic.

You cannot add, delete, or rename any custom properties of an embedded graphic or change the data type. However, you can:

- Revert the value and description of the custom property to its default as defined in the source graphic.
- Set the visibility of the custom property. This has an effect if the graphic containing the embedded graphic is embedded into another graphic.

### To override the value and description of a custom property

1. Select the embedded graphic on the canvas.
2. On the **Special** menu, click **Custom Properties**. The **Edit Custom Properties** dialog box appears.
3. Select the custom property you want to override with a new value or description.
4. In the **Default Value** box, type a new value.
5. In the **Description** box, type a new description.

### To revert the value and description of a custom property

1. Select the embedded graphic on the canvas.
2. On the **Special** menu, click **Custom Properties**. The **Edit Custom Properties** dialog box appears.
3. Select the custom property you want to revert.
4. Click the Revert icon. The value and description of the selected custom property are reverted to the value and description of the same custom property in the source graphic.

## Related Topics

[Embedding Graphics within Graphics](#)

### Restoring an Embedded Graphic to the Original Size of its Source Graphic

You can restore an embedded graphic to its original size as it is defined in the object or in the Industrial Graphic Editor.

#### To restore an embedded graphic to its original size

1. Select the embedded graphic that you want to restore to its original size.
2. On the **Edit** menu, point to **Embedded Symbol**, and then click **Symbol - Original Size**. The embedded graphic is restored to the original size of its source graphic.

## Related Topics

[Embedding Graphics within Graphics](#)

### Converting an Embedded Graphic to a Group

You can convert an embedded graphic to a group. A converted graphic is no longer associated with its source graphic. All configuration of the embedded graphic is preserved.

If you convert an embedded graphic to a group:

- Scripts of the embedded graphic are not converted.
- You can optionally move the custom properties to the group.
- Relative references of the embedded graphic are no longer valid.
- Wizard options are no longer supported.

## To convert an embedded graphic to a group

1. Select the embedded graphic that you want to convert to a group.
2. On the **Edit** menu, point to **Embedded Symbol**, and then click **Convert To Group**. The embedded graphic is converted to a group.

## Related Topics

[Embedding Graphics within Graphics](#)

## Detecting the Source Graphic of an Embedded Graphic

You can view the source of an embedded graphic by using the **SymbolReference** property.

### To detect the source of an embedded graphic

1. Select the embedded graphic on the canvas.
2. In the Properties Editor, view the **SymbolReference** property to see what object or environment contains the source and the name of the source graphic itself. This can be:
  - *Symbol:SymbolName*.
  - *Symbol:InstanceName.SymbolName*

## Related Topics

[Embedding Graphics within Graphics](#)

## Editing the Source of an Embedded Graphic

You can edit the source of an embedded graphic by opening it in a new session of the Industrial Graphic Editor.

### To edit the source of an embedded graphic

1. Select the embedded graphic.
2. On the **Edit** menu, point to **Embedded Symbol**, and then click **Edit Symbol**. The source of the embedded graphic is opened in a new session of the Industrial Graphic Editor.
3. Edit the source graphic as needed and click **Save and Close**. The new session of the Industrial Graphic Editor is closed and the **Symbol Changed** icon appears in the status bar.
4. Double-click the **Symbol Changed** icon. The change is reflected in the embedded graphic.  
If you do not accept the change, the embedded graphic is updated the next time you open it in the Industrial Graphic Editor.

## Related Topics

[Embedding Graphics within Graphics](#)

## Controlling Size Propagation of Embedded Graphics

You can control the way that size changes of the source graphic are propagated to its embedded instances, which are embedded graphics. For example, a size change is:

- Resizing one of the elements in the source graphic so that the graphic boundary changes.
- Adding elements to or removing elements from the source graphic so that the graphic's boundary changes.

This feature is called dynamic size change and can be enabled or disabled.

## Related Topics

[Embedding Graphics within Graphics](#)

[Setting the Anchor Point of a Source Graphic](#)

[Showing or Hiding the Anchor Points of Embedded Graphics](#)

[Enabling or Disabling Dynamic Size Change of Embedded Graphics](#)

## Setting the Anchor Point of a Source Graphic

You can set the position of the anchor point of a source graphic. The anchor point of a source graphic is by default the center point of all elements on the canvas.

You can change the position of the anchor point:

- Numerically by typing the absolute or relative anchor point position values in the Properties Editor.
- Graphically by dragging the anchor point on the canvas.

### To change the position of the anchor point numerically

1. Click on the canvas.
2. In the Properties Editor, type position values X,Y for:
  - **AbsoluteAnchor** property, where the position is relative to the top left corner of the canvas 0,0.
  - **RelativeAnchor** property, where the position is relative to the center point of all elements on the canvas.The anchor point is changed accordingly. The **AbsoluteAnchor** and **RelativeAnchor** property values are updated accordingly.

### To change the position of the anchor point graphically

1. Click on the canvas.
2. In the Properties Editor, click the **AbsoluteAnchor** or **RelativeAnchor** property label. The anchor point of the graphic is shown.
3. Drag the anchor point to the new position. The **AbsoluteAnchor** and **RelativeAnchor** property values are updated accordingly.

## Related Topics

[Controlling Size Propagation of Embedded Graphics](#)

### Showing or Hiding the Anchor Points of Embedded Graphics

You can show or hide the anchor points of embedded graphics. An anchor point shows the current absolute anchor position of the embedded graphic on the canvas.

#### To show or hide the anchor point of an embedded graphic

- On the toolbar, click the **Show/Hide Embedded Symbol Anchor Points** icon. The anchor of the embedded graphic appears or disappears.

## Related Topics

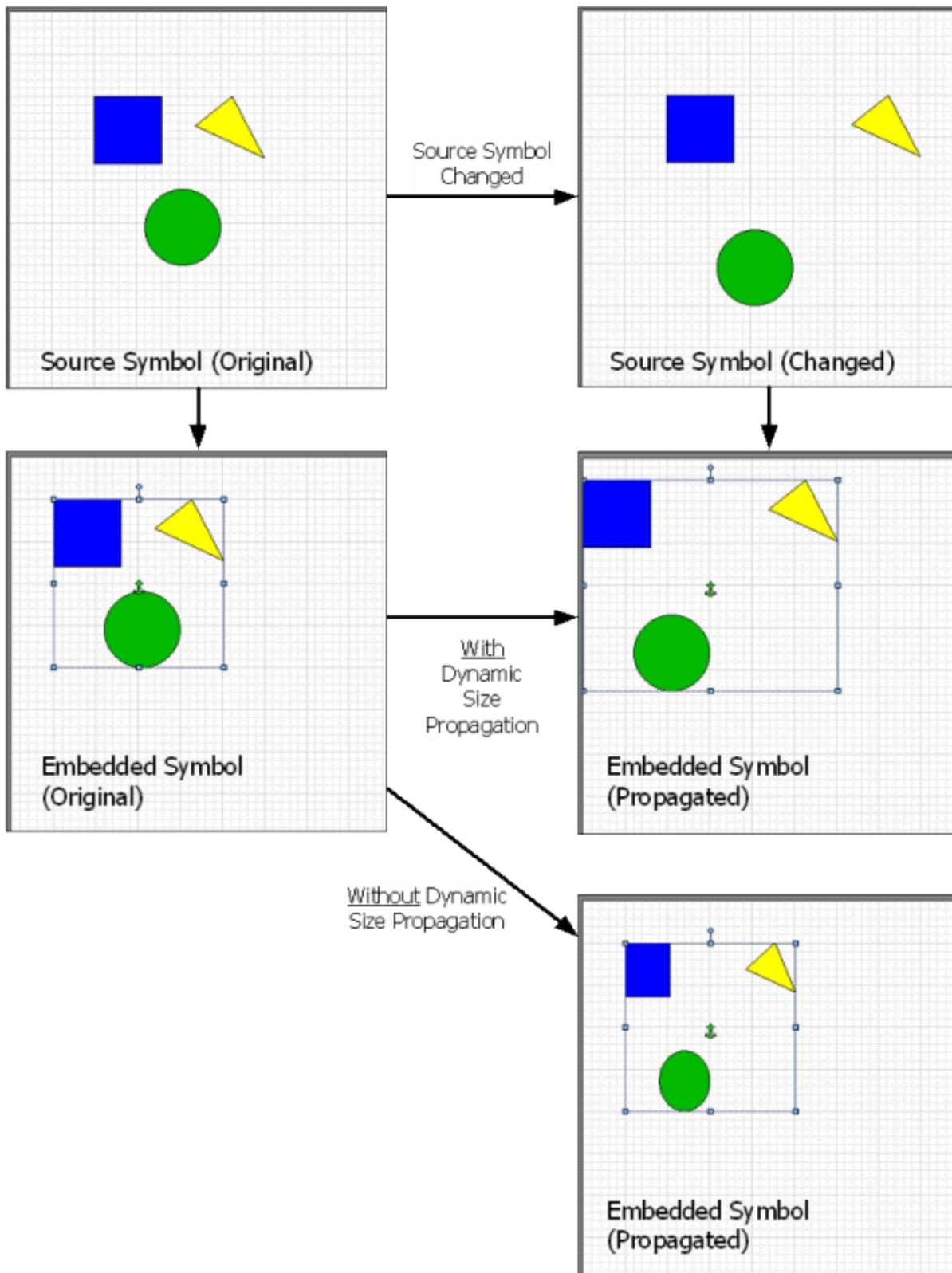
[Controlling Size Propagation of Embedded Graphics](#)

### Enabling or Disabling Dynamic Size Change of Embedded Graphics

You can enable or disable the dynamic size change of embedded graphics. The anchor points of the embedded instances are not changed by any size change to the source graphic.

If the source graphic size changes and the dynamic size change is enabled, the embedded graphic size adapts accordingly. If the dynamic size change is disabled, the embedded graphic size does not change.

In both cases the anchor points of its embedded instances do not move on the canvas.



#### To enable or disable dynamic size change of an embedded graphic

1. Select the embedded graphic on the canvas.
2. On the toolbar, click the **Enable/Disable Dynamic Size Change** icon. The dynamic size change is enabled or

disabled.

## Related Topics

[Controlling Size Propagation of Embedded Graphics](#)

## Working with Symbol Wizards

### In This Chapter

[Introduction](#)

[Understanding the Symbol Wizard Editor](#)

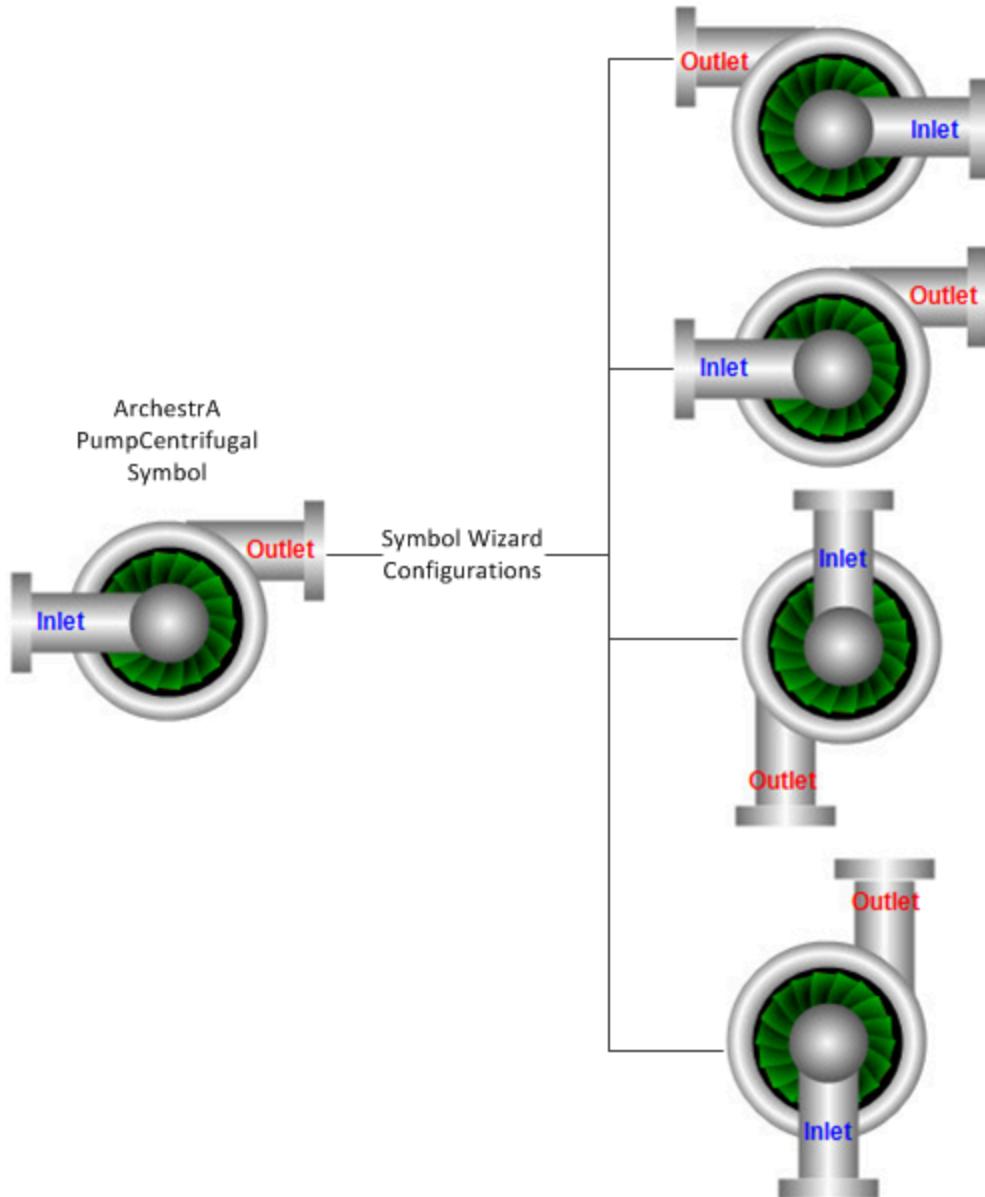
[Designing a Symbol Wizard](#)

[Using Symbol Wizards in an Application](#)

[Symbol Wizard Tips and Examples](#)

### Introduction

The Industrial Graphic Editor includes the Symbol Wizard Editor, which can be used to create reusable configurable graphics called Symbol Wizards. For example, a single pump symbol can be created with the Symbol Wizard Editor that includes different visual pump configurations based on the orientation of inlet and outlet pipes.



Incorporating multiple configurations in a single graphic reduces the number of graphics needed to develop an application.

## Related Topics

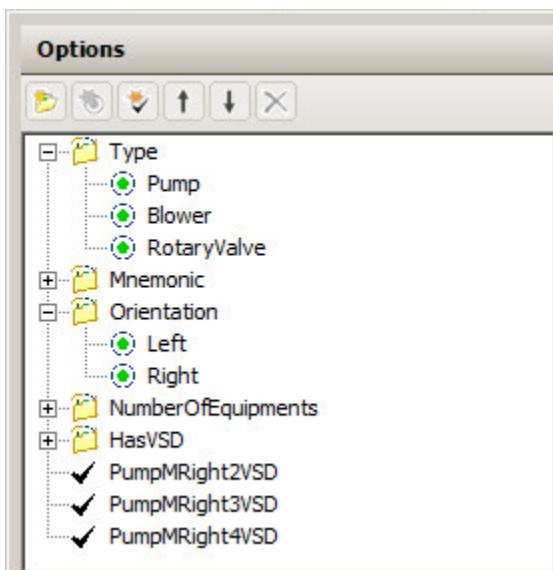
[Working with Symbol Wizards](#)

## Understanding the Symbol Wizard Editor

After enabling the Symbol Wizard Editor, the Industrial Graphic Editor window updates to show Symbol Wizard Editor panes at the left of the window.

- Beneath the **Tools** pane, separate tabbed panes show the graphic elements, custom properties, and named scripts that belong to a graphic.

- The tabbed **Options** pane shows a hierarchical list of Choice Groups, Choices, and Options that define graphic configurations.



The **Options** pane includes buttons to add, delete, and reorder Choice Groups, Choices, and Options.

- The tabbed **Layers** view includes a list of defined graphic layers. Beneath each layer, separate folders contain the graphic's elements, custom properties, and named scripts associated with the layer. A graphic's elements, custom properties, and named scripts are assigned to graphic layers by dragging them to corresponding folders in the **Layers** view.

## Related Topics

[Working with Symbol Wizards](#)

[Understanding Choice Groups and Choices](#)

[Understanding Symbol Wizard Layers](#)

[Defining Graphic Configuration Rules](#)

## Understanding Choice Groups and Choices

The Symbol Wizard Editor **Options** pane includes buttons to create Choice Groups, Choices, and Options.

- A Choice Group represents a unique property of a graphic and appears as the top level property node in the **Options** view.
- A Choice represents a possible value of a Choice Group property. Choices are indented beneath the associated Choice Group node in the **Options** view. Choices are mutually exclusive and only one choice can be selected from a Choice Group for a single configuration of a graphic.

An item shown in the **Options** view list can be moved by selecting it, and then clicking the **Up** or **Down** arrow. If no Choice is specified as the default value for a Choice Group, the first Choice added to the Choice Group is always the default value.

In the example of an centrifugal pump graphic, one possible Choice Group is Orientation for the different configurations of inlet and outlet pipes. The Left, Right, Bottom, and Top choices appear as the associated Choice

attributes of the Orientation Choice Group.

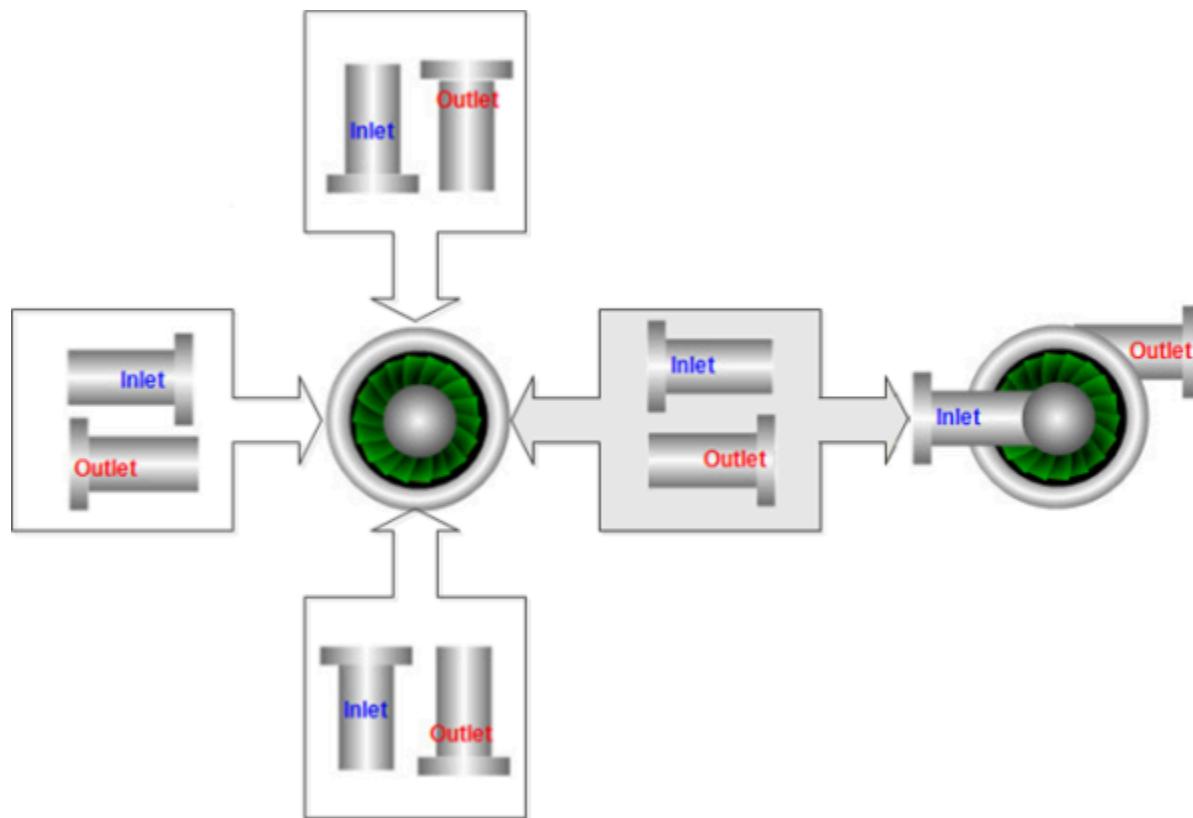
## Related Topics

[Understanding the Symbol Wizard Editor](#)

### Understanding Symbol Wizard Layers

Symbol Wizard layers associate graphic elements, custom properties, and named scripts to a unique graphic configuration defined by a rule. When the rule is True, the layer's graphic elements, custom properties, and named scripts are part of the Symbol Wizard's configuration.

In the example of a centrifugal pump graphic, a rule determines the orientation of the pump's inlet and outlet pipes. When the rule for the Right configuration is True, the Right layer containing the inlet and outlet pipes is part of the graphic's configuration.



The blade housing does not belong to a layer because it is common to all pump graphic configurations. Graphic elements of a graphic that do not belong to a layer appear in all graphic configurations. As a result, the pump's blade housing appears in the Left, Right, Top, and Bottom configurations of the pump by default.

Likewise, adding graphic elements, custom properties, and named scripts to a layer without a rule results in these elements appearing in all graphic configurations. Each layer must have a defined rule that specifies a True condition when the set of graphic elements, custom properties, and named scripts are part of a graphic configuration.

Associating graphic elements, named scripts, and custom properties to graphic layers involves working with the Symbol Wizard Editor **Layers** pane shown to the left of the graphic canvas.

## Related Topics

[Understanding the Symbol Wizard Editor](#)

### Defining Graphic Configuration Rules

A rule defines an expression that determines if a given choice or option and its associated graphic layer is visible or hidden based on the evaluation of the rule to true or false.

Rules can consist of a single expression or compound expressions using Boolean keywords or operator characters:

Boolean Keywords	AND, OR, NOT
Operator Characters	<b>Period (.)</b> A period concatenates a Choice Group to a Choice in a hierarchical expression. <b>Pipe ( )</b> A pipe evaluates to a Boolean OR. <b>Ampersand (&amp;)</b> An ampersand evaluates to a Boolean AND. <b>Exclamation point (!)</b> An exclamation point evaluates to a Boolean NOT. <b>Parentheses ( )</b> A compound expression enclosed within parentheses is evaluated before other expressions in a rule

Any other unlisted keywords or operator characters in a rule are treated as part of the references.

- Compound expressions that include a Boolean keyword need to include blank spaces around the keyword.  
*ConditionA OR ConditionB*
- Compound expressions that include an operator character that evaluates to a Boolean condition do not require blank spaces.  
*ConditionA | ConditionB*
- A property attribute needs to be referenced by its hierarchical Choice Group name.  
*ChoiceGroup.Choice*
- Rules cannot reference a Choice Group alone. Rule expressions need to reference Choices within a Choice Group.  
*ChoiceGroup.Choice*.
- When an Option is renamed, the name change is updated in all referenced rule expressions.
- An Option or a Choice can be deleted only if no graphics are associated with their default layers.

## Related Topics

[Understanding the Symbol Wizard Editor](#)

[Examples of Graphic Configuration Rules](#)

# Examples of Graphic Configuration Rules

The following examples explain how rules specify the layers that belong to a Symbol Wizard configuration.

- Orientation.Left&HasTach.True

When this rule is True, the Symbol Wizard's configuration includes a layer containing a pair of pipes with the inlet pipe oriented to the left and a tachometer.

- Orientation.Left AND HasTach.True

This rule is the same as the preceding rule except that a Boolean keyword is used rather than an operator character. There are blank spaces before and after the Boolean keyword in the rule.

- Orientation.Right&HasTach.False

When this rule is True, the Symbol Wizard's configuration includes a layer containing a pair of pipes with the inlet pipe oriented to the right and a layer that does not include a tachometer.

- (Orientation.Top&HasTach.True)|(Orientation.Bottom&HasTach.True)

When this rule is True, the Symbol Wizard's configuration includes two layers containing pipes with inlet pipe oriented at the top or the bottom. Both pipe layers include a tachometer. The selected option of the Orientation Wizard Option determines which pipe layer appears in the configuration.

For more practical examples of creating rules, see [Symbol Wizard Tips and Examples](#).

## Related Topics

[Defining Graphic Configuration Rules](#)

## Designing a Symbol Wizard

The process of creating and implementing a Symbol Wizard has two workflows:

- The first workflow, referred to as a designer workflow, uses the Symbol Wizard Editor to create Symbol Wizards containing multiple configurations.
- The second workflow, referred to as a consumer workflow, embeds a Symbol Wizard and then configures it for use in an application.

## Related Topics

[Working with Symbol Wizards](#)

[Creating Graphic Choice Groups, Choices, and Options](#)

[Assigning Graphic Configuration Rules](#)

[Updating Graphic Layers](#)

[Associating Configuration Elements to Graphic Layers](#)

[Verifying Graphic Configurations](#)

## Creating Graphic Choice Groups, Choices, and Options

The following list summarizes the tasks that need to be completed in a designer workflow to create a Symbol Wizard containing multiple configurations.

- Define a graphic's Choice Groups, their Choices, and Options
- Assign rules to Choice Groups, Choices, and Options
- Associate graphic elements, custom properties, and named scripts to graphic layers
- Verify each graphic configuration with Symbol Wizard Preview

After planning the possible configurations for a graphic, Designers should know the properties and the possible attributes associated with each configuration. Designers create Choice Groups, Choices, and Options to define a graphic's properties and attributes.

**Important:** Situational Awareness Library symbols have predefined Choice Groups, Choices, and Options.

### To create graphic choice groups, choices, and options

1. In Plant SCADA Studio, create a copy of a graphic in the Industrial Graphic Editor for which you want to create multiple configurations.  
You can also build an entirely unique symbol from scratch and create multiple configurations of it with Symbol Wizard.
2. Check out and open the copied graphic in the Industrial Graphic Editor's canvas drawing area.
3. Click the Symbol Wizard icon shown on the Industrial Graphic Editor menu bar.  
You can also show Symbol Wizard by pressing Alt+W or selecting it as an option from the **View** menu.  
The Industrial Graphic Editor updates to show the Symbol Wizard Editor's tabbed panes at the left of the window.
4. Click the **Options** tab.
5. Click **Add Choice Group** to create a Choice Group.  
A Choice Group folder appears in the **Options** window.
6. Rename the Choice Group to assign an easily identifiable name of a property used in a graphic configuration.  
Creating a Choice Group automatically sets it to rename mode. You can also manually rename a Choice Group by right-clicking on the Choice Group and select **Rename** from the menu.
7. Repeat steps 5-6 to create as many Choice Groups as needed to define all properties of a graphics that determine its configurations.
8. Select a Choice Group folder and click **Add Choice** to add a choice beneath the select Choice Group.
9. Rename the Choice to assign an easily identifiable name of a property attribute used in a graphic configuration.
10. Repeat steps 8-9 to assign all possible Choice attributes to the Choice Groups.
11. Click **Add Option** to add an Option, which appears in the window at the same hierarchical level as Choice Groups.
12. Right-click the Option and select **Rename** to assign a name.
13. Repeat steps 11-12 to create as many Options needed to define a graphic's configurations.

## Related Topics

[Designing a Symbol Wizard](#)

### Assigning Graphic Configuration Rules

In a designer workflow, you can specify rules for a graphic's defined Choices and Options. Choice Groups should not be included in graphic configuration rules.

These rules determine the graphic elements, custom properties, and scripts that belong to a graphic configuration. For more information about rule syntax, see [Defining Graphic Configuration Rules](#).

#### To define graphic configuration rules

1. Show the selected graphic in the Industrial Graphic Editor with the Symbol Wizard enabled.
2. Select a Choice from the **Options** view.

The **Properties** view updates to show **Option Properties** fields. The **Name** field shows the name of the Choice you selected from the **Options** view. The **Rule** field is blank.

3. If necessary, enter a rule for the Choice.

**Important:** Not all Choices require rules. Specify only those rules necessary to create graphic configurations. Choices without rules are always visible.

4. Repeat steps 2-3 to specify rules for the remaining Choices of the graphic.

5. Select an Option from the **Options** view.

The **Name** field of the **Option Properties** view updates to show the name of the Option you selected from the **Options** view.

6. Enter a rule for the Option that defines the conditions to show or hide the Choice Groups and Choices in a configuration.

7. Enter True or False in the **Default Value** field to set the Option as part of the graphic's default configuration or not.

8. In the **Description** field, enter a description of the Option.

The description appears when the Consumer embeds the graphic and clicks on the option to configure it.

9. Repeat steps 5-8 to specify rules and optional default values for the remaining Options of the graphic.

## Related Topics

[Designing a Symbol Wizard](#)

### Updating Graphic Layers

Symbol Wizard automatically creates a set of default layers that match the hierarchical set of Choices and Options defined for a graphic. Each Choice layer has an assigned default rule containing the expression `ChoiceGroup.Choice` that defines an attribute of a graphic's property.

The default rule for an Option layer is simply the name of the Option itself. Renaming an Option automatically renames any layer rules that reference the Option.

In a designer workflow, you can update layers by adding layers to or deleting layers from the set of default layers.

Also, layers can be renamed and the default rule assigned to a layer can be changed.

**Important:** Updating graphic layers may not be necessary if the default set of layers created for Choices and Options can create all graphic configurations.

If a graphic layer is renamed, it loses the link to the Option. When the Option name is updated, the layer name will not get updated with changed Option name.

### To add or delete a graphic layer

1. Show the selected graphic in the Industrial Graphic Editor with the Symbol Wizard selected.
2. Click the **Layers** tab to show the list of layers.
3. To add a layer, do the following:



- a. Click the **Add Layer** icon above the **Layers** list.

You can also add a layer by right-clicking within the layers list to show the action menu and selecting **Add**.

The new layer appears at the bottom of the list with an assigned default name.

- b. Click on the new layer to select it.
- c. Rename the new layer.

Creating a layer automatically sets it to rename mode. You can also manually rename a layer by right-clicking on the layer and select **Rename** from the menu.

4. To delete a layer, do the following:



- a. Click on the layer within the list to be deleted.
- b. Delete the layer by clicking the **Delete Layer** icon above the **Layers** list or right clicking to show the context menu and selecting **Delete**.

### To update a layer rule

1. Show the selected graphic in the Industrial Graphic Editor with the Symbol Wizard selected.
2. Click the **Layers** tab to show the list of layers.
3. Select a layer from the list whose rule needs to be updated.  
The **Layer Properties** view appears and shows the current rule assigned to the selected layer Choice or Option.
4. Click within the **Rule** field to select it.
5. Update the rule.
6. Click **Save** to save the changes to the layer rule.

## Related Topics

[Designing a Symbol Wizard](#)

## Associating Configuration Elements to Graphic Layers

The basic workflow to associate graphic elements, custom properties, or named scripts to a graphic layer consists of these general steps:

1. Select a graphic layer from the **Layers** view.
2. Select items from the tabbed **Elements**, **Named Scripts**, and **Custom Properties** views to associate with the selected layer.  
**Note:** Multiple graphic elements, custom properties, or named scripts can be selected using the Shift key to select a range of listed items or the Ctrl key to select individual items from a list.
3. Drag and drop the selected graphic elements, custom properties, or scripts into the **Layers** view.

Configuration elements can be associated with a graphic layer by two methods:

- **Active layer method:** Select the check box to the left of the layer name. Then, drag and drop the configuration element anywhere within the **Layers** view. The configuration element is automatically associated to the correct folder of the active layer.
- **Direct folder method:** Select a layer and expand it to show the folders for the different types of configuration elements. Then, drag and drop the configuration element directly on the folder that matches the type of configuration element.

## Related Topics

[Designing a Symbol Wizard](#)

[Associating Graphic Elements to Graphic Layers](#)

[Using Shortcut Menu Commands to Edit Graphic Layer Graphic Elements](#)

[Associating Custom Properties to Graphic Layers](#)

[Associating Named Scripts to Graphic Layers](#)

# Associating Graphic Elements to Graphic Layers

Graphic elements show the visual properties of a graphic. In a designer workflow, associate graphic elements to the defined layers of a graphic.

## To associate graphic elements to graphic layers

1. Show the symbol with the Symbol Wizard Editor selected.
2. Click the **Elements** tab to show the graphic elements that belong to the graphic.
3. Click the **Layers** tab.
4. Activate a layer from the **Layers** view by selecting the check box next to the layer.  
If you prefer to add graphic elements directly to a layer's **Graphic Elements** folder with the direct folder method, simply click the layer name from the list to select it.
5. Click the box to the left of the check box to expand the layer view and show the **Graphic Elements** folder.
6. Click on the graphic element in the **Elements** view to be associated with the active graphic layer.

You can also select the symbol element group by clicking it on the displayed graphic.

7. Using standard Windows drag and drop technique, drag the graphic element from the **Elements** view and drop it anywhere within the **Layers** view.

If you are using the direct folder method, you must drop the graphic element directly on the selected layer's **Graphic Elements** folder.

The selected element appears beneath the active layer's **Graphic Elements** folder.

8. Repeat steps 6-7 to select all element groups that belong to the graphic layer.

You can also select multiple graphic elements from the **Elements** view and drop them as a set.

9. Repeat steps 4-8 to select all elements for the different layers of a graphic.

The **Show/Hide** icon appears to the left of the **Graphic Elements** folder in the **Layers** view. Clicking the icon shows or hides the graphic elements in a layer's **Graphic Elements** folder on the graphic itself.

10. Click the **Show/Hide** icon to verify the graphic elements associated to a layer are correct for the graphic configuration.

11. Save your changes to the graphic.

## Related Topics

[Associating Configuration Elements to Graphic Layers](#)

# Using Shortcut Menu Commands to Edit Graphic Layer Graphic Elements

The Symbol Wizard Editor provides a set of shortcut menu commands to add graphic elements to graphic layers or remove them from layers. Using shortcut commands makes it easier to add or remove graphic elements when a complex Symbol Wizard contains many graphic elements and layers.

### Adding Graphic Elements to Active Symbol Layers

Adding graphic elements to an active layer involves selecting an active layer, selecting one or more graphic elements, and then using the **Add To Active Layers** shortcut command.

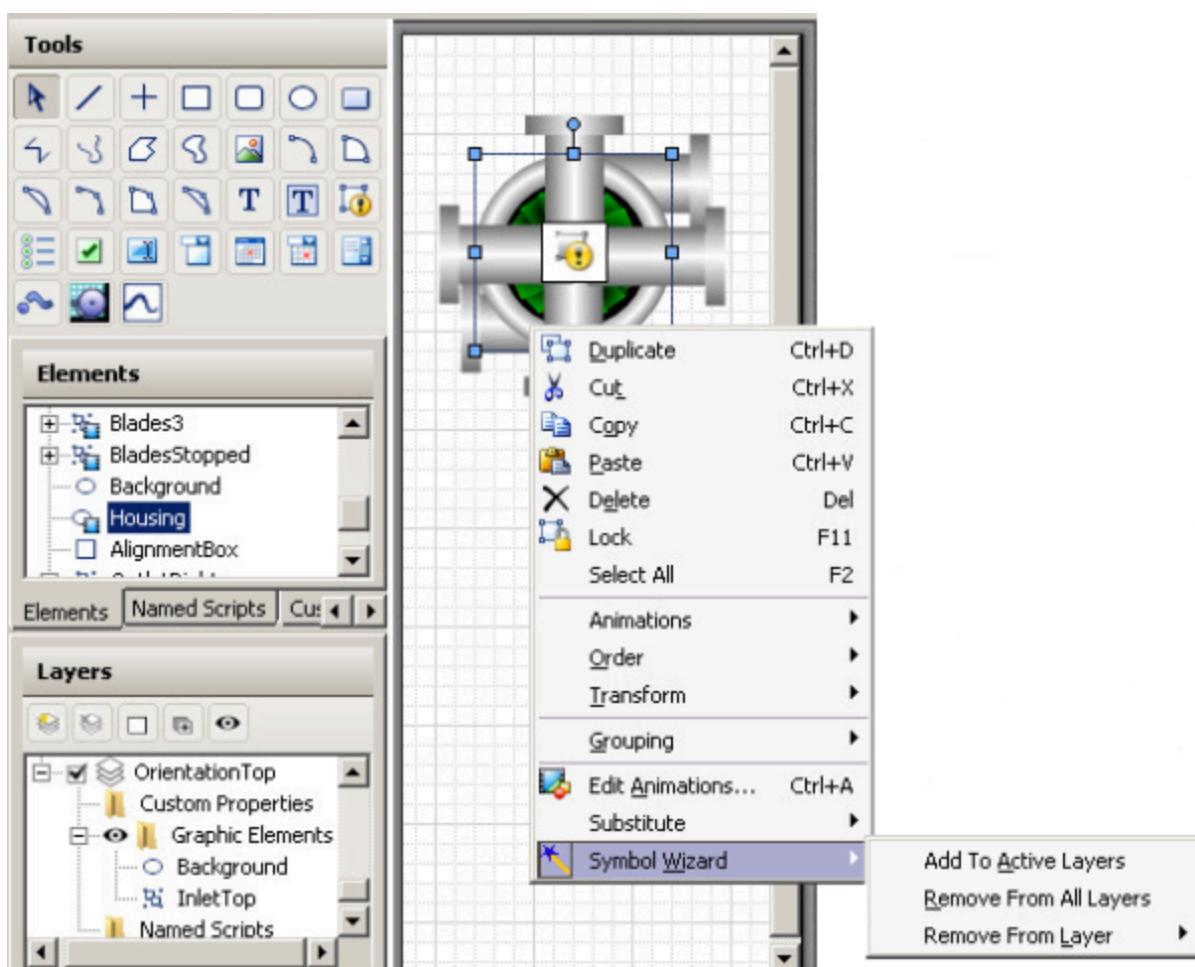
---

**Note:** All graphic elements should be created for all Symbol Wizard configurations before adding them to graphic layers.

---

### To add graphic elements to graphic layers

1. Show the graphic with Symbol Wizard Editor selected.
2. From the **Layers** pane, select the check box next to graphic layer to make it active.  
If you want to add a graphic element to multiple layers, select the check box next to each layer to make them active.
3. Select the graphic element to be added from the displayed Symbol Wizard.  
The graphic element can also be selected from the **Elements** pane.
4. Show the Symbol Wizard shortcut commands by right-clicking on the selected graphic element on the graphic or right-clicking on the graphic element name from the **Elements** pane.



5. Click **Add to Active Layers**.
6. Verify the graphic element has been added to the active layers.

### Removing Graphic Elements from Graphic Layers

Removing graphic elements from graphic layers follows a similar sequence of steps as adding graphic elements to layers. The Symbol Wizard shortcut menu includes separate commands to remove graphic elements from all layers or only from a selected layer.

#### To remove graphic elements from graphic layers

1. Show the graphic with Symbol Wizard Editor selected.
2. From the **Layers** pane, select the check box next to the graphic layer that contains a graphic element to be removed.

Selecting a layer is not necessary if the graphic element will be removed from all layers. Also, if a layer is not selected, the **Remove From Layer** command shows a list of layers that include the selected graphic element to be removed.

3. Select the graphic element to be added from the displayed Symbol Wizard.

The graphic element can also be selected from the **Elements** pane.

4. Show the Symbol Wizard shortcut commands by right-clicking on the selected graphic element on the graphic or right-clicking on the graphic element name from the **Elements** pane.

5. Click **Remove From All Layers** or **Remove From Layer** based on whether the graphic element needs to be removed from all layers or only the selected layer.

If a layer has not been selected, the **Remove From Layer** command shows a list of layers that include the graphic element selected to be removed. Click a layer from the list to remove a graphic element.



6. Verify the graphic element has been removed from the selected layers.

## Related Topics

[Associating Configuration Elements to Graphic Layers](#)

# Associating Custom Properties to Graphic Layers

Associating custom properties to a graphic layer uses a procedure similar to associating graphic elements. Selected custom properties are dragged and dropped on the **Custom Properties** folder to associate them to a graphic layer. You can associate custom properties to layers with the active layer or directory folder methods.

### To associate custom properties to graphic layers

1. Open the selected graphic in the Industrial Graphic Editor with the Symbol Wizard selected.
2. Click the **Custom Properties** grid to show the locally defined custom properties of the graphic.  
Custom properties of embedded graphics are not listed.
3. Click the **Layers** tab.
4. Select a layer from the **Layers** view to add custom properties by selecting the check box next to the layer.
5. Click the box to the left of the check box to expand the layer view and show the **Custom Properties** folder.
6. Click on a custom property in the **Custom Properties** view that belongs to the selected graphic layer.
7. Using standard Windows drag and drop technique, drag the custom property from the **Custom Properties** view and drop it on the **Custom Properties** folder.  
The selected custom property appears beneath the **Custom Properties** folder.
8. Repeat steps 6-7 to select all custom properties that belong to the graphic layer.
9. Repeat steps 4-7 to select the remaining custom properties for the different layers of a graphic.
10. Save your changes to the graphic.

## Related Topics

[Associating Configuration Elements to Graphic Layers](#)

# Associating Named Scripts to Graphic Layers

Associating named scripts to a graphic layer uses a similar procedure to associate graphic elements or custom properties. You can associate named scripts to layers with the active layer or directory folder methods.

## To associate named scripts to graphic layers

1. Show the selected graphic in the Industrial Graphic Editor with the Symbol Wizard selected.
2. Click the **Named Scripts** tab to show the scripts associated with the graphic.
3. Click the **Layers** tab.
4. Select a layer from the **Layers** view by selecting the check box next to the layer.
5. Click the box to the left of the check box to expand the layer view and show the **Named Scripts** folder.
6. Click on a script in the **Named Scripts** view that belongs to the selected graphic layer.
7. Using standard Windows drag and drop technique, drag the script from the **Named Scripts** view and drop it on the **Named Scripts** folder.  
The selected script appears beneath the **Named Scripts** folder.
8. Repeat steps 6-7 to select all scripts that belong to the graphic layer.
9. Repeat steps 4-7 to select the remaining scripts for the different layers of a graphic.
10. Save your changes to the graphic.

## Related Topics

[Associating Configuration Elements to Graphic Layers](#)

## Verifying Graphic Configurations

After creating the different configurations of a graphic, you can use the Symbol Wizard Preview to verify each configuration works as designed. Also, you can validate the graphic to identify any invalid references to other objects or values.

## To verify graphic configurations

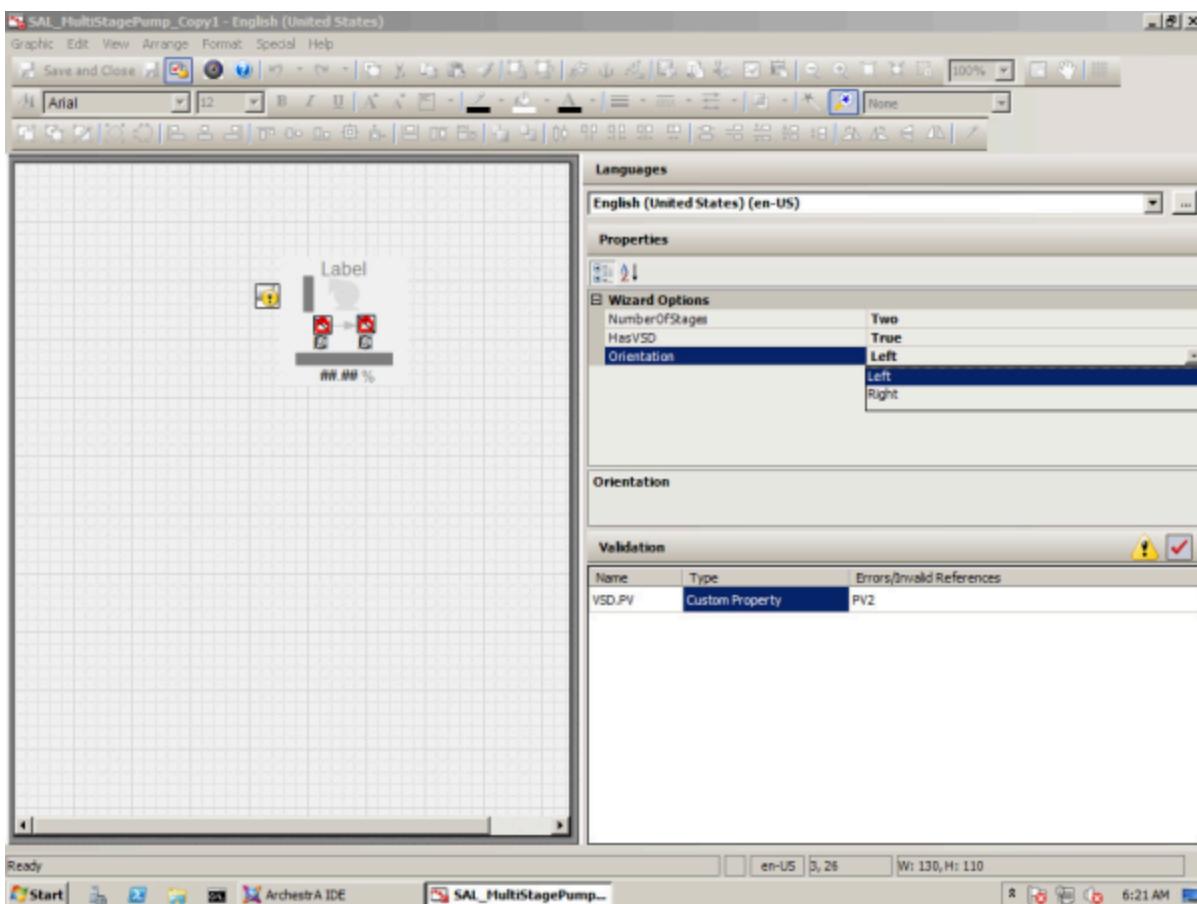
1. Open the graphic created with Symbol Wizard in the Industrial Graphic Editor.



2. Click **Symbol Wizard Preview** shown on the menu bar of the Industrial Graphic Editor.

You can also open the Symbol Wizard Preview as a **View** menu option or by pressing Alt+P.

The Industrial Graphic Editor updates to show the **Wizard Options** view with a set of drop-down lists to select different graphic property attributes and options. Select the default graphic configuration.



3. Select the different combinations of property values and view options from **Wizard Options** fields.
4. Verify the graphic that appears is correct for the specified configuration Choices and Option rule.



5. Click the **Validation** icon to see if the graphic contains any invalid references.

The **Validation** view lists any invalid references within the graphic that need to be corrected.

Invalid references also include references to properties or elements in hidden graphic layers.

## Related Topics

[Designing a Symbol Wizard](#)

## Using Symbol Wizards in an Application

Symbol Wizards are stored in a library just like standard Industrial Graphics. When you select a graphic and embed it into an HMI/SCADA application, the graphic's default configuration is selected.

In a consumer workflow, you can change a Symbol Wizard's configuration by changing the values assigned to the graphic's properties from the Symbol Wizard's **Wizard Options** section of the **Properties** view. After selecting a graphic configuration and changing any properties, save the graphic.

The Symbol Wizard appears as the configuration you have selected. A Symbol Wizard's configuration cannot be changed during application run time.

## Related Topics

[Working with Symbol Wizards](#)

[Embedding Symbol Wizards](#)

### Embedding Symbol Wizards

In a consumer workflow, you embed Symbol Wizards from the Industrial Graphic Editor. Embedding a Symbol Wizard is similar to embedding a standard Industrial Graphic.

A Symbol Wizard appears with its default configuration when it is embedded. The Consumer can select another configuration by changing the configuration values shown in the **Wizard Options** section of the **Properties** view.

#### To embed a graphic

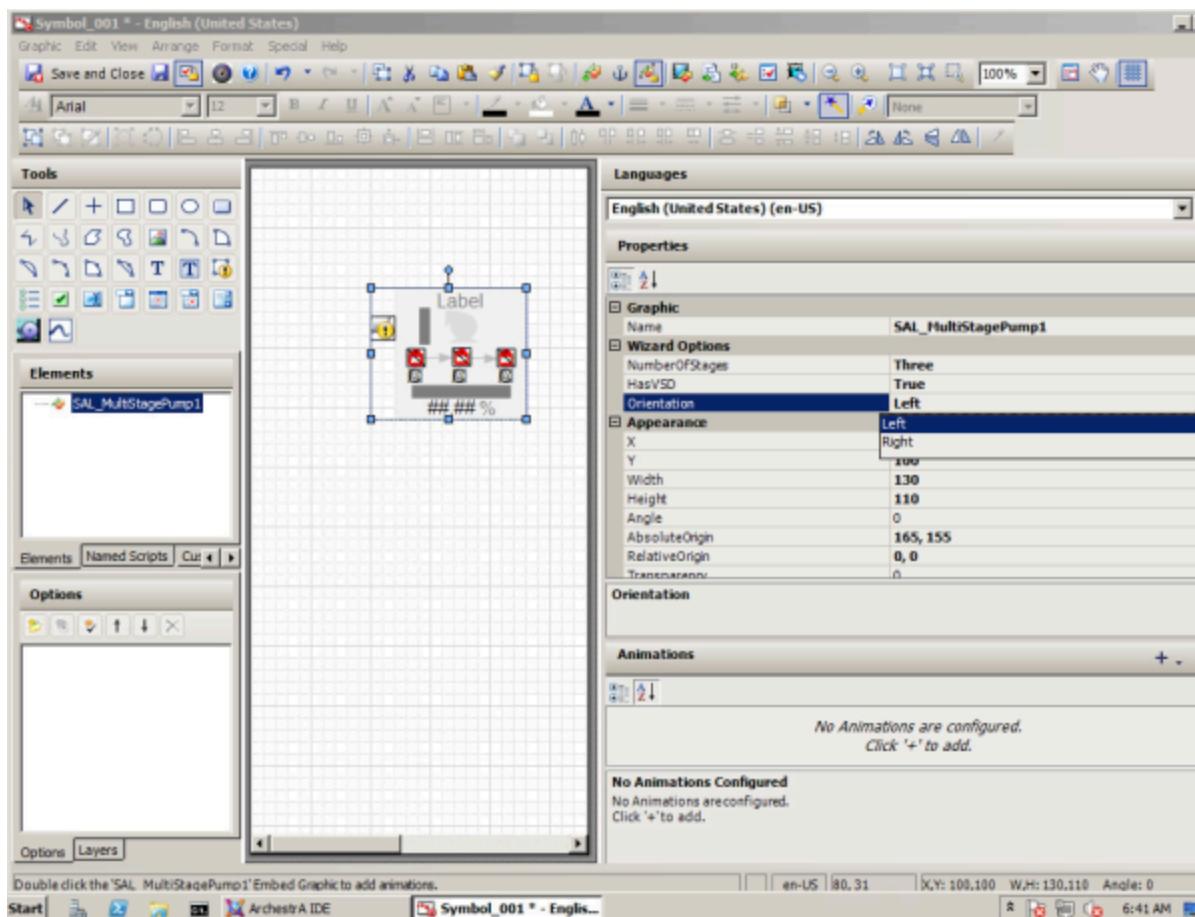
1. Create a new graphic or add a graphic to your HMI application.
2. Open the symbol to show the Industrial Graphic Editor.
3. On the **Edit** menu, click **Embed Graphic**.



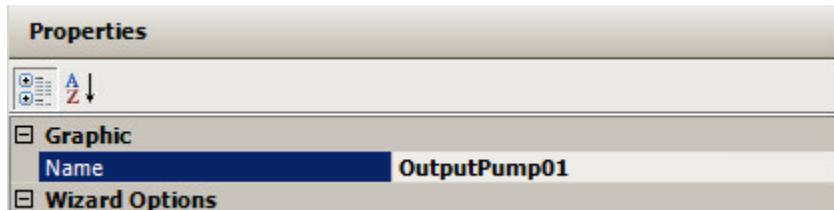
You can also click the **Embed Graphic** icon from the menu bar.

Your HMI's graphics browser appears.

4. Locate the folder containing the Symbol Wizard.
  5. Click the graphic to select it and click **OK**.
  6. Position the pointer at the location where the Symbol Wizard should be placed.
  7. Click once to embed the Symbol Wizard.
- An embedded Symbol Wizard appears with handles on the Industrial Graphic Editor canvas to show that it is selected.
8. Select the graphic's configuration by selecting values for the various options shown in the **Wizard Options** view.



- Rename the graphic.



- Right-click on the graphic and select **Custom Properties** from the menu.  
The **Edit Custom Properties** dialog box appears with the set of custom properties defined for the Symbol Wizard.
- Configure the custom properties with the required references for the application.
- Press [F10] to show the **Edit Scripts** dialog box.
- Verify if any changes need to be made to the graphic's named scripts to run within the application.
- Save the changes made to the graphic.

## Related Topics

[Using Symbol Wizards in an Application](#)

## Symbol Wizard Tips and Examples

This section describes a practical example of creating a Symbol Wizard. The example explains how to modify an centrifugal pump graphic to create a Symbol Wizard with Wizard Options that represent different orientations of inlet and outlet pumps for a tank farm application. The Symbol Wizard also includes a Wizard Option to show or hide a pump tachometer.

## Related Topics

[Working with Symbol Wizards](#)

[Creating Visual Configurations of an Industrial Graphic](#)

### Creating Visual Configurations of an Industrial Graphic

Complete the following tasks to create a Symbol Wizard:

- Plan the different configurations of a graphic and select a default configuration that represents the base Symbol Wizard.
- Identify the graphic elements needed to create each graphic configuration.
- Add graphic elements, named scripts, and custom properties for each configuration.
- Create graphic layers to group graphic elements, named scripts, and custom properties
- Specify rules to select the layers needed to create each Symbol Wizard configuration

## Related Topics

[Symbol Wizard Tips and Examples](#)

[Planning Symbol Wizard Configurations](#)

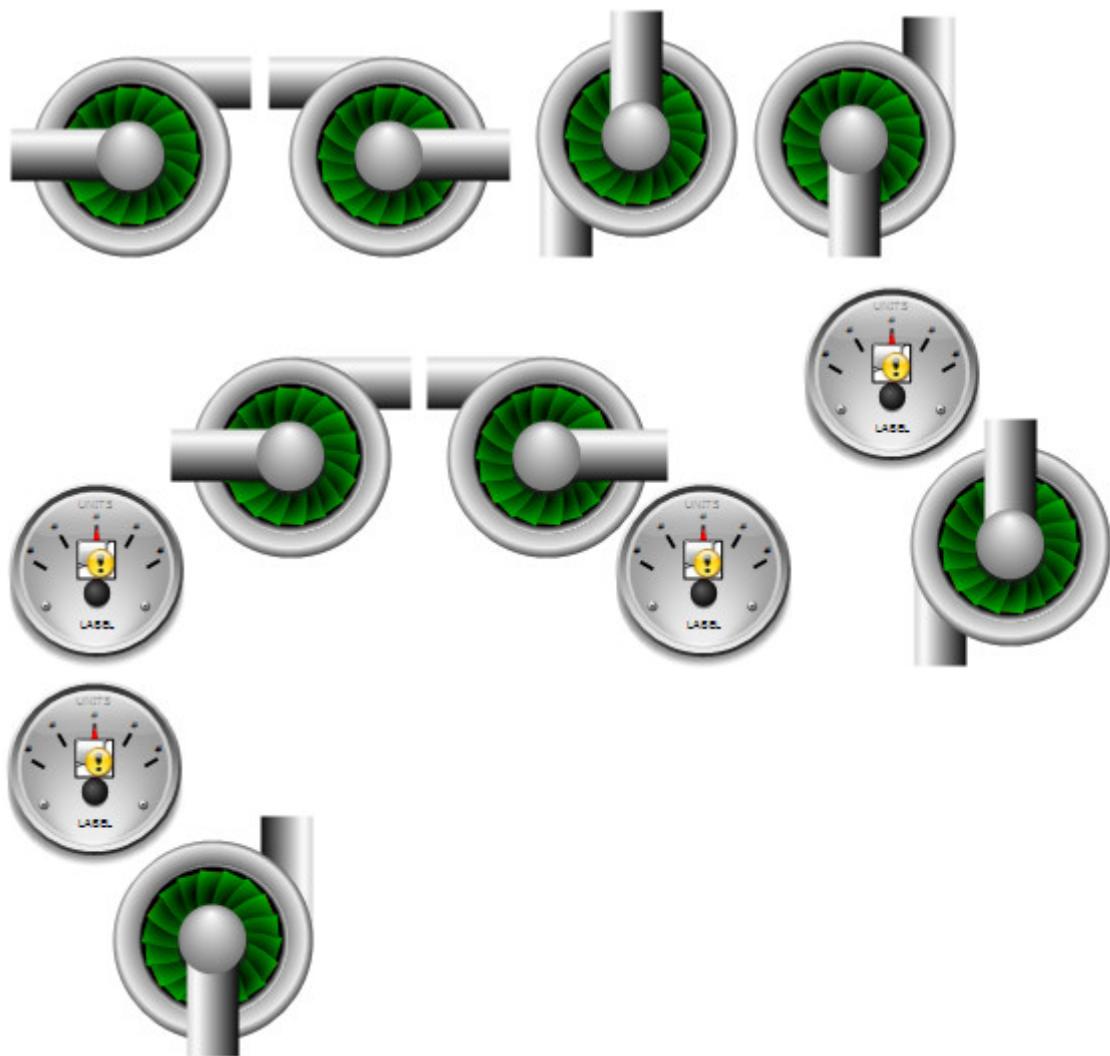
[Identify Graphic Elements](#)

[Build a Visual Representation of a Symbol Wizard](#)

[Specify Rules to select Graphic Layers](#)

# Planning Symbol Wizard Configurations

The first step in creating a Symbol Wizard is to identify the different configurations that should be included in a symbol. In the example of a centrifugal pump, a Symbol Wizard should represent a pump that has the inlet pipe at the left, right, top, and bottom of the pump's central housing. Also, the Symbol Wizard should be able to show or hide a tachometer for each orientation of the centrifugal pump.



After identifying all of the different configurations of a Symbol Wizard, identify the unique properties of each configuration. The example of a centrifugal pump includes two properties: inlet pipe orientation and whether a tachometer is shown with a pump or not.

The next step is to identify the properties and associated equipment.items for each configuration of the symbol.

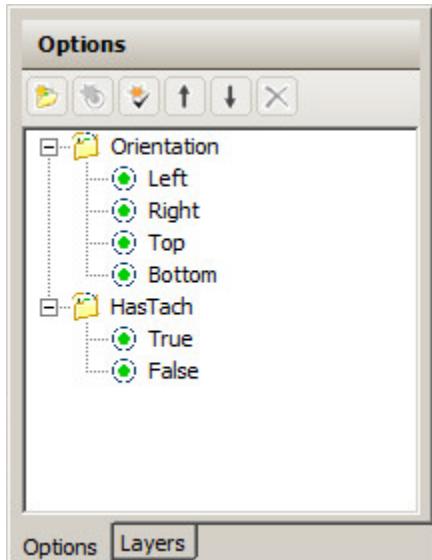
Symbol Configuration	Configuration Properties
	Orientation=Left HasTach=False

	Orientation=Left HasTach=True
	Orientation=Right HasTach=False
	Orientation=Right HasTach=True
	Orientation=Top HasTach=False
	Orientation=Top HasTach=True

	Orientation=Bottom HasTach=False
	Orientation=Bottom HasTach=True

Using the Symbol Wizard Editor, create the Choice Groups and Choices needed to represent all properties and equipment.items of a Symbol Wizard.

In the example of a centrifugal pump, the Choice Groups are Orientation and HasTach. The Orientation Choice Group includes Left, Right, Top, and Bottom Choices, which are the possible equipment.items of a pump's inlet pipe. The HasTach Choice Group includes Boolean True or False Choices that indicate whether a configuration includes a tachometer or not.



Initially, the top listed Choice is the default for a Choice Group. To assign another listed Choice as the default value for the Choice Group, assign the Choice in **Default Value** field of the **Option Properties** pane.

Option Properties	
Name	Orientation
Description	
Rule	
DefaultValue	Right

If the desired base configuration of a centrifugal pump has pipes oriented to the right and includes a tachometer, then Right should be assigned as the default Orientation Choice and True assigned as the default HasTach Choice.

#### Planning Tips

- Always decide the different configurations that should be incorporated into a Symbol Wizard as the first step.
- Identify those properties that define a symbol's configurations. These properties will be specified as the Choice Groups when building configurations with the Symbol Wizard Editor.

Identify all equipment.items of each property that define a symbol's configurations. These equipment.items will be the child Choices of the parent Choice Groups.

Select a default Symbol Wizard configuration at the planning stage to identify the symbol elements, named scripts, and custom properties that you want to include in the base configuration.

#### Related Topics

[Creating Visual Configurations of an Industrial Graphic](#)

## Identify Graphic Elements

Graphic elements are the graphics, named scripts, custom properties, and animations included with each configuration of a Symbol Wizard. The first step is to identify the graphic elements that need to be created for each Symbol Wizard configuration.

The following table shows the graphic elements needed to create a Symbol Wizard of a centrifugal pump.

Graphic Configuration	Configuration Properties	Required Graphic Elements
	Orientation=Left HasTach=False	Graphic elements: <ul style="list-style-type: none"><li>InletLeft</li><li>OutletRight</li></ul>
	Orientation=Left HasTach=True	Graphic elements: <ul style="list-style-type: none"><li>InletLeft</li><li>OutletRight</li><li>MeterLeft</li></ul>
	Orientation=Right HasTach=False	Graphic elements: <ul style="list-style-type: none"><li>InletRight</li><li>OutleftLeft</li></ul>
	Orientation=Right HasTach=True	Graphic elements: <ul style="list-style-type: none"><li>InletRight</li><li>OutleftLeft</li><li>MeterRight</li></ul>
	Orientation=Top HasTach=False	Graphic elements: <ul style="list-style-type: none"><li>InletTop</li><li>OutletBottom</li></ul>

Graphic Configuration	Configuration Properties	Required Graphic Elements
	Orientation=Top HasTach=True	Graphic elements: <ul style="list-style-type: none"> <li>• InletTop</li> <li>• OutletBottom</li> <li>• MeterTop</li> </ul>
	Orientation=Bottom HasTach=False	Graphic elements: <ul style="list-style-type: none"> <li>• InletBottom</li> <li>• OutletTop</li> </ul>
	Orientation=Bottom HasTach=True	Graphic elements: <ul style="list-style-type: none"> <li>• InletBottom</li> <li>• OutletTop</li> <li>• MeterTop</li> </ul>

### Identification Tips

- Assign short descriptive names to graphic elements. Default names are created for layers by concatenating Choice Group and Choice names. Shorter names reduce the number of Option and Layer rules that will extend beyond the borders of **Rule** field of the Symbol Wizard Editor.
- Use a standard naming convention for the graphic elements of a Symbol Wizard. Using a standard naming convention groups similar functional graphic elements together in the list shown in the **Elements** pane. This makes it easier to find graphic elements when a Symbol Wizard contains many graphic elements. Also, the names of layers appear in alphabetic order.

### Related Topics

[Creating Visual Configurations of an Industrial Graphic](#)

# Build a Visual Representation of a Symbol Wizard

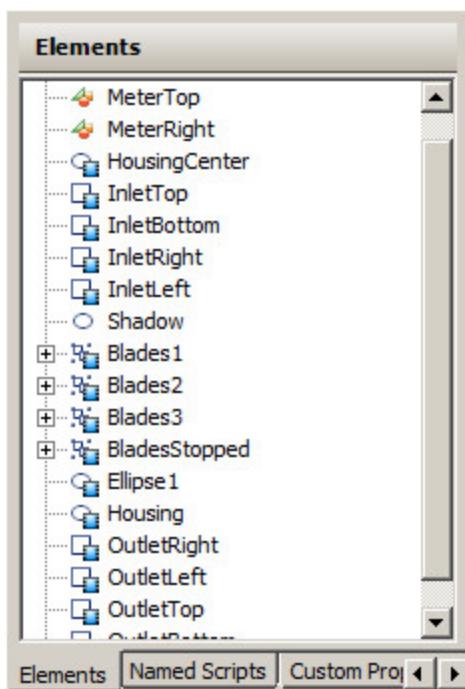
The major steps to add the necessary graphic elements to a Symbol Wizard consist of the following:

1. Check out and open an instance of a graphic from the Industrial Graphic Editor in the Industrial Graphic Editor, or create a new graphic.
2. Create the graphic elements required for each Symbol Wizard configuration by doing one of the following:
  - Embed other graphics from the Industrial Graphic Editor into the symbol.
  - Duplicate graphic elements from the symbol and edit them as necessary for each Symbol Wizard configuration.



In the example of the centrifugal pump Symbol Wizard, a meter has been embedded into centrifugal pump graphic, and then duplicated for the different configuration positions. The inlet and outlet pipes are created by duplicating the graphic pipe elements from the centrifugal pump graphic.

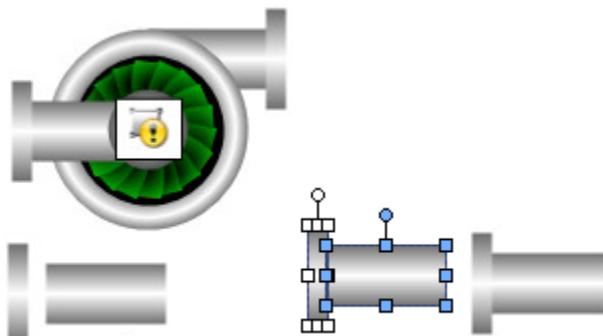
3. Rename each graphic element to easily associate it with a Symbol Wizard configuration.



4. Position the graphic elements to accurately represent the different visual representations of each configuration of a Symbol Wizard.

#### Visualization Tips

- If the same graphic element will be placed at different positions within a Symbol Wizard based on different configurations, create a copy of the graphic element for each position. Each graphic element can be placed into a separate layer, making it easier to specify rules to show the element at the desired position.
- If a graphic element included in a specific Symbol Wizard configuration consists of two or more elements, group the elements together. Grouping related elements makes it easier to assign graphics to Symbol Wizard layers.

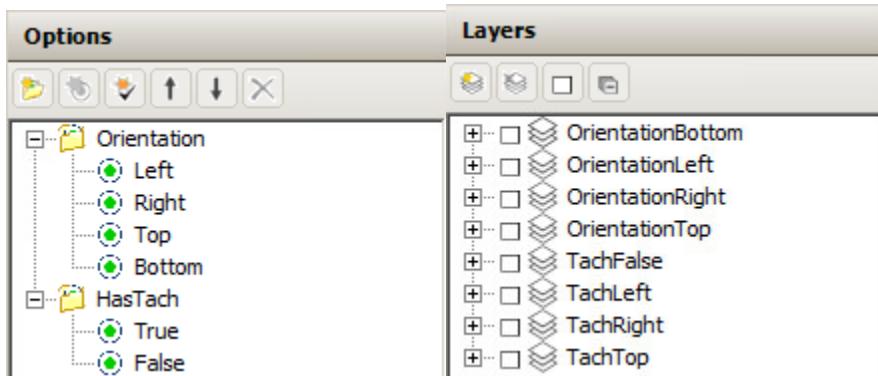


#### Related Topics

[Creating Visual Configurations of an Industrial Graphic](#)

# Assign Graphic Elements, Named Scripts, and Custom Properties to Graphic Layers

By default, the **Layers** pane shows a layer for each Choice Group/Choice combination listed in the **Options** pane. In the example of a centrifugal pump Symbol Wizard, there are unique layers for each orientation of the inlet pipe of the pump graphic. Inlet and outlet graphic element pairs are added to each Orientation layer.



Layers need to be added for the left, right, and top positions of the tachometer when the HasTach Choice Group is True. Copies of the embedded tachometer graphic are added to the TachLeft, TachRight, and TachTop layers, which map to the different positions of the tachometer shown in the pump Symbol Wizard. The TachFalse layer does not contain any graphic elements because it selects the Symbol Wizard configurations without a tachometer.

## Layer Tips

- Use the Active Layer method to quickly drag and drop elements, scripts, and custom properties to a layer folder. Selected elements can be dropped anywhere within the **Layers** view and automatically placed in the correct folder of the active layer.
- If not created by default, create an empty layer without graphic elements for a Choice Group with Boolean True/False Choices. This makes it easier to write layer rules to hide graphic elements when a Choice Group is False.
- After adding graphic elements to a layer, toggle the **Show/Hide** icon on and off to verify the correct graphic elements have been added to the layer.
- Toggle the **Expand All/Collapse All** button above the **Layers** pane to show or hide all of the folders beneath each layer.

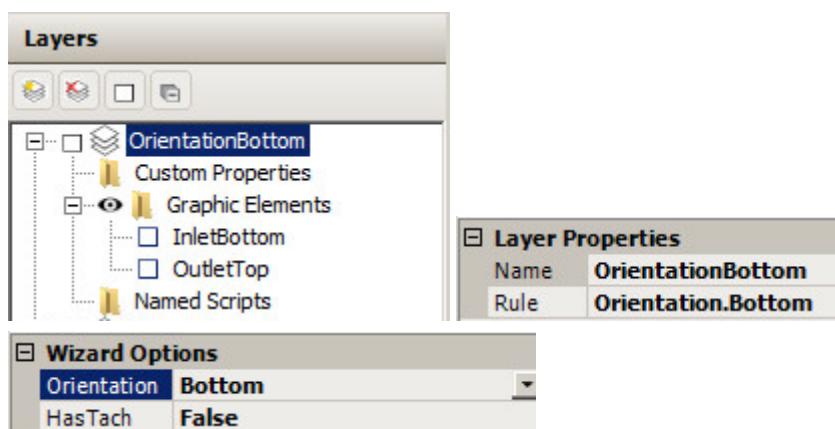
## Related Topics

[Creating Visual Configurations of an Industrial Graphic](#)

# Specify Rules to select Graphic Layers

A default rule is assigned to each layer based on the Choice Group Choice pair. In the example of the centrifugal pump Symbol Wizard, the Orientation Choice Group layer rules map directly to **Wizard Options** choices.

Selecting an Orientation option displays the graphic elements of the selected layer in the pump's configuration.



Modify the default layer rules to show or hide the tachometer. In the case of the Left or Right pump orientation, we recommend that the layer rules select the appropriate Orientation layer and tachometer layers. The TachLeft and TachRight layer rules include an AND statement that selects the tachometer and the Symbol Wizard's pipe orientation:

TachLeft: Orientation.Left&HasTach.True

TachRight: Orientation.Right&HasTach.True

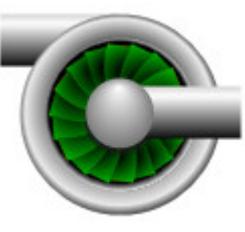
The layer rule to select the Top and Bottom Orientation configurations with a tachometer is more complex because the position of the tachometer is the same in both orientations. The TachTop layer rule includes separate Top and Bottom compound expressions joined with an OR statement.

(Orientation.Top&HasTach.True)|(Orientation.Bottom&HasTach.True)

Using Symbol Wizard Preview, verify each set of layer rules defines only a single unique Symbol Wizard configuration. Rule errors become apparent when a Symbol Wizard includes elements from other configurations, or elements are missing.

Symbol Wizard Configuration	Wizard Options and Corresponding Active Configuration Layer Rules			
	<table border="1"> <tr> <td data-bbox="869 1311 1400 1347"><b>Wizard Options</b></td> </tr> <tr> <td data-bbox="869 1347 1400 1383">Orientation: <b>Left</b></td> </tr> <tr> <td data-bbox="869 1383 1400 1419">HasTach: <b>False</b></td> </tr> </table>	<b>Wizard Options</b>	Orientation: <b>Left</b>	HasTach: <b>False</b>
<b>Wizard Options</b>				
Orientation: <b>Left</b>				
HasTach: <b>False</b>				

Orientation.Left&HasTach.False

	<table border="1"><tr><td colspan="2"><b>Wizard Options</b></td></tr><tr><td>Orientation</td><td><b>Right</b></td></tr><tr><td>HasTach</td><td><b>True</b></td></tr></table>	<b>Wizard Options</b>		Orientation	<b>Right</b>	HasTach	<b>True</b>
<b>Wizard Options</b>							
Orientation	<b>Right</b>						
HasTach	<b>True</b>						
Orientation.Left&HasTach.True							
	<table border="1"><tr><td colspan="2"><b>Wizard Options</b></td></tr><tr><td>Orientation</td><td><b>Right</b></td></tr><tr><td>HasTach</td><td><b>False</b></td></tr></table>	<b>Wizard Options</b>		Orientation	<b>Right</b>	HasTach	<b>False</b>
<b>Wizard Options</b>							
Orientation	<b>Right</b>						
HasTach	<b>False</b>						
Orientation.Right&HasTach.False							
	<table border="1"><tr><td colspan="2"><b>Wizard Options</b></td></tr><tr><td>Orientation</td><td><b>Right</b></td></tr><tr><td>HasTach</td><td><b>True</b></td></tr></table>	<b>Wizard Options</b>		Orientation	<b>Right</b>	HasTach	<b>True</b>
<b>Wizard Options</b>							
Orientation	<b>Right</b>						
HasTach	<b>True</b>						
Orientation.Right&HasTach.True							
	<table border="1"><tr><td colspan="2"><b>Wizard Options</b></td></tr><tr><td>Orientation</td><td><b>Top</b></td></tr><tr><td>HasTach</td><td><b>False</b></td></tr></table>	<b>Wizard Options</b>		Orientation	<b>Top</b>	HasTach	<b>False</b>
<b>Wizard Options</b>							
Orientation	<b>Top</b>						
HasTach	<b>False</b>						
Orientation.Top&HasTach.False							

	<table border="1"><tr><td colspan="2"><b>Wizard Options</b></td></tr><tr><td>Orientation</td><td><b>Top</b></td></tr><tr><td>HasTach</td><td><b>True</b></td></tr></table>	<b>Wizard Options</b>		Orientation	<b>Top</b>	HasTach	<b>True</b>
<b>Wizard Options</b>							
Orientation	<b>Top</b>						
HasTach	<b>True</b>						
(Orientation.Top&HasTach.True) (Orientation.Bottom&HasTach.True)							
	<table border="1"><tr><td colspan="2"><b>Wizard Options</b></td></tr><tr><td>Orientation</td><td><b>Bottom</b></td></tr><tr><td>HasTach</td><td><b>False</b></td></tr></table>	<b>Wizard Options</b>		Orientation	<b>Bottom</b>	HasTach	<b>False</b>
<b>Wizard Options</b>							
Orientation	<b>Bottom</b>						
HasTach	<b>False</b>						
Orientation.Bottom&HasTach.False							
	<table border="1"><tr><td colspan="2"><b>Wizard Options</b></td></tr><tr><td>Orientation</td><td><b>Bottom</b></td></tr><tr><td>HasTach</td><td><b>True</b></td></tr></table>	<b>Wizard Options</b>		Orientation	<b>Bottom</b>	HasTach	<b>True</b>
<b>Wizard Options</b>							
Orientation	<b>Bottom</b>						
HasTach	<b>True</b>						
(Orientation.Top&HasTach.True) (Orientation.Bottom&HasTach.True)							

### Rule Tips

- Symbol Wizard rules are evaluated simultaneously. Place parentheses around compound expressions, which are evaluated before other operators outside of parentheses in a rule.
- Rules cannot reference a Choice Group alone. Always write rule expressions that reference Choices within a Choice Group in a hierarchical manner: ChoiceGroup.Choice.
- Use operator characters (&, |, !) rather than Boolean keywords (AND, OR, and NOT) to save space when writing rules. Using operator characters reduces the likelihood that a long rule will extend beyond the borders of the **Rule** field.

## Related Topics

[Creating Visual Configurations of an Industrial Graphic](#)

## List of Element Properties

### In This Appendix

[Alphabetical List of Properties](#)

[List by Functional Area](#)

[Order of Precedence for Property Styles](#)

### Alphabetical List of Properties

This section shows you the properties of elements, the canvas, element groups, and embedded graphics.

Each property has a purpose, a category it belongs to, where it is used if it can be used in scripting at run time, and where to find more information on how to use it.

The first part of this section contains an alphabetical list of all properties, the second part shows a table for each category of properties.

The following table contains a list of properties used by the:

- Elements.
- Canvas.
- Element groups.
- Embedded graphics.

Asterisk (\*) marks properties that are specific to only one type of element or the canvas, a group or an embedded graphic.

Property	Purpose, category, usage and further information
<b>AbsoluteAnchor*</b>	Purpose: Defines the absolute anchor point of the source graphic. By default, this is the center point of all elements on the canvas but can be changed.
	Category: Appearance
	Used by: Canvas
	Can be read by script at run time: No
	Info: <a href="#">Size Propagation and Anchor Points</a>
<b>AbsoluteOrigin</b>	Purpose: Defines an X, Y location relative to the top, left (0, 0) origin of the graphic or window.
	Category: Appearance

	<p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Embedded Symbol, Group, Path, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box.</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Changing Points of Origin in the Properties Editor</a></p>
<b>Alignment</b>	<p>Purpose: Controls the location of the text relative to the bounding rectangle of the element.</p> <p>Category: Text Style</p> <p>Used by: Button, Text, Text Box</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Text Alignment</a></p>
<b>AnchorFixedTo</b>	<p>Purpose: Determines if the anchor point is fixed to the canvas when you resize, delete, or add elements (Absolute), or if the anchor point is recalculated relative to the element sizes and positions (Relative).</p> <p>Category: Appearance</p> <p>Used by: Embedded Symbol, Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Size Propagation and Anchor Points</a></p>
<b>AnchorPoint*</b>	<p>Purpose: Defines the anchor X, Y location of the embedded graphic.</p> <p>Category: Appearance</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Size Propagation and Anchor Points</a></p>
<b>Angle</b>	<p>Purpose: Defines the current angle of rotation of the element. 0 is always the top of the element relative</p>

	<p>to the canvas. Angle is always determined relative to the top of the element and rotates in a clockwise direction.</p>
	<p>Category: Appearance</p>
	<p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Group, Embedded Symbol</p>
	<p>Can be read by script at run time: Yes</p>
	<p>Info: <a href="#">Rotating Elements by Changing the Angle Property</a></p>
<b>AutoScale</b>	<p>Purpose: If this property is set to True then the text is stretched horizontally and vertically (larger or smaller) to fit the bounding rectangle.</p>
	<p>Category: Appearance</p>
	<p>Used by: Button, Text Box</p>
	<p>Can be read by script at run time: Yes</p>
	<p>Info: <a href="#">Setting Auto Scaling and Word Wrapping for a Text Box</a></p>
<b>ButtonStyle*</b>	<p>Purpose: Determines if the button appears as a standard button or as an image.</p>
	<p>Category: Appearance</p>
	<p>Used by: Button</p>
	<p>Can be read by script at run time: No</p>
	<p>Info: <a href="#">Configuring Buttons with Images</a></p>
<b>CalendarColumns*</b>	<p>Purpose: Defines the number of columns the calendar object has.</p>
	<p>Category: Appearance</p>
	<p>Used by: Calendar</p>
	<p>Can be read by script at run time: No</p>
	<p>Info: <a href="#">Setting the Number of Calendar Month Sheets</a></p>

<b>CalendarRows*</b>	<p>Purpose: Defines the number of rows the calendar object has.</p> <p>Category: Appearance</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Number of Calendar Month Sheets</a></p>
<b>Caption*</b>	<p>Purpose: Defines the text shown on the Check Box at design time and at run time when the caption property is not bound to a reference in the checkbox animation panel.</p> <p>Category: Text Style</p> <p>Used by: Check Box</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Caption Text of a Check Box Control</a></p>
<b>Checked*</b>	<p>Purpose: Sets or gets the value of check box. This is the initial value of the check box when the control is not connected to a reference and is overridden at run time with value of reference.</p> <p>Category: Appearance</p> <p>Used by: Check Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Setting the Default State of a Check Box Control</a></p>
<b>.Color1</b>	<p>Purpose: Color1 is a sub-property of a FillColor, UnfilledColor, LineColor or TextColor property. It is used to change the first color of the fill, unfill, line or text style if applicable.</p> <p>Category: Depends on its source property</p> <p>Used by: Depends on its source property</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Enabling and Disabling Elements for Run-Time Interaction</a></p>
<b>.Color2</b>	<p>Purpose: Color2 is a sub-property of a FillColor,</p>

	<p>UnfilledColor, LineColor or TextColor property. It is used to change the second color of the fill, unfill, line or text style if applicable.</p>
	<p>Category: Depends on its source property</p>
	<p>Used by: Depends on its source property</p>
	<p>Can be read by script at run time: No</p>
	<p>Info: <a href="#">Enabling and Disabling Elements for Run-Time Interaction</a></p>
<b>.Color3</b>	<p>Purpose: Color3 is a sub-property of a FillColor, UnfilledColor, LineColor or TextColor property. It is used to change the third color of the fill, unfill, line or text style if applicable.</p>
	<p>Category: Depends on its source property</p>
	<p>Used by: Depends on its source property</p>
	<p>Can be read by script at run time: No</p>
	<p>Info: <a href="#">Enabling and Disabling Elements for Run-Time Interaction</a></p>
<b>ControlStyle</b>	<p>Purpose: Defines the control style as Flat or 3D.</p>
	<p>Category: Appearance</p>
	<p>Used by: Radio Button Group, Check Box</p>
	<p>Can be read by script at run time: No</p>
	<p>Info: <a href="#">Setting the Layout of the Radio Button Group Options</a> and <a href="#">Setting the 3D appearance of a Check Box Control</a></p>
<b>Count</b>	<p>Purpose: Indicates how many items there are in a list.</p>
	<p>Category: not available at design time</p>
	<p>Used by: Radio Button Group, Combo Box, List Box</p>
	<p>Can be read by script at run time: Yes</p>
	<p>Info: <a href="#">Using Radio Button Group-Specific Properties at Run Time</a>, <a href="#">Using Combo Box-Specific Properties at Run Time</a> and <a href="#">Using List Box-Specific Properties at Run Time</a></p>

<b>CustomFormat*</b>	Purpose: Defines the format to be used in the DateTime Picker control for input of a date or time. ***inappropriateUI***  Category: Appearance  Used by: DateTime Picker  Can be read by script at run time: No  Info: <a href="#">Configuring DateTime Picker Controls</a>
<b>CustomProperties</b>	Purpose: The collection of CustomProperties defined by the graphic.  Category: Custom Properties  Used by: Canvas, Embedded Symbol  Can be read by script at run time: No  Info: <a href="#">Using Custom Properties</a>
<b>Description*</b>	Purpose: Contains a meaningful description of the graphic.  Category: Graphic  Used by: Canvas  Can be read by script at run time: No  Info: <a href="#">Setting the Radius of Rounded Rectangles</a>
<b>DefaultValue</b>	Purpose: The default time value to use for the control.  Category: Appearance  Used by: Calendar, DateTime Picker  Can be read by script at run time: No  Info: <a href="#">Setting the Default Value of the Calendar Control</a> and <a href="#">Configuring DateTime Picker Controls</a>
<b>DownImage*</b>	Purpose: Defines the image that is rendered in the button element when it is clicked or held down.  Category: Appearance  Used by: Button

	<p>Can be read by script at run time: No</p> <p>Info: <a href="#">Configuring Buttons with Images</a></p>
<b>DropDownType*</b>	<p>Purpose: Defines the type of combo box: simple, drop-down or drop-down list.</p> <p>Category: Appearance</p> <p>Used by: Combo Box</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Type of Combo Box Control</a></p>
<b>DropDownWidth*</b>	<p>Purpose: Defines the width of the drop-down list.</p> <p>Category: Appearance</p> <p>Used by: Combo Box</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Width of the Drop-Down List</a></p>
<b>DynamicSizeChange*</b>	<p>Purpose: Determines if the embedded graphic propagates the size changes from the source graphic.</p> <p>Category: Appearance</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Enabling or Disabling Dynamic Size Change of Embedded Graphics</a></p>
<b>Enabled</b>	<p>Purpose: When set to True enables the element at run time and allows the user to interact with it. If the property is set to False the user cannot use the mouse or keyboard to interact with the element. Data changes as a result of an animation or script still execute.</p> <p>Category: Runtime Behavior</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Radio Button Group, Check Box, Edit</p>

	<p>Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p>
	<p>Can be read by script at run time: Yes</p>
	<p>Info: <a href="#">Enabling and Disabling Elements for Run-Time Interaction</a></p>
<b>End</b>	<p>Purpose: Defines the end of a line or H/V line as X, Y location.</p>
	<p>Category: Appearance</p>
	<p>Used by: Line, H/V Line</p>
	<p>Can be read by script at run time: No</p>
	<p>Info: <a href="#">Setting Start or End Points of a Line</a></p>
<b>EndCap</b>	<p>Purpose: Defines the cap used at the end of the line of an open element.</p>
	<p>Category: Line Style</p>
	<p>Used by: Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc</p>
	<p>Can be read by script at run time: No</p>
	<p>Info: <a href="#">Setting Line End Shape and Size</a></p>
<b>FillBehavior</b>	<p>Purpose: Determines how the Fill (Horizontal, Vertical or Both) should be applied to the element.</p>
	<p>Category: Fill Style</p>
	<p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p>
	<p>Can be read by script at run time: No</p>
	<p>Info: <a href="#">Setting Fill Behavior</a></p>
<b>FillColor</b>	<p>Purpose: Defines the fill style used for the filled portion of the element.</p>
	<p>Category: Fill Style</p>
	<p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Radio</p>

	<p>Button Group, Check Box, Edit Box, Combo Box, Calendar, List Box, Path</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting Fill Style and Changing Background Color and Text Color of Windows Common Controls</a></p>
<b>FillOrientation</b>	<p>Purpose: Determines the orientation of the fill when the element orientation is any value other than 0.</p> <p>Category: Fill Style</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting Fill Orientation</a></p>
<b>FirstDayOfWeek*</b>	<p>Purpose: Defines the first day of the week used for the display of the columns in the calendar.</p> <p>Category: Appearance</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the First Day of the Week</a></p>
<b>Font</b>	<p>Purpose: Defines the basic text font as defined by the operating system.</p> <p>Category: Text Style</p> <p>Used by: Button, Text, Text Box, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Text Font</a></p>
<b>Format*</b>	<p>Purpose: Defines the format of the reference values. This is only available for array mode.</p> <p>Category: Appearance</p> <p>Used by: DateTime Picker</p>

	<p>Can be read by script at run time: No</p> <p>Info: <a href="#">Configuring DateTime Picker Controls</a></p>
<b>HasTransparentColor*</b>	<p>Purpose: Indicates whether or not the image applies a transparent color. If True the image is rendered transparent wherever a color in the image matches the TransparentColor property.</p> <p>Category: Appearance</p> <p>Used by: Image</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Setting the Image Color Transparency</a></p>
<b>Height</b>	<p>Purpose: Defines the height of the element.</p> <p>Category: Appearance</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Resizing Elements by Changing Size Properties</a></p>
<b>HorizontalDirection</b>	<p>Purpose: Determines the horizontal direction of the fill for the element. Can be "Right" or "Left".</p> <p>Category: Fill Style</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting Horizontal Fill Direction and Percentage</a></p>
<b>HorizontalPercentFill</b>	<p>Purpose: Determines the percentage of horizontal fill for the element.</p> <p>Category: Fill Style</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse,</p>

	Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path
	Can be read by script at run time: Yes
	Info: <a href="#">Setting Horizontal Fill Direction and Percentage</a>
<b>HorizontalScrollbar</b>	Purpose: Determines if a horizontal scroll bar appears on a list box control to allow the user to scroll the list box items horizontally.  Category: Appearance  Used by: List Box  Can be read by script at run time: No  Info: <a href="#">Using a Horizontal Scroll Bar in a List Box Control</a>
<b>Image*</b>	Purpose: Defines the image that is rendered in the element. Any image format supported by the application can be used.  Category: Appearance  Used by: Image  Can be read by script at run time: No  Info: <a href="#">Selecting a Different Image</a>
<b>ImageAlignment*</b>	Purpose: Controls the location of the image relative to the bounding rectangle of the graphic. This property is only applicable when the ImageStyle is set to Normal.  Category: Appearance  Used by: Image  Can be read by script at run time: No  Info: <a href="#">Setting the Image Alignment</a>
<b>ImageStyle</b>	Purpose: Defines how the image is rendered relative to its bounding rectangle.  Category: Appearance  Used by: Button, Image

	<p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Image Display Mode</a></p>
<b>IntegralHeight</b>	<p>Purpose: Determines if the List Box size is an integral multiple of the Font Size so that a finite number of items fit in it without being clipped.</p> <p>Category: Appearance</p> <p>Used by: Combo Box, List Box</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Avoiding Clipping of Items in the Simple Combo Box Control</a></p>
<b>Language</b>	<p>Purpose: Defines the current language of the graphic.</p> <p>Category: Runtime Behavior</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Selecting a Language for a Graphic.</a></p>
<b>LanguageID</b>	<p>Purpose: Defines the current language ID of the graphic.</p> <p>Category: Runtime Behavior</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Selecting a Language for a Graphic.</a></p>
<b>Layout*</b>	<p>Purpose: Defines the way the radio buttons are arranged in the group (Horizontal or Vertical).</p> <p>Category: Appearance</p> <p>Used by: Radio Button Group</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Layout of the Radio Button Group Options</a></p>
<b>LineColor</b>	<p>Purpose: Defines the color and affects of the line or</p>

	<p>border.</p>
	<p>Category: Line Style</p>
	<p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Text Box, Path</p>
	<p>Can be read by script at run time: No</p>
	<p>Info: <a href="#">Setting the Line Style</a></p>
<b>LinePattern</b>	<p>Purpose: Defines the pattern of the line or border.</p>
	<p>Category: Line Style</p>
	<p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Text Box, Path</p>
	<p>Can be read by script at run time: No</p>
	<p>Info: <a href="#">Setting the Line Pattern</a></p>
<b>LineWeight</b>	<p>Purpose: Determines the weight of the element's line or border. A value of 0 means that there is no line or border.</p>
	<p>Category: Line Style</p>
	<p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Text Box, Path</p>
	<p>Can be read by script at run time: Yes</p>
	<p>Info: <a href="#">Setting the Line Weight</a></p>
<b>Locked</b>	<p>Purpose: Locks or unlocks the element's size, position, orientation and origin. Other properties that can have an affect on element size, position, orientation and origin are also locked. These are element-specific.</p>
	<p>Category: Appearance</p>
	<p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2</p>

	<p>Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Locking and Unlocking Elements</a></p>
<b>MaxDropDownItems*</b>	<p>Purpose: Defines the maximum number of items the drop-down list shows.</p> <p>Category: Appearance</p> <p>Used by: Combo Box</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Maximum Number of Items to Appear in the Combo Box Drop-Down List</a></p>
<b>Multiline*</b>	<p>Purpose: Determines if the control shows several lines of text that automatically wrap up when reaching the right border of the control.</p> <p>Category: Appearance</p> <p>Used by: Edit Box</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Configuring the Text to Wrap in an Edit Box Control</a></p>
<b>MultiplePopupsAllowed*</b>	<p>Purpose: If False, ShowSymbol animations only show within a single dialog window no matter how many animations are invoked and regardless of how the animations are configured. If True, ShowSymbol animations show in separate dialog windows.</p> <p>Category: Runtime Behavior</p> <p>Used by: Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Radius of Rounded Rectangles</a></p>
<b>Name</b>	<p>Purpose: Gives the element a meaningful unique name.</p>

	<p>Category: Graphic</p>
	<p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Embedded Symbol, Path</p>
	<p>Can be read by script at run time: No</p>
	<p>Info: <a href="#">Setting the Radius of Rounded Rectangles</a></p>
<b>NewIndex</b>	<p>Purpose: Returns the index of the last value added to the list. This is provided for migration of HMI application windows common controls.</p> <p>Category: not available at design time</p> <p>Used by: Combo Box, List Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Using Combo Box-Specific Properties at Run Time</a> and <a href="#">Using List Box-Specific Properties at Run Time</a></p>
<b>OwningObject*</b>	<p>Purpose: Used as the object reference to replace all "Me." references in expressions and scripts. Everywhere there is a "Me." reference this object reference is used instead. The object name can be set either using a tag or equipment reference.</p> <p>Category: Runtime Behavior</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: Yes</p>
<b>Radius*</b>	<p>Purpose: Defines the radius of the corners of the Rounded Rectangle.</p> <p>Category: Appearance</p> <p>Used by: Rounded Rectangle</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Setting the Radius of Rounded Rectangles</a></p>

<b>ReadOnly*</b>	<p>Purpose: Determines if the user can type data into the edit box.</p> <p>Category: Appearance</p> <p>Used by: Edit Box</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Configuring the Text to be Read-Only in an Edit Box Control</a></p>
<b>RelativeAnchor*</b>	<p>Purpose: Relative anchor point of the source graphic. By default, this is 0,0.</p> <p>Category: Appearance</p> <p>Used by: Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Size Propagation and Anchor Points</a></p>
<b>RelativeOrigin</b>	<p>Purpose: Defines the relative origin as X, Y location. The location is relative to the center point of the element (0, 0).</p> <p>Category: Appearance</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Group, Path, Embedded Symbol</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Changing Points of Origin in the Properties Editor</a></p>
<b>Scripts*</b>	<p>Purpose: Defines a collection of scripts configured for the graphic.</p> <p>Category: Runtime Behavior</p> <p>Used by: Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Adding and Maintaining Graphic Scripts</a></p>
<b>SelectedValue</b>	<p>Purpose: Reads the value of the selected item, or</p>

	<p>selects the item with that value if it exists.</p> <p>Category: not available at design time</p> <p>Used by: Radio Button Group, List Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Using Radio Button Group-Specific Properties at Run Time</a> and <a href="#">Using List Box-Specific Properties at Run Time</a></p>
<b>ShowToday*</b>	<p>Purpose: Determines if today's date is shown on the calendar control.</p> <p>Category: Appearance</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Showing or Hiding Today's Date on a Calendar Control</a></p>
<b>Smoothing*</b>	<p>Purpose: When False the graphics are rendered normally, when True graphics are rendered with anti-aliasing which produces a smoother appearing graphic.</p> <p>Category: Appearance</p> <p>Used by: Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Radius of Rounded Rectangles</a></p>
<b>Start</b>	<p>Purpose: Defines the start of a line or H/V line as X, Y location.</p> <p>Category: Appearance</p> <p>Used by: Line, H/V Line</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting Start or End Points of a Line</a></p>
<b>StartAngle</b>	<p>Purpose: Defines the starting angle of an Arc, Pie or Chord. 0 is always the top of the graphic relative to its orientation. A positive number is clockwise from 0 and a negative number is counter clockwise from</p>

	<p>0. If a negative number is used to set the property it is automatically converted to a positive value.</p>
	<p>Category: Appearance</p>
	<p>Used by: 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, 2 Point Arc, 3 Point Arc</p>
	<p>Can be read by script at run time: Yes</p>
	<p>Info: <a href="#">Changing Angles of Arcs, Pies and Chords</a></p>
<b>StartCap</b>	<p>Purpose: Defines the cap used at the start of the line of an open graphic.</p>
	<p>Category: Line Style</p>
	<p>Used by: Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc</p>
	<p>Can be read by script at run time: No</p>
	<p>Info: <a href="#">Setting Line End Shape and Size</a></p>
<b>SweepAngle</b>	<p>Purpose: Defines the ending angle of the Arc, Pie or Chord. This angle is always measured from the start angle.</p>
	<p>Category: Appearance</p>
	<p>Used by: 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, 2 Point Arc, 3 Point Arc</p>
	<p>Can be read by script at run time: Yes</p>
	<p>Info: <a href="#">Changing Angles of Arcs, Pies and Chords</a></p>
<b>SymbolReference*</b>	<p>Purpose: Contains the exact location that the embedded graphic is linked to. This can help the user in locating the original definition for editing purposes.</p>
	<p>This property is always disabled.</p>
	<p>Category: Runtime Behavior</p>
	<p>Used by: Embedded Symbol</p>
	<p>Can be read by script at run time: No</p>
	<p>Info: <a href="#">Detecting the Source Graphic of an Embedded Graphic</a></p>

<b>TabOrder</b>	<p>Purpose: Defines the tab order for the element. The tab order is only used when navigating by the keyboard. This property is valid only when the TabStop property is set to true.</p> <p>Category: Runtime Behavior</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Editing the Tab Order of an Element</a></p>
<b>TabStop</b>	<p>Purpose: Determines if the element can be navigated to and can receive focus at run time.</p> <p>Category: Runtime Behavior</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Editing the Tab Order of an Element</a></p>
<b>Tension</b>	<p>Purpose: Specifies how tightly the curve bends through the control points of the curve.</p> <p>Category: Appearance</p> <p>Used by: Closed Curve, Curve</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Changing the Tension of Curves and Closed Curves</a></p>
<b>Text</b>	<p>Purpose: Defines the unicode text that is shown by the element.</p>

	<p>Category: Appearance</p> <p>Used by: Button, Text, Text Box, Edit Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Setting the Displayed Text</a></p>
<b>TextColor</b>	<p>Purpose: Defines the color and affects applied to the text.</p> <p>Category: Text Style</p> <p>Used by: Button, Text, Text Box, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Text Color</a> and <a href="#">Changing Background Color and Text Color of Windows Common Controls</a></p>
<b>TextFormat</b>	<p>Purpose: Defines the formatting string that is applied to the text when it is shown.</p> <p>Category: Appearance</p> <p>Used by: Button, Text, Text Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Setting the Text Display Format</a></p>
<b>TitleFillColor*</b>	<p>Purpose: Determines the background solid color in the title bar of the calendar control.</p> <p>Category: Fill Style</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting Title Fill Color and Text Color on a Calendar Control</a></p>
<b>TitleTextColor*</b>	<p>Purpose: Determines the text solid color in the title bar of the calendar control.</p> <p>Category: Text Style</p> <p>Used by: Calendar</p>

	<p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting Title Fill Color and Text Color on a Calendar Control</a></p>
<b>TopIndex*</b>	<p>Purpose: Returns the index of the top most item in the list. This is provided for migration of HMI application windows common controls.</p> <p>Category: not available at design time</p> <p>Used by: List Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Using List Box-Specific Properties at Run Time</a></p>
<b>TrailingTextColor*</b>	<p>Purpose: Determines the text solid color of the text for the trailing days. The trailing days are days outside the current month.</p> <p>Category: Text Style</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Text Color for Trailing Dates in a Calendar Control</a></p>
<b>Transparency</b>	<p>Purpose: Defines the transparency of the element. A value of 0 means fully opaque and a value of 100 means fully transparent.</p> <p>Category: Appearance</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Group, Path, Embedded Symbol</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Setting the Transparency Level of an Element</a></p>
<b>.Transparency</b>	<p>Purpose: Transparency is a sub-property of a FillColor, UnfilledColor, LineColor or TextColor property. It is used to change the transparency of the fill, unfill, line or text style if applicable. The transparency acts in addition to the transparency of</p>

	<p>the element.</p> <p>Category: Depends on its source property</p> <p>Used by: Depends on its source property</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Enabling and Disabling Elements for Run-Time Interaction</a></p>
<b>TransparentColor*</b>	<p>Purpose: Defines the RGB color value that is used as the transparent color.</p> <p>Category: Appearance</p> <p>Used by: Image</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Image Color Transparency</a></p>
<b>TreatAsIcon</b>	<p>Purpose: If this property is set to False, the animations defined on the graphics within the group or embedded graphic take precedence over an animation defined on the group or embedded graphic. If there are no animations or the user clicked on an area of the group or embedded graphic that does not have an animation, then the group or embedded graphic animation executes.</p> <p>If the property is set to True, only the animation on the group or embedded graphic is executed. The interactive animations within the group or embedded graphic never execute.</p> <p>Category: Runtime Behavior</p> <p>Used by: Group, Embedded Symbol</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Editing the Embedded Graphic</a></p>
<b>UnFilledColor</b>	<p>Purpose: Determines the element's unfilled area appearance.</p> <p>Category: Fill Style</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2</p>

	<p>Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting Unfilled Style</a></p>
<b>UpImage*</b>	<p>Purpose: Defines the image that is used in the button element when it is un-clicked or released.</p> <p>Category: Appearance</p> <p>Used by: Button</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Configuring Buttons with Images</a></p>
<b>Value</b>	<p>Purpose: Reads the value of the selected item, or selects the item with that value if it exists. Its data type depends on the control.</p> <p>Category: not available at design time</p> <p>Used by: Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Reading and Writing the Selected Value at Run Time</a></p>
<b>VerticalDirection</b>	<p>Purpose: Defines the vertical direction of the fill. Can be "Top" or "Bottom".</p> <p>Category: Fill Style</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting Vertical Fill Direction and Percentage</a></p>
<b>VerticalPercentFill</b>	<p>Purpose: Determines the percentage of vertical fill for the element.</p> <p>Category: Fill Style</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p>

	<p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Setting Vertical Fill Direction and Percentage</a></p>
<b>Visible</b>	<p>Purpose: Determines the visibility of the element. This property is configured at design time and used only at runtime. At design time all elements are visible irrespective of this setting.</p> <p>Category: Runtime Behavior</p>
	<p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p>
	<p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Changing the Visibility of Elements</a></p>
<b>Width</b>	<p>Purpose: Defines the width of the element.</p> <p>Category: Appearance</p>
	<p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p>
	<p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Resizing Elements by Changing Size Properties</a></p>
<b>WordWrap</b>	<p>Purpose: When set to True, the text in the button or text box is formatted to fit as much text on a single line within the horizontal bounding area of the element and then continued to the next line. This continues as long as there is vertical space.</p> <p>Category: Appearance</p>
	<p>Used by: Button, Text Box</p>
	<p>Can be read by script at run time: Yes</p>

	Info: <a href="#">Wrapping Text in Buttons</a>
X	<p>Purpose: Defines the left position of the element.</p> <p>Category: Appearance</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p>
	Can be read by script at run time: Yes
	Info: <a href="#">Moving Elements</a>
Y	<p>Purpose: Defines the top position of the element.</p> <p>Category: Appearance</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p>
	Can be read by script at run time: Yes
	Info: <a href="#">Moving Elements</a>

## Related Topics

[List of Element Properties](#)

## List by Functional Area

Each property of the elements, the canvas, element groups and embedded objects belongs to one of the following property categories:

- Graphic
- Appearance
- Fill Style
- Line Style
- Text Style

- Runtime Behavior
- Custom Properties

## Related Topics

- [List of Element Properties](#)
- [Graphic Category Properties](#)
- [Appearance Category Properties](#)
- [Fill Style Group Properties](#)
- [Line Style Group Properties](#)
- [Text Style Group Properties](#)
- [Runtime Behavior Group Properties](#)
- [Custom Properties Group Properties](#)

### Graphic Category Properties

The following table contains a list of properties in the Graphic property category used by the:

- Elements.
- Canvas.
- Element groups.
- Embedded graphics.

It shows their purpose, where they are used and where to find more information on how to use them.

An asterisk (\*) identifies properties that are specific to only one type of element or the canvas, a group or an embedded graphic.

Property	Purpose, category, usage and further information
<b>Description*</b>	Purpose: Contains a meaningful description of the graphic. Category: Graphic Used by: Canvas Can be read by script at run time: No Info: <a href="#">Setting the Radius of Rounded Rectangles</a>
<b>Name</b>	Purpose: Gives the element a meaningful unique name. Category: Graphic Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box,

	<p>Combo Box, Calendar, DateTime Picker, List Box, Embedded Symbol, Path</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Radius of Rounded Rectangles</a></p>
--	---

## Related Topics

[List by Functional Area](#)

### Appearance Category Properties

The following table contains a list of properties in the Appearance property category used by the:

- Elements.
- Canvas.
- Element groups.
- Embedded graphics.

It shows their purpose, where they are used and where to find more information on how to use them.

Asterisk (\*) marks properties that are specific to only one type of element or the canvas, a group or an embedded graphic.

Property	Purpose, category, usage and further information
<b>AbsoluteAnchor*</b>	<p>Purpose: Defines the absolute anchor point of the source graphic. By default, this is the center point of all elements on the canvas but can be changed.</p> <p>Category: Appearance</p> <p>Used by: Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Size Propagation and Anchor Points</a></p>
<b>AnchorFixedTo</b>	<p>Purpose: Determines if the anchor point is fixed to the canvas when you resize, delete, or add elements (Absolute), or if the anchor point is recalculated relative to the element sizes and positions (Relative).</p> <p>Category: Appearance</p> <p>Used by: Embedded Symbol, Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Size Propagation and Anchor Points</a></p>

<b>AbsoluteOrigin</b>	<p>Purpose: Defines an X, Y location relative to the top, left (0, 0) origin of the graphic or window.</p> <p>Category: Appearance</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Embedded Symbol, Group, Path, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box.</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Changing Points of Origin in the Properties Editor</a></p>
<b>AnchorPoint*</b>	<p>Purpose: Defines the anchor X, Y location of the embedded graphic.</p> <p>Category: Appearance</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Size Propagation and Anchor Points</a></p>
<b>Angle</b>	<p>Purpose: Defines the current angle of rotation of the element. 0 is always the top of the element relative to the canvas. Angle is always determined relative to the top of the element and rotates in a clockwise direction.</p> <p>Category: Appearance</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Group, Embedded Symbol</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Rotating Elements by Changing the Angle Property</a></p>
<b>AutoScale</b>	<p>Purpose: If this property is set to True then the text is stretched horizontally and vertically (larger or smaller) to fit the bounding rectangle.</p>

	<p>Category: Appearance</p> <p>Used by: Button, Text Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Setting Auto Scaling and Word Wrapping for a Text Box</a></p>
<b>ButtonStyle*</b>	<p>Purpose: Determines if the button appears as a standard button or as an image.</p> <p>Category: Appearance</p> <p>Used by: Button</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Configuring Buttons with Images</a></p>
<b>CalendarColumns*</b>	<p>Purpose: Defines the number of columns the calendar object has.</p> <p>Category: Appearance</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Number of Calendar Month Sheets</a></p>
<b>CalendarRows*</b>	<p>Purpose: Defines the number of rows the calendar object has.</p> <p>Category: Appearance</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Number of Calendar Month Sheets</a></p>
<b>Checked*</b>	<p>Purpose: Sets or gets the value of check box. This is the initial value of the check box when the control is not connected to a reference and is overridden at run time with value of reference.</p> <p>Category: Appearance</p> <p>Used by: Check Box</p> <p>Can be read by script at run time: Yes</p>

	<a href="#">Info: Setting the Default State of a Check Box Control</a>
<b>ControlStyle</b>	<p>Purpose: Defines the control style as Flat or 3D.</p> <p>Category: Appearance</p> <p>Used by: Radio Button Group, Check Box</p> <p>Can be read by script at run time: No</p> <p><a href="#">Info:Setting the Layout of the Radio Button Group Options</a> and <a href="#">Setting the 3D appearance of a Check Box Control</a></p>
<b>CustomFormat*</b>	<p>Purpose: Defines the format to be used in the DateTime Picker control for input of a date or time.</p> <p>Category: Appearance</p> <p>Used by: DateTime Picker</p> <p>Can be read by script at run time: No</p> <p><a href="#">Info: Configuring DateTime Picker Controls</a></p>
<b>DefaultValue</b>	<p>Purpose: The default time value to use for the control.</p> <p>Category: Appearance</p> <p>Used by: Calendar, DateTime Picker</p> <p>Can be read by script at run time: No</p> <p><a href="#">Info: Setting the Default Value of the Calendar Control</a> and <a href="#">Configuring DateTime Picker Controls</a></p>
<b>DownImage*</b>	<p>Purpose: Defines the image that is rendered in the button element when it is clicked or held down.</p> <p>Category: Appearance</p> <p>Used by: Button</p> <p>Can be read by script at run time: No</p> <p><a href="#">Info: Configuring Buttons with Images</a></p>
<b>DropDownType*</b>	<p>Purpose: Defines the type of combo box: simple, drop-down or drop-down list.</p> <p>Category: Appearance</p>

	<p>Used by: Combo Box</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Type of Combo Box Control</a></p>
<b>DropDownWidth*</b>	<p>Purpose: Defines the width of the drop-down list.</p> <p>Category: Appearance</p> <p>Used by: Combo Box</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Width of the Drop-Down List</a></p>
<b>DynamicSizeChange*</b>	<p>Purpose: Determines if the embedded graphic propagates the size changes from the source graphic.</p> <p>Category: Appearance</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Enabling or Disabling Dynamic Size Change of Embedded Graphics</a></p>
<b>End</b>	<p>Purpose: Defines the end of a line or H/V line as X, Y location.</p> <p>Category: Appearance</p> <p>Used by: Line, H/V Line</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting Start or End Points of a Line</a></p>
<b>FirstDayOfWeek*</b>	<p>Purpose: Defines the first day of the week used for the display of the columns in the calendar.</p> <p>Category: Appearance</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the First Day of the Week</a></p>
<b>Format*</b>	<p>Purpose: Defines the format of the reference values. This is only available for array mode.</p>

	<p>Category: Appearance</p> <p>Used by: DateTime Picker</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Configuring DateTime Picker Controls</a></p>
<b>HasTransparentColor*</b>	<p>Purpose: Indicates whether or not the image applies a transparent color. If True the image is rendered transparent wherever a color in the image matches the TransparentColor property.</p> <p>Category: Appearance</p> <p>Used by: Image</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Setting the Image Color Transparency</a></p>
<b>Height</b>	<p>Purpose: Defines the height of the element.</p> <p>Category: Appearance</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Resizing Elements by Changing Size Properties</a></p>
<b>HorizontalScrollbar</b>	<p>Purpose: Determines if a horizontal scroll bar appears on a list box control to allow the user to scroll the list box items horizontally.</p> <p>Category: Appearance</p> <p>Used by: List Box</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Using a Horizontal Scroll Bar in a List Box Control</a></p>
<b>Image*</b>	<p>Purpose: Defines the image that is rendered in the element. Any image format supported by the application can be used.</p>

	<p>Category: Appearance</p> <p>Used by: Image</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Selecting a Different Image</a></p>
<b>ImageAlignment*</b>	<p>Purpose: Controls the location of the image relative to the bounding rectangle of the graphic. This property is only applicable when the ImageStyle is set to Normal.</p> <p>Category: Appearance</p> <p>Used by: Image</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Image Alignment</a></p>
<b>ImageStyle</b>	<p>Purpose: Defines how the image is rendered relative to its bounding rectangle.</p> <p>Category: Appearance</p> <p>Used by: Button, Image</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Image Display Mode</a></p>
<b>IntegralHeight</b>	<p>Purpose: Determines if the List Box size is an integral multiple of the Font Size so that a finite number of items fit in it without being clipped.</p> <p>Category: Appearance</p> <p>Used by: Combo Box, List Box</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Avoiding Clipping of Items in the Simple Combo Box Control</a></p>
<b>Layout*</b>	<p>Purpose: Defines the way the radio buttons are arranged in the group (Horizontal or Vertical).</p> <p>Category: Appearance</p> <p>Used by: Radio Button Group</p> <p>Can be read by script at run time: No</p>

	<p>Info: <a href="#">Setting the Layout of the Radio Button Group Options</a></p>
<b>Locked</b>	<p>Purpose: Locks or unlocks the element's size, position, orientation and origin. Other properties that can have an affect on element size, position, orientation and origin are also locked. These are element-specific.</p> <p>Category: Appearance</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Locking and Unlocking Elements</a></p>
<b>MaxDropDownItems*</b>	<p>Purpose: Defines the maximum number of items the drop-down list shows.</p> <p>Category: Appearance</p> <p>Used by: Combo Box</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Maximum Number of Items to Appear in the Combo Box Drop-Down List</a></p>
<b>Multiline*</b>	<p>Purpose: Determines if the control shows several lines of text that automatically wrap up when reaching the right border of the control.</p> <p>Category: Appearance</p> <p>Used by: Edit Box</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Configuring the Text to Wrap in an Edit Box Control</a></p>
<b>Radius*</b>	<p>Purpose: Defines the radius of the corners of the Rounded Rectangle.</p> <p>Category: Appearance</p>

	<p>Used by: Rounded Rectangle</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Setting the Radius of Rounded Rectangles</a></p>
<b>ReadOnly*</b>	<p>Purpose: Determines if the user can type data into the edit box.</p> <p>Category: Appearance</p> <p>Used by: Edit Box</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Configuring the Text to be Read-Only in an Edit Box Control</a></p>
<b>RelativeAnchor*</b>	<p>Purpose: Relative anchor point of the source graphic. By default, this is 0,0.</p> <p>Category: Appearance</p> <p>Used by: Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Size Propagation and Anchor Points</a></p>
<b>RelativeOrigin</b>	<p>Purpose: Defines the relative origin as X, Y location. The location is relative to the center point of the element (0, 0).</p> <p>Category: Appearance</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Group, Path, Embedded Symbol</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Changing Points of Origin in the Properties Editor</a></p>
<b>ShowToday*</b>	<p>Purpose: Determines if today's date is shown on the calendar control.</p> <p>Category: Appearance</p> <p>Used by: Calendar</p>

	<p>Can be read by script at run time: No</p> <p>Info: <a href="#">Showing or Hiding Today's Date on a Calendar Control</a></p>
<b>Smoothing*</b>	<p>Purpose: When False the graphics are rendered normally, when True graphics are rendered with anti-aliasing which produces a smoother appearing graphic.</p> <p>Category: Appearance</p> <p>Used by: Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Radius of Rounded Rectangles</a></p>
<b>Start</b>	<p>Purpose: Defines the start of a line or H/V line as X, Y location.</p> <p>Category: Appearance</p> <p>Used by: Line, H/V Line</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting Start or End Points of a Line</a></p>
<b>StartAngle</b>	<p>Purpose: Defines the starting angle of an Arc, Pie or Chord. 0 is always the top of the graphic relative to its orientation. A positive number is clockwise from 0 and a negative number is counter clockwise from 0. If a negative number is used to set the property it is automatically converted to a positive value.</p> <p>Category: Appearance</p> <p>Used by: 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, 2 Point Arc, 3 Point Arc</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Changing Angles of Arcs, Pies and Chords</a></p>
<b>SweepAngle</b>	<p>Purpose: Defines the ending angle of the Arc, Pie or Chord. This angle is always measured from the start angle.</p> <p>Category: Appearance</p> <p>Used by: 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point</p>

	<p>Chord, 2 Point Arc, 3 Point Arc</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Changing Angles of Arcs, Pies and Chords</a></p>
<b>Tension</b>	<p>Purpose: Specifies how tightly the curve bends through the control points of the curve.</p> <p>Category: Appearance</p> <p>Used by: Closed Curve, Curve</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Changing the Tension of Curves and Closed Curves</a></p>
<b>Text</b>	<p>Purpose: Defines the unicode text that is shown by the element.</p> <p>Category: Appearance</p> <p>Used by: Button, Text, Text Box, Edit Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Setting the Displayed Text</a></p>
<b>TextFormat</b>	<p>Purpose: Defines the formatting string that is applied to the text when it is shown.</p> <p>Category: Appearance</p> <p>Used by: Button, Text, Text Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Setting the Text Display Format</a></p>
<b>Transparency</b>	<p>Purpose: Defines the transparency. A value of 0 means fully opaque and a value of 100 means fully transparent.</p> <p>Category: Appearance</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Group, Path, Embedded Symbol</p>

	<p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Setting the Transparency Level of an Element</a></p>
<b>TransparentColor*</b>	<p>Purpose: Defines the RGB color value that is used as the transparent color.</p> <p>Category: Appearance</p> <p>Used by: Image</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Image Color Transparency</a></p>
<b>UpImage*</b>	<p>Purpose: Defines the image that is used in the button element when it is un-clicked or released.</p> <p>Category: Appearance</p> <p>Used by: Button</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Configuring Buttons with Images</a></p>
<b>Width</b>	<p>Purpose: Defines the width of the element.</p> <p>Category: Appearance</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Resizing Elements by Changing Size Properties</a></p>
<b>WordWrap</b>	<p>Purpose: When set to True, the text in the button or text box is formatted to fit as much text on a single line within the horizontal bounding area of the element and then continued to the next line. This continues as long as there is vertical space.</p> <p>Category: Appearance</p> <p>Used by: Button, Text Box</p>

	Can be read by script at run time: Yes
	Info: <a href="#">Wrapping Text in Buttons</a>
X	<p>Purpose: Defines the left position of the element.</p> <p>Category: Appearance</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p>
	Can be read by script at run time: Yes
	Info: <a href="#">Moving Elements</a>
Y	<p>Purpose: Defines the top position of the element.</p> <p>Category: Appearance</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p>
	Can be read by script at run time: Yes
	Info: <a href="#">Moving Elements</a>

## Related Topics

[List by Functional Area](#)

### Fill Style Group Properties

The following table contains a list of properties in the Fill Style property category used by the:

- Elements.
- Canvas.
- Element groups.
- Embedded graphics.

It describes their purpose, where they are used and where to find more information on how to use them.

An asterisk (\*) identifies properties that apply to only one type of element or the canvas, a group or an embedded graphic.

Property	Purpose, category, usage and further information
<b>FillBehavior</b>	<p>Purpose: Determines how the Fill (Horizontal, Vertical or Both) should be applied to the element.</p> <p>Category: Fill Style</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting Fill Behavior</a></p>
<b>FillColor</b>	<p>Purpose: Defines the fill style used for the filled portion of the element.</p> <p>Category: Fill Style</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, List Box, Path</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting Fill Style and Changing Background Color and Text Color of Windows Common Controls</a></p>
<b>FillOrientation</b>	<p>Purpose: Determines the orientation of the fill when the element orientation is any value other than 0.</p> <p>Category: Fill Style</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting Fill Orientation</a></p>
<b>HorizontalDirection</b>	<p>Purpose: Determines the horizontal direction of the fill for the element. Can be "Right" or "Left".</p> <p>Category: Fill Style</p>

	<p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting Horizontal Fill Direction and Percentage</a></p>
<b>HorizontalPercentFill</b>	<p>Purpose: Determines the percentage of horizontal fill for the element.</p> <p>Category: Fill Style</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Setting Horizontal Fill Direction and Percentage</a></p>
<b>TitleFillColor*</b>	<p>Purpose: Determines the background solid color in the title bar of the calendar control.</p> <p>Category: Fill Style</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting Title Fill Color and Text Color on a Calendar Control</a></p>
<b>UnFilledColor</b>	<p>Purpose: Determines the element's unfilled area appearance.</p> <p>Category: Fill Style</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting Unfilled Style</a></p>
<b>VerticalDirection</b>	<p>Purpose: Defines the vertical direction of the fill. Can be "Top" or "Bottom".</p> <p>Category: Fill Style</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse,</p>

	Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path
	Can be read by script at run time: No
	Info: <a href="#">Setting Vertical Fill Direction and Percentage</a>
<b>VerticalPercentFill</b>	Purpose: Determines the percentage of vertical fill for the element.
	Category: Fill Style
	Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path
	Can be read by script at run time: Yes
	Info: <a href="#">Setting Vertical Fill Direction and Percentage</a>

## Related Topics

[List by Functional Area](#)

### Line Style Group Properties

The following table contains a list of properties in the Line Style property category used by the:

- Elements.
- Canvas.
- Element groups.
- Embedded graphics.

It shows their purpose, where they are used and where to find more information on how to use them.

Asterisk (\*) marks properties that are specific to only one type of element or the canvas, a group or an embedded graphic.

Property	Purpose, category, usage and further information
<b>EndCap</b>	<p>Purpose: Defines the cap used at the end of the line of an open element.</p> <p>Category: Line Style</p> <p>Used by: Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting Line End Shape and Size</a></p>
<b>LineColor</b>	<p>Purpose: Defines the color and affects of the line or border.</p> <p>Category: Line Style</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Text Box, Path</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Line Style</a></p>
<b>LinePattern</b>	<p>Purpose: Defines the pattern of the line or border.</p> <p>Category: Line Style</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Text Box, Path</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Line Pattern</a></p>
<b>LineWeight</b>	<p>Purpose: Determines the weight of the element's line or border. A value of 0 means that there is no line or border.</p> <p>Category: Line Style</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Text Box, Path</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Setting the Line Weight</a></p>
<b>StartCap</b>	<p>Purpose: Defines the cap used at the start of the line of an open graphic.</p> <p>Category: Line Style</p> <p>Used by: Line, H/V Line, Polyline, Curve, 2 Point Arc, 3</p>

Property	Purpose, category, usage and further information
	<p>Point Arc</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting Line End Shape and Size</a></p>

## Related Topics

[List by Functional Area](#)

### Text Style Group Properties

The following table contains a list of properties in the Text Style property category used by the:

- Elements.
- Canvas.
- Element groups.
- Embedded graphics.

It shows their purpose, where they are used and where to find more information on how to use them.

An asterisk (\*) indicates properties that are specific to only one type of element or the canvas, a group or an embedded graphic.

Property	Purpose, category, usage and further information
<b>Alignment</b>	<p>Purpose: Controls the location of the text relative to the bounding rectangle of the element.</p> <p>Category: Text Style</p> <p>Used by: Button, Text, Text Box</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Text Alignment</a></p>
<b>Caption*</b>	<p>Purpose: Defines the text shown on the Check Box at design time and at run time when the caption property is not bound to a reference in the checkbox animation panel.</p> <p>Category: Text Style</p> <p>Used by: Check Box</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Caption Text of a Check Box Control</a></p>

<b>Font</b>	<p>Purpose: Defines the basic text font as defined by the operating system.</p> <p>Category: Text Style</p> <p>Used by: Button, Text, Text Box, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Text Font</a></p>
<b>TextColor</b>	<p>Purpose: Defines the color and affects applied to the text.</p> <p>Category: Text Style</p> <p>Used by: Button, Text, Text Box, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Text Color</a> and <a href="#">Changing Background Color and Text Color of Windows Common Controls</a></p>
<b>TitleTextColor*</b>	<p>Purpose: Determines the text solid color in the title bar of the calendar control.</p> <p>Category: Text Style</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting Title Fill Color and Text Color on a Calendar Control</a></p>
<b>TrailingTextColor*</b>	<p>Purpose: Determines the text solid color of the text for the trailing days. The trailing days are days outside the current month.</p> <p>Category: Text Style</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Text Color</a></p>

## Related Topics

[List by Functional Area](#)

### Runtime Behavior Group Properties

The following table contains a list of properties in the Runtime Behavior property category used by the:

- Elements.
- Canvas.
- Element groups.
- Embedded graphics.

The table shows the purpose of Runtime Behavior properties, where they are used, and where to find more information on how to use them.

Asterisk (\*) marks properties that are specific to only one type of element or the canvas, a group or an embedded graphic.

Property	Purpose, category, usage and further information
Enabled	<p>Purpose: When set to True enables the element at run time and allows the user to interact with it. If the property is set to False the user cannot use the mouse or keyboard to interact with the element. Data changes as a result of an animation or script still execute.</p> <p>Category: Runtime Behavior</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Enabling and Disabling Elements for Run-Time Interaction</a></p>
Language	<p>Purpose: Defines the current language of the graphic.</p> <p>Category: Runtime Behavior</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: Yes</p>

	<p>Info: <a href="#">Selecting a Language for a Graphic</a>.</p>
LanguageID	<p>Purpose: Defines the current language ID of the graphic.</p> <p>Category: Runtime Behavior</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: Yes</p> <p>Info: <a href="#">Selecting a Language for a Graphic</a>.</p>
MultiplePopupsAllowed*	<p>Purpose: If False, ShowSymbol animations only show within a single dialog window no matter how many animations are invoked and regardless of how the animations are configured. If True, ShowSymbol animations show in separate dialog windows.</p> <p>Category: Runtime Behavior</p> <p>Used by: Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Setting the Radius of Rounded Rectangles</a></p>
OwningObject*	<p>Purpose: Used as the object reference to replace all "Me." references in expressions and scripts. Everywhere there is a "Me." reference this object reference is used instead. The object name can be set either using a tag or equipment reference.</p> <p>Category: Runtime Behavior</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: Yes</p>
Scripts*	<p>Purpose: Defines a collection of scripts configured for the graphic.</p> <p>Category: Runtime Behavior</p> <p>Used by: Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Adding and Maintaining Graphic Scripts</a></p>
SymbolReference*	<p>Purpose: Contains the exact location that the Embedded Symbol is linked to. This can help the user</p>

	<p>in locating the original definition for editing purposes.</p>
	<p>This property is always disabled.</p>
	<p>Category: Runtime Behavior</p>
	<p>Used by: Embedded Symbol</p>
	<p>Can be read by script at run time: No</p>
	<p>Info: <a href="#">Detecting the Source Graphic of an Embedded Graphic</a></p>
TabOrder	<p>Purpose: Defines the tab order for the element. The tab order is only used when navigating by the keyboard. This property is valid only when the TabStop property is set to true.</p> <p>Category: Runtime Behavior</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Editing the Tab Order of an Element</a></p>
TabStop	<p>Purpose: Determines if the element can be navigated to and can receive focus at run time.</p> <p>Category: Runtime Behavior</p> <p>Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Editing the Tab Order of an Element</a></p>
TreatAsIcon	<p>Purpose: If this property is set to False, the animations defined on the graphics within the group or</p>

	embedded graphic take precedence over an animation defined on the group or embedded graphic. If there are no animations or the user clicked on an area of the group or embedded graphic that does not have an animation, then the group or embedded graphic animation executes.
	If the property is set to True, only the animation on the group or embedded graphic is executed. The interactive animations within the group or embedded graphic never execute.
	Category: Runtime Behavior
	Used by: Group, Embedded Symbol
	Can be read by script at run time: Yes
	Info: <a href="#">Editing the Embedded Graphic</a>
Visible	Purpose: Determines the visibility of the element. This property is configured at design time and used only at runtime. At design time all elements are visible irrespective of this setting.  Category: Runtime Behavior  Used by: Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol  Can be read by script at run time: Yes  Info: <a href="#">Changing the Visibility of Elements</a>

## Related Topics

[List by Functional Area](#)

### Custom Properties Group Properties

The following table contains a list of properties in the Custom Properties property category used by the:

- Elements.
- Canvas.

- Element groups.
- Embedded graphics.

It shows their purpose, where they are used and where to find more information on how to use them.

The Custom Properties group contains also any other custom property you define.

Property	Purpose, category, usage and further information
CustomProperties	<p>Purpose: The collection of CustomProperties defined by the graphic.</p> <p>Category: Custom Properties</p> <p>Used by: Canvas, Embedded Symbol</p> <p>Can be read by script at run time: No</p> <p>Info: <a href="#">Using Custom Properties</a></p>

## Related Topics

[List by Functional Area](#)

### Order of Precedence for Property Styles

The order of precedence for property styles from high to low is:

1. Quality and Status
2. Element Style Animation
3. Style Animations
4. Group-level Element Style
5. Element-level Element Style
6. Local element-level style

## Related Topics

[List of Element Properties](#)

## Switching Languages for Graphic Elements

This section describes how to switch the language shown for Industrial Graphics, the effects of different language settings on an individual industrial graphic, and language switching behaviors for certain features, such as embedded graphics, custom properties, and string substitution.

The language settings configured in Plant SCADA Studio control which languages are available to graphics (see [Configure Languages for Industrial Graphics Applications](#)). You cannot add a language at the graphic level.

Using the Industrial Graphic Editor, you select which of the configured languages you want to show for your graphic. By default, text for the graphic is shown in the default language and font specified in Plant SCADA Studio.

## In This Chapter

- [Selecting a Language for a Graphic](#)
- [Removing a Language from a Graphic](#)
- [Creating Elements When Multiple Languages are Defined](#)
- [How Fonts are Applied at Design Time](#)
- [Language Switching for Embedded Graphics](#)
- [String Substitutions and Language Switching](#)
- [Translating String Custom Properties](#)
- [Support for Empty Strings](#)
- [Language Switching Example](#)
- [Overriding Translated Strings for Industrial Graphics](#)
- [Language Switching at Run Time](#)

### Selecting a Language for a Graphic

When you select a language for a graphic, all graphic elements show the translated text associated with the selected language, if it is available. You can switch languages even if you open the graphic in read-only mode.

You can only select languages that are currently configured in Plant SCADA Studio.

---

**Note:** The current Element Style assigned to a text font does not change when you switch to a different language.

---

#### To select the language for a graphic

1. Open the graphic in the Industrial Graphic Editor.
2. In the **Languages** panel, select the language from the list.

### Related Topics

[Switching Languages for Graphic Elements](#)

### Removing a Language from a Graphic

If you remove a language from Plant SCADA Studio, the language is still available for a graphic until you specifically remove it.

You cannot remove a language from a graphic if the language is configured at the HMI/SCADA product level.

#### To remove the language for a graphic

1. Open the graphic in the Industrial Graphic Editor.
2. On the **Special** menu, point to **Configure** and then click **Locales**. The **Configure Languages** dialog box appears.
3. Select the language to remove and click **Remove**.
4. Click **OK**.

## Related Topics

[Switching Languages for Graphic Elements](#)

### Creating Elements When Multiple Languages are Defined

You must select the default language for your HMI/SCADA product before you create a new graphic element. You cannot create an element if you have a secondary language selected in the Industrial Graphic Editor. This includes:

- Duplicating an element
- Pasting an element
- Embedding a graphic
- Grouping or ungrouping elements
- Combining paths
- Breaking paths
- Convert a graphic to a group

## Related Topics

[Switching Languages for Graphic Elements](#)

### How Fonts are Applied at Design Time

When you create a graphic element that supports visible text, it is created in the default language of your HMI/SCADA product. The font used is the last font persisted in the editor. However, if you provide a specific translation for an element in a secondary language, the configured font for the language is applied to the element.

For example:

1. You configure three languages: English (Default, Font = Arial), French (Font = Courier New) and German (Font = Times New Roman).
2. You open a graphic S1 in the English language. The editor default is Arial. You create a textbox in English. It is created with the Arial font.
3. You switch to German and translate the text. The font changes to Times New Roman, which is the font configured for German. The original font size and style remains the same.
4. You switch to French. The font is Arial because the text in French is not translated yet.

## Related Topics

[Switching Languages for Graphic Elements](#)

## Language Switching for Embedded Graphics

When you embed a graphic into another graphic, any translations are also loaded for the embedded graphic. Switching the language for a graphic also switches the language for all embedded graphics.

For example, a graphic G1 contains a text graphic with the text "English String." English is the default language. The text is set to "French String" for the French language. The following steps describe language switching for the embedded graphic:

1. You embed the graphic G1 into the graphic G2.  
You see the text showing the "English String."
2. You switch the language to French in the editor of G2.  
The text in the embedded graphic G1 switches to "French String."

If you convert an embedded graphic to a group, any translations defined for the embedded graphic are migrated to the new elements created in the graphic. If the embedded graphic supports languages that are not defined in the base graphic, those translations are removed during the conversion.

If you open a graphic containing one or more embedded graphics and the current language of the Industrial Graphic Editor is not available in the embedded graphics, then the embedded graphic uses the configured default language, if available. If the configured default language configured in your application, is not available, then the embedded graphics use the last saved default language of that graphic.

For example:

1. You create graphic S1 with French and German languages. The default language for your HMI/SCADA application is German.
2. You embed G1 inside G2 and then save and close G2.
3. You change the default application language to French.
4. You add English as another language.

If you open the graphic G2 in the English language, then the graphic G1 is shown using the French text. This is because French is the default language configured in the application.

Embedded graphics support translations for custom properties. For more information on translating custom properties, see [Translating String Custom Properties](#).

Embedded graphics also support translations for string substitutions. For more information on translation and string substitutions, see [String Substitutions and Language Switching](#).

## Related Topics

[Switching Languages for Graphic Elements](#)

## String Substitutions and Language Switching

You can substitute strings for textual elements within a graphic. For general information about string substitution, see [Substituting Strings](#).

If you perform a first-time string substitution on an embedded graphic in the primary language, that substitution is shown in the secondary languages. You can then perform a substitution in the secondary language to create a

string substitution specific to the secondary language.

If you perform a first-time string substitution on an embedded graphic in a secondary language, the substitution is also applied to the primary language, because the translated string that previously existed for the primary language is no longer valid. Because the primary language value is changed in the graphic, this string applies to all secondary languages configured. You can then perform a second substitution in the primary language, which will apply to all secondary languages except the ones that have had a specific substitution set.

If you perform a string substitution in a secondary language with an existing string substitution in the primary language, the new substitution is applied to the secondary language only.

The following design time and run-time rules are applied during a language switch to properly update an embedded graphic with the current substitutions for the language:

- Apply the string substitutions from the default language.
- Apply the string substitutions from the secondary language, if switching to a secondary language.

For example, an embedded graphic contains a text graphic with the text "English String." English is the default language. The following steps describe how changing the language affects string substitution for the embedded graphic:

1. While editing in the default language, you select an embedded graphic and open the string substitution dialog box.
2. You can see the old column with a value of "English String."
3. You replace the "English String" with "New English String."
4. You close the string substitution dialog box.
5. You switch to the French language.
6. You open the string substitution dialog box and see the string "New English String" in the old column.
7. You now replace the "New English String" with "New French String."
8. You close the string substitution dialog box.
9. You switch to the German language.
10. You open the string substitution dialog box and see the string "New English String" in the old column.
11. You now replace the "New English String" with "New German String."
12. You close the string substitution dialog box.
13. You switch to the French language.
14. You open the string substitution dialog box and see the string "New French String" in the old column.

If you select an alternate graphic, the string substitutions made on the initial graphic are reapplied to the new graphic across all languages.

## Related Topics

[Switching Languages for Graphic Elements](#)

## Translating String Custom Properties

You can translate custom properties that are defined as static strings.

If the custom property is configured with a reference, then that reference applies across all languages in the graphic. If you change that reference in any language to a static string, that string is set for all languages, and you can provide specific translations in the other languages.

For example, you create a custom property CP1 of type string with a default value of "Hello." You can now translate this custom property. You switch to another language in the editor and modify the default value of CP1 to UD1.str1 (changed from string to reference). Now CP1 cannot be translated. If you go back and change CP1 from a reference to a string, you can translate it again. The value you place in the default value is the value shown for all other languages if you do not specify a different string in that language.

When the custom property dialog box opens, it shows the appropriate translated values for the constant string custom properties, as determined by the translation precedence rules. For more information on these rules, see Precedence Rules for Showing the Language and Font.

## Related Topics

[Switching Languages for Graphic Elements](#)

## Support for Empty Strings

You cannot substitute an empty string in the primary locale. Use space characters. When you set an empty string for a primary locale, the empty string is propagated to all other locales that do not have translations.

Performing a first time substitution of an empty string in a secondary locale puts a space character in the primary and the current locale. The remaining locales will match the primary value if they do not already have a specific value.

If a primary locale contains an empty string, it will be exported for translation.

If you substitute an empty string for a secondary locale, the element shows as empty. However, if you switch to the primary locale and then back to the secondary locale, the element shows the primary string substitution again.

## Related Topics

[Switching Languages for Graphic Elements](#)

## Language Switching Example

The following table describes the effects of language switching on the various parts of the system. In this example:

- The application is configured with two languages: English and French.
- The default language is English.
- There are several graphics, some of which contain partial or mismatched language configurations compared to your HMI/SCADA product's configured languages.

Action	Effect on the Languages Configured for the Graphic	Effect on Elements that contain English Translations	Effect on Elements that contain French Translations
You open a new graphic.	English language is added to the graphic.	None	None
You open an existing graphic that only has English defined.	None	None	None
You open an existing graphic that only has the French language defined.	English language is added to the graphic.	French strings are transferred into the English language.	French strings are marked as specific translations for French.
You open an existing graphic that only has the German language defined.	English language is added to the graphic.	German strings are transferred into the English language. German strings are marked as specific translations for German.	None
You change a text string in the English language.	None	New string is set for the English language.	None
You switch to the French language for the first time in a graphic that only had the English language.	French language is added to the graphic.	None	The default language strings are shown unless a specific French translation exists.
You change a string while viewing the French language.	None	None	The new string is set as the specific translation for the French language and used for display.
You create a new text element in a graphic that has English only while viewing the English language.	None	The new element's string value is saved for translation.	None
You delete an element with translations in English and French.	None	English translations are removed.	French translations are removed.
You copy animations from an element.	None	English animation translation strings are put into the clipboard.	French animation translation strings are put into the clipboard.
You paste animations to	None	English animation	French animation

an element.		translation strings are put into the destination animations.	translation strings are put into the destination animations.
You clear animations for an element.	None	English animation strings are removed.	French animation strings are removed.
You copy and paste elements from a German-only graphic into a graphic containing English and French.	None	German strings are placed in the English language.	None
You copy and paste elements from an English and German graphic into a symbol containing English and French.	None	English strings from the source graphic are copied during the paste. The German strings are dropped.	None
You export the French language, having never switched to the French language.	The French language is added to the graphic for the purposes of the export. The language is not saved back to the graphic during the export.	English strings are exported in the "Phrase" XML attribute field.	If specific strings exist for French, they are exported in the "Translation" XML field.
You import the French language.	The French language is added.	None	Any translations provided in the import are marked as specific translations for the French language. If the translation is empty, the default language value is shown.
You convert a graphic to a group.	Same logic as copy/paste.	Same logic as copy/paste.	Same logic as copy/paste.
You delete the German language.	German is removed.	None	None

## Related Topics

[Switching Languages for Graphic Elements](#)

## Overriding Translated Strings for Industrial Graphics

After you embed a graphic into an application, you can override the translations for:

- Strings on the substitutable graphic elements within the Industrial Graphic.

- String type custom properties on the graphic.

To translate these overrides, export and import the strings.

## Related Topics

[Switching Languages for Graphic Elements](#)

### Language Switching at Run Time

At run time, languages can be switched by selecting the language from the **Special** menu. The list of languages shown is based on the languages configured for your HMI/SCADA product.

## Related Topics

[Switching Languages for Graphic Elements](#)

## Windows Common Control List Methods

You can use the methods of the Windows common controls to manipulate the controls at run time by using them in scripting.

The following table contains a list of methods you can use in scripting to:

- Load and save the contents of the Edit Box control from and to a file.
- Manipulate items in the lists of the List Box control and Combo Box control.
- Manipulate items in the lists of the List Box control and Combo Box control.

Method	Purpose, syntax and information
AddItem	<b>Purpose:</b> Add an item (coerced to String) to the list. If the list is sorted, then the new item is inserted at the right position and selected. If the list is unsorted, the item is added to the bottom of the list. <b>Used by:</b> Combo Box, List Box <b>Note:</b> This function does not work when using an Enum or Array to populate the List Box. <b>Syntax:</b> ControlName.AddItem(CaptionString); <b>Info:</b> <a href="#">Adding and Inserting Items into a List</a>
Clear	<b>Purpose:</b> Removes all items from the List. If the list is bound, it clears the bound reference (array or enum). <b>Note:</b> This function does not work when using an Enum or Array to populate the List Box. <b>Used by:</b> Combo Box, List Box

Method	Purpose, syntax and information
	<p><b>Syntax:</b> ControlName.Clear();</p> <p><b>Info:</b> <a href="#">Deleting Items from a List</a></p>
DeleteItem	<p><b>Purpose:</b> Accepts an index as a parameter and removes that item from the list. The first item in the list has an index of 0.</p> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Syntax:</b> ControlName.DeleteItem(Index);</p> <p><b>Info:</b> <a href="#">Deleting Items from a List</a></p>
DeleteSelection	<p><b>Purpose:</b> Delete the currently selected item from the list.</p> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Syntax:</b> ControlName.DeleteSelection();</p> <p><b>Info:</b> <a href="#">Deleting Items from a List</a></p>
FindItem	<p><b>Purpose:</b> Accepts a string as a parameter and returns the index of the first item that matches the string. The first item in the list has an index of 0.</p> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Syntax:</b> ControlName.FindItem(SearchString);</p> <p><b>Info:</b> <a href="#">Finding an Item in a List</a></p>
GetItem	<p><b>Purpose:</b> Returns the item associated with an index supplied as a parameter to this function. The first item in the list has an index of 0.</p> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Syntax:</b> ItemCaption = ControlName.GetItem(Index);</p> <p><b>Info:</b> <a href="#">Reading the Caption of a Selected Item in a List</a></p>
InsertItem	<p><b>Purpose:</b> Inserts the supplied string after the current selection in the List. Does not work if list is sorted.</p> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Syntax:</b> ControlName.InsertItem(String);</p> <p><b>Info:</b> <a href="#">Adding and Inserting Items into a List</a></p>
SetItemData	<p><b>Purpose:</b> Associates a value with an item in the list which index is provided to the function. The first item in the list has an index of 0.</p> <p><b>Note:</b> This function only works when UseValuesAsItems is set to false. It does not work when using an Enum or Array to populate the List Box control.</p>

Method	Purpose, syntax and information
	<p><b>Used by:</b> Combo Box, List Box</p> <p><b>Syntax:</b> ControlName.SetItemData(Index,Value);</p> <p><b>Info:</b> <a href="#">Associating Items with Values in a List</a></p>
GetItemData	<p><b>Purpose:</b> Returns the value associated with the item in the list which index is supplied to the function. The first item in the list has an index of 0.</p> <p><b>Note:</b> This function only works when UseValuesAsItems is set to false. It does not work when using an Enum or Array to populate the List Box control.</p> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Syntax:</b> Value = ControlName.GetItemData(Index);</p> <p><b>Info:</b> <a href="#">Associating Items with Values in a List</a></p>
LoadList	<p><b>Purpose:</b> Loads a list of strings from a file which name is passed as parameter to the function. The default location for files is the users folder, for example: c:\documents and settings\username.</p> <p><b>Note:</b> The LoadList method does not work when using an Enum or Array to populate the List Box control.</p> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Syntax:</b> ControlName.LoadList(FileName);</p> <p><b>Info:</b> <a href="#">Loading and Saving Item Lists</a></p>
LoadText	<p><b>Purpose:</b> Loads a text from a file into the Edit Box control. The default location for files is the users folder, for example: c:\documents and settings\username.</p> <p><b>Used by:</b> Edit Box</p> <p><b>Syntax:</b> ControlName.LoadText(FileName);</p> <p><b>Info:</b> <a href="#">Configuring Edit Box Methods</a></p>
SaveList	<p><b>Purpose:</b> Save a list to a file which name is passed as parameter to the function. The default location for files is the users folder, for example: c:\documents and settings\username.</p> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Syntax:</b> ControlName.SaveList(FileName);</p> <p><b>Info:</b> <a href="#">Loading and Saving Item Lists</a></p>
SaveText	<p><b>Purpose:</b> Saves the current text in the Edit Box control to a file. The default location for files is the users</p>

Method	Purpose, syntax and information
	folder, for example: c:\documents and settings\username. <b>Used by:</b> Edit Box <b>Syntax:</b> ControlName.SaveText(FileName); <b>Info:</b> <a href="#">Configuring Edit Box Methods</a>

## Related Topics

[Windows Common Control List Methods](#)

## QuickScript References

### In This Appendix

[Script Functions](#)

[QuickScript .NET Variables](#)

[QuickScript .NET Control Structures](#)

[QuickScript .NET Operators](#)

## Script Functions

This section describes the script functions available in the HMI/SCADA development environment. The function documentation is organized into a set of folders that represents the same organization of the functions in the Script Function Browser.

Also provided are additional references for standard QuickScript .NET variables, control structures, and operators.

Other Microsoft .NET script functions, are not documented. Refer to Microsoft .NET documentation for descriptions of the functions.

## Related Topics

[QuickScript References](#)

[Graphic Client Functions](#)

[Math Functions](#)

[Miscellaneous Functions](#)

[String Functions](#)

[System Functions](#)

## Graphic Client Functions

Use graphic client functions to hide and show symbols, open and close popup windows, log in and log off users, or to query custom properties contained in a symbol.

## Related Topics

[Script Functions](#)

[GetCPQuality\(\)](#)

[GetCPTimeStamp\(\)](#)

[HideSelf\(\)](#)

# GetCPQuality()

Returns the Quality value of a custom property. This function is available within any Industrial Graphics client script, but may not be supported by your HMI. For more information, consult your HMI documentation.

## Syntax

```
Int GetCPQuality(String name)
```

Where *String name* is the name of the custom property whose quality is to be retrieved.

This script function takes the name of a custom property on the symbol. This argument is of type string and it can be a reference or a constant.

If the custom property is type constant, GOOD is the quality always returned.

---

**Note:** For use with custom properties only. It does not apply to HMI tags.

---

## Return Value

The GetCPQuality() script function returns a value 0-255 of type Integer, as per the OPC quality standard. 192 is GOOD.

Example

```
cp2 = GetCPQuality("cp1");
```

Where cp1 and cp2 are custom properties and the data type of cp2 is Integer.

## Related Topics

[Graphic Client Functions](#)

# GetCPTimeStamp()

Returns the time stamp of a custom property. This function is available within any Industrial Graphics client

script.

## Syntax

```
DateTime GetCPTimeStamp(String name)
```

Where *String name* is the name of the custom property whose time stamp is to be retrieved.

This script function takes the name of a custom property on the symbol. This argument is of type string and it can be a reference or a constant.

**Note:** For use with custom properties only. It does not apply to HMI tags.

## Return Value

The GetCPTimeStamp() script function returns the time stamp of the custom property's current value of type DateTime. If the custom property value is a constant, then the return value is the time the value was created.

Example

```
cp2 = GetCPTimeStamp("cp1");
```

Where cp1 and cp2 are custom properties and the data type of cp2 is DateTime.

## Related Topics

[Graphic Client Functions](#)

# HideSelf()

Closes the displayed graphic or layout for which this script is configured. This script function is available within any Industrial Graphics client script.

## Category

Graphic Client

## Syntax

```
HideSelf();
```

## Remarks

For an Industrial Graphics script, call the script function within the symbol to hide the popup.

## Example

```
HideSelf();
```

## Related Topics

[Graphic Client Functions](#)

### Math Functions

Use math functions to return the answer to the specified mathematical expression.

In QuickScript, all mathematical operations are calculated internally as double, regardless of the operand data type. Following standard mathematical rules, the result is always rounded in division operations to maintain accuracy. Rounding only occurs on the end result, not intermediate values, and the quotient will match the target data type. This is the standard methodology for SCADA and DCS systems, and provides the data integrity, precision retention, time stamps, and overall data quality propagation and aggregation needed for these systems.

If you want to round at each step instead of only at the final result, you can leverage the support built into QuickScript for .NET libraries and utilize the System.Math.Floor and System.Math.Round methods to explicitly round the intermediate steps. As an example, consider the following script:

```
dim dividend as integer;
dim divisor as integer;
dim quotient as integer;
dim remainder as integer;
dividend = 8;
divisor = 3;
LogMessage("Value of dividend = " + dividend);
LogMessage("Value of divisor = " + divisor);
    quotient = dividend/divisor;
LogMessage("Value of quotient = " + quotient);
remainder = dividend mod divisor;
LogMessage ("Value of remainder = " + remainder);
dividend = divisor*quotient +remainder;
LogMessage ("Value of dividend = " + dividend);
```

The result is: 8 / 3 = 3

If, instead, you want to drop the remainder (not rounding the final result to the nearest integer), you could add a call to the Math.Floor method and use the following:

```
dim dividend as integer;
dim divisor as integer;
dim quotient as integer;
dim remainder as integer;
dividend = 8;
divisor = 3;
LogMessage("Value of dividend = " + dividend);
LogMessage("Value of divisor = " + divisor);
// *** Add call to Math.Floor. This drops the remainder rather than rounding the internal
Double result to integer
    quotient = System.Math.Floor(dividend/divisor);
LogMessage("Value of quotient = " + quotient);
remainder = dividend mod divisor;
LogMessage ("Value of remainder = " + remainder);
dividend = divisor*quotient +remainder;
LogMessage ("Value of dividend = " + dividend);
```

The result is: 8 / 3 = 2 (remainder 2)

## Related Topics

[Script Functions](#)

[Abs\(\)\(\)](#)

[ArcCos\(\)\(\)](#)

[ArcSin\(\)\(\)](#)

[ArcTan\(\)\(\)](#)

[Cos\(\)\(\)](#)

[Exp\(\)\(\)](#)

[Int\(\)\(\)](#)

[Log\(\)\(\)](#)

[Log10\(\)\(\)](#)

[LogN\(\)\(\)](#)

[Pi\(\)\(\)](#)

[Round\(\)\(\)](#)

[Sgn\(\)\(\)](#)

[Sin\(\)\(\)](#)

[Sqrt\(\)\(\)](#)

[Tan\(\)\(\)](#)

[Trunc\(\)\(\)](#)

# Abs()

Returns the absolute value (unsigned equivalent) of a specified number.

## Category

Math

## Syntax

```
Result = Abs( Number );
```

## Parameter

*Number*

Any number or numeric equipment.item.

## Examples

```
Abs(14); ' returns 14
```

```
Abs(-7.5); ' returns 7.5
```

## Related Topics

[Math Functions](#)

# ArcCos()

Returns an angle between 0 and 180 degrees whose cosine is equal to the number specified.

## Category

Math

## Syntax

```
Result = ArcCos( Number );
```

## Parameter

*Number*

Any number or numeric equipment.item with a value between -1 and 1 (inclusive).

## Examples

```
ArcCos(1); ' returns 0
ArcCos(-1); ' returns 180
```

## See Also

[Cos\(\)](#), [Sin\(\)](#), [Tan\(\)](#), [ArcSin\(\)](#), [ArcTan\(\)](#)

## Related Topics

[Math Functions](#)

# ArcSin()

Returns an angle between -90 and 90 degrees whose sine is equal to the number specified.

## Category

Math

## Syntax

```
Result = ArcSin( Number );
```

## Parameter

*Number*

Any number or numeric equipment.item with a value between -1 and 1 (inclusive).

## Examples

```
ArcSin(1); ' returns 90  
ArcSin(-1); ' returns -90
```

## See Also

[Cos\(\)](#), [Sin\(\)](#), [Tan\(\)](#), [ArcCos\(\)](#), [ArcTan\(\)](#)

## Related Topics

[Math Functions](#)

# ArcTan()

Returns an angle between -90 and 90 degrees whose tangent is equal to the number specified.

## Category

Math

## Syntax

```
Result = ArcTan( Number );
```

## Parameter

*Number*

Any number or numeric equipment.item.

## Examples

```
ArcTan(1); ' returns 45  
ArcTan(0); ' returns 0
```

## See Also

[Cos\(\)](#), [Sin\(\)](#), [Tan\(\)](#), [ArcCos\(\)](#), [ArcSin\(\)](#)

## Related Topics

[Math Functions](#)

# Cos()

Returns the cosine of an angle in degrees.

## Category

Math

## Syntax

```
Result = Cos( Number );
```

## Parameter

*Number*

Any number or numeric equipment.item.

## Examples

```
Cos(90); ' returns 0  
Cos(0); ' returns 1
```

This example shows how to use the function in a math equation:

```
Wave = 50 * Cos(6 * Now().Second);
```

## See Also

[Sin\(\)](#), [Tan\(\)](#), [ArcCos\(\)](#), [ArcSin\(\)](#), [ArcTan\(\)](#)

## Related Topics

[Math Functions](#)

# Exp()

Returns the result of the exponent  $e$  raised to a power.

## Category

Math

## Syntax

```
Result = Exp( Number );
```

## Parameter

*Number*

Any number or numeric equipment.item.

## Example

```
Exp(1); ' returns 2.718...
```

## Related Topics

[Math Functions](#)

# Int()

Returns the next integer less than or equal to a specified number.

## Category

Math

## Syntax

```
IntegerResult = Int( Number );
```

## Parameter

*Number*

Any number or numeric equipment.item.

## Remarks

When handling negative real (float) numbers, this function returns the integer farthest from zero.

## Examples

```
Int(4.7); ' returns 4
Int(-4.7); ' returns -5
```

## Related Topics

[Math Functions](#)

# Log()

Returns the natural log (base e) of a number.

## Category

Math

## Syntax

```
RealResult = Log( Number );
```

## Parameter

*Number*

Any number or numeric equipment.item.

## Remarks

Natural log of 0 is undefined.

## Examples

```
Log(100); ' returns 4.605...
Log(1); ' returns 0
```

## See Also

[LogN\(\)](#), [Log10\(\)](#)

## Related Topics

[Math Functions](#)

# Log10()

Returns the base 10 log of a number.

## Category

Math

## Syntax

```
Result = Log10( Number );
```

## Parameter

*Number*

Any number or numeric equipment.item.

## Example

```
Log10(100); ' returns 2
```

## See Also

[Log\(\)](#), [LogN\(\)](#)

## Related Topics

[Math Functions](#)

# LogN()

Returns the values of the logarithm of x to base n.

## Category

Math

## Syntax

```
Result = LogN( Number, Base );
```

## Parameters

### Number

Any number or numeric equipment.item.

### Base

Integer to set log base. You could also specify an integer equipment.item.

## Remarks

Base 1 is undefined.

## Examples

```
LogN(8, 3); ' returns 1.89279  
LogN(3, 7); ' returns 0.564
```

## See Also

[Log\(\)](#), [Log10\(\)](#)

## Related Topics

[Math Functions](#)

# Pi()

Returns the value of Pi.

## Category

Math

## Syntax

```
RealResult = Pi();
```

## Example

```
Pi(); ' returns 3.1415926
```

## Related Topics

[Math Functions](#)

# Round()

Rounds a real number to a specified precision and returns the result.

## Category

Math

## Syntax

```
RealResult = Round( Number, Precision );
```

## Parameters

*Number*

Any number or numeric equipment.item.

*Precision*

Sets the precision to which the number is rounded. This value can be any number or a numeric equipment.item.

## Examples

```
Round(4.3, 1); ' returns 4
Round(4.3, .01); ' returns 4.30
Round(4.5, 1); ' returns 5
Round(-4.5, 1); ' returns -4
Round(106, 5); ' returns 105
Round(43.7, .5); ' returns 43.5
```

## See Also

[Trunc\(\)](#)

## Related Topics

[Math Functions](#)

# Sgn()

Determines the sign of a value (whether it is positive, zero, or negative) and returns the result.

## Category

Math

## Syntax

```
IntegerResult = Sgn( Number );
```

## Parameter

*Number*

Any number or numeric equipment.item.

## Return Value

If the input number is positive, the result is 1. Negative numbers return a -1, and 0 returns a 0.

## Examples

```
Sgn(425); ' returns 1;  
Sgn(0); ' returns 0;  
Sgn(-37.3); ' returns -1;
```

## Related Topics

[Math Functions](#)

# Sin()

Returns the sine of an angle in degrees.

## Category

Math

## Syntax

```
Result = Sin( Number );
```

## Parameter

*Number*

Angle in degrees. Any number or numeric equipment.item.

## Examples

```
Sin(90); ' returns 1;  
Sin(0); ' returns 0;
```

This example shows how to use the function in a math expression:

```
wave = 100 * Sin (6 * Now().Second);
```

## See Also

[Cos\(\)](#), [Tan\(\)](#), [ArcCos\(\)](#), [ArcSin\(\)](#), [ArcTan\(\)](#)

## Related Topics

[Math Functions](#)

# Sqrt()

Returns the square root of a number.

## Category

Math

## Syntax

```
RealResult = Sqrt( Number );
```

## Parameter

*Number*

Any number or numeric equipment.item.

## Example

This example takes the value of me.PV and returns the square root as the value of x:

```
x=Sqrt(me.PV);
```

## Related Topics

[Math Functions](#)

# Tan()

Returns the tangent of an angle given in degrees.

## Category

Math

## Syntax

```
Result = Tan( Number );
```

## Parameter

*Number*

The angle in degrees. Any number or numeric equipment.item.

## Examples

```
Tan(45); ' returns 1;  
Tan(0); ' returns 0;
```

This example shows how to use the function in a math expression:

```
Wave = 10 + 50 * Tan(6 * Now().Second);
```

## See Also

[Cos\(\)](#), [Sin\(\)](#), [ArcCos\(\)](#), [ArcSin\(\)](#), [ArcTan\(\)](#)

## Related Topics

[Math Functions](#)

# Trunc()

Truncates a real (floating point) number by simply eliminating the portion to the right of the decimal point, including the decimal point, and returns the result.

## Category

Math

## Syntax

```
NumericResult = Trunc( Number );
```

## Parameter

*Number*

Any number or numeric equipment.item.

## Remarks

This function accomplishes the same result as placing the contents of a float type equipment.item into an integer type equipment.item.

## Examples

```
Trunc(4.3); ' returns 4;  
Trunc(-4.3); ' returns -4;
```

## See Also

[Round\(\)](#)

## Related Topics

[Math Functions](#)

## Miscellaneous Functions

Functions in the miscellaneous group perform a variety of purposes, such as logging data or querying equipment.items.

## Related Topics

[Script Functions](#)

[DateTimeGMT\(\)](#)

[IsBad\(\)](#)

[IsGood\(\)](#)

[IsInitializing\(\)](#)

[IsUncertain\(\)](#)

[IsUsable\(\)](#)

[LogDataChangeEvent\(\)](#)

[LogError\(\)](#)

[LogMessage\(\)](#)[.LogWarning\(\)](#)

## DateTimeGMT()

Returns a number representing the number of days and fractions of days since January 1, 1970, in Coordinated Universal Time (UTC), regardless of the local time zone.

### Category

Miscellaneous

### Syntax

```
Result=DateTimeGMT();
```

### Parameters

None

### Example

```
MessageTag = StringFromTime(DateTimeGMT() * 86400.0, 3);
```

### Related Topics

[Miscellaneous Functions](#)

## IsBad()

Returns a Boolean value indicating if the quality of the specified equipment.item is Bad.

### Category

Miscellaneous

### Syntax

```
BooleanResult = IsBad( equipmentitem1, equipmentitem2, ... );
```

### Parameter(s)

*equipment.item1, equipment.item2, ...equipment.itemN*

Names of one or more equipment.items for which you want to determine Bad quality. You can include a variable-length list of equipment.items.

## Return Value

If any of the specified equipment.items has Bad quality, then true is returned. Otherwise, false is returned.

## Examples

```
IsBad(TIC101.PV);  
IsBad(TIC101.PV, PIC102.PV);
```

## See Also

[IsGood\(\)](#), [IsInitializing\(\)](#), [IsUncertain\(\)](#), [IsUsable\(\)](#)

## Related Topics

[Miscellaneous Functions](#)

# IsGood()

Returns a Boolean value indicating if the quality of the specified equipment.item is Good.

## Category

Miscellaneous

## Syntax

```
BooleanResult = IsGood( equipment.item1, equipment.item2, ... );
```

## Parameter(s)

*equipment.item1, equipment.item2, and so on*

Name of the equipment.item(s) for which you want to determine Good quality. You can include a variable-length list of equipment.items.

## Return Value

If all of the specified equipment.items have Good quality, then true is returned. Otherwise, false is returned.

## Examples

```
IsGood(TIC101.PV);  
IsGood(TIC101.PV, PIC102.PV);
```

## See Also

[IsBad\(\)](#), [IsInitializing\(\)](#), [IsUsable\(\)](#)

## Related Topics

[Miscellaneous Functions](#)

# IsInitializing()

Returns a Boolean value indicating if the quality of the specified equipment.item is Initializing.

## Category

Miscellaneous

## Syntax

```
BooleanResult = IsInitializing( equipment.item1, equipment.item2, ... );
```

## Parameter(s)

*equipment.item1, equipment.item2, and so on*

Name of the equipment.item(s) for which to determine Initializing quality. You can include a variable-length list of equipment.items.

## Return Value

If any of the specified equipment.items has Initializing quality, then true is returned. Otherwise, false is returned.

## Examples

```
IsInitializing(TIC101.PV);  
IsInitializing(TIC101.PV, PIC102.PV);
```

## See Also

[IsBad\(\)](#), [IsGood\(\)](#), [IsUncertain\(\)](#), [IsUsable\(\)](#)

## Related Topics

[Miscellaneous Functions](#)

# IsUncertain()

Returns a Boolean value indicating if the quality of the specified equipment.item is Uncertain.

## Category

Miscellaneous

## Syntax

```
BooleanResult = IsUncertain( equipment.item1, equipment.item2, ... );
```

## Parameter(s)

*equipment.item1, equipment.item2, and so on*

Name of the equipment.item(s) to determine Uncertain quality. You can include a variable-length list of equipment.items.

## Return Value

If all of the specified equipment.items have Uncertain quality, then true is returned. Otherwise, false is returned.

## Examples

```
IsUncertain(TIC101.PV);  
IsUncertain(TIC101.PV, PIC102.PV);
```

## See Also

[IsBad\(\)](#), [IsGood\(\)](#), [IsInitializing\(\)](#), [IsUsable\(\)](#)

## Related Topics

[Miscellaneous Functions](#)

# IsUsable()

Returns a Boolean value indicating if the specified equipment.item is usable for calculations.

## Category

Miscellaneous

## Syntax

```
BooleanResult = IsUsable( equipment.item1, equipment.item2, ... );
```

## Parameter(s)

*equipment.item1, equipment.item2, ...equipment.itemN*

Name of one or more equipment.items for which you want to determine unusable quality. You can include a variable-length list of equipment.items.

## Return Value

If all of the specified equipment.items have either Good or Uncertain quality, then true is returned. Otherwise, false is returned.

## Remarks

The attributes having Good or Uncertain quality qualifies as usable. In addition, each float or double equipment.item cannot be a NaN (not a number).

## Examples

```
IsUsable(TIC101.PV);  
IsUsable(TIC101.PV, PIC102.PV);
```

## See Also

[IsBad\(\)](#), [IsGood\(\)](#), [IsInitializing\(\)](#), [IsUncertain\(\)](#)

## Related Topics

[Miscellaneous Functions](#)

# LogCustom()

Writes a user-defined custom flag message in the [Log Viewer](#).

## Category

Miscellaneous

## Syntax

```
LogCustom( CustomFlag, msg );
```

## Parameter

*CustomFlag*

Creates a new log flag based on the first parameter string. The first call creates the custom flag.

*msg*

The message to write to the Log Viewer. Actual string or a string equipment.item.

## Remarks

The log flag is disabled by default.

The message is always logged under the component "ObjectName.ScriptName". For example, "WinPlatform\_001.script1: msg", which identifies what object and what script within the object logged the error.

LogCustom() is similar to LogMessage(), but displays the message in the custom log flag when Log Custom is enabled.

The parameter help tooltip and Function Browser sample parameter list will show "LogCustom( CustomFlag, msg )" rather than "LogCustom( CustomFlag, Message )". "Message" is a reserved keyword.

## Example

```
LogCustom(EditBox1.text, "User-defined message.");
```

This statement writes to the Log Viewer as follows:

```
10/24/2005 12:49:14 PM ScriptRuntime
<ObjectName.ScriptName>: <LogFlag EditBox1> User-defined message.
```

## Related Topics

[Miscellaneous Functions](#)

# LogDataChangeEvent()

Logs an application change event to the application Historian.

---

**Note:** The LogDataChangeEvent() function works only in object scripts, not in symbol scripts.

---

## Category

Miscellaneous

## Syntax

```
LogDataChangeEvent(EquipmentitemName, Description, OldValue, NewValue, TimeStamp);
```

## Parameters

*EquipmentitemName*

Equipment.item name as a tag name.

*Description*

Description of the object.

*OldValue*

Old value of the equipment.item.

*NewValue*

New value of the equipment.item.

*TimeStamp*

The time stamp associated with the logged event. The timestamp can be UTC or local time. The TimeStamp parameter is optional. The timestamp of the logged event defaults to Now() if a TimeStamp parameter is not included.

## Remarks

A symbol script still compiles if the LogDataChangeEvent() function is included. However, a warning message is written to the log at run time that the function is inoperable.

## Example

This example logs an event when a pump starts or stops with a timestamp of the current time when the event occurred.

```
LogDataChangeEvent(TC104.pumpstate, "Pump04", OldState, NewState);
```

## Related Topics

[Miscellaneous Functions](#)

# .LogError()

Writes a user-defined error message in the [Log Viewer](#) with a red error log flag.

## Category

Miscellaneous

## Syntax

```
.LogError( msg );
```

## Parameter

*msg*

The message to write to the Log Viewer. Actual string or a string equipment.item.

## Remarks

The log flag is enabled by default.

The message is always logged under the component "ObjectName.ScriptName". For example, "WinPlatform\_001.script1: msg", which identifies what object and what script within the object logged the error.

`.LogError()` is similar to `LogMessage()`, but displays the message in red.

The parameter help tooltip and Function Browser sample parameter list will show "LogError( msg )" rather than "LogError( Message )". "Message" is a reserved keyword.

## Example

```
.LogError("User-defined error message.");
```

This statement writes to the Log Viewer as follows:

```
10/24/2005 12:49:14 PM ScriptRuntime  
<ObjectName.ScriptName>: User-defined error message.
```

## Related Topics

[Miscellaneous Functions](#)

# LogMessage()

Writes a user-defined message to the [Log Viewer](#).

## Category

Miscellaneous

## Syntax

```
LogMessage( msg );
```

## Parameter

*msg*

The message to write to the Log Viewer. Actual string or a string equipment.item.

## Remarks

This is a very powerful function for troubleshooting scripting. By strategically placing LogMessage() functions in your scripts, you can determine the order of script execution, performance of scripts, and identify the value of equipment.items both before they are changed and after they are affected by the script.

Each message posted to the Log Viewer is stamped with the exact date and time. The message always begins with the component "Tagname.ScriptName" so you can tell what object and what script within the object posted the message to the log.

## Examples

```
LogMessage("Report Script is Running");
```

The above statement writes the following to the Log Viewer:

```
10/24/2005 12:49:14 PM ScriptRuntime <Tagname.ScriptName>:Report Script is Running.  
MyTag=MyTag + 10;  
LogMessage("The Value of MyTag is " + Text(MyTag, "#"));
```

## Related Topics

[Miscellaneous Functions](#)

# LogTrace()

Writes a user-defined trace message in the [Log Viewer](#).

## Category

Miscellaneous

## Syntax

```
LogTrace( msg );
```

## Parameter

*msg*

The message to write to the Log Viewer. Actual string or a string equipment.item.

## Remarks

The log flag is disabled by default.

The message is always logged under the component "ObjectName.ScriptName". For example, "WinPlatform\_001.script1: msg", which identifies what object and what script within the object logged the error.

LogTrace() is similar to LogMessage(), but displays the message as Trace when Log Trace is enabled.

The parameter help tooltip and Function Browser sample parameter list will show "LogTrace( msg )" rather than "LogTrace( Message )". "Message" is a reserved keyword.

## Example

```
LogTrace("User-defined trace message.");
```

This statement writes to the Log Viewer as follows:

```
10/24/2005 12:49:14 PM ScriptRuntime
<ObjectName.ScriptName>: User-defined trace message.
```

## Related Topics

[Miscellaneous Functions](#)

# LogWarning()

Writes a user-defined error message in the [Log Viewer](#) with a yellow warning log flag.

## Category

Miscellaneous

## Syntax

```
LogWarning( msg );
```

## Parameter

*msg*

The message to write to the Log Viewer. Actual string or a string equipment.item.

## Remarks

The log flag is disabled by default.

The message is always logged under the component "ObjectName.ScriptName". For example, "WinPlatform\_001.script1: msg", which identifies what object and what script within the object logged the error.

LogWarning() is similar to LogMessage(), but displays the message as a yellow warning message.

The parameter help tooltip and Function Browser sample parameter list will show "LogWarning( msg )" rather than "LogWarning( Message )". "Message" is a reserved keyword.

## Example

```
LogWarning("User-defined warning message.")
```

This statement writes to the Log Viewer as follows:

```
10/24/2005 12:49:14 PM ScriptRuntime
<ObjectName.ScriptName>: User-defined warning message.
```

## Related Topics

[Miscellaneous Functions](#)

### String Functions

Use string functions to work with character strings and string values.

## Related Topics

[Script Functions](#)

[DText\(\)](#)

[StringASCII\(\)](#)

[StringChar\(\)](#)

[StringCompareNoCase\(\)](#)

[StringFromGMTTimeToLocal\(\)](#)

[StringFromIntg\(\)](#)

[StringFromReal\(\)](#)

[StringFromTime\(\)](#)

[StringFromTimeLocal\(\)](#)

[StringInString\(\)](#)

[StringLeft\(\)](#)

[StringLen\(\)](#)

[StringLower\(\)](#)

[StringMid\(\)](#)

[StringReplace\(\)](#)

[StringRight\(\)](#)

[StringSpace\(\)](#)

[StringTest\(\)](#)

[StringToIntg\(\)](#)

[StringToReal\(\)](#)

[StringTrim\(\)](#)  
[StringUpper\(\)](#)  
[Text\(\)](#)

## DText()

Returns one of two possible strings, depending on the value of the *Discrete* parameter.

### Category

String

### Syntax

```
StringResult = DText( Discrete, OnMsg, OffMsg );
```

### Parameters

*Discrete*

A Boolean value or Boolean equipment.item.

*OnMsg*

The message that is shown when the value of *Discrete* equals true.

*OffMsg*

The message shown when *Discrete* equals false.

### Example

```
StringResult = DText(me.temp > 150, "Too hot", "Just right");
```

### Related Topics

[String Functions](#)

## StringASCII()

Returns the ASCII value of the first character in a specified string.

### Category

String

## Syntax

```
IntegerResult = StringASCII( Char );
```

## Parameter

*Char*

Alphanumeric character or string or string equipment.item.

## Remarks

When this function is processed, only the single character is tested or affected. If the string provided to StringASCII contains more than one character, only the first character of the string is tested.

## Examples

```
StringASCII("A"); ' returns 65;  
StringASCII("A Mixer is Running"); ' returns 65;  
StringASCII("a mixer is running"); ' returns 97;
```

## See Also

[StringChar\(\)](#), [StringFromIntg\(\)](#), [StringFromReal\(\)](#), [StringFromTime\(\)](#), [StringInString\(\)](#), [StringLeft\(\)](#), [StringLen\(\)](#),  
[StringLower\(\)](#), [StringMid\(\)](#), [StringReplace\(\)](#), [StringSpace\(\)](#), [StringTest\(\)](#), [StringToIntg\(\)](#), [StringToReal\(\)](#), [StringTrim\(\)](#),  
[StringUpper\(\)](#), [Text\(\)](#)

## Related Topics

[String Functions](#)

# StringChar()

Returns the character corresponding to a specified ASCII code.

## Category

String

## Syntax

```
StringResult = StringChar( ASCII );
```

## Parameter

*ASCII*

ASCII code or an integer equipment.item.

## Remarks

Use the StringChar function to add ASCII characters not normally represented on the keyboard to a string equipment.item.

This function is also useful for SQL commands. The where expression sometimes requires double quotation marks around string values, so use StringChar(34).

## Example

In this example, a [Carriage Return (13)] and [Line Feed (10)] are added to the end of StringAttribute and passed to ControlString. Inserting characters out of the normal 32-126 range of displayable ASCII characters can be very useful for creating control codes for external devices such as printers or modems.

```
ControlString = StringAttribute+StringChar(13)+StringChar(10);
```

## Related Topics

[String Functions](#)

# StringCompare()

Compares a string value with another string.

## Category

String

## Syntax

```
StringCompare( Text1, Text2 );
```

## Parameters

*Text1*

First string in the comparison.

*Text2*

Second string in the comparison.

## Return Value

The return value is zero if the strings are identical, -1 if Text1's value is less than Text2, or 1 if Text1's value is greater than Text2.

## Example

```
Result = StringCompare ("Text1","Text2"); (or)
Result = StringCompare (MText1,MText2);
Where Result is an Integer or Real tag and MText1 and MText2 are Memory Message tags.
```

## See Also

[StringASCII\(\)](#), [StringChar\(\)](#), [StringFromIntg\(\)](#), [StringFromReal\(\)](#), [StringFromTime\(\)](#), [StringFromTimeLocal\(\)](#),  
[StringInString\(\)](#), [StringLeft\(\)](#), [StringLen\(\)](#), [StringLower\(\)](#), [StringMid\(\)](#), [StringReplace\(\)](#), [StringRight\(\)](#), [StringSpace\(\)](#),  
[StringTest\(\)](#), [StringToIntg\(\)](#), [StringToReal\(\)](#), [StringTrim\(\)](#), [StringUpper\(\)](#), [Text\(\)](#)

## Related Topics

[String Functions](#)

# StringCompareNoCase()

C.compares a string value with another string and ignores the case.

## Category

String

## Syntax

```
SStringCompareNoCase( Text1, Text2 );
```

## Parameters

*Text1*

First string in the comparison.

*Text2*

Second string in the comparison.

## Return Value

The return value is zero if the strings are identical (ignoring case), -1 if Text1's value is less than Text2 (ignoring case), or 1 if Text1's value is greater than Text2 (ignoring case).

## Example

```
Result = StringCompareNoCase ("Text1","TEXT1"); (or)
Result = StringCompareNoCase (MText1,MText2);
Where Result is an Integer or Real tag and MText1 and MText2 are Memory Message tags.
```

## See Also

[StringASCII\(\)](#), [StringChar\(\)](#), [StringFromIntg\(\)](#), [StringFromReal\(\)](#), [StringFromTime\(\)](#), [StringInString\(\)](#), [StringLeft\(\)](#), [StringLen\(\)](#), [StringLower\(\)](#), [StringMid\(\)](#), [StringReplace\(\)](#), [StringSpace\(\)](#), [StringTest\(\)](#), [StringToIntg\(\)](#), [StringToReal\(\)](#), [StringTrim\(\)](#), [StringUpper\(\)](#), [Text\(\)](#)

## Related Topics

[String Functions](#)

# StringFromGMTTimeToLocal()

Converts a time value (in seconds since Jan-01-1970) to a particular string representation. This is the same as [StringFromTime\(\)\(\)](#).

## Category

String

## Syntax

```
MessageResult=StringFromGMTTimeToLocal(SecsSince1-1-70,StringType);
```

## Parameters

*SecsSince1-1-70*

Is converted to the *StringType* specified and the result is stored in *MessageResult*.

*StringType*

Determines the display method:

1 = Displays the date in the same format set from the windows control Panel. (Similar to that displayed for \$DateString.)

2 = Displays the time in the same format set from the Windows control Panel. (Similar to that displayed for \$TimeString.)

3 = Displays a 24-character string indicating both the date and time: "Wed Jan 02 02:03:55 1993"

4 = Displays the short form for the day of the week: "Wed"

5 = Displays the long form for the day of the week: "Wednesday"

## Remarks

Any adjustments necessary due to Daylight Savings Time are automatically applied to the return result. Therefore, it is not necessary to make any manual adjustments to the input value to convert to DST.

## Example

This example assumes that the time zone on the local node is Pacific Standard Time (UTC-0800). The UTC time passed to the function is 12:00:00 AM on Friday, 1/2/1970. Since PST is 8 hours behind UTC, the function will return the following results:

```
StringFromGMTTimeToLocal(86400, 1); ' returns "1/1/1970"  
StringFromGMTTimeToLocal(86400, 2); ' returns "04:00:00 PM"  
StringFromGMTTimeToLocal(86400, 3); ' returns "Thu Jan 01 16:00:00 1970"  
StringFromGMTTimeToLocal(86400, 4); ' returns "Thu"  
StringFromGMTTimeToLocal(86400, 5); ' returns "Thursday"
```

## See Also

[StringASCII\(\)](#), [StringChar\(\)](#), [StringFromIntg\(\)](#), [StringFromReal\(\)](#), [StringFromTime\(\)](#), [StringInString\(\)](#), [StringLeft\(\)](#), [StringLen\(\)](#), [StringLower\(\)](#), [StringMid\(\)](#), [StringReplace\(\)](#), [StringSpace\(\)](#), [StringTest\(\)](#), [StringToIntg\(\)](#), [StringToReal\(\)](#), [StringTrim\(\)](#), [StringUpper\(\)](#), [Text\(\)](#)

## Related Topics

[String Functions](#)

# StringFromIntg()

Converts an integer value into its string representation in another base and returns the result.

## Category

String

## Syntax

```
SringResult = StringFromIntg( Number, numberBase );
```

## Parameters

*Number*

Number to convert. Any number or an integer equipment.item.

*numberBase*

Base to use in conversion. Any number or an integer equipment.item.

## Examples

```
StringFromIntg(26, 2); ' returns "11010"  
StringFromIntg(26, 8); ' returns "32"  
StringFromIntg(26, 16); ' returns "1A"
```

## See Also

[StringASCII\(\)](#), [StringChar\(\)](#), [StringFromReal\(\)](#), [StringFromTime\(\)](#), [StringInString\(\)](#), [StringLeft\(\)](#), [StringLen\(\)](#),  
[StringLower\(\)](#), [StringMid\(\)](#), [StringReplace\(\)](#), [StringSpace\(\)](#), [StringTest\(\)](#), [StringToIntg\(\)](#), [StringToReal\(\)](#), [StringTrim\(\)](#),  
[StringUpper\(\)](#), [Text\(\)](#)

## Related Topics

[String Functions](#)

# StringFromReal()

Converts a real value into its string representation, either as a floating-point number or in exponential notation, and returns the result.

## Category

String

## Syntax

```
StringResult = StringFromReal( Number, Precision, Type );
```

## Parameters

*Number*

Converted to the Precision and Type specified. Any number or a float equipment.item.

*Precision*

Specifies how many decimal places is shown. Any number or an integer equipment.item.

*Type*

A string value that determines the display method. Possible values are:

f = Display in floating-point notation.

e = Display in exponential notation with a lowercase "e."

E = Display in exponential notation with an uppercase "E" followed by a plus sign and at least three exponential digits.

## Examples

```
StringFromReal(263.355, 2,"f"); ' returns "263.36";
StringFromReal(263.355, 2,"e"); ' returns "2.63e2";
StringFromReal(263.355, 2,"E"); ' returns "2.63 E+002";
```

## See Also

[StringASCII\(\)](#), [StringChar\(\)](#), [StringFromIntg\(\)](#), [StringFromTime\(\)](#), [StringInString\(\)](#), [StringLeft\(\)](#), [StringLen\(\)](#),  
[StringLower\(\)](#), [StringMid\(\)](#), [StringReplace\(\)](#), [StringSpace\(\)](#), [StringTest\(\)](#), [StringToIntg\(\)](#), [StringToReal\(\)](#), [StringTrim\(\)](#),  
[StringUpper\(\)](#), [Text\(\)](#)

## Related Topics

[String Functions](#)

# StringFromTime()

Converts a time value (in seconds since January 1, 1970) into a particular string representation and returns the result.

## Category

String

## Syntax

```
StringResult = StringFromTime( SecsSince1-1-70, StringType );
```

## Parameters

*SecsSince1-1-70*

Converted to the *StringType* specified.

*StringType*

Determines the display method. Possible values are:

1 = Shows the date in the same format set from the Windows Control Panel.

2 = Shows the time in the same format set from the Windows Control Panel.

3 = Shows a 24-character string indicating both the date and time: "Wed Jan 02 02:03:55 1993"

4 = Shows the short form for a day of the week: "Wed"

5 = Shows the long form for a day of the week: "Wednesday"

## Remarks

The time value is UTC equivalent: number of elapsed seconds since January 1, 1970 GMT. The value returned reflects the local time.

## Examples

```
StringFromTime(86400, 1); ' returns "1/2/1970"  
StringFromTime(86400, 2); ' returns "12:00:00 AM"  
StringFromTime(86400, 3); ' returns "Fri Jan 02 00:00:00 1970"  
StringFromTime(86400, 4); ' returns "Fri"  
StringFromTime(86400, 5); ' returns "Friday"
```

## See Also

[StringASCII\(\)](#), [StringChar\(\)](#), [StringFromIntg\(\)](#), [StringFromReal\(\)](#), [StringInString\(\)](#), [StringLeft\(\)](#), [StringLen\(\)](#),  
[StringLower\(\)](#), [StringMid\(\)](#), [StringReplace\(\)](#), [StringSpace\(\)](#), [StringTest\(\)](#), [StringToIntg\(\)](#), [StringToReal\(\)](#), [StringTrim\(\)](#),  
[StringUpper\(\)](#), [Text\(\)](#)

## Related Topics

[String Functions](#)

# StringFromTimeLocal()

Converts a time value (in seconds since Jan-01-1970) into a particular string representation. The value returned also represents local time.

## Category

String

## Syntax

```
MessageResult=StringFromTimeLocal(SecsSince1-1-70, StringType);
```

## Parameters

*SecsSince1-1-70*

Is converted to the *StringType* specified and the result is stored in *MessageResult*.

*StringType*

Determines the display method:

1 = Displays the date in the same format set from the windows control Panel. (Similar to that displayed for \$DateString.)

2 = Displays the time in the same format set from the Windows control Panel. (Similar to that displayed for \$TimeString.)

3 = Displays a 24-character string indicating both the date and time: "Wed Jan 02 02:03:55 1993"

4 = Displays the short form for the day of the week: "Wed"

5 = Displays the long form for the day of the week: "Wednesday"

## Remarks

Any adjustments necessary due to Daylight Savings Time will automatically be applied to the return result. Therefore, it is not necessary to make any manual adjustments for DST to the input value.

## Example

```
StringFromTimeLocal (86400, 1); ' returns "1/2/1970"  
StringFromTimeLocal (86400, 2); ' returns "12:00:00 AM"  
StringFromTimeLocal (86400, 3); ' returns "Fri Jan 02 00:00:00 1970"  
StringFromTimeLocal (86400, 4); ' returns "Fri"  
StringFromTimeLocal (86400, 5); ' returns "Friday"
```

## See Also

[StringASCII\(\)](#), [StringChar\(\)](#), [StringFromIntg\(\)](#), [StringFromReal\(\)](#), [StringFromTime\(\)](#), [StringInString\(\)](#), [StringLeft\(\)](#), [StringLen\(\)](#), [StringLower\(\)](#), [StringMid\(\)](#), [StringReplace\(\)](#), [StringSpace\(\)](#), [StringTest\(\)](#), [StringToIntg\(\)](#), [StringToReal\(\)](#), [StringTrim\(\)](#), [StringUpper\(\)](#), [Text\(\)](#)

## Related Topics

[String Functions](#)

# StringInString()

Returns the position in a string of text where a specified string first occurs.

## Category

String

## Syntax

*IntegerResult* = StringInString( *Text*, *SearchFor*, *StartPos*, *CaseSens* );

## Parameters

*Text*

The string that is searched. Actual string or a string equipment.item.

#### *SearchFor*

The string to be searched for. Actual string or a string equipment.item.

#### *StartPos*

Determines the position in the text where the search begins. Any number or an integer equipment.item.

#### *CaseSens*

Determines whether the search is case-sensitive.

0 = Not case-sensitive

1 = Case-sensitive

Any number or an integer equipment.item.

## Remarks

If multiple occurrences of *SearchFor* are found, the location of the first is returned.

## Examples

```
StringInString("The mixer is running", "mix", 1, 0) ' returns 5;
StringInString("Today is Thursday", "day", 1, 0) ' returns 3;
StringInString("Today is Thursday", "day", 10, 0) ' returns 15;
StringInString("Today is Veteran's Day", "Day", 1, 1) ' returns 20;
StringInString("Today is Veteran's Day", "Night", 1, 1) ' returns 0;
```

## See Also

[StringASCII\(\)](#), [StringChar\(\)](#), [StringFromIntg\(\)](#), [StringFromReal\(\)](#), [StringFromTime\(\)](#), [StringLeft\(\)](#), [StringLen\(\)](#),  
[StringLower\(\)](#), [StringMid\(\)](#), [StringReplace\(\)](#), [StringRight\(\)](#), [StringSpace\(\)](#), [StringTest\(\)](#), [StringToIntg\(\)](#),  
[StringToReal\(\)](#), [StringTrim\(\)](#), [StringUpper\(\)](#), [Text\(\)](#)

## Related Topics

[String Functions](#)

# StringLeft()

Returns a specified number of characters in a string value, starting with the leftmost string character.

## Category

String

## Syntax

*StringResult* = StringLeft( *Text*, *Chars* );

## Parameters

*Text*

Actual string or a string equipment.item.

*Chars*

Number of characters to return or an integer equipment.item.

## Remarks

If *Chars* is set to 0, the entire string is returned.

## Examples

```
StringLeft("The Control Pump is On", 3) ' returns "The";
StringLeft("Pump 01 is On", 4) ' returns "Pump";
StringLeft("Pump 01 is On", 96) ' returns "Pump 01 is On";
StringLeft("The Control Pump is On", 0) ' returns "The Control Pump is On";
```

## See Also

[StringASCII\(\)](#), [StringChar\(\)](#), [StringFromIntg\(\)](#), [StringFromReal\(\)](#), [StringFromTime\(\)](#), [StringInString\(\)](#), [StringLen\(\)](#),  
[StringLower\(\)](#), [StringMid\(\)](#), [StringReplace\(\)](#), [StringRight\(\)](#), [StringSpace\(\)](#), [StringTest\(\)](#), [StringToIntg\(\)](#),  
[StringToReal\(\)](#), [StringTrim\(\)](#), [StringUpper\(\)](#), [Text\(\)](#)

## Related Topics

[String Functions](#)

# StringLen()

Returns the number of characters in a string.

## Category

String

## Syntax

*IntegerResult* = StringLen( *Text* );

## Parameter

*Text*

Actual string or a string equipment.item.

## Remarks

All the characters in the string equipment.item are counted, including blank spaces and those not normally shown on the screen.

## Examples

```
StringLen("Twelve percent") ' returns 14;  
StringLen("12%") ' returns 3;  
StringLen("The end." + StringChar(13)) ' returns 9;
```

The carriage return character is ASCII 13.

## See Also

[StringASCII\(\)](#), [StringChar\(\)](#), [StringFromIntg\(\)](#), [StringFromReal\(\)](#), [StringFromTime\(\)](#), [StringInString\(\)](#), [StringLeft\(\)](#), [StringLower\(\)](#), [StringMid\(\)](#), [StringReplace\(\)](#), [StringRight\(\)](#), [StringTest\(\)](#), [StringToIntg\(\)](#), [StringToReal\(\)](#), [StringTrim\(\)](#), [StringUpper\(\)](#), [Text\(\)](#)

## Related Topics

[String Functions](#)

# StringLower()

Converts all uppercase characters in text string to lowercase and returns the result.

## Category

String

## Syntax

```
StringResult = StringLower( Text );
```

## Parameter

*Text*

String to be converted to lowercase. Actual string or a string equipment.item.

## Remarks

Lowercase characters, symbols, numbers, and other special characters are not affected.

## Examples

```
StringLower("TURBINE") ' returns "turbine";
StringLower("22.2 Is The Value") ' returns "22.2 is the value";
```

## See Also

[StringASCII\(\)](#), [StringChar\(\)](#), [StringFromIntg\(\)](#), [StringFromReal\(\)](#), [StringFromTime\(\)](#), [StringInString\(\)](#), [StringLeft\(\)](#), [StringLen\(\)](#), [StringMid\(\)](#), [StringReplace\(\)](#), [StringRight\(\)](#), [StringTest\(\)](#), [StringToIntg\(\)](#), [StringToReal\(\)](#), [StringTrim\(\)](#), [StringUpper\(\)](#), [Text\(\)](#)

## Related Topics

[String Functions](#)

# StringMid()

Extracts a specific number of characters from a starting point within a string and returns the extracted character string as the result.

## Category

String

## Syntax

```
StringResult = StringMid( Text, StartChar, Chars );
```

## Parameters

*Text*

Actual string or a string equipment.item to extract a range of characters.

*StartChar*

The position of the first character within the string to extract. Any number or an integer equipment.item.

*Chars*

The number of characters within the string to return. Any number or an integer equipment.item.

## Remarks

This function is slightly different than the [StringLeft\(\)](#) function and [StringRight\(\)](#) function in that it allows you to specify both the start and end of the string that is to be extracted.

## Examples

```
StringMid("The Furnace is Overheating",5,7); ' returns "Furnace";
StringMid("The Furnace is Overheating",13,3); ' returns "is ";
StringMid("The Furnace is Overheating",16,50); ' returns "Overheating"
```

## See Also

[StringASCII\(\)](#), [StringChar\(\)](#), [StringFromReal\(\)](#), [StringFromTime\(\)](#), [StringInString\(\)](#), [StringLeft\(\)](#), [StringLen\(\)](#),  
[StringLower\(\)](#), [StringReplace\(\)](#), [StringSpace\(\)](#), [StringTest\(\)](#), [StringToIntg\(\)](#), [StringToReal\(\)](#), [StringTrim\(\)](#),  
[StringUpper\(\)](#), [Text\(\)](#)

## Related Topics

[String Functions](#)

# StringReplace()

Replaces or changes specific parts of a provided string and returns the result.

## Category

String

## Syntax

```
StringResult = StringReplace( Text, SearchFor, ReplaceWith, CaseSens, NumToReplace, MatchWholeWords );
```

## Parameters

### *Text*

The string in which characters, words, or phrases will be replaced. Actual string or a string equipment.item.

### *SearchFor*

The string to search for and replace. Actual string or a string equipment.item.

### *ReplaceWith*

The replacement string. Actual string or a string equipment.item.

### *CaseSens*

Determines whether the search is case-sensitive. (0=no and 1=yes) Any number or an integer equipment.item.

#### *NumToReplace*

Determines the number of occurrences to replace. Any number or an integer equipment.item. To indicate all occurrences, set this value to -1.

#### *MatchWholeWords*

Determines whether the function limits its replacement to whole words. (0=no and 1=yes) Any number or an integer equipment.item. If MatchWholeWords is turned on (set to 1) and the SearchFor is set to "and", the "and" in "handle" are not replaced. If the MatchWholeWords is turned off (set to 0), it is replaced.

## Remarks

Use this function to replace characters, words, or phrases within a string.

The StringReplace() function does not recognize special characters, such as @ # \$ % & \* ( ). It reads them as delimiters. For example, if the function StringReplace() (abc#,abc#,1234,0,1,1) is processed, there is no replacement. The # sign is read as a delimiter instead of a character.

## Examples

```
StringReplace("In From Within","In","Out",0,1,0) ' returns "Out From Within" (replaces only  
the first one);  
StringReplace("In From Within","In","Out",0,-1,0) ' returns "Out From without" (replaces  
all occurrences);  
StringReplace("In From Within","In","Out",1,-1,0) ' returns "Out From Within" (replaces all  
that match case);  
StringReplace("In From Within","In","Out",0,-1,1) ' returns "Out From Within" (replaces all  
that are whole words);
```

## See Also

[StringASCII\(\)](#), [StringChar\(\)](#), [StringFromIntg\(\)](#), [StringFromReal\(\)](#), [StringFromTime\(\)](#), [StringInString\(\)](#), [StringLeft\(\)](#),  
[StringLen\(\)](#), [StringLower\(\)](#), [StringMid\(\)](#), [StringSpace\(\)](#), [StringTest\(\)](#), [StringToIntg\(\)](#), [StringToReal\(\)](#), [StringTrim\(\)](#),  
[StringUpper\(\)](#), [Text\(\)](#)

## Related Topics

[String Functions](#)

# StringRight()

Returns the specified number of characters starting at the right-most character of text.

## Category

String

## Syntax

```
StringResult = StringRight( Text, Chars );
```

## Parameters

*Text*

Actual string or a string equipment.item.

*Chars*

The number of characters to return or an integer equipment.item.

## Remarks

If *Chars* is set to 0, the entire string is returned.

## Examples

```
StringRight("The Pump is On", 2) ' returns "On";
StringRight("The Pump is On", 5) ' returns "is On";
StringRight("The Pump is On", 87) ' returns "The Pump is On";
StringRight("The Pump is On", 0) ' returns "The Pump is On";
```

## See Also

[StringASCII\(\)](#), [StringChar\(\)](#), [StringFromIntg\(\)](#), [StringFromReal\(\)](#), [StringFromTime\(\)](#), [StringInString\(\)](#), [StringLeft\(\)](#), [StringLen\(\)](#), [StringLower\(\)](#), [StringMid\(\)](#), [StringReplace\(\)](#), [StringSpace\(\)](#), [StringTest\(\)](#), [StringToIntg\(\)](#), [StringToReal\(\)](#), [StringTrim\(\)](#), [StringUpper\(\)](#), [Text\(\)](#)

## Related Topics

[String Functions](#)

# StringSpace()

Generates a string of spaces either within a string equipment.item or within an expression and returns the result.

## Category

String

## Syntax

```
StringResult = StringSpace( NumSpaces );
```

## Parameter

### *NumSpaces*

Number of spaces to return. Any number or an integer equipment.item.

## Examples

```
All spaces are represented by the "x" character:  
StringSpace(4) ' returns "xxxx";  
"Pump" + StringSpace(1) + "Station" ' returns "PumpxStation";
```

## See Also

[StringASCII\(\)](#), [StringChar\(\)](#), [StringFromIntg\(\)](#), [StringFromReal\(\)](#), [StringFromTime\(\)](#), [StringInString\(\)](#), [StringLeft\(\)](#), [StringLen\(\)](#), [StringLower\(\)](#), [StringMid\(\)](#), [StringReplace\(\)](#), [StringTest\(\)](#), [StringToIntg\(\)](#), [StringToReal\(\)](#), [StringTrim\(\)](#), [StringUpper\(\)](#), [Text\(\)](#)

## Related Topics

[String Functions](#)

# StringTest()

Tests the first character of text to determine whether it is of a certain type and returns the result.

## Category

String

## Syntax

```
DiscreteResult = StringTest( Text, TestType );
```

## Parameters

### *Text*

String that function acts on. Actual string or a string equipment.item.

### *TestType*

Determines the type of test. Possible values are:

- 1 = Alphanumeric character ('A-Z', 'a-z' and '0-9')
- 2 = Numeric character ('0- 9')
- 3 = Alphabetic character ('A-Z' and 'a-z')
- 4 = Uppercase character ('A-Z')

- 5 = Lowercase character ('a'-'z')
- 6 = Punctuation character (0x21-0x2F)
- 7 = ASCII characters (0x00 - 0x7F)
- 8 = Hexadecimal characters ('A-F' or 'a-f' or '0-9')
- 9 = Printable character (0x20-0x7E)
- 10 = Control character (0x00-0x1F or 0x7F)
- 11 = White Space characters (0x09-0x0D or 0x20)

## Remarks

`StringTest()` function returns true to `DiscreteResult` if the first character in `Text` is of the type specified by `TestType`. Otherwise, false is returned. If the `StringTest()` function contains more than one character, only the first character of the `equipment.item` is tested.

## Examples

```
StringTest("ACB123",1)  ' returns 1;  
StringTest("ABC123",5)  ' returns 0;
```

## See Also

[StringASCII\(\)](#), [StringChar\(\)](#), [StringFromIntg\(\)](#), [StringFromReal\(\)](#), [StringFromTime\(\)](#), [StringInString\(\)](#), [StringLeft\(\)](#), [StringLen\(\)](#), [StringLower\(\)](#), [StringMid\(\)](#), [StringReplace\(\)](#), [StringSpace\(\)](#), [StringToIntg\(\)](#), [StringToReal\(\)](#), [StringTrim\(\)](#), [StringUpper\(\)](#), [Text\(\)](#)

## Related Topics

[String Functions](#)

# StringToIntg()

Converts the numeric value of a string to an integer value and returns the result.

## Category

String

## Syntax

```
IntegerResult = StringToIntg( Text );
```

## Parameter

*Text*

String that function acts on. Actual string or a string equipment.item.

## Remarks

When this statement is evaluated, the system reads the first character of the string for a numeric value. If the first character is other than a number, the string's value is equated to zero (0). Blank spaces are ignored. If the first character is a number, the system continues to read the subsequent characters until a non-numeric value is detected.

## Examples

```
StringToIntg("ABCD"); ' returns 0;
StringToIntg("22.2 is the Value"); ' returns 22 (since integers are whole numbers);
StringToIntg("The Value is 22"); ' returns 0;
```

## See Also

[StringASCII\(\)](#), [StringChar\(\)](#), [StringFromIntg\(\)](#), [StringFromReal\(\)](#), [StringFromTime\(\)](#), [StringInString\(\)](#), [StringLeft\(\)](#), [StringLen\(\)](#), [StringLower\(\)](#), [StringMid\(\)](#), [StringReplace\(\)](#), [StringSpace\(\)](#), [StringTest\(\)](#), [StringToReal\(\)](#), [StringTrim\(\)](#), [StringUpper\(\)](#), [Text\(\)](#)

## Related Topics

[String Functions](#)

# StringToReal()

Converts the numeric value of a string to a real (floating point) value and returns the result.

## Category

String

## Syntax

```
RealResult = StringToReal( Text );
```

## Parameter

*Text*

String that function acts on. Actual string or a string equipment.item.

## Remarks

When this statement is evaluated, the system reads the first character of the string for a numeric value. If the first character is other than a number (blank spaces are ignored), the string's value is equated to zero (0). If the first character is found to be a number, the system continues to read the subsequent characters until a non-numeric value is encountered.

## Examples

```
StringToReal("ABCD"); ' returns 0;
StringToReal("22.261 is the value"); ' returns 22.261;
StringToReal("The Value is 2"); ' returns 0;
```

## See Also

[StringASCII\(\)](#), [StringChar\(\)](#), [StringFromIntg\(\)](#), [StringFromReal\(\)](#), [StringFromTime\(\)](#), [StringInString\(\)](#), [StringLeft\(\)](#), [StringLen\(\)](#), [StringLower\(\)](#), [StringMid\(\)](#), [StringReplace\(\)](#), [StringSpace\(\)](#), [StringTest\(\)](#), [StringToIntg\(\)](#), [StringTrim\(\)](#), [StringUpper\(\)](#), [Text\(\)](#)

## Related Topics

[String Functions](#)

# StringTrim()

Removes unwanted spaces from text and returns the result.

## Category

String

## Syntax

```
StringResult = StringTrim( Text, TrimType );
```

## Parameter

*Text*

String that is trimmed of spaces. Actual string or a string equipment.item.

*TrimType*

Determines how the string is trimmed. Possible values are:

1 = Remove leading spaces to the left of the first non-space character

2 = Remove trailing spaces to the right of the last non-space character

3 = Remove all spaces except for single spaces between words

## Remarks

The text is searched for white-spaces (ASCII 0x09-0x0D or 0x20) that are to be removed. TrimType determines the method used by the function:

## Examples

All spaces are represented by the "x" character.

```
StringTrim("xxxxxThisxisxxxxtestxxxx", 1) ' returns "Thisxisxxxxtestxxxx";  
StringTrim("xxxxxThisxisxxxxtestxxxx", 2) ' returns "xxxxxThisxisxxxxtest";  
StringTrim("xxxxxThisxisxxxxtestxxxx", 3) ' returns "Thisxisxxxxtest";
```

The [StringReplace\(\)](#) function can remove ALL spaces from a specified a string equipment.item. Simply replace all the space characters with a "null."

## See Also

[StringASCII\(\)](#), [StringChar\(\)](#), [StringFromIntg\(\)](#), [StringFromReal\(\)](#), [StringFromTime\(\)](#), [StringInString\(\)](#), [StringLeft\(\)](#), [StringLen\(\)](#), [StringLower\(\)](#), [StringMid\(\)](#), [StringReplace\(\)](#), [StringSpace\(\)](#), [StringTest\(\)](#), [StringToIntg\(\)](#), [StringToReal\(\)](#), [StringUpper\(\)](#), [Text\(\)](#)

## Related Topics

[String Functions](#)

# StringUpper()

Converts all lowercase text characters to uppercase and returns the result.

## Category

String

## Syntax

```
StringResult = StringUpper( Text );
```

## Parameter

*Text*

String to be converted to uppercase. Actual string or a string equipment.item.

## Remarks

Uppercase characters, symbols, numbers, and other special characters are not affected.

## Examples

```
StringUpper("abcd"); ' returns "ABCD";
StringUpper("22.2 is the value"); ' returns "22.2 IS THE VALUE";
```

## See Also

[StringASCII\(\)](#), [StringChar\(\)](#), [StringFromIntg\(\)](#), [StringFromReal\(\)](#), [StringFromTime\(\)](#), [StringInString\(\)](#), [StringLeft\(\)](#), [StringLen\(\)](#), [StringLower\(\)](#), [StringMid\(\)](#), [StringReplace\(\)](#), [StringSpace\(\)](#), [StringTest\(\)](#), [StringToIntg\(\)](#), [StringToReal\(\)](#), [StringTrim\(\)](#), [Text\(\)](#)

## Related Topics

[String Functions](#)

# Text()

Converts a number to text based on a specified format.

## Category

String

## Syntax

```
StringResult = Text( Number, Format );
```

## Parameters

*Number*

Any number or numeric equipment.item.

*Format*

Format to use in conversion. Actual string or a string equipment.item.

## Examples

```
Text(66,"#.00"); ' returns 66.00;
Text(22.269,"#.00"); ' returns 22.27;
Text(9.999,"#.00"); ' returns 10.00;
```

The following example shows how to use this function within another function:

```
LogMessage("The current value of FreezerRoomTemp is:" + Text (FreezerRoomTemp, "#.#"));  
In the following example, MessageTag is set to "One=1 Two=2".  
MessageTag = "One + " + Text(1,"#") + StringChar(32) + "Two +" + Text(2,"#");
```

## See Also

[StringFromIntg\(\)](#), [StringFromReal\(\)](#), [StringToIntg\(\)](#), [StringToReal\(\)](#)

## Related Topics

[String Functions](#)

## System Functions

Use system functions to interact with the operating system or other core system functions, such as ActiveX objects.

## Related Topics

[Script Functions](#)

[CreateObject\(\)](#)

[Now\(\)](#)

# CreateObject()

Creates an ActiveX (COM) object.

## Category

System

## Syntax

*ObjectResult* = CreateObject( *ProgID* );

## Parameter

*ProgID*

The program ID (as a string) of the object to be created.

## Example

```
CreateObject("ADODB.Connection");
```

## Related Topics

[System Functions](#)

# Now()

Returns the current time.

## Category

System

## Syntax

*TimeValue* = Now();

## Remarks

The return value can be formatted using .NET functions.

## Related Topics

[System Functions](#)

## QuickScript .NET Variables

QuickScript .NET variables must be declared before they can be used in QuickScript .NET scripts. Variables can be used on both the left and right side of statements and expressions.

Local variables or equipment.items can be used together in the same script. Variables declared within the script body lose their value after the script is executed. Those declared in the script body cannot be accessed by other scripts.

Variables declared in the **Declarations** area maintain their values throughout the lifetime of the object that the script is associated with.

Declare each variable in the script by a separate DIM statement followed by a semicolon. Enter DIM statements in the **Declarations** area of the **Script** tab page. The DIM statement syntax is as follows:

```
DIM <variable_name> [ ( <upper_bound>
[, <upper_bound>[, <upper_bound>]] ) ]
[ AS <data_type> ];
```

Where:

DIM	Required keyword.
<variable_name>	Name that begins with a letter (A-Z or a-z) and whose remaining characters can be any combination of

	letters (A-Z or a-z), digits (0-9) and underscores (_). The variable name is limited to 255 Unicode characters.
<upper_bound>	Reference to the upper bound (a number between 1 and 2,147,483,647, inclusive) of an array dimension. Three dimensions are supported in a DIM statement, each being nested in the syntax structure. After the upper bound is specified, it is fixed after the declaration. A statement similar to Visual Basic's ReDim is not supported.  The lower bound of each array dimension is always 1.
AS	Optional keyword for declaring the variable's datatype.
<data_type>	Any one of the following 11 datatypes: Boolean, Discrete, Integer, ElapsedTime, Float, Real, Double, String, Message, Time or Object.  Data_type can also be a .Net data_type like System.Xml.XmlDocument or a type defined in an imported script library.  If you omit the AS clause from the DIM statement, the variable, by default, is declared as an Integer datatype. For example:  DIM LocVar1;  is equivalent to:  DIM LocVar1 AS Integer;

In contrast to equipment.item names, variable names must not contain dots. Variable names and the data type identifiers are not case sensitive. If there is a naming conflict between a declared variable and another named entity in the script (for example, equipment.item name, alias or name of an object leveraged by the script), the variable name takes precedence over the other named entities. If the variable name is the same as an alias name, a warning message appears when the script is validated to indicate that the alias is ignored.

The syntax for specifying the entire array is "[ ]" for both local array variables and for equipment.item references. For example, to assign an equipment.item array to a local array, the syntax is:

locarr[] = tag.attr[];

DIM statements can be located anywhere in the script body, but they must precede the first referencing script statement or expression. If a local variable is referenced before the DIM statement, script validation done when you save the object containing the script prompts you to define it.

---

**Caution:** The validation mentioned above occurs only when you save the object containing the script. This is not the script syntax validation done when you click the **Validate Script** button.

Do not cascade DIM statements. For example, the following examples are invalid:

DIM LocVar1 AS Integer, LocVar2 AS Real;

```
DIM LocVar3, LocVar4, LocVar5, AS Message;
```

To declare multiple variables, you must enter separate DIM statements for each variable.

When used on the right side of an equation, declared local variables always cause expressions on the left side to have Good quality. For example :

```
dim x as integer;
dim y as integer;
x = 5;
y = 5;
me.attr = 5;
me.attr = x;
me.attr = x+y;
```

In each case of me.attr, quality is Good.

When you use a variable in an expression to the right of the operator, its Quality is treated as Good for the purpose of data quality propagation.

You can use null to indicate that there is no object currently assigned to a variable. Using null has the same meaning as the keyword "null" in C# or "nothing" in Visual Basic. Assigning null to a variable makes the variable eligible for garbage collection. You may not use a variable whose value is null. If you do, the script terminates and an error message appears in the logger. You may, however, test a variable for null. For example:

```
IF myvar == null THEN ...
```

It is not possible to pass equipment.items as parameters for system objects. To work around this issue, use a local variable as an intermediary or explicitly convert the equipment.item to a string using an appropriate function call when calling the system object.

## Related Topics

[QuickScript References](#)

[Numbers and Strings](#)

### Numbers and Strings

Allowed format for integer constants in decimal format is as follows:

```
IntegerConst = 0 or [sign] <non-zero_digit> <digit>*;
```

Where:

- sign ::= + | -
- non-zero\_digit ::= 1-9
- digit ::= 0-9

For example, an integer constant is a zero or consists of an optional sign followed by one or more digits. Leading zeros are not allowed. Integer constants outside the range -2147483648 to 2147483647 cause an overflow error.

Prepending either 0x or 0X causes a literal integer constant to be interpreted as hexadecimal notation. The +/- sign is supported.

The acceptable float for integers in hexadecimal is as follows:

```
IntegerHexConst = [<sign>] <0><x (or X)> <hexdigit>*
```

Where:

- sign ::= + or -
- hexdigit ::= 0-9, A-F, a-f (only eight hexdigits [32-bits] are allowed)

Allowed format for floats is as follows:

FloatConst ::= [<sign>] <digit>\* .<digit>+ [<exponent>;]

or

[<sign>] <digit>+ [.<digit>\* [<exponent>]];

Where:

- sign ::= + or -
- digit ::= 0-9 (can be one or more decimal digits)
- exponent = e (or E) followed by a sign and then digit(s)

Float constants are applicable as values for variables of type float, real, or double. For example, float constants do not take the number of bytes into account. Script validation detects an overflow when a float, real, or double variable has been assigned a float constant that exceeds the maximum value.

If no digits appear before the period (.), at least one has to appear after it. If neither an exponent part nor the period appears, a period is assumed to follow the last digit in the string.

If an equipment.item reference exists that has a format similar to a float constant with an exponent (such as "5E3"), then use the equipment.item qualifier, as follows:

equipment.item("5E3")

Strings have to be surrounded by double quotation marks. They are referred to as quoted strings. The double-double quote indicates a single double-quote in the string. For example, the string:

Joe said, "Look at that."

can be represented in QuickScript .NET as:

"Joe said, ""Look at that."""

## Related Topics

[QuickScript .NET Variables](#)

## QuickScript .NET Control Structures

QuickScript .NET provides five primary control structures in the scripting environment:

- [IF ...THEN ... ELSEIF ... ELSE ... ENDIF](#)
- [FOR ... TO ... STEP... NEXT Loop](#)
- [FOR EACH ... IN ... NEXT](#)
- [TRY ... CATCH](#)
- [WHILE Loop](#)

## Related Topics

[QuickScript References](#)  
[IF ...THEN ... ELSEIF ... ELSE ... ENDIF](#)  
[IF... THEN ... ELSEIF ... ELSE ... ENDIF and Equipment.item Quality](#)  
[FOR ... TO ... STEP... NEXT Loop](#)  
[FOR EACH ... IN ... NEXT](#)  
[TRY ... CATCH](#)  
[WHILE Loop](#)

### IF ...THEN ... ELSEIF ... ELSE ... ENDIF

IF-THEN-ELSE-ENDIF conditionally executes various instructions based on the state of an expression. The syntax is as follows:

```
IF <Boolean_expression> THEN
[statements];
[ { ELSEIF
    [statements] } ];
[ ELSE
    [statements] ];
ENDIF;
```

Where Boolean\_expression is an expression that can be evaluated as a Boolean.

Depending on the data type returned by the expression, the expression is evaluated to constitute a True or False state according to the following table:

Data Type	Mapping
Boolean, Discrete	Directly used (no mapping needed).
Integer	Value = 0 evaluated as False. Value != 0 evaluated as True.
Float, Real	Value = 0 evaluated as False. Value != 0 evaluated as True.
Double	Value = 0 evaluated as False. Value != 0 evaluated as True.
String, Message	Cannot be mapped. Using an expression that results in a string type as the Boolean_expression results in a script validation error.
Time	Cannot be mapped. Using an expression that results in a time type as the Boolean_expression results in a script validation error.

Data Type	Mapping
ElapsedTime	Cannot be mapped. Using an expression that results in an elapsed time type as the Boolean_expression results in a script validation error.
Object	Using an expression that results in an object type. Validates, but at run time, the object is converted to a Boolean. If the type cannot be converted to a Boolean, a run-time exception is raised.

The first block of statements is executed if Boolean\_expression evaluates to True. Optionally, a second block of statements can be defined after the keyword ELSE. This block is executed if the Boolean\_expression evaluates to False.

To help decide between multiple alternatives, an optional ELSEIF clause can be used as often as needed. The ELSEIF clause mimics switch statements offered by other programming languages. For example:

```
IF value == 0 Then
    Message = "Value is zero";
ELSEIF value > 0 Then
    Message = "Value is positive";ELSEIF value < 0 Then
    Message = "Value is negative";
ELSE
    {Default. Should never occur in this example};
ENDIF;
```

The following approach nests a second IF compound statement within a previous one and requires an additional ENDIF:

```
IF (X1 == 1) THEN
    X1 = 5;
{ ELSEIF <X1 == 2> THEN
    X1 = 10;
ELSEIF X1 == 3 THEN
    X1 = 20 ;
ELSEIF X1 == 4 THEN
    X1 = 30 };
    IF X1 == 99 THEN
        X1 = 0;
    ENDIF;
ENDIF;
```

See Sample Scripts for more ideas about using this type of control structure.

## Related Topics

[QuickScript .NET Control Structures](#)

### IF... THEN ... ELSEIF ... ELSE ... ENDIF and Equipment.item Quality

When an equipment.item value is copied to another equipment.item of the same type, the equipment.item's quality is also copied. This can be especially relevant when working with I/O equipment.items. For example, the following two statements copy both value and quality:

```
me.Attr2 = me.Attr1;
me.Attr2.value = me.Attr1.value;
```

If only the value needs to be copied and the equipment.item has the quality BAD, you can use a temporary variable to hold the value. For example:

```
Dim temp as Integer;
temp = me.Attr1;
me.Attr2 = temp;
```

If there is a comparison such as Attr1 <> Attr2 and one of the equipment.items has the quality BAD, then the statements within the IF control block are not executed. For example, assuming Attr1 has the quality BAD:

```
if me.Attr1<> me.Attr2 then
    me.Attr2 = me.Attr1;
endif;
```

In this script, the statement me.Attr2 = me.Attr1 is not executed because Attr1 has the quality BAD and comparing a BAD quality value with a good quality value is not defined/not possible.

The recommended approach is to first verify the quality of Attr1, as shown in the following example:

```
if(IsBad(me.Attr1)) then
LogMessage("Attr1 quality is bad, its value is not copied to Attr2");
else
if me.Attr1<> me.Attr2 then
me.AttrA2 = me.Attr1;
endif;
endif;
```

An alternative method of verifying quality is to use the "==" operator:

```
if Me.Attr1 == TRUE then
```

Or, you can add the "value" property to the simplified IF THEN statement:

```
if Me.Attr1.value then
```

Using any of the above methods to verify data quality will ensure that your scripts execute correctly.

## Related Topics

[QuickScript .NET Control Structures](#)

### FOR ... TO ... STEP... NEXT Loop

FOR-NEXT performs a function (or set of functions) within a script several times during a single execution of a script. The general format of the FOR-NEXT loop is as follows:

```
FOR <analog_var> = <start_expression> TO <end_expression> [STEP <change_expression>];
[statements];
[EXIT FOR];
[statements];
NEXT;
```

Where:

- `analog_var` is a variable of type Integer, Float, Real, or Double.
- `start_expression` is a valid expression to initialize `analog_var` to a value for execution of the loop.
- `end_expression` is a valid expression. If `analog_var` is greater than `end_expression`, execution of the script jumps to the statement immediately following the `NEXT` statement.

This holds true if loop is incrementing up, otherwise, if loop is decrementing, loop termination occurs if analog\_var is less than end\_expression.

- change\_expression is an expression that defines the increment or decrement value of analog\_var after execution of the NEXT statement. The change\_expression can be either positive or negative.
  - If change\_expression is positive, start\_expression must be less than or equal to end\_expression or the statements in the loop do not execute.
  - If change\_expression is negative, start\_expression must be greater than or equal to end\_expression for the body of the loop to be executed.
- If STEP is not set, then change\_expression defaults to 1 for increasing increments, and defaults to -1 for decreasing increments.

Exit the loop from within the body of the loop with the EXIT FOR statement.

The FOR loop is executed as follows:

1. analog\_var is set equal to start\_expression.
2. If change\_expression is positive, the system tests to see if analog\_var is greater than end\_expression. If so, the loop exits. If change\_expression is negative, the system tests to see if analog\_var is less than end\_expression. If so, program execution exits the loop.
3. The statements in the body of the loop are executed. The loop can potentially be exited via the EXIT FOR statement.
4. analog\_var is incremented by 1,-1, or by change\_expression if it is specified.
5. Steps 2 through 4 are repeated.

FOR-NEXT loops can be nested. The number of levels of nesting possible depends on memory and resource availability.

## Related Topics

[QuickScript .NET Control Structures](#)

### FOR EACH ... IN ... NEXT

FOR EACH loops can be used only with collections exposed by OLE Automation servers. A FOR-EACH loop performs a function (or set of functions) within a script several times during a single execution of a script. The general format of the FOR-EACH loop is as follows:

```
FOR EACH <object_variable> IN <collection_object >
    [statements];
    [EXIT FOR;];
    [statements];
NEXT;
```

Where:

- object\_variable is a dimmed variable.
- collection\_object is a variable holding a collection object.

As in the case of the FOR ... TO loop, it is possible to exit the execution of the loop through the statement EXIT

FOR from within the loop.

## Related Topics

[QuickScript .NET Control Structures](#)

### TRY ... CATCH

TRY ... CATCH provides a way to handle some or all possible errors that may occur in a given block of code, while still running rather than terminating the program. The TRY part of the code is known as the try block. Deal with any exceptions in the CATCH part of the code, known as the catch block.

The general format for TRY ... CATCH is as follows:

```
TRY
  [try statements] 'guarded section
CATCH
  [catch statements]
ENDTRY
```

Where:

*tryStatements*

Statement(s) where an error can occur. Can be a compound statement. The tryStatement is a guarded section.

*catchStatements*

Statement(s) to handle errors occurring in the associated Try block. Can be a compound statement.

Statements inside the Catch block may reference the reserved ERROR variable, which is a .NET System.Exception thrown from the Try block. The statements in the Catch block run only if an exception is thrown from the Try block.

TRY ... CATCH is executed as follows:

1. Run-time error handling starts with TRY. Put code that might result in an error in the try block.
2. If no run-time error occurs, the script will run as usual. Catch block statements will be ignored.
3. If a run-time error occurs, the rest of the try block does not execute.
4. When a run-time error occurs, the program immediately jumps to the CATCH statement and executes the catch block.

The simplest kind of exception handling is to stop the program, write out the exception message, and continue the program.

The error variable is not a string, but a .NET object of System.Exception. This means you can determine the type of exception, even with a simple CATCH statement. Call the GetType() method to determine the exception type, and then perform the operation you want, similar to executing multiple catch blocks.

## Example:

```
dim command = new System.Data.SqlClient.SqlCommand;
dim reader as System.Data.SqlClient.SqlDataReader;
command.Connection = new System.Data.SqlClient.SqlConnection;
try
  command.Connection.ConnectionString = "Integrated Security=SSPI";
```

```
command.CommandText="select * from sys.databases";
command.Connection.Open();
reader = command.ExecuteReader();

while reader.Read()
    me.name = reader.GetString(0);
    LogMessage(me.name);
endWhile;
catch
    LogMessage(error);
endtry;
if reader <> null and not reader.IsClosed then
    reader.Close();
endif;
if command.Connection.State == System.Data.ConnectionState.Open then
    command.Connection.Close();
endif;
```

## Related Topics

[QuickScript .NET Control Structures](#)

### WHILE Loop

WHILE loop performs a function or set of functions within a script several times during a single execution of a script while a condition is true. The general format of the WHILE loop is as follows:

```
WHILE <Boolean_expression>
[statements]
[EXIT WHILE;]
[statements]
ENDWHILE;
```

Where: Boolean\_expression is an expression that can be evaluated as a Boolean as defined in the description of IF...THEN statements.

It is possible to exit the loop from the body of the loop through the EXIT WHILE statement.

The WHILE loop is executed as follows:

1. The script evaluates whether the Boolean\_expression is true or not. If not, program execution exits the loop and continues after the ENDWHILE statement.
2. The statements in the body of the loop are executed. The loop can be exited through the EXIT WHILE statement.
3. Steps 1 through 2 are repeated.

WHILE loops can be nested. The number of levels of nesting possible depends on memory and resource availability.

## Related Topics

[QuickScript .NET Control Structures](#)

## QuickScript .NET Operators

The following QuickScript .NET operators require a single operand:

Operator	Short Description
~	Complement
-	Negation
NOT	Logical NOT

The following QuickScript .NET operators require two operands:

Operator	Short Description
+	Addition and concatenation
-	Subtraction
&	Bitwise AND
*	Multiplication
**	Power
/	Division
^	Exclusive OR
	Inclusive OR
<	Less than
<=	Less than or equal to
<>	Not equal to
=	Assignment
==	Equivalency (is equivalent to); not supported for entire array compares. Arrays must be compared one element at a time using ==.
>	Greater than
>=	Greater than or equal to
AND	Logical AND
MOD	Modulo
OR	Logical OR

Operator	Short Description
SHL	Left shift
SHR	Right shift

The following table shows the precedence of QuickScript .NET operators:

Precedence	Operator
1 (highest)	( )
2	- (negation), NOT, ~
3	**
4	* , /, MOD
5	+ , - (subtraction)
6	SHL, SHR
7	<, >, <=, >=
8	==, <>
9	&
10	^
11	
12	=
13	AND
14 (lowest)	OR

The arguments of the listed operators can be numbers or equipment.item values. Putting parentheses around an argument is optional. Operator names are not case-sensitive.

## Related Topics

- [QuickScript References](#)
- [Parentheses \( \)](#)
- [Negation \( - \)](#)
- [Complement \( ~ \)](#)
- [Power \( \\*\\* \)](#)
- [Multiplication \( \\* \), Division \( / \), Addition \( + \), Subtraction \( - \)](#)
- [Modulo \(MOD\)](#)

Shift Left (SHL), Shift Right (SHR)

Assignment ( = )

Comparisons ( <, >, <=, >=, ==, <> )

## Parentheses ( )

Parentheses specify the correct order of evaluation for the operator(s). They can also make a complex expression easier to read. Operator(s) in parentheses are evaluated first, preempting the other rules of precedence that apply in the absence of parentheses. If the precedence is in question or needs to be overridden, use parentheses.

In the example below, parentheses add B and C together before multiplying by D:

( B + C ) \* D;

## Related Topics

### Negation ( - )

Negation is an operator that acts on a single component. It converts a positive integer or real number into a negative number.

## Related Topics

[QuickScript .NET Operators](#)

### Complement ( ~ )

This operator yields the one's complement of a 32-bit integer. It converts each zero-bit to a one-bit and each one-bit to a zero-bit. The one's complement operator is an operator that acts on a single component, and it accepts an integer operand.

## Related Topics

[QuickScript .NET Operators](#)

### Power ( \*\* )

The Power operator returns the result of a number (the base) raised to the power of a second number (the power). The base and the power can be any real or integer numbers, subject to the following restrictions:

- A zero base and a negative power are invalid.  
Example: "0 \*\* -2" and "0 \*\* -2.5"
- A negative base and a fractional power are invalid.  
Example: "-2 \*\* 2.5" and "-2 \*\* -2.5"
- Invalid operands yield a zero result.

The result of the operation should not be so large or so small that it cannot be represented as a real number.

Example:

```
1 ** 1 = 1.0
3 ** 2 = 9.0
10 ** 5 = 100,000.0
```

## Related Topics

[QuickScript .NET Operators](#)

### Multiplication ( \* ), Division ( / ), Addition ( + ),Subtraction ( - )

These binary operators perform basic mathematical operations. The plus (+) can also concatenate String datatypes.

For example, in the data change script below, each time the value of "Number" changes, "Setpoint" changes as well:

```
Number=1;
Setpoint.Name = "Setpoint" + Text(Number, "#");
```

Where: The result is "Setpoint1."

## Related Topics

[QuickScript .NET Operators](#)

### Modulo (MOD)

MOD is a binary operator that divides an integer quantity to its left by an integer quantity to its right. The remainder of the quotient is the result of the MOD operation. Example:

```
97 MOD 8 yields 1
63 MOD 5 yields 3
```

## Related Topics

[QuickScript .NET Operators](#)

### Shift Left (SHL), Shift Right (SHR)

SHL and SHR are binary operators that use only integer operands. The binary content of the 32-bit word referenced by the quantity to the left of the operator is shifted (right or left) by the number of bit positions specified in the quantity to the right of the operator.

Bits shifted out of the word are lost. Bit positions vacated by the shift are zero-filled. The shift is an unsigned shift.

## Related Topics

[QuickScript .NET Operators](#)

### Bitwise AND ( & )

A bitwise binary operator compares 32-bit integer words with each other, bit for bit. Typically, this operator masks a set of bits. The operation in this example "masks out" (sets to zero) the upper 24 bits of the 32-bit word. For example:

```
result = name & 0xff;
```

## Related Topics

[QuickScript .NET Operators](#)

### Exclusive OR (^) and Inclusive OR ( | )

The ORs are bitwise logical operators compare 32-bit integer words to each other, bit for bit. The Exclusive OR compare the status of bits in corresponding locations. If the corresponding bits are the same, a zero is the result. If the corresponding bits differ, a one is the result. Example:

```
0 ^ 0 yields 0
0 ^ 1 yields 1
1 ^ 0 yields 1
1 ^ 1 yields 0
```

The Inclusive OR examines the corresponding bits for a one condition. If either bit is a one, the result is a one. Only when both corresponding bits are zeros is the result a zero. For example:

```
0 | 0 yields 0
0 | 1 yields 1
1 | 0 yields 1
1 | 1 yields 1
```

## Related Topics

[QuickScript .NET Operators](#)

### Assignment ( = )

Assignment is a binary operator which accepts integer, real, or any type of operand. Each statement can contain only one assignment operator. Only one name can be on the left side of the assignment operator.

Read the equal sign (=) of the assignment operator as "is assigned to" or "is set to."

---

**Note:** Do not confuse the equal sign with the equivalency sign (==) used in comparisons.

---

## Related Topics

[QuickScript .NET Operators](#)

## Comparisons ( <, >, <=, >=, ==, <> )

Comparisons in IF-THEN-ELSE statements execute various instructions based on the state of an expression.

### Related Topics

[QuickScript .NET Operators](#)

### AND, OR, and NOT

These operators work only on discrete equipment.items. If these operators are used on integers or real numbers, they are converted as follows:

- Real to Discrete: If real is 0.0, discrete is 0, otherwise discrete is 1.
- Integer to Discrete: If integer is 0, discrete is 0, otherwise discrete is 1.

If the statement is: "Disc1 = Real1 AND Real2;" and Real1 is 23.7 and Real2 is 0.0, Disc1 has 0 assigned to it, since Real1 is converted to 1 and Real2 is converted to 0.

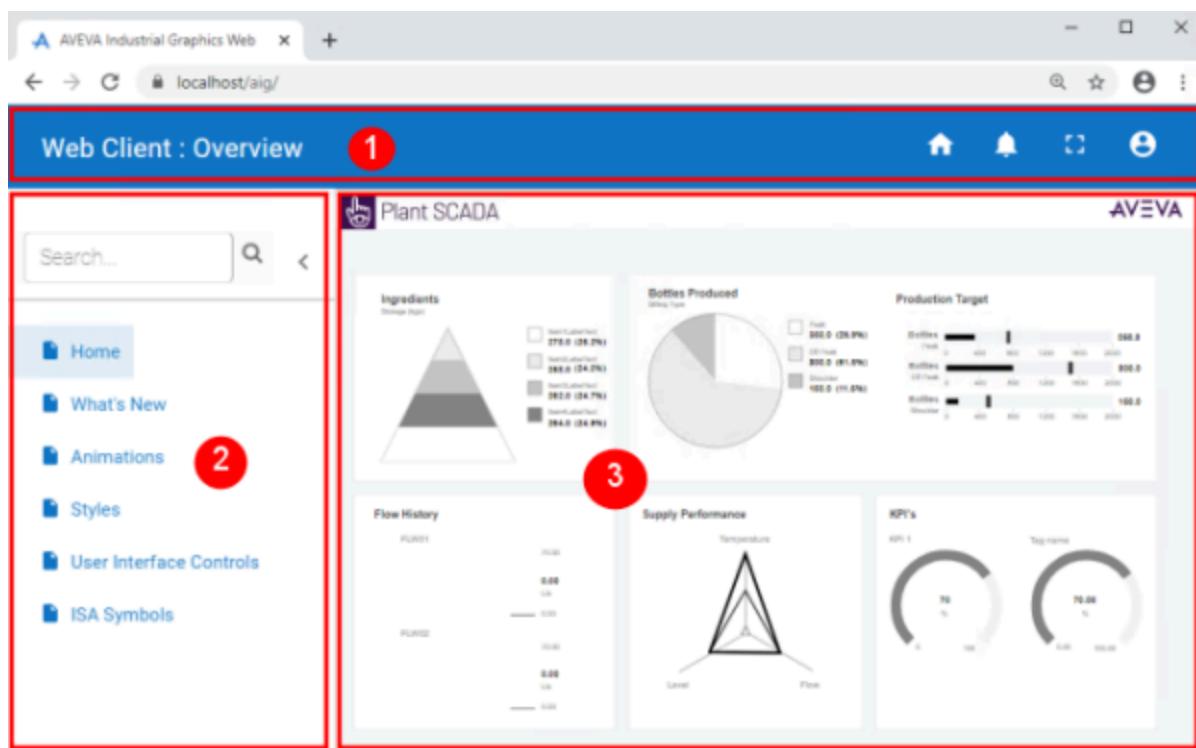
When assigning the floating-point result of a mathematical operation to an integer, the value is rounded to the nearest integer instead of truncating it. This means that an operation like IntAttr = 32/60 results in IntAttr having a value of 1, not 0. If truncation is needed, use the Trunc() function.

### Related Topics

[QuickScript .NET Operators](#)

## Use the Industrial Graphics Web Client

The Industrial Graphics Web Client allows you to view selected Industrial Graphics used within a Plant SCADA project on any supported HTML5 web browser. The following diagram describes its layout.



Element	Description	
1	The <b>Title Bar</b> displays the Industrial Graphics application name, page name, home icon, notifications icon, full screen icon and profile icon.	
	Application Name: Symbol Name	The Title Bar will display the application and page name.  For example, <i>Web Client: Overview</i> . Where: <i>Web Client</i> = the application name. <i>Overview</i> = the page name.
		Click to view the page set as the web client home page.
		Click the Notifications icon messages related to License status change and other status change messages. The number of notifications will be overlayed on the icon.
		Click to maximize the canvas area and display the web page in full screen mode. You can also use the F11 key. The title bar and navigation overlay will not be

		<p>visible.</p> <p><b>To view the title bar:</b></p> <ul style="list-style-type: none"><li>• Move your cursor to the top of the page, the title bar will slide down.</li><li>• Click  to pin the title bar on the page.</li><li>• Click  to hide the title bar.</li></ul>
		Click to minimize the canvas area and exit the full screen mode.
		The profile icon displays the user logged in.
	 <b>Sign Out</b>	Click the profile icon and then select the <b>Sign Out</b> icon. The Sign Out will only be available if System Management Server option is configured. For more information, see <a href="#">Log On to the Industrial Graphics Web Client</a> .
		From the Title Bar click  , and then click the <b>Language</b> icon. Select the language. The tooltip, translated static text and custom properties will display in the selected language. <b>Note:</b> When the user changes the language, a new session will be created causing the data in custom properties to be reset.
2		<p>Click the following icon to view the <b>menu navigation</b> overlay.</p> <p>All folders and symbols under the selected Web Client root folder are displayed.</p> <p>You can use the search box at the top of the overlay to search for pages. The search box is enabled with the autocomplete feature, where page names matching the input search term will be listed as you are typing. Select the page name to view the page in the canvas area.</p>

3	<p>The <b>canvas area</b> displays the page selected from the navigation overlay. The page will be scaled to fit the browser viewport size, while maintaining the aspect ratio. The navigation and notifications overlays do not consume any space on the canvas.</p>
	<p> Click the warning icon to view the list of animations or properties not supported by the Industrial Graphics Web Client. This icon is available only during application development and when the browser points to the exact URL of <a href="http://localhost/aig">http://localhost/aig</a>.</p>

## See Also

[View Application Graphics in a Web Browser](#)

[Application Graphics and Browser Resizing](#)

[Pan and Zoom Support](#)

[Customize the Web Client](#)

## View Application Graphics in a Web Browser

After you configure an Industrial Graphics application you can view the graphics in a web browser. For a list of tested and supported browsers, see [Browser and Mobile Support](#).

1. Launch an HTML5 supported browser, enter the URL: <http://<hostname>/aig> or <http://<IPAddress>/aig>. If your credentials are authenticated and a valid Industrial Graphics Web Server license is acquired, the Industrial Graphics Web Client page will display. The home page will be displayed by default. You can also access the web client using the URL: <https://<hostname>/aig> or <https://<IPAddress>/aig>. The Web Client will automatically use the HTTPS protocol when the System Management Server is configured with a certificate (local or remote server) for encrypted communication between the server and client. If the Web Client page is not displayed, refer to [Troubleshooting the Industrial Graphics Web Client](#).
2. Click the  icon. The overlay on the left displays the hierarchical structure of the pages under the selected Web Client Root folder within the running application.
3. Click a folder name. The pages within the folder are displayed.
4. Click a page. The selected graphic is displayed in the Web Client.

The user account accessing the graphics must be Authenticated and Authorized, for more information see

## Licensing for Industrial Graphics Applications in a Web Browser.

Plant SCADA Runtime must be running on the host machine for symbols displayed in the Web Client to receive live data from Plant SCADA tags. You can also run Plant SCADA as a service.

## See Also

[Customize the Web Client](#)

## Application Graphics and Browser Resizing

The Industrial Graphics Web Client page can be viewed in different screen sizes. As the browser is adjusted, the graphics resize to fit the new browser viewport dimensions. The graphic will also resize when the mobile device is rotated between portrait and landscape orientation.

For very small screens, the icons on the right collapse into a vertical bar. The icons will be hidden to save screen space. You can view Industrial Graphics on any mobile device, and use the pan and zoom gestures to view larger graphics.

## See Also

[Pan and Zoom Support](#)

## Pan and Zoom Support

The Industrial Graphics Web Client supports the Pan and Zoom gestures for all supported browsers and devices. When a graphic in the web browser is zoomed in from a non-mobile device, the zoom percentage is displayed in the lower left corner of the horizontal scroll bar. The zoom percentage is limited up to 500%.

### Pan and Zoom using Touch

You can zoom-in and zoom-out on a focused display by using two fingers.

- To zoom-in, place two fingers on the screen and expand. To zoom-out, pinch the display with two fingers.
- Double-tap to restore the display to 100% zoom.

To pan the display, press one finger on the display and move your finger.

### Pan and Zoom using Mouse

You can zoom-in and zoom-out on a focused display by using the mouse's scroll wheel.

- Hold the **Ctrl** key and scroll the wheel up or down to zoom-in or zoom-out. If the mouse pointer is outside of a pane, then zooming will not be allowed.
- Double-click the left mouse button to restore the display to 100% zoom.

To pan a display, keep the center wheel pressed, move the mouse to the area and then release the center wheel.

### Zoom using Keyboard

You can zoom-in and zoom-out on a focused display by using keyboard.

To zoom-in, use the **Ctrl** and + keys together. To zoom-out, use **Ctrl** and - keys together.

### Zoom by Location

You can also zoom in the graphic from the location you click on.

Pan and Zoom toolbars are not supported. Pan and Zoom gestures are not supported for graphics displayed using the ShowSymbol animation, Button Click and Slider.

## See Also

[Use the Industrial Graphics Web Client](#)

## Customize the Web Client

Users can customize the Web Client settings using Configurator on the computer that hosts the Industrial Graphics Server.

1. Open Configurator and navigate to **Industrial Graphics Server | Client Settings** tab.
2. In the **Graphic Refresh Rate (ms)** specify the refresh rate in milliseconds.
3. Click **Apply**. This will restart the Industrial Graphics Server.

---

**Note:** Alarm Refresh Rate is not supported by Industrial Graphics applications in this version of Plant SCADA.

## See Also

[Use the Industrial Graphics Web Client](#)

## Manage the Industrial Graphics Web Client

### In This Chapter

[Secure the Industrial Graphics Web Client](#)

[Licensing for Industrial Graphics Applications in a Web Browser](#)

## Secure the Industrial Graphics Web Client

A System Management Server (SMS) allows an Industrial Graphics Web Client to be used securely with the HTTPS protocol. You will need to connect an Industrial Graphics Server to an SMS to secure client access.

As an administrator, there are two configuration options available for user authentication and security. The option you chose will depend on whether or not you need to expose the Industrial Graphic Server to clients that exist outside your SCADA system network.

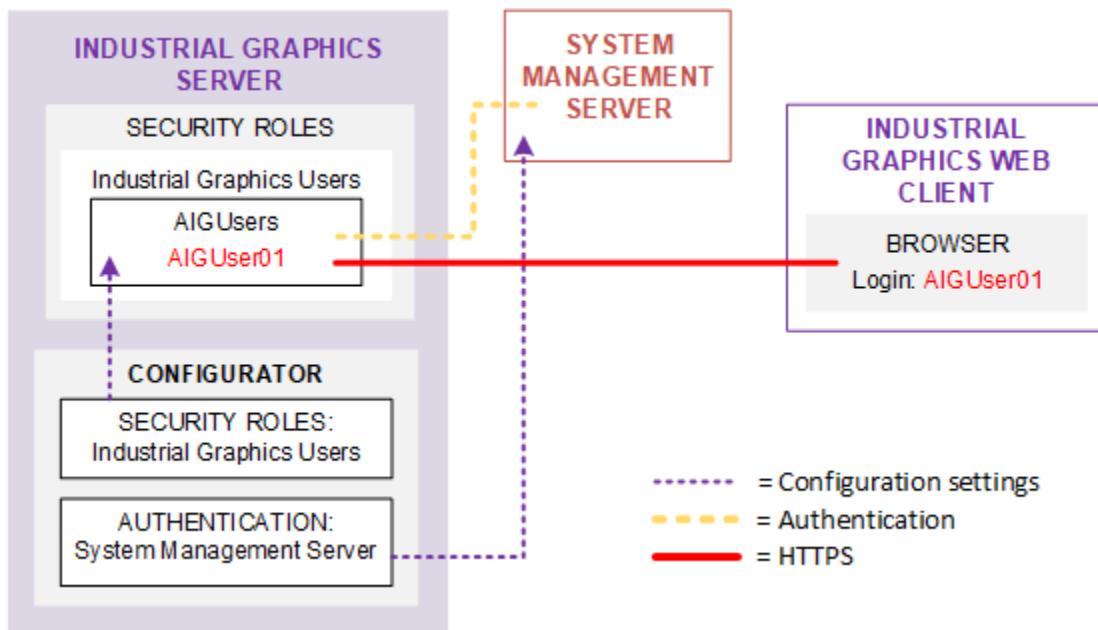
If this is the case, you need to use a Secure Gateway installed in a demilitarized zone (DMZ) to intercept requests from clients. This will use a reverse proxy server to protect the identity of the Industrial Graphics Server from external computers connecting via the Internet.

## Securing clients within your plant network

To secure Industrial Graphics Web Clients that operate within your SCADA system network, use Configurator to set up the following:

- **Authentication** - Connect the Industrial Graphics Server to an SMS.  
See [Use Configurator to Set Up an Industrial Graphics Server](#).
- **Security Roles** - Add the required users to one of the Industrial Graphics security roles on the Industrial Graphics Server.  
See [Configure User Access for an Industrial Graphics Web Client](#).

Once this configuration is complete, the SMS can authenticate a Web Client user and provide an HTTPS connection to the Industrial Graphics Server.

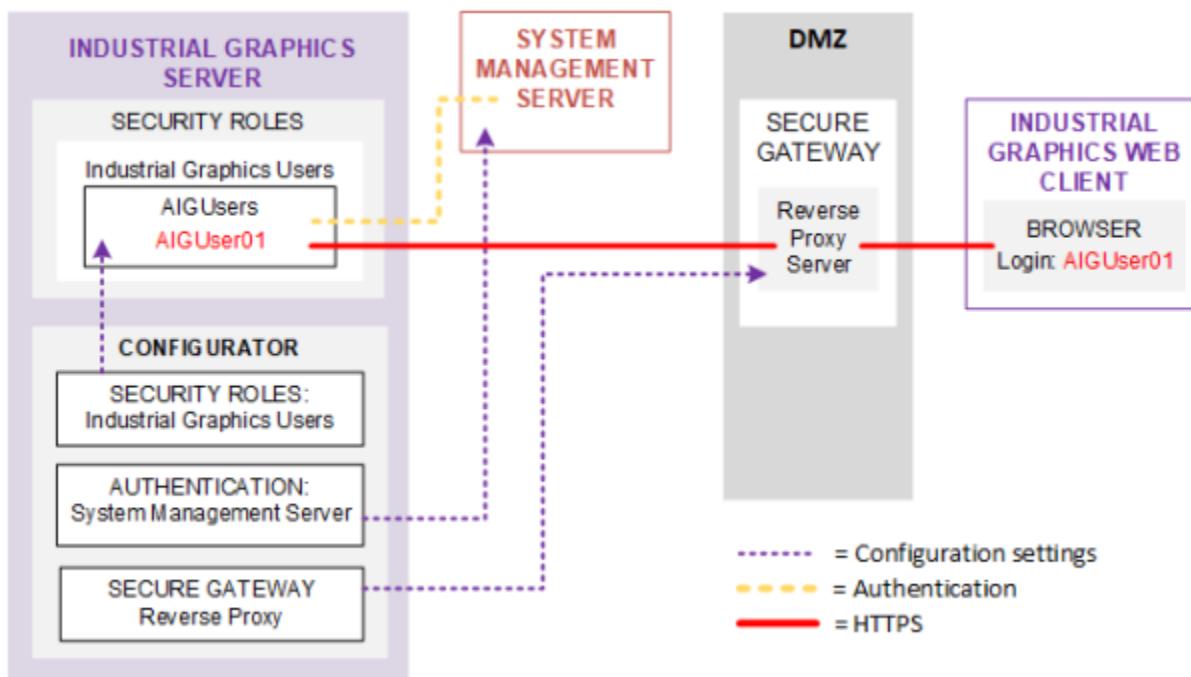


## Securing clients via a Secure Gateway server

To secure Industrial Graphics Web Clients that exist beyond your SCADA system network, use Configurator to set up the following:

- **Authentication** - Connect the Industrial Graphics Server to an SMS.  
For instructions, see [Configure an Industrial Graphics Server using a Secure Gateway](#).
- **Secure Gateway** - Provide the Fully Qualified Domain Name of the reverse proxy server in Configurator's **Secure Gateway** field.  
For detailed configuration instructions, see [Configure an Industrial Graphics Server using a Secure Gateway](#).
- **Security Roles** - Add the required users to one of the Industrial Graphics security roles on the Industrial Graphics Server.  
See [Configure User Access for an Industrial Graphics Web Client](#).

Once this configuration is complete, the SMS can authenticate a Web Client user and provide an HTTPS connection to the Industrial Graphics Server via the Secure Gateway server.



If the Web Client page loads at runtime without a valid security token, the following will occur.

- The Web Client will redirect to a login page.
- The credentials entered by the user will be checked against the Active Directory.
- If the credentials are valid, then Active Directory will provide a security token and return it to the Web Client.
- The Web Client will then grant access to user with the token.

If a security token already exists, then the user will be granted access.

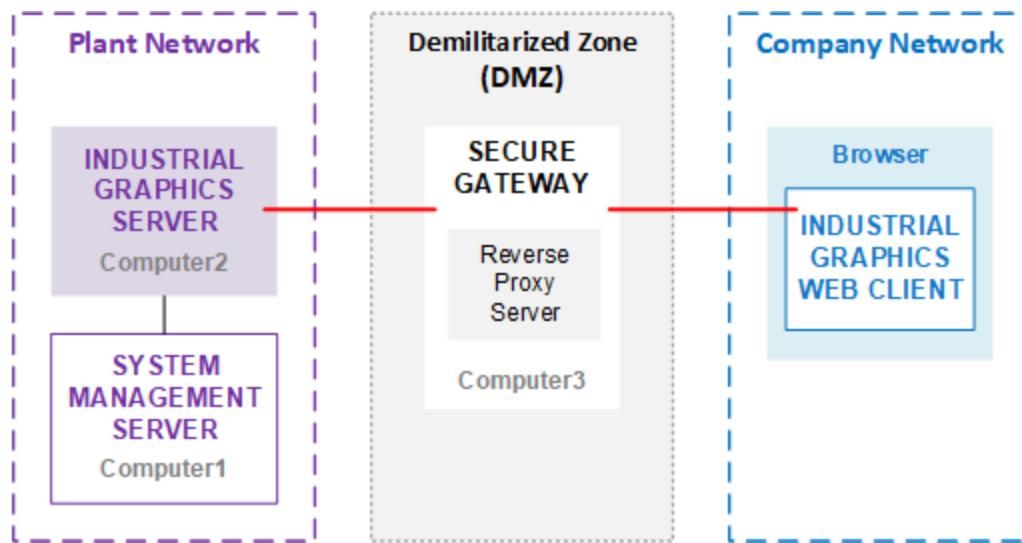
## See Also

[Log On to the Industrial Graphics Web Client](#)

## Configure an Industrial Graphics Server using a Secure Gateway

This topic provides the instructions required to set up an Industrial Graphics Server that needs to communicate with web clients located outside your SCADA system network. It explains how to secure the server with a Secure Gateway using a reverse proxy server.

The examples in this document are based on the following system architecture:



- **computer1** is the computer name of the System Management Server.
- **computer2** is the computer name of the Industrial Graphics Server.
- **computer3** is the computer name of the Secure Gateway.

## Confirm the Industrial Graphics Server is working

Before you configure the Secure Gateway, you need to confirm that the Industrial Graphics Server is set up correctly.

To do this, enter the following address into a browser on a computer within the plant network.

<https://computer2/aig>

If the Industrial Graphics Web Client does not display, see the section on securing clients within your plant network in the topic [Secure the Industrial Graphics Web Client](#).

## Set up the Secure Gateway

The first step is to set up computer3, the Secure Gateway installed in a demilitarized zone (DMZ). This requires the installation of a reverse proxy server to protect the identity of the Industrial Graphics Server from external computers connecting via the Internet.

For this, you will need a valid SSL certificate (.key and .pem files) issued by an authorized entity. This certificate will be used later to set up the System Management Server.

---

**Note:** Self-signed certificates are not recommended for production environments.

---

## Set up the System Management Server

Make the following changes on the System Management Server (computer1).

1. Add the SSL certificate from the Secure Gateway (computer3).  
You need to import the certificate to the Trusted Root.
2. Open the **appsettings.json** file for editing.

This file is part of the Platform Common Services installation on the System Management Server. It is located here:

C:\Program Files (x86)\AVEVA\Platform Common Services\Management Server

3. Add the "PublicOrigin" property to the AppSetting section of the file in the format specified below. Use a fully-qualified domain name (FQDN) to identify the Secure Gateway computer.

```
"PublicOrigin": "https://computer3.mydmzdomain.com"
```

Specify the port number if it is not 443.

```
"PublicOrigin": "https://computer3.mydmzdomain.com:<portNumber>"
```

```
".AppSettings": {
    "AllowGroups": [
        "aaAdministrators"
    ],
    "UserGroupFilters": [
    ],
    "SelfSignedCertificate": "",
    "BaseUrl": "http://+:80/identitymanager",
    "ServiceSource": "Token Host",
    "PublicOrigin": "https://computer3.mydmzdomain.com",
    "CustomOptions": {
        "UseNativeLogon": "true",
        "UsePrincipalSearcher": "false",
        "CheckNestedGroups": "false"
    },
    "SsoSyncOptions": {
        "MaxEntitiesPerChunk": 100,
        "MaxMessageBufferSize": 5242880,
        "InternalLogging": {
            "LogLevel": {
                "Default": "Information",
                "System": "Information",
                "Microsoft": "Information"
            }
        }
    }
},
```

4. Reconfigure the System Management Server.

Go to the System Management Server page in Configurator and confirm that **This machine is the System Management Server** is selected.

Click the **Configure** button.

The changes you have made to the PublicOrigin property will be evident in the Configuration Messages panel. For example, the System Management Server will now be identified by the FQDN of the Secure Gateway computer.

#### Configuration Messages

```
Checking if binding a site to the SSL(HTTPS) port is allowed.  
Ensuring the SSL(HTTPS) port has the correct ACL.  
Updating HTTPS Port 443 in the registry.  
Configuring the certificate binding.  
Certificates are configured.  
Connecting to the System Management Server 'https://computer3.mydmzdomain.com:443'
```

5. You will need to enter the login details for a domain user to connect to the Secure Gateway computer.

If the configuration process is not successful, try the following:

- Ping the Secure Gateway computer before you attempt to configure the System Management Server.
- Confirm that the Secure Gateway SSL certificate has been added correctly.

## Set up the Industrial Graphics Server

---

**Note:** Before you commence, you should confirm that your Plant SCADA project includes the "BUILTIN\Administrators" role. Check the **Roles** view in Plant SCADA Studio's **Security** activity.

---

Make the following changes on the Industrial Graphics Server (computer2).

1. Add the SSL certificate from the Secure Gateway.

You need to import the certificate to the Trusted Root.

2. Open the **appsettings.json** file for editing.

This file is part of the Platform Common Services installation on the Industrial Graphics Server. It is located here:

C:\Program Files (x86)\AVEVA\Platform Common Services\Management Server

3. Add the "PublicOrigin" property to the AppSetting section of the file in the format specified below. Use a fully-qualified domain name (FQDN) to identify the Secure Gateway computer.

"PublicOrigin": "https://computer3.mydmzdomain.com"

Specify the port number if it is not 443.

"PublicOrigin": "https://computer3.mydmzdomain.com:<portNumber>"

```
"AppSetting": {  
    "AllowGroups": [  
        "aaAdministrators"  
    ],  
    "UserGroupFilters": [  
    ],  
    "SelfSignedCertificate": "",  
    "BaseURL": "http://+:80/identitymanager",  
    "ServiceSource": "Token Host",  
    "PublicOrigin": "https://computer3.mydmzdomain.com",  
    "CustomOptions": {  
        "UseNativeLogon": "true",  
        "UsePrincipalSearcher": "false",  
        "CheckNestedGroups": "false"  
    },  
    "SsoSyncOptions": {  
        "MaxEntitiesPerChunk": 100,  
        "MaxMessageBufferSize": 5242880,  
        "InternalLogging": {  
            "LogLevel": {  
                "Default": "Information",  
                "System": "Information",  
                "Microsoft": "Information"  
            }  
        }  
    }  
},
```

4. Open Configurator on the Industrial Graphics Server.
5. Go to the System Management Server page and confirm that **Connect to an existing System Management Server** is selected.
6. Enter the FQDN for the Secure Gateway computer (for example, computer3.mydmzdomain.com).
7. Click the **Configure** button.

The changes you have made to the PublicOrigin property should be evident in the Configuration Messages panel.

8. In Configurator, go to Authentication Settings page.

---

**Note:** The changes to the PublicOrigin setting on the System Management Server will be reflected on the Authentication Settings page. While the FQDN of the Secure Gateway computer is displayed in the System Management Server field, the SMS will continue to operate on the computer on which it was originally configured.

---

9. Enter the FQDN for the Secure Gateway computer in the **Secure Gateway** field.

To enable authentication of users, the Industrial Graphics Server must be registered with your System Management Server.

System Management Server

computer3.mydmzdomain.com:443

Secure Gateway

computer3.mydmzdomain.com

A secure gateway server is a reverse proxy server designed to protect the identity of your Industrial Graphics servers from clients connecting outside your plant network. It should be installed in a DMZ.

10. Click the **Configure** button.

When these steps are complete, you can log in to the Industrial Graphics Web Client via the Secure Gateway computer using the following address:

<https://computer3/aig>

If you attempt to log in from a computer within the plant network, you can access the Industrial Graphics Server directly using the following address:

<https://computer2/aig>

## See Also

[Log On to the Industrial Graphics Web Client](#)

## Configure User Access for an Industrial Graphics Web Client

**Security Roles** are used to manage authentication for Industrial Graphics applications. Any user that requires access to the Industrial Graphics Server via the Web Client needs to be associated with an Industrial Graphics security role.

Two Industrial Graphics security roles are created locally by Plant SCADA when you install an Industrial Graphics Server:

- **Industrial Graphics Users** — provides read-only access to a Plant SCADA system.  
The Windows® User Groups "AIGUsers" is created and associated with this role.
- **Industrial Graphics R/W Users** — provides read and write access to a Plant SCADA system.  
The Windows® User Groups "AIGUsersRW" is created and associated with this role.  
For more information about write access, see [Enable Tag Writes for Industrial Graphics Applications](#).

### To enable access for Industrial Graphics Web Client users:

1. Add the required users to the AIGUsers (read only) or AIGUsersRW (read and write) domain group on the Industrial Graphics Server.  
Or:  
Add the required users to a Windows domain group you have created, and then associate this group with the

appropriate Industrial Graphics security role.

For further instructions, see [Modifying the Members of a Security Role](#).

---

**Note:** To allow authorization in a distributed system, only add domain groups to an Industrial Graphics security role. You should avoid adding individual users to these roles.

2. Any users also need to be a member of a Windows group that is mapped to a valid role within your Plant SCADA project. This is required to enable I/O server requests. See [Roles](#).

---

**Note:** Additional configuration is required if you want to allow Industrial Graphics users to write to variable tags in your Plant SCADA system. See [Enable Tag Writes for Industrial Graphics Applications](#).

## See Also

[Use the Industrial Graphics Web Client](#)

### Log On to the Industrial Graphics Web Client

When you have completed the configurations described in [Secure the Industrial Graphics Web Client](#), authenticated users will be able log in to the Industrial Graphics Web Client. The System Management Server will provide HTTPS access to any users that can be validated from Microsoft Active Directory.

#### To log on to the Industrial Graphics Web Client:

1. Launch the Web Client in a supported browser (see [Browser and Mobile Support](#)).  
The login page appears.
2. Enter your user name and password.
3. Click **Login**.  
 Optionally, you could click on **Windows Integrated** to use the credentials of the user currently logged in to Windows.
4. Use **Sign Out** to log out of the session.

## See Also

[Configure User Access for an Industrial Graphics Web Client](#)

### Licensing for Industrial Graphics Applications in a Web Browser

You need a valid Industrial Graphics Web Server license to login and view application graphics from a web browser. Licensing also extends to hosting application graphics on external websites. For detailed information on licensing, see AVEVA Enterprise Licensing.

Acquiring the license is a two-stage process. After the Web Server starts up, it will first authenticate the user and then determine the user's authorization.

- **Authentication**

The Industrial Graphics Server will verify that a Web Client user is part of a domain group associated with the "Industrial Graphics Users" Security Role. See [Configure User Access for an Industrial Graphics Web Client](#).

If you add a new user to the group, the new user must log off and log in again for the change to take effect.

- **Authorization**

After a user is authenticated, the web server will attempt to acquire a license. If one is not available, then the web client will operate in a [Single Session Mode](#).

If the Web Server has acquired a valid license but did not renew it, the Web Client will be in [Grace Period](#) mode. A notification message will be displayed in the notification page. The message will be logged every time Web Client unsuccessfully attempts to renew the license.

---

**Note:** If the Industrial Graphics Server is already running when a license becomes available, it will not be able to acquire the license until after you have restarted the **AVEVA Industrial Graphics Service** in the Windows Component Services console.

If the Industrial Graphics Server is shutdown, it will release all the licenses it has acquired regardless of the number of client sessions and the current state of the license. All web client sessions will also be terminated immediately.

## Single Session Mode

If the Industrial Graphics Server does not acquire any license at startup, it will operate in the Single Session Mode, which is assigned on a first-come-first-serve basis.

- If a valid license is unavailable and the first session license has been acquired, the Web Client will notify the user that no licenses are available.
- The Industrial Graphics Server allows sessions a [Grace Period](#) if the server has acquired an unlimited license, but subsequently loses the license.

## See Also

[Licensing for Industrial Graphics Applications in a Web Browser](#)

## Grace Period

The Industrial Graphics Web Server will enter the grace period under the below conditions:

- If a valid license is acquired and later expires.
- If a valid license is acquired, but cannot be acquired again.

During the grace period, all Web Client features will be available for connected and new sessions. The Industrial Graphics Web page will display a visual notification that the Industrial Graphics Server is in grace period mode. The grace period is 14 days. After the grace period has ended, the server will continue to function and immediately switch to a single license for one session, on a first come first serve basis. If the same license is acquired, then the Web Server will exit the Grace Period mode.

## See Also

[Licensing for Industrial Graphics Applications in a Web Browser](#)

# Supported Graphical Elements and Known Limitations

## In This Chapter

[Known Limitations](#)

[Properties Supported by All Graphical Elements](#)

[Properties Supported by All Graphic Elements With Some Limitations](#)

[Supported Graphic Elements and Additional Properties](#)

[Supported Animations](#)

## Known Limitations

The Industrial Graphics Web Client will be enhanced with each product release. The following list summarizes the major limitations in the current release.

- Only English locale is supported by the Industrial Graphics Server.
  - For decimal points you need to enter "." (period).
  - For thousands separators you need to enter "," (comma).
- The status of write operations is not reflected by the status element or quality styles applied to elements. Design your pages so that an operator can determine if a write has been set or rejected by displaying the current value.
- Frame windows with special (unsupported) characters in the window name are not supported.
- Indirect tags are not supported.
- Windows Common Controls (WCC) refer to RadioButton Group, CheckBox, Editbox, ComboBox, Calendar, DateTime Picker, and ListBox.
- Bit state animations are not supported.
- Truth table animations are not supported for:
  - Visibility animations
  - Blink animations
  - Value display animations
  - Disable animations.
- Array radio button group animations are not supported.
- Hierarchical Name and Contained Name Display Animations are not supported.
- Multi pen trends are not supported.
- The following script functions are not supported:
  - SendKeys
  - SetBad
  - SetGood
  - SetInitializing
  - SetUncertain

- SignedAlarmAck
- SignedWrite
- WriteStatus
- WWControl
- WWExecute
- WWPoke
- WWRequest
- WWStringFromTime.

---

**Important:** When loading a graphic in the Web Client, unsupported features are logged as warning messages in the System Management Console (SMC) Logger. A generated report will include the warning path.

---

## See Also

[Properties Supported by All Graphical Elements](#)

[Properties Supported by All Graphic Elements With Some Limitations](#)

[Supported Graphic Elements and Additional Properties](#)

[Supported Animations](#)

## Properties Supported by All Graphical Elements

The following properties are supported by all graphic elements in a symbol:

- Angle
- X
- Y
- Width
- Height
- Enable
- Visible
- AbsoluteAnchor
- RelativeAnchor
- Transparency
- ElementStyle
- OwningObject
- Start
- End
- Radius
- Tension
- Custom Property Overrides

**Note:** The ElementStyle property enables users to select a defined element style and apply it to a graphic element. An element style includes options that define color properties like FillColor, LineColor, TextColor and OutlineColor.

## See Also

[Supported Graphical Elements and Known Limitations](#)

## Properties Supported by All Graphic Elements With Some Limitations

The following table lists the properties of graphic elements that can be incorporated with some HTML5 rendering limitations.

Graphic Element Property	Property Limitations
FillColor	Texture is not supported. Some other limitations on FillColor might apply to individual elements.
UnFilledColor	The same limitations of the FillColor property apply to the UnFilledColor property.
LineColor	Texture is not supported.
Font	Only supports the Font Size, Font Type, and Bold or Regular Font Style options of the Font property.
TextColor	Only the Solid Color option of the TextColor property is supported.
FillOrientation	RelativeToScreen is supported only when Color is set as 'Solid' or 'Fill' or 'UnFill'.
FillBehavior	Will always be set to Both.
Runtime Behavior	TabOrder, TabStop, ZoomPercent are not supported on any graphical element.
AutoScale	Is not supported.
ShowGraphic	Is supported.
Line	If the Line graphic is configured with the LineWeight property value more than 1px on the rim of the symbol, it will be Truncated.
Group/Embed Graphic	Supports Visibility, Blink, Disable, Push Button, User Input/Value Display, ShowSymbol and ActionScript animation. Changing the 'Treat as icon', and 'Enabled' properties by script is not supported.

Graphic Element Property	Property Limitations
	<p>%FillHorizontal and %FillVertical animations:</p> <ul style="list-style-type: none"> <li>- Only applies on the group. Not supported for sub elements.</li> <li>- for Button and Image are not supported .</li> <li>- 'Related To Screen' for both group and sub elements are not supported.</li> <li>- when the angle of the sub elements is changed is not supported</li> </ul>

## See Also

[Supported Graphical Elements and Known Limitations](#)

## Supported Graphic Elements and Additional Properties

The following table lists the graphic elements that can be incorporated with some limitations. Any graphic element not listed in this table is not supported.

The table includes a column that lists any limitations of the supported graphic element properties. When using graphics that incorporate color properties, the colors have the same limitations applicable for FillColor, LineColor, and other related graphic properties.

Graphic Elements	Supported Properties	Limitation Notes
Line Polyline Curve 2 Point Arc 3 Point Arc Connector	Line Style: <ul style="list-style-type: none"> <li>• LineWeight</li> <li>• LinePattern</li> <li>• StartCap</li> <li>• EndCap</li> </ul>	StartCap, EndCap: Supported but there maybe slight differences in rendering.
Button	ButtonStyle Text Word Wrap FillOrientation FillColor Alignment	ButtonStyle: Only Standard button style is supported. Word Wrap: Any caption that exceeds the button width will be truncated. FillOrientation:'RelativeToScreen' is only the Color is set. FillColor: Only Solid color is supported. If you have gradients with multiple colors selected, they will be converted to a single color using the first color.

Graphic Elements	Supported Properties	Limitation Notes
		Alignment: Only Centers alignment is supported.
Text	Alignment	Alignment: Only left-top supported.
Image	HasTransparentColor TransparentColor ImageStyle	FillColor, FillPercent and unfilledColor are not supported.
Text Box	Text TextFormat WordWrap LineWeight LinePattern Alignment Font	Flip is not supported
Status	Graphics Expression	Status Style: Only Default configuration is supported.
Trend Pen	Data Source	Trend time Period: Does not support Fixed Type.
Check Box		Selected Value Changes: Irrespective of user settings, only values submitted immediately are supported. Checked: Does not support Checked property.
Radio Button Group	Static Values and Captions States	Selected Value Changes: Irrespective of user settings, only values submitted immediately are supported.
Edit Box		Selected Value Changes: Irrespective of user settings, only values submitted immediately are supported.
Combo Box		Selected Value Changes: Irrespective of user settings, only values submitted immediately are supported. Type: Does not support

Graphic Elements	Supported Properties	Limitation Notes
		MaximumLength
Calendar		<p>Selected Value Changes: Irrespective of user settings, only values submitted immediately are supported.</p> <p>Bold Dates: Is not supported</p> <p>ShowToday: Will always display, even if checked out.</p> <p>Calendar Colors: Is not supported</p>
DateTime Picker		<p>Selected Value Changes: Irrespective of user settings, only values submitted immediately are supported.</p> <p>Configuration: Is not supported.</p> <p>Calendar DropDown Colors: Not supported.</p>
ListBox		<p>Selected Value Changes: Irrespective of user settings, only values submitted immediately are supported.</p>
ConnectorPoint		Moving the ConnectorPoint is not supported.

## See Also

[Supported Graphical Elements and Known Limitations](#)

## Supported Animations

The following table lists animations supported by the Web Client. When using animations that incorporate color properties, the selected animation colors have the same limitations applicable for FillColor, LineColor, and other related graphic element properties.

**Note:** The following table lists only supported animations. Any animations not listed are not supported by the Web Client.

Animation	Limitations
Element Style	Refer to the graphic elements limitations regarding color, line, and font. The same limitations applicable to graphic elements also apply to Element Style

Animation	Limitations
	<p>animation.</p> <p>Expression Or Reference: Time and Elapsed Time are not supported.</p> <p>Changes to an element style definition become effective only after redeploying or publishing the application.</p> <p>Does not support Windows Common Controls.</p>
Fill Style Line Style Text Style	<p>Refer to graphic element limitations regarding color, line, and font. The same limitations applicable to graphic elements also apply to Element Style animation.</p> <p>Expression Or Reference: Time and Elapsed Time states are not supported.</p> <p>Does not support Windows Common Controls.</p> <p>TextColor: Only Solid Color is supported.</p>
% Fill Horizontal % Fill Vertical	<p>Only RelativeToGraphic is supported.</p>
Slider Horizontal Slider Vertical	<p>A slider can support data binding with a custom property.</p> <p>Slider only supports On Mouse Release when writing to a custom property.</p> <p>Slider does not support Cursor anchor</p> <p>Slider does not support Show Tooltip.</p>
Width Height	<p>Does not support Windows Common Controls.</p>
Orientation	<p>The Relative Anchor or RelativeOrigin are not supported when setting Orientation animation. (Only supported anchor at center).</p>
Value Display	<p>The following display value animations are supported:</p> <ul style="list-style-type: none"> <li>• Boolean value</li> <li>• Analog value</li> <li>• String value</li> <li>• Time value</li> </ul>
Disable	<p>The RadioButton graphic element does not support Disable animation.</p>

Animation	Limitations
Point	Point animation on Curve and ClosedCurve are not supported.
Tooltip	<p>Industrial Graphics Web Client supports Tooltip animation except for Image, Radio Button, Button, and Text graphic elements.</p> <p>Tooltip animation is not supported on Group or embedded symbol.</p> <p>Does not support Windows Common Controls.</p>
Action Scripts	Actions scripts are supported except for functions that invoke dialogs. The ShowGraphic animation is only supported for a Frame window.
Show Symbol	<p>Title: Only supports the default option.</p> <p>Type: Supports Modal and Modeless where the Close button option is checked.</p> <p>Position: Only supports the center where the Client Area x,y value is 0,0</p> <p>Size: Only supports Relative To symbol option.</p> <p>Shortcut: Is not supported.</p>
ShowGraphic	<p>Title: Only supports the default option.</p> <p>Type: Support Modal and Modeless with the Close Button option checked</p> <p>Position: Only supports center, Client Area and x,y value is 0,0</p> <p>Size: Only supports the Relative To Symbol option</p> <p>Shortcut: Is not supported</p>
Hyperlink	Does not support on windows common controls, button, text, and textbox.

## See Also

[Supported Graphical Elements and Known Limitations](#)

## Troubleshooting the Industrial Graphics Web Client

If you are having problems displaying the Industrial Graphics Web Client, you should firstly check the settings described in the topic [Confirm the Settings for an Industrial Graphics Server](#).

**If the Web Client is not displayed, verify the following settings:**

- Confirm that the AVEVA Industrial Graphics Service is running in the Windows® Component Services console.
- Confirm that the AVEVA Plant SCADA Connectivity Service is running in the Component Services console.
- Confirm the user account accessing the Industrial Graphics Web page is associated with the "Industrial Graphics Users" security role (see [Configure User Access for an Industrial Graphics Web Client](#)).
- Confirm that the browser settings have been configured to allow cookies for this web site. Private browsing modes are not supported as they are designed to block cookies by default.
- If IIS is running on the same machine, it may conflict with the Industrial Graphics Web Server.  
If IIS is running and it is not being used, stop and disable the service. If IIS must run, configure to use a port other than Port 80 to avoid conflict.

**If you get a "HTTP Error 404.0 - Not Found" error message when you attempt to launch the Industrial Graphics Web Client:**

- Ensure that Web Client is enabled on the Industrial Graphics Server. Contact your administrator.

**If any graphic primitives or animations are not visible or working as expected:**

- Check the list of unsupported features to see if it is not yet supported by the Web Client.
- Run the Web Client locally via <http://localhost/aig>, browse to the symbol, and look for the compatibility icon to view a list of the unsupported items in the graphic.
- If you use a primitive or animation in a graphic that is not supported, it will be removed and a message logged. The logged message contains the path to a generated CSV report file. You can view the Logger message, and navigate to the report for more information.

**If the selected graphic is displayed but no data is shown or animations are not updated:**

- Ensure Plant SCADA runtime is running if the graphics point to Plant SCADA tags. You should also check if the I/O servers are running.

**If I/O updates are not appearing in the Web Client:**

- This could mean Runtime Manager is not running as a service on your Plant SCADA servers. Check the [Log Viewer](#) for the following message:  
"Failed to connect to 'scada'. Error description: No connection was established with a service endpoint".  
To configure Runtime Manager to run as a service, see [Configure a Runtime Computer for Encryption](#).

**If you get a "Certificate Error: Your connection to this site is not secure" message when accessing the Web Client:**

- This means a valid certificate is not available on the client. Review the process described in the topic [Use SMS Certificates with Web Applications](#).

## See Also

[Manage the Industrial Graphics Web Client](#)

## Industrial Graphics Web Client Error Handling

Industrial Graphics Web Client provides a uniform error page for displaying expected errors. The error page will be displayed for the following scenarios:

- **No Application** - The Industrial Graphics Web Client could not detect the necessary supporting files for the last application.
- **Permission Denied** - The user currently logged in does not have adequate permission to view the web page. Add the user to the Industrial Graphics Users security role (see [Configure User Access for an Industrial Graphics Web Client](#)).
- **No License Available** - A valid license status is not obtained for the current web session. Confirm that a valid web client license is available.

## See Also

[Troubleshooting the Industrial Graphics Web Client](#)

## Browser and Mobile Support

### Browser Support

Industrial Graphics Web Client has been tested with the following web browsers:

- Google Chrome
  - Chrome on desktop version 64.0.3282.186 or later
  - Chrome on Android Phone & Pad version 64.0.3282.137 or later
- Microsoft Edge
  - Microsoft Edge on Windows 10 version 41.16299.15.0 or later
  - Microsoft Edge on Surface version 41.16299.248.0 or later
- Apple Safari for IOS version 11.2.6

You will receive an error message if the web browser is not supported.

### Mobile Device Support

Industrial Graphics Web Client supports the following mobile devices and their default browsers:

- Apple iPhone
- Apple iPad

- Android Phone & Tablet
- Microsoft Surface

## See Also

[Log On to the Industrial Graphics Web Client](#)

## Situational Awareness Projects

A Situational Awareness project is designed to support abnormal situation management for operators. It can be used to create a Plant SCADA project that provides accurate information to an operator in a way that can be perceived and acted upon quickly, without overwhelming their cognitive abilities.

This objective is achieved through:

- A consistent look and feel for graphical content.
- The use of standardized color settings, optimized for perceiving process data and alarms.
- The use of objects that conform to an optimized display hierarchy.
- Easy access to information related to the selected context.
- The inclusion of navigation aids and an operator dashboard.

To support these principles, a default layout is applied to projects created using a Situational Awareness Starter project. The layout includes common elements such as a header bar, content area, and a dashboard that displays contextual information and navigational tools. See [Default Layout](#).

Situational Awareness projects differ to other pure template-based projects as they use "Workspace" technology to manage the content of a Master Page that appears on each configured client screen. Each pane in a Master Page displays a page that can be updated independently at runtime based on navigational or contextual changes. The Workspace is responsible for managing contextual updates to those panes by using the association between the hierarchical location of a detected change, and the type of content each pane is configured to display. See [Key Components of a Situational Awareness Project](#).

A set of library objects are also available via an included library project (named "SA\_Library"). The included objects support the Workspace autofill functionality at runtime and maintain a consistent look and feel across a project's display pages (see [Libraries](#)).

---

**Note:** Avoid making changes to the included SA\_Include project for use as a runtime project. A Plant SCADA upgrade will install a new version of the project, which will overwrite any changes you make.

## See Also

[Alarms](#)

[Interlocks](#)

[Create a Situational Awareness Project](#)

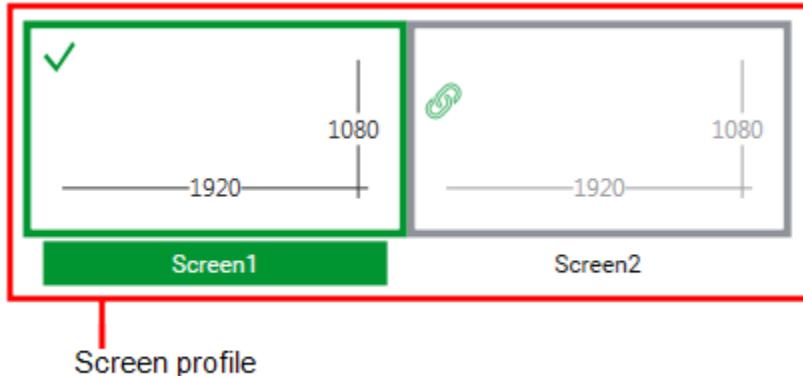
[Customize a Situational Awareness Project](#)

## Key Components of a Situational Awareness Project

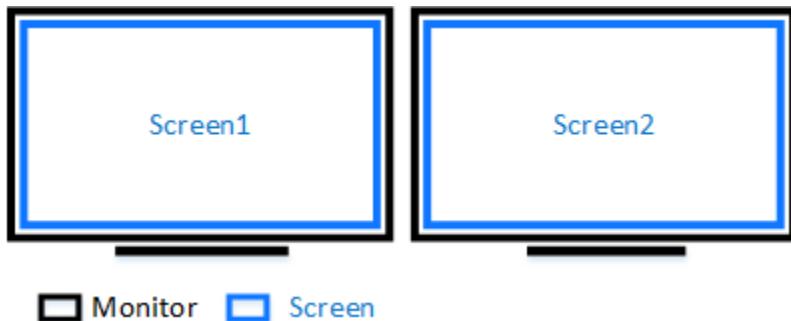
A project created from Situational Awareness Starter Project uses the Workspace architecture to enable the runtime environment to respond to contextual changes. This topic provides an introduction to the key components of this architecture.

- **Screen Profile**

A screen profile defines the arrangement of screens used by a Plant SCADA display client with multiple monitors. You can create a screen profile in the Setup activity in Plant SCADA Studio (see [Screen Profiles](#)).



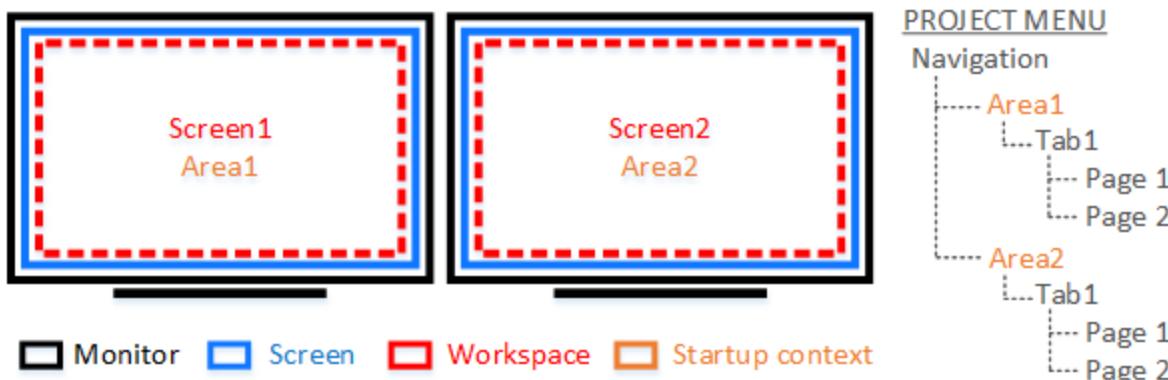
You can then apply a screen profile to a display client using the Screen Setup page of the Computer Setup Wizard.



- **Workspace**

The Workspace is the engine that manages the contextual content changes within each screen. It enables contextual navigation, and optionally provides a tailored, situational awareness-aligned operator experience.

A startup context is set for each Workspace by linking each screen to a Level 1 entry in a project menu named "Navigation". This provides a starting point for any contextual navigation that occurs within the Workspace.

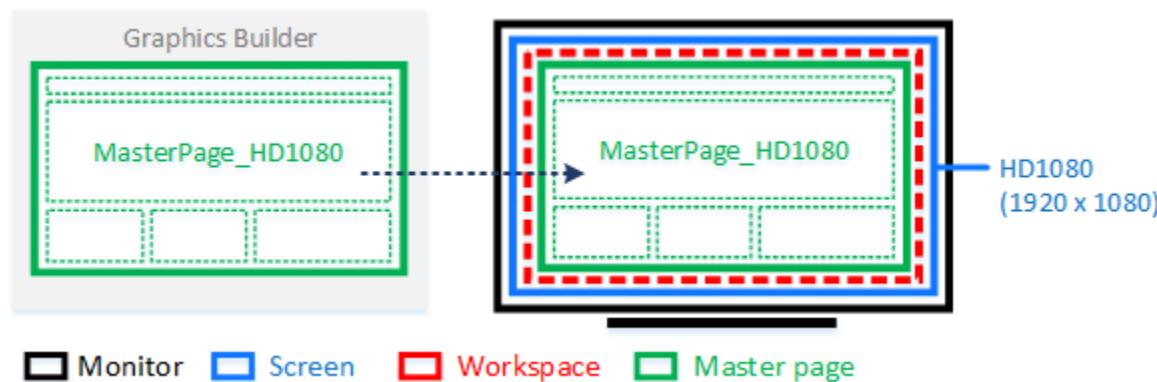


Startup context is set for each screen via the **Screen Setup** page of the Computer Setup Wizard.

For information on how to configure an appropriate navigation menu, see [Prepare the Navigation Menu](#).

- **Master Page**

A master page provides the layout for the content that is managed by a Workspace. It differs from other graphics pages, as it is made up of a set of Panes that each host individual pages at runtime (see "Panes" below).



The size of a master page needs to match the resolution of the screen on which it will display. The master pages that are provided with Plant SCADA come in two resolutions:

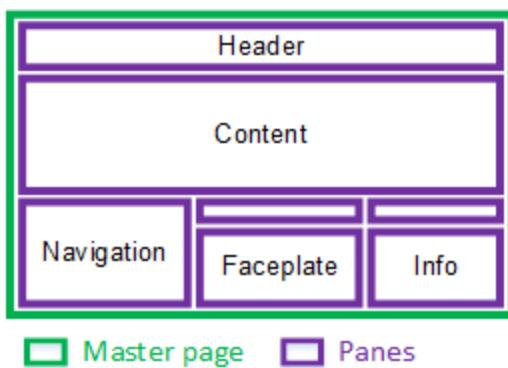
- HD1080 - 16:9 (1920 x 1080)
- UHD4K - 16:9 (3840 x 2160)

To specify the master page that displays on a screen when runtime starts, use the **Screen Setup** page of the Computer Setup Wizard.

To create your own master page, see [Create a Master Page for a Customized Workspace](#).

- **Panes**

Panes are sections within a master page that form a framework for the content that appears at runtime. Each pane hosts a page that can be dynamically updated.



The properties associated with a pane allow you to specify how it behaves when the context of a client changes, the type of content it can display, and how a page fills the space within the pane (see [Configure Panes on a Master Page](#)).

If your project is based on Plant SCADA's Situational Awareness Starter Project, a default layout of panes is available via the included master pages (see [Default Layout](#)).

There are two ways to initiate an update within this architecture:

- Commands — typically used on the Header Bar or a navigation menu to display a page.
- Contextual updates — the content of a pane may update when a piece of equipment comes into context.

Contextual updates are enabled by associating "Content Types" with the equipment and pages in a project. This allows a pane to update automatically when equipment of a particular type comes into context. See [Autofill](#).

## See Also

[Configure Your Project](#)

## Autofill

The Workspace autofill system allows the content on a screen to be automatically updated at runtime based on the current context of a client. Autofill works for all windows that are registered with the workspace, such as the master page window and any popup windows.

For example, if a pump is selected on a display page, the autofill process will update any relevant panes with content that directly relates to the pump, and optionally its parental or child hierarchy.

Autofill works by associating a particular type of content (such as a faceplate or page) with each equipment definition in a project's equipment hierarchy. To align with this, the panes in a workspace can be configured to automatically display matching content types.

When a piece of equipment comes into context, any pane that has autofill enabled will check if the associated content type is the same as the content it is configured to display. If a match is made, the pane will display the content.

To engage Plant SCADA's autofill functionality, you need to perform the following tasks:

- Associate content types with equipment (see [Add Equipment Using Equipment Editor](#)).
- Specify the type of content each pane will display (see [Configure Panes on a Master Page](#))

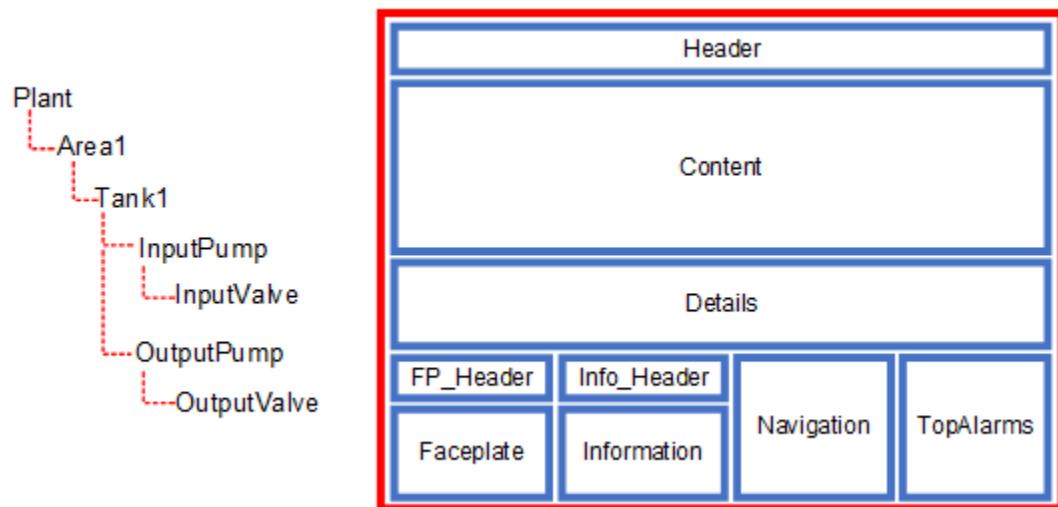
- Specify the type of content for pages in your projects (see [Assign a Content Type to a Page](#))
- Choose the Autofill behavior you want for your system, referred to as the "Context Mode". The default mode is "Current, Then Up, Then Down".

The Context Mode determines if the workspace(s) use content from equipment on higher and lower levels of the equipment hierarchy when context changes, rather than just the selected piece of equipment. To specify how autofill behaves for your system, see [Set the Context Mode for a Workspace](#).

## Example

The following example demonstrates how Context Modes and the configuration of equipment and workspaces allows content to be automatically filled on context change.

The equipment hierarchy below features a tank (Tank1) with an input pump and output pump. Assume that Plant SCADA's autofill functionality will present the content associated with this hierarchy on the following arrangement of panes.



To enable autofilling, the following settings are configured.

### Equipment hierarchy

When you define equipment in a Plant SCADA project, you use the **Content** property to associate each piece of equipment with a particular type of content. In the case of the example above, the following equipment hierarchy is configured.

Equipment	Level	Content	Content Type	Description
Plant	1	Overview_L1	L1	Associated with a level 1 page.
Plant.Area1	2	Area_L2	L2	Associated with a level 2 page.
Plant.Area1.Tank1	3	Tank1_L3	L3	Associated with a level 3 page.
Plant.Area1.Tank1.InputPump	4	Drive_FP	FP	Associated with a drive faceplate.

Equipment	Level	Content	Content Type	Description
Plant.Area1.Tank1.InputPump.InputValve	5	Valve_FP	FP	Associated with a valve faceplate.
Plant.Area1.Tank1.OutputPump	4	Drive_FP	FP	Associated with a drive faceplate.
Plant.Area1.Tank1.OutputPump.OutputValve	5	Valve_FP	FP	Associated with a valve faceplate.

For more information, see [Add Equipment Using Equipment Editor](#).

### Panes

Panes have two properties that determine the outcome of the autofill process.

- **ContentTypes** — defines the type of content the pane will display. You can specify more than one, separated by commas.
- **FillMode** — determines how a pane behaves as context changes. Four fill modes are supported:
  - Static — The content does not change.
  - Autofill — The content will update if the equipment in context has a matching content type.
  - AutofillContextMustMatch — The content will update if the equipment is a direct contextual match and it has content of the type required by the pane (see below).
  - StaticContextMustMatch — The content does not change, but any associations are updated if the equipment is a direct contextual match (see below).

The "ContextMustMatch" options allow you to exclude a pane from a Context Mode setting that allows a workspace to autofill at higher and lower levels of the equipment hierarchy. This means only content that is directly associated with the selected equipment will display.

In the case of the example above, the panes are configured as follows:

Pane	Content Types	Fill Mode	Context Update
Header	—	0 = Static	The pane does not change.
Content	L1, L2	1 = Autofill	The pane will automatically display a level 1 or level 2 page.
Details	L3, L4	1 = Autofill	The pane will automatically display a level 3 or level 4 page.
Faceplate	FP	3 = AutofillContextMustMatch	The pane will only display content associated with the equipment if the

Pane	Content Types	Fill Mode	Context Update
			current context is a direct match.
FP_Header	—	4 = StaticContextMustMatch	The content of the pane will remain static however, any associations will be updated.
Information	—	4 = StaticContextMustMatch	The content of the pane will remain static however, any tag associations will be updated.
Info_Header	—	0 = Static	The pane does not change.
Navigation	—	0 = Static	The pane does not change.
TopAlarms	—	0 = Static	The pane does not change.

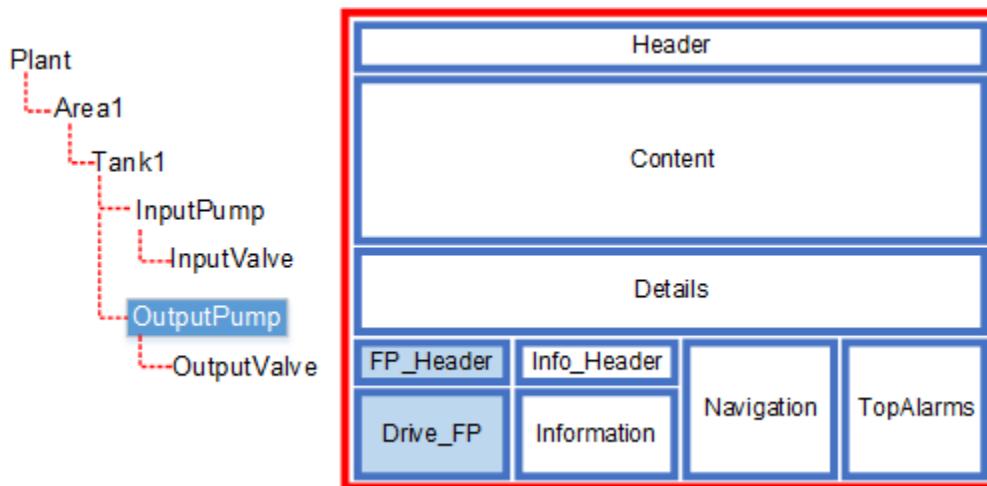
For more information, see [Configure Panes on a Master Page](#).

**Note:** A page can be classified as a particular content type. This allows the system to know what it is, and to match it with a pane content type configuration. See [Assign a Content Type to a Page](#).

The following demonstrates the expected outcome for the above example based on the three available context modes.

- **WS\_CONTEXTMODE\_CurrentOnly**

Only the current piece of equipment is considered when panes are matched with content.



In the case of the example above, the following panes will update when the "OutputPump" is in context.

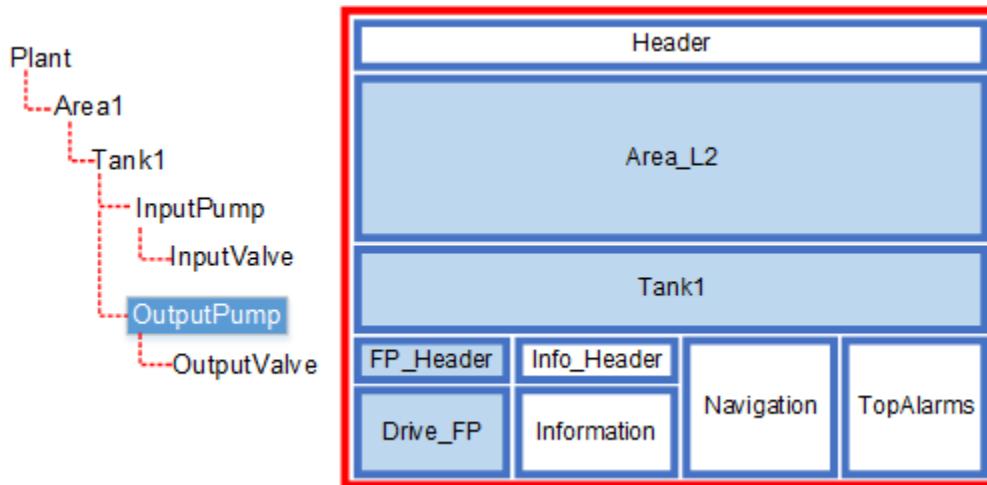
- **Faceplate pane** — 'Plant.Area1.Tank1.OutputPump' is associated with a drive faceplate, which matches

the content type defined for the Faceplate pane. As the FillMode is set to "AutofillContextMustMatch", the faceplate will automatically display in this pane.

- **FP\_Header pane** — The FillMode for this pane is set to "StaticContextMustMatch" which means the displayed page will not change, however any associations will update (for example, the Display Name field).

- **WS\_CONTEXTMODE\_CurrentThenUp**

The current piece of equipment is considered when panes are matched with content. When this process is complete, content types associated with higher levels of the equipment hierarchy are matched with any remaining panes.



In the case of the example above, the following panes will update when the "OutputPump" is in context.

- **Faceplate pane** — 'Plant.Area1.Tank1.OutputPump' is associated with a drive faceplate, which matches the content type defined for the Faceplate pane. As the FillMode is set to "AutofillContextMustMatch", the faceplate will automatically display in this pane.
- **FP\_Header pane** — The FillMode for this pane is set to "StaticContextMustMatch" which means the displayed page will not change, however any associations will update (for example, the Display Name field).

Moving up the hierarchy, 'Plant.Area1.Tank1' is considered next. The following panes will update:

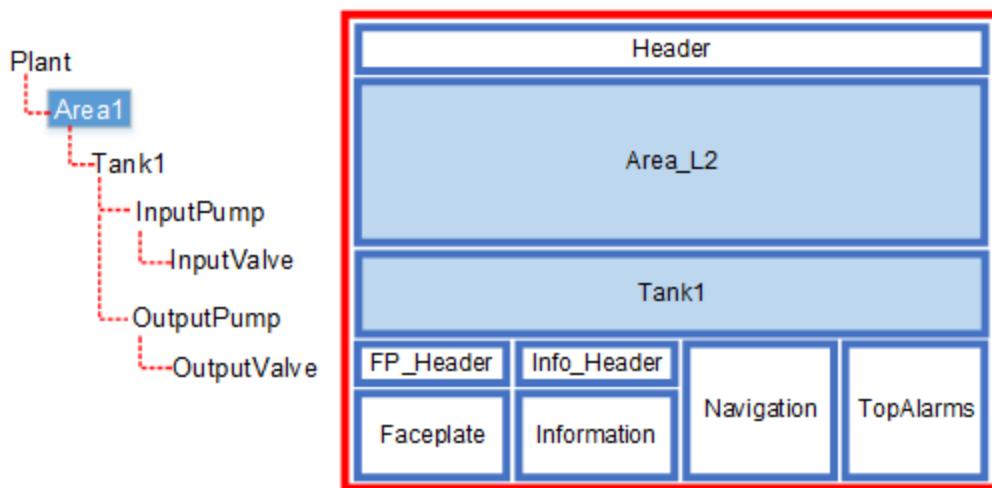
- **Details pane** — 'Plant.Area1.Tank1' is associated with a level three page ("Details\_L3"), which matches the content type defined for the Details pane. As the FillMode is set to "Autofill", the page will automatically display.

Moving up the hierarchy again, 'Plant.Area1' is considered next. The following panes will update:

- **Content pane** — 'Plant.Area1' is associated with a level two page ("Area\_L2"), which matches the content type defined for the Content pane. As the FillMode is set to "Autofill", the page will display.

- **WS\_CONTEXTMODE\_CurrentThenUpThenDown**

The current piece of equipment is considered when panes are matched with content. When this process is complete, equipment on higher levels of the hierarchy are matched with any remaining panes, then equipment on lower levels of the hierarchy are considered.



In the case of the example above, the following panes would be updated when 'Plant.Area1' is in context.

- **Content pane** — 'Plant.Area1' is associated with a level two page ("Area\_L2"), which matches the content type defined for the Content pane. As the FillMode is set to "Autofill", the page will automatically display.

As there are still Autofill panes that are not filled (for example, "Details Pane"), the system begins moving up the hierarchy looking for content.

'Plant' is considered next. It only has content "Overview\_L1" of content type L1, and the only pane that supports that ("Content") has already been filled. The system will then go back to the original context, 'Tank1', and move down the hierarchy to its first child. Only the first child is ever considered when moving down the hierarchy.

Moving down the hierarchy, 'Plant.Area1.Tank1' is considered next. The following pane will update:

- **Details pane** — 'Plant.Area1.Tank1' is associated with a level three page ("Details\_L3"), which matches the content type defined for the Details pane. As the FillMode is set to "Autofill", the page will display.

Moving further down the hierarchy to 'Plant.Area1.Tank1.OutputPump', the FillMode setting for the remaining panes takes affect as the current context is no longer a direct match. The following updates will occur:

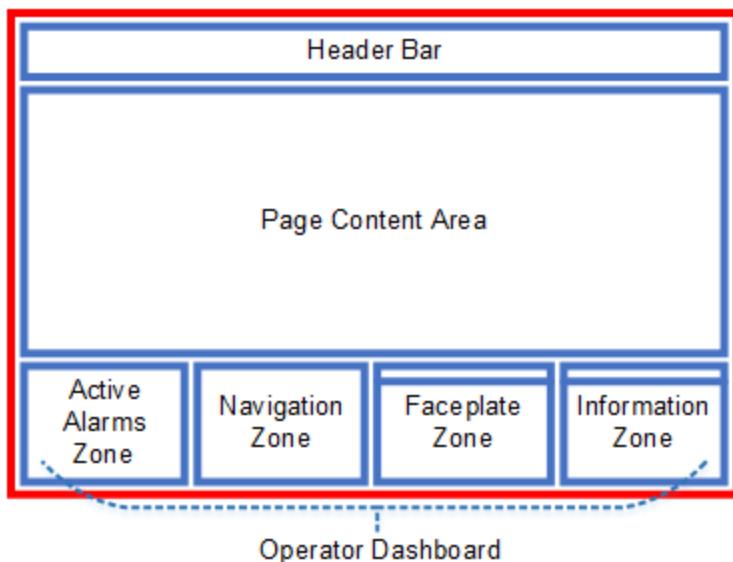
- **Faceplate pane** — As the FillMode is set to "AutofillContextMustMatch", the pane will display its default blank page.
- **FP\_Header pane** — As the FillMode is set to "StaticContextMustMatch", the pane will display its default blank page.

## See Also

[Key Components of a Situational Awareness Project](#)

## Default Layout

When you create a project using the Situational Awareness Starter project, the default workspace will include the following arrangement of panes.



The [Header Bar](#) includes a set of commands that an operator may require regular access to during runtime.

The Page Content Area displays the pages you create to represent your plant or production facility (see [Create Content Pages](#)).

An operator dashboard also appears along the bottom of the interface. It includes the following sections:

- [Active Alarms Zone](#) (4K only)
- [Navigation Zone](#)
- [Faceplate Zone](#)
- [Information Zone](#).

A set of [Default Alarm Pages](#) can also be launched via the Header Bar.

The master page that is a foundation for this arrangement of panes is available in two screen sizes:

- HD1080 (1920 x 1080, 16:9)
- UHD4K (3840 x2160, 16:9)

---

**Note:** The Active Alarms Zone is not available on the HD1080 version of the workspace. Instead, an HD1080 project includes a **Top 5 Alarms** button on the [Header Bar](#).

In a project created from the Starter Project, the default layout is only applied to a single screen profile for a primary monitor. If you would like to create a screen profile for a display client with multiple monitors, see [Set Up a Screen Profile for Multiple Monitors](#).

## See Also

[Key Components of a Situational Awareness Project](#)

## Header Bar

The default layout for a Situational Awareness project comes with a header bar. It includes a set of commands

that an operator may require regular access to during runtime. These commands can be customized.

Icons	Common tools	Description
	Header Logo	<p>By default the Plant SCADA logo is used. However, this is configurable. To display your own logo in the allocated space, import the logo into Graphics Builder and save it as a symbol in your project. Assign the name of the symbol (in the format "library.symbol") to the project INI parameter [Page]Logo. See <a href="#">Customize the Logo on the Header Bar</a>.</p>
	Home	<p>Displays the home page. The home page is specified in a project's Navigation Menu. The menu's highest Level 1 entry sets the context for the workspace, the Target Page field for the entry identifies the home page. See <a href="#">Prepare the Navigation Menu</a>.</p>
	Page Navigation	<p>Back, Forward and Up page navigation commands. Back and Forward allow you to move between pages in the current browse history or stack. Up allows you to move up a level in a set of hierarchical pages. In multi-monitor systems, the stack is per monitor and not linked. When a log out occurs, the stack is deleted.</p>
	Trends	<p>Opens the trend summary page for the operational area, allowing you to access predefined trends (see <a href="#">Default Trend Pages</a>).</p>
	Active Alarms	<p>Displays alarms that are unacknowledged, or acknowledged and still in an alarm state (see <a href="#">Default Alarm Pages</a>).</p>
	Historical Alarms	<p>Displays an historical log of alarms (see <a href="#">Default Alarm Pages</a>).</p>

Icons	Common tools	Description
	Shelved Alarms	Lists alarms that are temporarily disabled (see <a href="#">Default Alarm Pages</a> ).
	Hardware Alarm	Displays a list of hardware alarms that are unacknowledged, or acknowledged and still in alarm state (see <a href="#">Default Alarm Pages</a> ). A flashing circle underneath the hardware alarm icon indicates there are unacknowledged hardware alarms.  
	Top 5 Alarms	Displays the top 5 active alarms visible to the operator. This button is not included on a header bar for a UHD4K workspace.
<b>Overview (L2)</b> MyPlant > F & B	Page Name	Title of page selected in navigation zone and displayed in page content area. If page is not in the navigation menu the message "Page does not exist in Navigation menu" will be displayed. Underneath the title of the page the menu breadcrumb displays.
<b>User: John Smith</b>	User	Name of logged in user. For example, John Smith is the current logged in user.
	Login	Allows a user to log in to a runtime system, providing access to additional functionality based on the user group or groups to which they belong. The drop-down menu provides access to the Logout command.
	Options	Allows you to toggle visibility of some display options for the objects in a runtime system. Select the relevant option from the drop down list.

Icons	Common tools	Description
		 Show or hide Control Links (see <a href="#">Add a Control Link</a> ).  Show or hide engineering/PV values.  Show or hide labels.
	Help	Opens a pane that displays runtime help.
	Date/Time	The current date and time.
	Shutdown	Shuts down the runtime system.

#### To customize a menu command:

1. In the **Visualization** activity, select **Menu Configuration**.
2. In the **Page** column, locate the menu entries that use the name "Headerbar".
3. Select the item that you want to customize. The **Level 1** column should provide a description of the header bar commands.
4. Go to the **Menu Command** field.
5. Enter a Cicode function, or select one from the drop-down list.
6. Modify the parameters as required.
7. Click **Save**.

**Note:** If you want to change the content of the header bar, you should make a copy of the header bar page (for example, "DefaultHeaderbar\_HD1080" or "DefaultHeaderbar\_UHD4K") and modify it. You can then assign the updated page as the Default Page in the master page's header pane.

#### See also

[Default Layout](#)

#### Active Alarms Zone

**Note:** The Active Alarms Zone only appears on the operator dashboard if the workspace is based on the high definition 4K master page ("Master\_PageMenu1\_UHD4K"). For an HD1080 project, you can view active alarms via the **Top 5 Alarms** button on the [Header Bar](#) or on the [Alarm page](#).

The Active Alarms Zone displays a list of alarms for a runtime system.

Date	Time	Name	Desc	State
◆ 23/08/2017 09:04:...		Drive10_Running_Fa...	Drive10_No_Runn...	ON
◆ 23/08/2017 09:04:...		Drive11_Running_Fa...	Drive11_No_Runn...	ON
◆ 23/08/2017 09:04:...		Drive12_Running_Fa...	Drive12_No_Runn...	ON
◆ 23/08/2017 09:04:...		Drive13_Running_Fa...	Drive13_No_Runn...	ON
◆ 23/08/2017 09:04:...		Drive14_Running_Fa...	Drive14_No_Runn...	ON
◆ 23/08/2017 09:04:...		Drive15_Running_Fa...	Drive15_No_Runn...	ON
◆ 24/07/2017 11:46:...		Drive4_Drive5_Runn...	Drive4_Drive5_N...	ON
◆ 24/07/2017 11:46:...		Drive5_Running_Fal...	Drive5_No_Runni...	ON
◆ 24/07/2017 11:46:...		Drive6_Running_Fal...	Drive6_No_Runni...	ON
◆ 24/07/2017 11:46:...		Drive7a_Running_Fa...	Drive7a_No_Runn...	ON

By default, the alarms are sorted according to the Highest Priority Order setting for the Alarm Server in Plant SCADA Studio. Within each priority, they are listed in the following order:

- Unacknowledged and active
- Unacknowledged and inactive
- Acknowledged and active
- Disabled or Shelved

Within these groups, alarms are listed according to the time they occurred.

If your project is based on the Situational Awareness Starter Project, the default columns presented for each row in the alarms list will include:

- Date – the date when the alarm became active
- Time – the time when the alarm became active
- Name – the name of the alarm tag
- Description – a description of the alarm
- State – the current state of the alarm.

---

**Note:** The default arrangement of columns in the alarms list is based on a display format named "TopActiveAlarms\_UHD4K". You can adjust this format in your project using the **Parameters** view in the **Setup** activity.

---

Each alarm also has an icon that indicates the severity and state of the alarm. The appearance of an icon represents the following:

## Color and shape

The color and shape of an alarm icon indicates the priority of an alarm.

If your project is based on the Situational Awareness Starter Project, the following default icons will be used to indicate priority:

-  High priority alarm
-  Medium priority
-  Low priority alarm

If these icons have been customized, their appearance will be determined by the display properties specified for each alarm priority (see [Configure Display Properties for an Alarm Priority](#)).

## Flashing icon

If an alarm icon is flashing it means the alarm is unacknowledged.

## Transparent icon

If the color of a flashing icon appears at half-intensity, it indicates that the alarm has cleared but remains unacknowledged. The icon will remain transparent and continue to flash until the alarm is acknowledged or reactivated.

You can use the Active Alarms Zone to perform the following actions at runtime:

## Navigate to the associated object

To bring the object associated with an alarm into context, right-click on it and select **Navigate** from the menu that appears.

## Acknowledge an alarm

To acknowledge an alarm, right-click on it and select **Acknowledge** from the menu that appears.

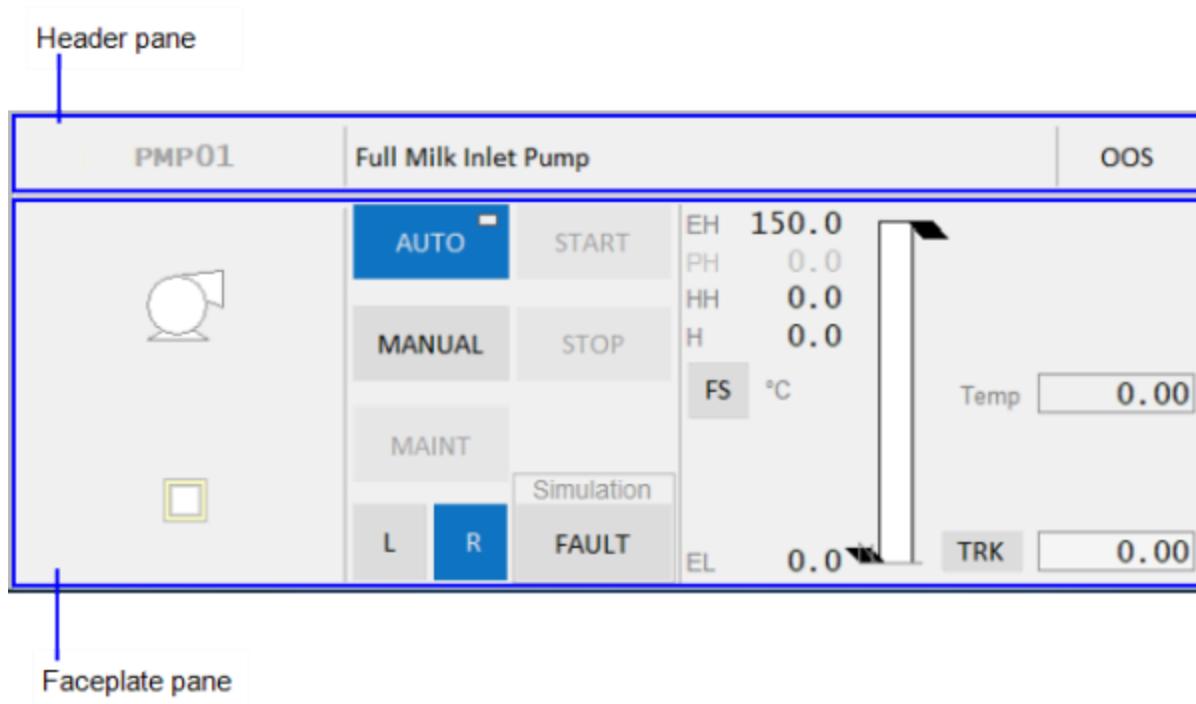
## See Also

[Default Layout](#)

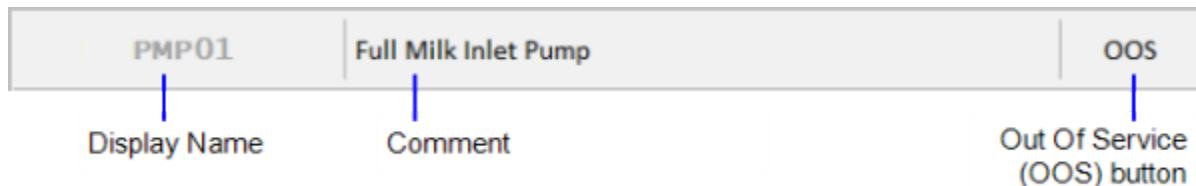
## Faceplate Zone

If an object on a display page has a faceplate associated with it, the faceplate will display in this section of the dashboard when the object is selected.

The area the faceplate section occupies is constructed using two panes.



## Header pane



The header pane includes three parts:

- **Display name** — If a Display Name is configured for the selected piece of equipment, it will display here. Cluster context is also included.  
If a Display Name is not configured for the object, a truncated equipment path for the object will appear. If required, the equipment path can be displayed as a tool tip.
- **Comment** — If a Comment is configured for the selected piece of equipment, it will display here.
- **Out Of Service (OOS) button** — Allows you to put the selected piece of equipment out of service, for example, for maintenance or repair.

While the equipment is out of service, its alarms are suppressed.

The OOS button has three operational states:

- If the button is light grey, it means the equipment is currently operational and cannot be placed out of service. You will need to stop the piece of equipment before you can use the OOS button.
- If the button is active, it indicates that you can use the button to place the piece of equipment out of service.
- If the button is dark grey, it indicates that the piece of equipment is currently out of service.

## Faceplate pane

The faceplate pane behaves in the following ways:

- If the selected object has a faceplate associated with it, it will display here.
- If the selected object does not have a faceplate associated with it, the pane will display "No faceplate available".
- If no object is selected, this pane will display "No object is selected".

## See also

[Default Layout](#)

## Information Zone

The Information Zone appears on the Operator Dashboard. It includes a set of tabs that present different types of information about the object that is currently in context. These tabs include **Alarms**, **Trends** and **Interlocks**.

## Alarms

The Alarms tab displays alarms associated with the object that is currently in context. Alarms are sorted in order of importance with the highest priority alarms displayed at the top of the list. When an alarm clears, it is removed from the list.

ALARM	TREND	INTERLOCKS
◆ 03:49:26 PM V_OP		Drive1_V_OP_HH_P1
◆ 10:10:46 AM S_Running		Drive1_Running_False_P1
▲ 03:49:26 PM OP_TRK		Drive1_OP_TRK_H_P2
▼ 03:49:26 PM V_Feedback		Drive1_V_Feedback_H_P3

Cause	Drive is overheating.
Response	Check the thermostat.
Response Time	00:05:00
Consequence	Drive may fail.

If your project is based on the Situational Awareness Starter Project, the default columns presented for each row in the alarms list will include:

- Time – the time the alarm became active.
- Item – the associated equipment item.
- Tag Name – the name of the alarm tag.

**Note:** The default arrangement of columns is based on a display format that is defined in the SA\_Include

---

system project. The format will be named "InfoAlarm\_HD1080" or "InfoAlarm\_UHD4K", depending on the resolution of the workspace master page. You can adjust this format by modifying [Format]FormatName in the **Parameters** view of the **Setup** activity (see [Project Database Parameters](#)).

---

Each alarm also has an icon that indicates the severity and state of the alarm. If your project is based on the Situational Awareness Starter Project, the following default icons will be used to indicate priority:

-  High priority alarm
-  Medium priority
-  Low priority alarm

If these icons have been customized, their appearance will be determined by the display properties specified for each alarm priority (see [Configure Display Properties for an Alarm Priority](#)).

If an alarm icon is flashing it means the alarm is unacknowledged. If the color of a flashing icon appears at half-intensity, it indicates that the alarm has cleared but remains unacknowledged. The icon will remain transparent and continue to flash until the alarm is acknowledged or reactivated.

You can use the right-click menu on the alarms list to perform the following actions at runtime:

1. **Acknowledge** an alarm.
2. Shelve an alarm. Select one of the following options:
  - **Shelve for ...** — switches the view to the Shelve form, which allows you to disable an alarm for a specified period of time, or indefinitely.
  - **Shelve until ...** — switches the view to the Shelve form, which allows you to disable an alarm until a specified date and time.
3. **Restore** an alarm.

The panel below the alarms list shows user-configured [Provide Cause and Response Information to Operators](#) information for the selected alarm. Up to eight sets of cause and response information can be associated with an alarm; the arrow buttons to the right of the panel allow an operator to navigate through the suggested responses. To add cause and response information to alarms, see [Add Cause and Response Information to Alarms](#).

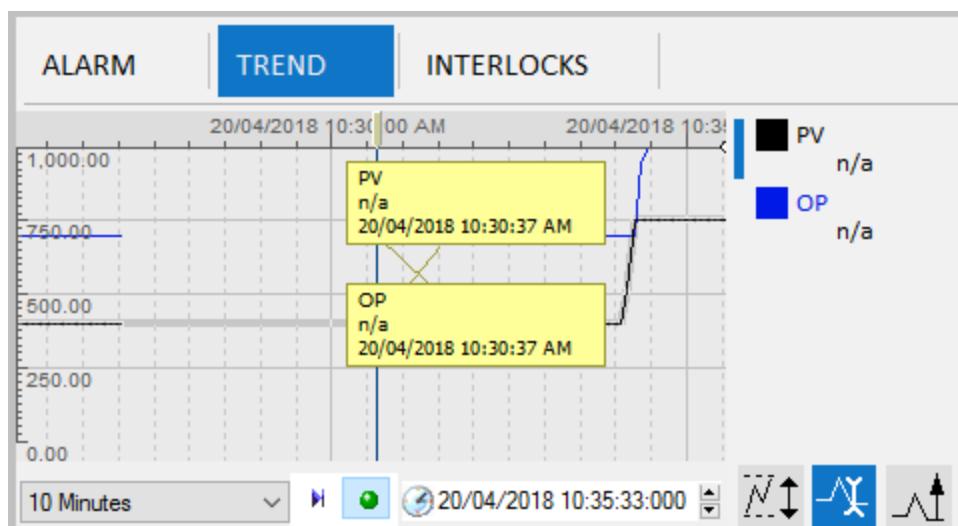
---

**Note:** If an object with an active alarm indicator does not display any alarms in the Information Zone when selected at runtime, it may mean that the alarm indicator has been configured to respond to "child" levels down the equipment hierarchy, or referenced equipment. These indirect equipment locations are not reported in the Alarms Information Zone for a selected piece of equipment. If this situation occurs, use a higher-level alarm count or an alarm page to determine the location of the active alarm.

---

## Trends

The Trend tab displays a graphical view of data associated with the object that is currently in context. This is presented using a compact version of Plant SCADA's Process Analyst.



The chart view displays data for the selected object using a set of pens that are color-coded and listed to the right of the chart. Each pen draws sample values against time. By default, the view will show the first five trends in alphabetical order by tag name.

Pens can be configured in one of the following ways:

- For each equipment item to be displayed on the chart: add an [Define Equipment Runtime Parameters](#) named "InfoZoneTrends" for each equipment to be displayed on the chart.
- As a global default: Configure the INI parameter [Workspace]InfoZoneTrends to display the required [Items](#).
- By a specific Equipment Type: Configure the INI parameter [Workspace]InfoZoneTrends.<Type> where <Type> matches a name in the Equipment Type view. For example, "InfoZoneTrends.Drive". The value of the parameters is a comma-separated list of equipment item names.

The following display commands are available from the navigation bar that runs along the bottom of the chart view.

Tool	Description
<b>10 Minutes</b>	<b>Span Picker</b> The Span Picker contains commonly used predefined time spans that you can select from a drop-down menu. The time span of the trend display represents the difference between the start of the chart and the end time. Selecting a time span adjusts the start time and leaves the end time as it is.
	<b>Synchronize to Now</b> The Synchronize to Now command synchronizes every pen such that the date/time reflects "Now", which is positioned on the right edge of the chart. "Now" is calculated using the current system time.
	<b>Toggle Autoscrolling</b> When Autoscroll is turned on, as time passes the

Tool	Description
	position of the pens moves by the same amount to keep pace. By default, the display is updated every second. When Autoscroll is turned off, as time passes the position in time of pens remain fixed.
 20/04/2018 10:35:33:000 ▲▼	<b>End Date/Time Picker</b> You can use the End Date/Time Picker to specify an end time for the chart view. You can type in a date or time explicitly, or you can use the up/down arrow buttons to increment or decrement the value respectively.

You can also use the following command buttons:

Tool	Description
	<b>Edit Vertical Scale</b> This command allows you to edit the vertical scale of a selected analog pen. It opens a dialog that allows you to enter new minimum and maximum values for Limits or Engineering Scale. The Engineering Scale values are obtained from the source tag. <b>Note:</b> The engineering scale is used by default.
	<b>Toggle Cursor</b> A cursor allows you to determine the value of a pen at a given point in time by dragging the cursor to a specific point on the pen line. A cursor label is used to display the value. This button allows you to toggle the cursor on and off.
	<b>Display in Trend Page</b> This button opens the <a href="#">Default Trend Pages</a> in the main display area, allowing you to view the current trend chart on a larger scale. <b>Note:</b> If you have navigated to the default Trend page using this button and you return to the graphic page and click the <b>Back</b> button, the graphic page that you browsed to last will be displayed (instead of the Trend page).

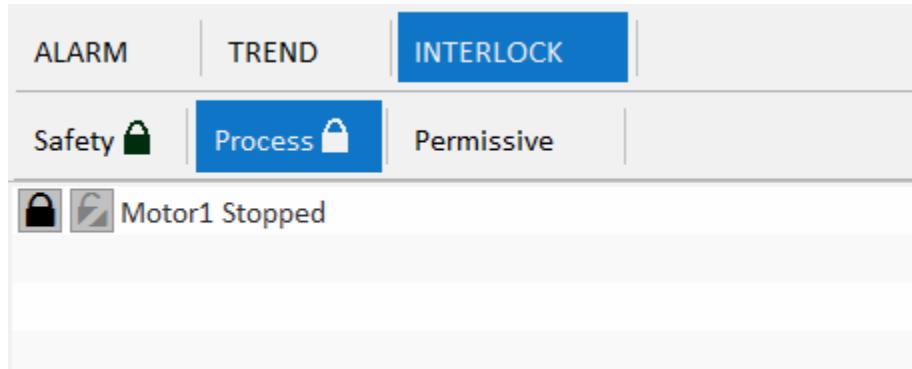
## Interlocks

An interlock is a physical connection between two pieces of equipment that links their operation, so that an event in one piece of equipment will trigger an action in the other (see [Interlocks](#)).

The **Interlocks** tab provides details of any active interlocks that may be associated with the currently selected piece of equipment. It includes a secondary set of tabs that reflect the categories used to identify the nature of an interlock. The Situational Awareness Starter Project provides the following default categories.

- Safety — conditions that cause equipment to stop running.
- Process — conditions that cause equipment within a process to stop running.
- Permissive — conditions that stop equipment from starting.

If there are any active interlocks within a category, a lock symbol is displayed on the associated category tab.



The interlocks are listed on each tab from oldest (triggered first) to most recent. The list includes the following set of default columns.

Column	Description
State	If an interlock is active, a black lock icon is displayed.  If an interlock has cleared, a white box is displayed. 
Bypass	Interlocks can be configured to support a bypass. This means an operator can manually override the interlock, forcing the locked process to be available to start. See <a href="#">Configure a Bypass for an Interlock</a> . If an interlock is capable of being bypassed (but is currently not bypassed), a gray lock icon is displayed.  <b>To bypass an interlock:</b> <ol style="list-style-type: none"><li>1. Right-click on the interlock and select <b>Bypass</b>.</li><li>2. In the Bypass form, enter a <b>Reason</b> for the bypass (if required).</li></ol>

Column	Description
	<p>3. Click <b>Yes</b>.</p> <p>An event will be added to the sequence of event journal in the following format:</p> <p>&lt;Cluster&gt;.&lt;Equipment&gt;.&lt;Item&gt; @ (INTERLOCK BYPASSED:)&lt;Reason&gt;</p> <p>If the interlock is currently bypassed, a purple lock icon is displayed.</p>  <p><b>To remove a bypass:</b></p> <ol style="list-style-type: none"> <li>1. Right-click on the interlock and select <b>Remove Bypass</b>.</li> <li>2. In the Bypass form, enter a <b>Reason</b> for the removing the bypass (if required).</li> <li>3. Click <b>Yes</b>.</li> </ol> <p>An event will be added to the sequence of event journal in the following format:</p> <p>&lt;Cluster&gt;.&lt;Equipment&gt;.&lt;Item&gt; @ (INTERLOCK BYPASS REMOVED:)&lt;Reason&gt;</p> <p>If an interlock is not configured to support a bypass, no icon is displayed in the column.</p>
Comment	A description of the interlock. The description is sourced from the <b>Comment</b> field for the associated Equipment Reference.

**Note:** The default arrangement of columns is based on a display format that is defined in the SA\_Include system project. The format will be named "Interlock\_HD1080" or "Interlock\_UHD4K", depending on the resolution of the workspace master page. You can adjust this format by modifying [\[Format\]FormatName](#) in the **Parameters** view of the **Setup** activity (see [Project Database Parameters](#)).

To go directly to the page on which the interlocking piece of equipment is homed, right-click on an interlock and select **Navigate**.

## See Also

[Default Layout](#)

## Navigation Zone

The Navigation Zone on the operator dashboard serves two purposes:

- Page navigation — the tabs and buttons can be used to switch between different display pages.
- Alarm counts — each tab and button displays an alarm count for the location it opens, providing an overview of the current alarm conditions.



**Note:** To enable alarm counts in the Navigation Zone, there are a number of project settings that need to be configured correctly. For more information, see [Enable Navigation Zone Alarm Counts](#).

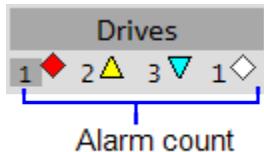
## Page Navigation

The layout of the tabs and buttons is determined by a project's page menu.

Level 1 of the menu sets the overall context for the workspace. The tabs along the navigation section header represent the menu's level 2 entries. The buttons on each tab represent up to 20 lower-level menu entries, displayed from left to right in arrangement determined by the Order property.

## Alarm Counts

Every tab and button in the navigation zone supports an alarm count that indicates how many alarms are currently active for equipment homed on the associated page.



By default, the alarm count represents the following:

- The number of active alarms in the highest priority (that is, the priority with the lowest value)
- The number of active alarms in the next highest priority
- The number of active alarms in the third highest priority
- The number of alarms that are currently shelved.

**Note:** If required, you can use the parameter [Workspace]NumberOfTopPriorities to display an additional alarm count for the fourth highest priority. For more information, see [Add an Additional Alarm Count to the Navigation Zone](#).

The maximum value that can be displayed for each alarm count is 99. If more than 99 alarms are active in a particular category, "99+" will display. You can confirm the exact number of alarms via a tool tip.

Each value appears to the left of an animated icon that indicates the priority the value represents. These icons are configured via the **Small Genie Name** property for each alarm priority. For more information, see [Configure Display Properties for an Alarm Priority](#).

A grey bar is used to highlight the tab and button that contain the most recent unacknowledged alarm that is currently in an ON state. The number next to the priority which is most important will also be highlighted.

For more information on how to prepare a project's menu for the navigation zone of the operator dashboard, see [Prepare the Navigation Menu](#).

## See Also

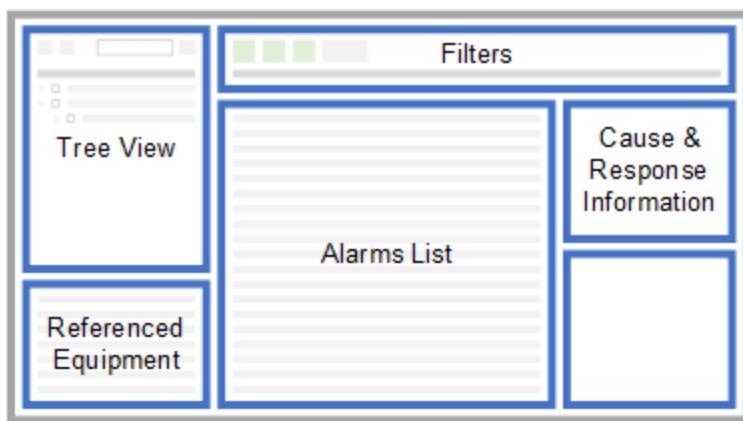
[Default Layout](#)

## Default Alarm Pages

A project created using the Situational Awareness Starter Project will include a set of default alarm pages that are accessible from the runtime [Header Bar](#). Select one of the buttons described in the table below to display an alarm page in the page content area (see [Default Layout](#)).

	Active Alarms	Alarms that are unacknowledged, or acknowledged and still in an alarm state.
	Historical Events	An historical log of alarms and operator activity.
	Shelved Alarms	Alarms that are temporarily shelved/disabled.
	Hardware Alarms	Hardware alarms that are either unacknowledged or acknowledged and still in alarm state.

Apart from the Hardware Alarms page (which includes just a basic alarms list), these pages share a common layout that features the following sections:



## Alarms List

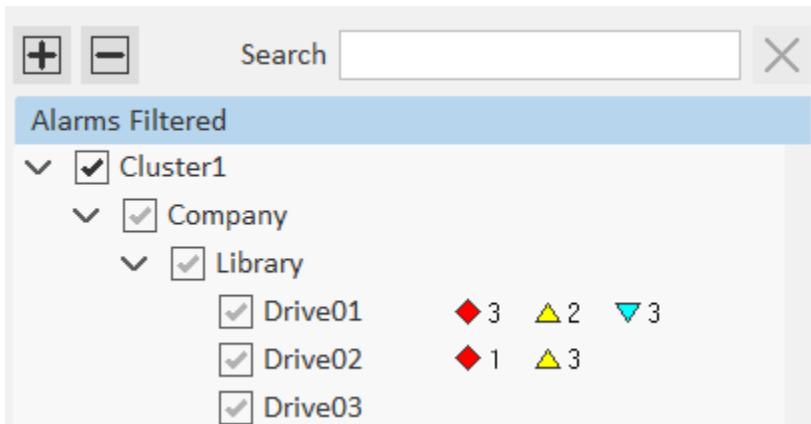
The content of the alarms list is determined by the following:

- The filters that are currently enabled (see *Filters* below).
- Any current selections in the Tree View. The list will only include alarms associated with the selected items (see *Tree View* below) and any selected referenced equipment (see *Referenced Equipment* below).

The default arrangements of columns is determined by the type of alarm page that is displayed and the default format specified for it by the [\[Format\]FormatName](#) parameter. You can rearrange the columns if required.

## Tree View

The tree view provides a hierarchical view of a project's equipment hierarchy. You can use it to filter the displayed list of alarms.



Each item in the tree view includes:

- Check boxes allow you to select one or more items in the tree.
- Alarms counts indicate the number of associated alarms in the top three alarm priorities, as well as the number of shelved alarms.

The alarms list will be filtered to display alarms associated with the selected items.

---

**Note:** You can use the parameter [Workspace]NumberOfTopPriorities to display alarm counts for the top four alarm priorities (instead of the top three). For more setup information, see [Configure Display Properties for a Fourth Alarm Priority](#).

---

To customize a tree view, see [Add a Tree View to a Page](#).

## Filters

This section of an alarms page allows you to filter the alarms list according to priority, state and acknowledgment. The **Current Filter** description indicates if any filters are currently active for the alarms list.

---

**Note:** You can add your own filters buttons to an alarm page using a set of provided Genies. See [Add an Alarm Filter Button](#).

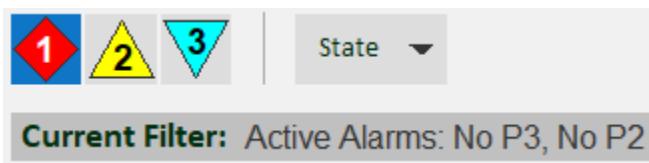
---

### Priority Filters

The alarm priority buttons allow you to filter out alarms included in the top three alarm priorities. By default, the three buttons are selected (as indicated by a colored background).



If you toggle a button off, the alarms assigned to the alarm priority will be removed from the list. The current filter description will indicate that the P1, P2 or P3 alarms are no longer displayed.



The Historical Events page also features a button that allows you to filter out shelved alarms.



You can add a fourth-highest alarm priority filter to alarm pages. See [Add a Fourth Alarm Priority Filter to Alarm Pages](#).

### State Filters

The State filter button allows you to filter out alarms that are in a specific state.

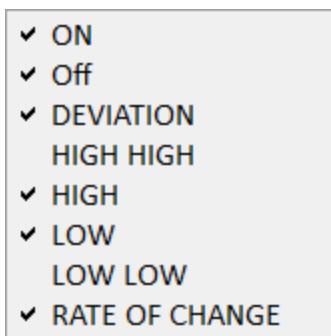


---

**Note:** This button is not available on the Shelved Alarms page.

---

When you click on the button, a check list of alarm states appears.



Select the alarm states that you would like to include in the alarm list. The current filter description will indicate if any states are not selected.

**Current Filter:** Active Alarms: No HIGH HIGH, No LOW LOW

#### Acknowledgment Filters

The Historical Events page also features button that allows you to filter **Acknowledged** and **Unacknowledged** alarms.

Acknowledged

Unacknowledged

By default, both buttons are selected (as indicated by a colored background). If you toggle a button off, the associated alarms will be removed from the list.

#### Cause & Response Information

This section of an alarms page displays cause and response information for a selected alarm (see [Provide Cause and Response Information to Operators](#)).

**Cause and Response for selected alarm:** < >

Cause	Drive1 is overheating.
Response	Check the thermostat.
Response Time	00:05:00
Consequence	Drive may fail.

The information presented includes:

- Cause — a description of the cause of an alarm.
- Response — a description of the appropriate response to an alarm.
- Response Time — the period of time in which the specified response needs to be acted upon.
- Consequence — a description of the likely outcome if the suggested response is not acted upon within the specified response time.

If multiple sets of cause and response properties are configured for an alarm, you can use the arrow buttons at the top right corner to scroll through them.

## Referenced Equipment

This section displays any [Equipment References](#) that are associated with the equipment that is currently selected in the Tree View.

Referenced Equipment		
Equipment	Full Name	Reference Type
<input checked="" type="checkbox"/> PMP01	MyPlant.FullMilk....	Interlock_Proce...

The **Referenced Equipment** header indicates the piece of equipment that is currently selected. The list beneath displays any equipment for which an associated equipment reference has been configured.

You can use the check box at the start of each row to add the referenced equipment to the current filter for the main alarms list. You can also use the **Clear All>Select All** button to toggle the selection of all the equipment references in the list.

If the check box has a grey background, it means the piece of equipment is already selected in the Tree View. If you select **Clear All** while a piece of equipment is in this state, it will deselect the piece of equipment in both the Referenced Equipment list and the Tree View.

### **See Also**

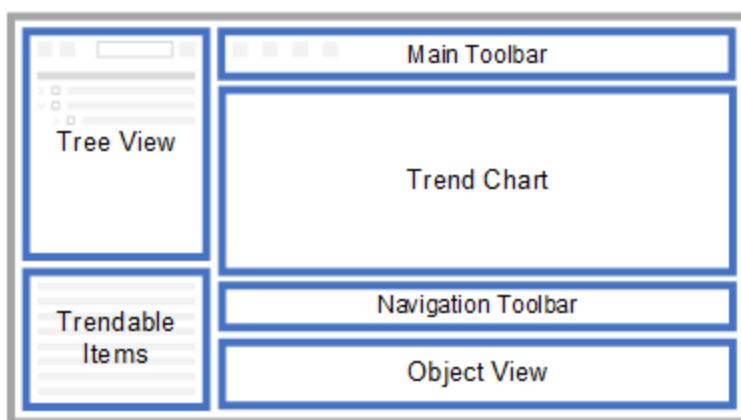
## Default Layout

## Default Trend Pages

A project created using the Situational Awareness Starter Project will include a default trend page that is accessible from the runtime [Header Bar](#). Click on the icon below to display a trend page in the page content area (see [Default Layout](#)).



The default trend page allows you to analyze specific trends and alarms for equipment. The layout of the default trend page features the following sections:



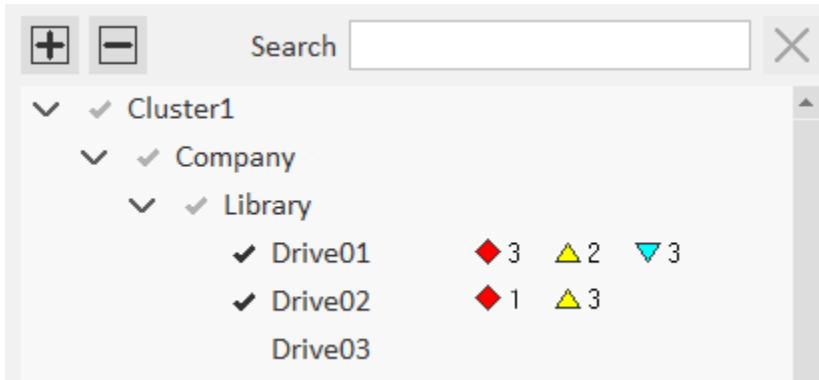
Behaviors that you will observe when you navigate between pages include:

- Navigating to a different page within the Workspace will not affect the trend page. Your pens, currently selected object in the Object Tree and any text entered in the Search box will be retained.
- Navigating to a different page and selecting a different equipment and then navigating back to the trend page will clear the entry in the Search box and also select the equipment in the equipment tree.
- Navigating to a different page and selecting a different equipment and clearing that selection before navigating to the trend page will retain your configuration on the trend page.

**Note:** You can launch the trend page from other pages using the buttons in the Information Zone's Trend tab. This will launch the trend page with the trend for the object selected on that page.

## Tree View

The tree view provides a hierarchical view of a project's equipment hierarchy.

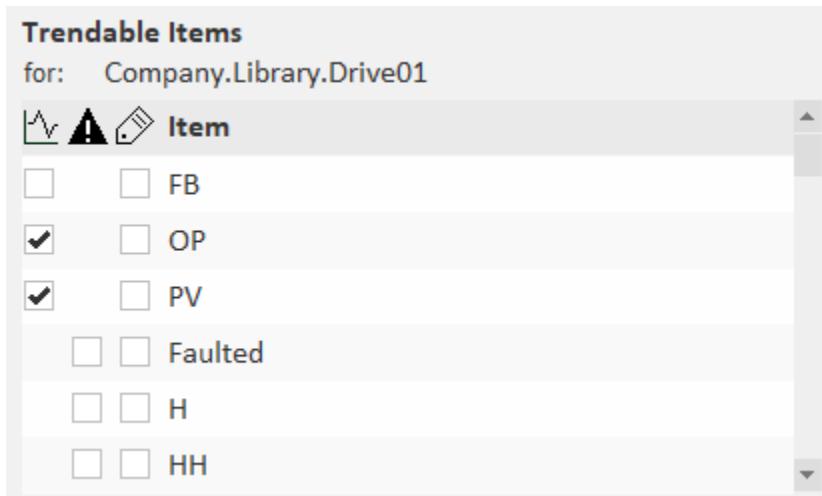


The tree view displays:

- Check marks for objects that have an active trend.
- Gray check marks for objects whose children have an active trend.
- Alarm counts that indicate the number of associated alarms in the top three alarm priorities, as well as the number of shelved alarms.
- Buttons to expand (+) and collapse (-) the tree.
- Search box that you can use to filter the displayed list of objects.

## Trendable Items

The Trendable Items section lists the items associated with each piece of equipment that you can add to the trend chart.



The following icons indicate the 'type' of an item (tag):

	Trend items
	Alarm items
	Instant items (variable)

A check box is displayed for each applicable type. Select the required check boxes to add them to the chart. Up to 16 trends can be added. When the limit is reached, the check boxes are hidden.

## Trend Chart

Displays the trend chart based on the selected equipment and options.

## Main Toolbar

The Main toolbar is located above the chart. It contains commands that allow you to perform general operations, such as save and open views, print reports, and so on. The table below describes the buttons that are included on the main toolbar.

Item	Description
	<b>Load View.</b> Loads a saved view from file. For details, see <a href="#">Load a View</a> .
	<b>Save View.</b> Stores a view to file. For details, see <a href="#">Saving</a>

Item	Description
	<a href="#">a View.</a>
	<b>Print.</b> Displays the standard Windows Print dialog box for printing trend reports. For details, see <a href="#">Printing and Exporting</a> .
	<b>Copy to Clipboard.</b> Copies visible pens to the Windows Clipboard. For details, see <a href="#">Copying Data to the Clipboard</a> .
	<b>Copy to File.</b> Exports visible pens to an Excel-compatible file. For details, see <a href="#">Copying Data to File</a> .
	<b>Add Pen.</b> Displays the Add New Pen(s) dialog box for adding a pen. For details, see <a href="#">Add Pens</a> . <b>Note:</b> Pens added with this button will not be synchronized with the equipment tree and trendable items list.
	<b>Remove Pen.</b> Deletes the currently selected pen from the trend display. For details, see <a href="#">Delete Pens</a> .
	<b>Remove All Pens.</b> Deletes the currently displayed pens from the trend display.
	<b>Lock/Unlock Pens.</b> Toggles the locking of pens. For details, see <a href="#">Lock/Unlock Pens</a> .
	<b>Show/Hide Points.</b> Toggles the display of points representing where sample data was recorded in the archive. For details, see <a href="#">Pens</a> .
	<b>Show/Hide Cursors.</b> Toggles the display of cursors. For details, see <a href="#">Cursors</a> .
	<b>Show/Hide Cursor Labels.</b> Toggles the display of cursor labels. For details, see <a href="#">Using Cursor Labels</a> .
	<b>Find in Model.</b> Locates the equipment for the currently selected pen in the equipment tree.

## Navigation Toolbar

The Navigation toolbar contains commands to allow you to navigate forward or backward through trends, as well as other navigation-related tasks.

A **Start Time Picker** is located on the left-hand side of the navigation toolbar, an **End Time Picker** is located on the right. You can use these to Specify a Start Time and End Time. If a time picker control is using Daylight Savings time, the clock to the left of the control will have a shaded segment.



The **Span Picker** contains commonly used predefined time spans that you can select from a drop-down menu. The time span of the trend display represents the difference between the start time and the end time. Selecting a time span adjusts the start time and leaves the end time as it is. To set a time span that is not included in the list of predefined times, see [Set a Nonstandard Time Span](#). See Also [Lock/Unlock the Time Span](#).

The Navigation toolbar also includes a set of command buttons. You can use these buttons to perform the following tasks:

- [Navigate Time](#)
- [Synchronize to Now](#)
- [Toggle Autoscrolling](#)
- [Zoom In/Zoom Out](#)
- [Undo Last Zoom](#)
- [Toggle Box Zoom](#)
- [Edit Vertical Scale.](#)

## Object View

The Object View displays the list of objects (equipment) displayed in the trend chart along with properties of the pens such as name, color, scale and so on. The following properties are displayed:

Column	Description
Scale	Vertical axis start and end position of the pen.
Engineering Units	Engineering units associated with the pen.
Comment	The trend/alarm comment defined for the pen.
Alarm Name	Name of associated alarm tag.
Cluster	Name of the cluster in which the object is defined.
Error	Displays the error of the last data request. Blank if last data request succeeded.

## Alarms

The principles of Situational Awareness specify that alarm conditions should be signaled using easily recognizable shapes and high-intensity color. This ensures that alarms stand out against the primarily achromatic color scheme used for the runtime interface.

It is also necessary to provide notification for all active alarms, even if they occur on a page that is currently not displayed. Direct access to the page that represents the source of an alarm should also be available when notification occurs.

In the case of the Situational Awareness Starter Project, a default set of six Genies (three large and three small)

are included in the "sa\_priorities" library to facilitate the signaling of alarms in accordance with these principles.



Priority 1  
Emergency



Priority 2  
High



Priority 3  
Low

Genies are also provided to represent when an alarm is "shelved", which means the alarm is disabled for a specified period of time. See [Shelve Alarms](#).



Shelved

---

**Note:** If required, you can customize the appearance of these Genies.

The Genies are used to highlight alarm conditions in the following locations.

#### Alarm Indicators

Alarm indicators are used on content pages to highlight an alarm condition for a specific object or group of objects. They comprise of an alarm flag (the large alarm priority Genie) and an alarm border.



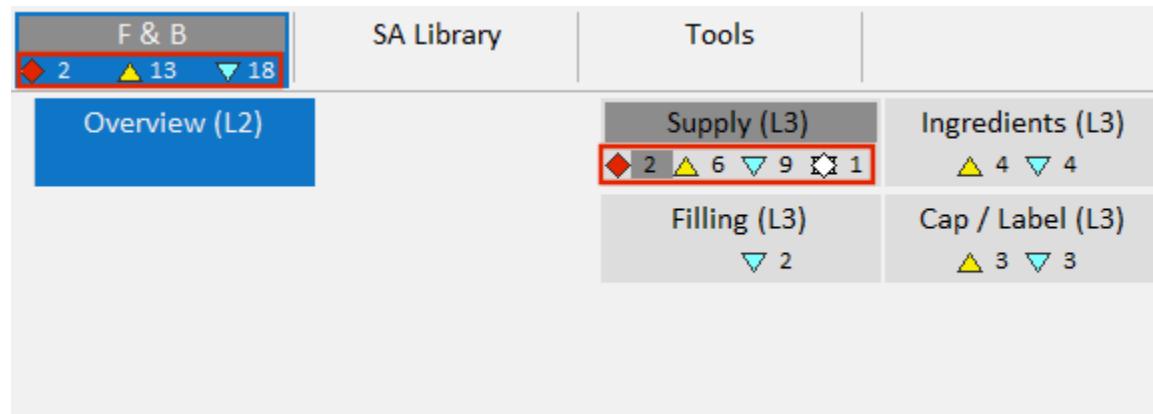
For more information, see [Use Alarm Indicators](#).

#### Navigation Zone

The Navigation Zone provides a system-wide view of current alarms across all pages.

Each tab provides a tally of the top three priority alarms that are active on the pages included on the tab.

Each button provides a tally of the top three priority alarms and shelved alarms for the location it opens.



For more information, see [Enable Navigation Zone Alarm Counts](#).

## Alarm Lists

Small icons are used to highlight the top three alarm priorities in alarms lists.

ALARM	TREND	INTERLOCK
◆ 03:38:27 PM HH		TS01 Full Milk Temperature High H...
▲ 10:40:12 AM LL		TS01 Full Milk Temperature Low Low
▼ 04:32:48 PM L		TS01 Full Milk Temperature Low
▼ 02:59:11 PM H		TS01 Full Milk Temperature High

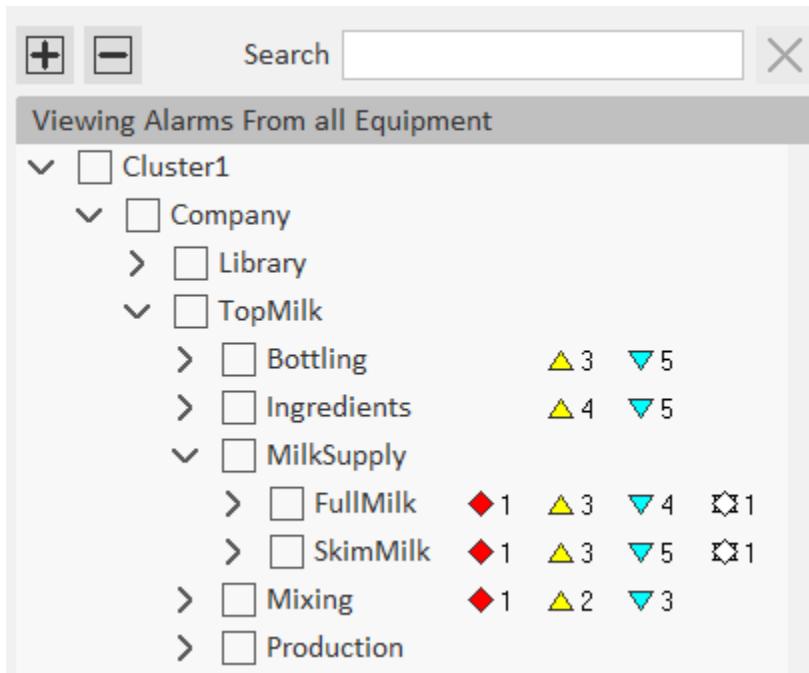
If an alarm is shelved, the priority icon will display a white fill.

This occurs in the following locations:

- Alarms pages (see [Default Alarm Pages](#)).
- The alarms list in the .
- the [Active Alarms Zone](#) (if a UHD4K workspace is used).

## Alarm Page Tree View

The tree view on each alarms page provides an alarm count for the top three alarm priorities and shelved alarms within each branch of the equipment hierarchy (see [Default Alarm Pages](#)).



As well as using color to indicate the priority of an alarm, the appearance of an alarm indicator or icon can be used to show the current state of an alarm.

## Alarm States

The following table shows how different alarm states are indicated for a priority 1 (Emergency) alarm in a project based on the Situational Awareness Starter Project.

State	Alarm Indicator	Alarm icon
On Unacknowledged	 <ul style="list-style-type: none"> <li>The border and flag both flash between colored and white fill.</li> </ul>	 <ul style="list-style-type: none"> <li>The icon flash between colored and white fill.</li> </ul>
On Acknowledged	 <ul style="list-style-type: none"> <li>The alarm border is filled with a half-intensity color.</li> <li>There is no flashing for either element.</li> <li>If the alarm condition clears, the alarm indicator is no longer displayed.</li> </ul>	 <ul style="list-style-type: none"> <li>The icon stops flashing.</li> </ul>
Off Unacknowledged	 <ul style="list-style-type: none"> <li>The alarm border appears in a lighter color.</li> <li>The alarm flag flashes.</li> <li>The alarm border does not flash.</li> </ul>	 <ul style="list-style-type: none"> <li>The icon flash between half-intensity color and white fill.</li> </ul>
Shelved	 <ul style="list-style-type: none"> <li>A white alarm border is shown with a generic alarm flag symbol.</li> </ul>	 <ul style="list-style-type: none"> <li>In an alarms list, the priority icon will appear with a white fill.</li> </ul>  <ul style="list-style-type: none"> <li>In an alarm tree view and the Navigation Zone, a generic shelved icon will appear.</li> </ul>

By default, the six alarm priority Genies are associated with the top three alarm priorities via the display

properties for each priority.

The screenshot shows a software interface with a top navigation bar containing tabs: Setup, Alarming (which is selected and highlighted in blue), Parameters, Screen Profiles, Devices, and Events. Below the navigation bar is a toolbar with buttons for Save, Discard, Copy, Paste, Delete Row(s), and Export All. The main area is titled "Alarm Priorities" and contains a table with three rows of data. The columns are labeled: Priority, Display Name, Library Name, Genie Name, and Small Genie Name. The data rows are:

Priority	Display Name	Library Name	Genie Name	Small Genie Name
1	Emergency	sa_priorities	sa_p1_normal	sa_p1_small
2	High	sa_priorities	sa_p2_normal	sa_p2_small
3	Low	sa_priorities	sa_p3_normal	sa_p3_small

The **Genie Name** property specifies the larger Genie used as an alarm flag; **Small Genie Name** specifies the smaller Genie that is used as an alarm icon. See [Configure Display Properties for an Alarm Priority](#).

The Genies that are used to represent a shelved alarm are associated with an Alarm Mode named "Shelved/Disabled" (see [Configure Display Properties for an Alarm Mode](#)).

## See Also

[Add an Additional Alarm Count to the Navigation Zone](#)

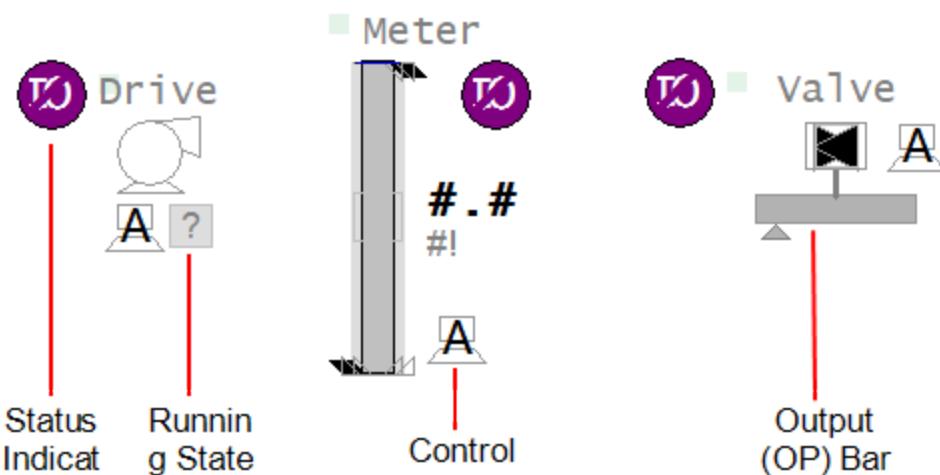
[Add a Fourth Alarm Priority Filter to Alarm Pages](#)

[Create a Custom Flag for an Alarm Indicator](#)

## Libraries

A project created with the Situational Awareness Starter Project will contain "SA\_Library" as an included project. SA\_Library provides a large set of objects that you can use to add content to your graphics pages and faceplates. These objects apply a consistent look and feel to a wide variety of common user interface controls.

The objects include Genies that represent physical pieces of equipment (such as drives, meters, and valves), and a set of common library elements that support them (such as status indicators, running state indicators, control mode indicators, and output bars).



For a complete list of the libraries included in the SA\_Library project, see Situational Awareness Library Project in the System Projects Reference section of the help.

## Composite Genies

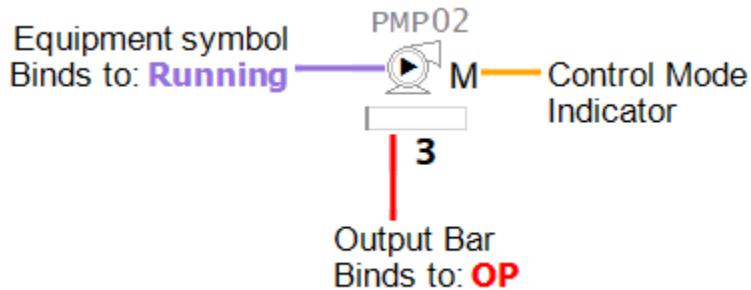
What makes the Situational Awareness Library unique is that many of library objects can be built using a [Composite Genie](#). A Composite Genie is an XML file that defines a set of Genies, options, parameters and layout display rules to determine what is displayed on a page. This is based on user-selected presentation options. The collection of components can be configured as a single entity, greatly reducing the size and complexity of your libraries.

To use a Composite Genie, simply open the XML template that best suits the piece of equipment you want to represent on a page. Ideally, the XML template should match the equipment type that was used to create the equipment instance the Composite Genie will represent. This is recommended, as the components of a Composite Genie are designed to bind to specific equipment items that cannot be changed.

This is straightforward if you have created your equipment instances using the default equipment types provided with the Situational Awareness Starter Project.

## Example

The following drive (PMP02) is from the ExampleSA project. It was created using the "Drive.xml" Composite Genie, which means it binds to the following equipment items.



The equipment items "Running", "CtrlMode", and "OP" are all associated with variable tags that link to the I/O

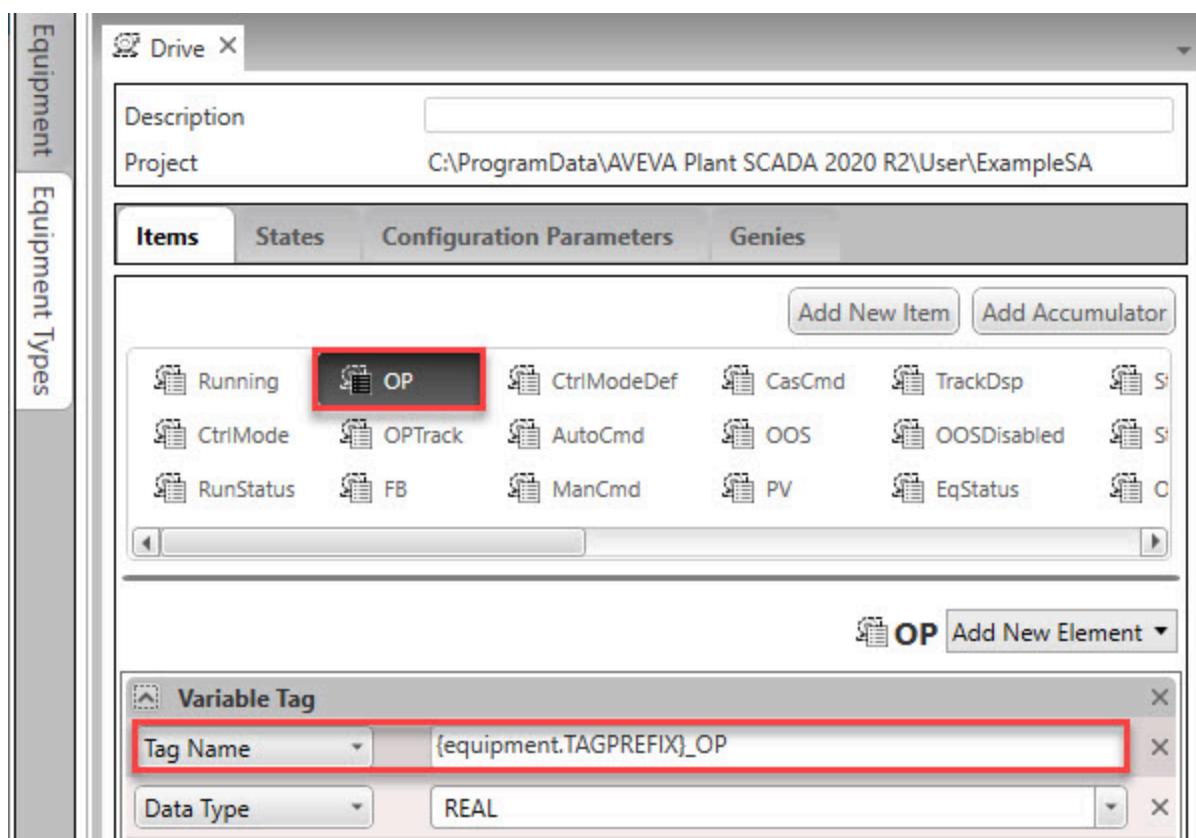
device.

Variables			
	Equipment	Item Name	Tag Name
29	Company.TopMilk.MilkSupply.FullMilk.PMP02	FB	TopMilk_PMP02_FB
30	Company.TopMilk.MilkSupply.FullMilk.PMP02	OP	TopMilk_PMP02_OP
31	Company.TopMilk.MilkSupply.FullMilk.PMP02	ORLow	TopMilk_PMP02_ORLow
32	Company.TopMilk.MilkSupply.FullMilk.PMP02	RunStatus	TopMilk_PMP02_RunStatus
33	Company.TopMilk.MilkSupply.FullMilk.PMP02	ORHigh	TopMilk_PMP02_ORHigh
34	Company.TopMilk.MilkSupply.FullMilk.PMP02	Stopping	TopMilk_PMP02_Stopping
35	Company.TopMilk.MilkSupply.FullMilk.PMP02	StopCmd	TopMilk_PMP02_StopCmd
36	Company.TopMilk.MilkSupply.FullMilk.PMP02	CtrlMode	TopMilk_PMP02_CtrlMode
37	Company.TopMilk.MilkSupply.FullMilk.PMP02	EqStatus	TopMilk_PMP02_EqStatus
38	Company.TopMilk.MilkSupply.FullMilk.PMP02	PRLow	TopMilk_PMP02_PRLow
39	Company.TopMilk.MilkSupply.FullMilk.PMP02	Starting	TopMilk_PMP02_Starting
40	Company.TopMilk.MilkSupply.FullMilk.PMP02	StartCmd	TopMilk_PMP02_StartCmd
41	Company.TopMilk.MilkSupply.FullMilk.PMP02	Stopped	TopMilk_PMP02_Stopped
42	Company.TopMilk.MilkSupply.FullMilk.PMP02	Running	TopMilk_PMP02_Running

These variables tags were generated by Equipment Editor, as PMP02 was based on the "Drive" Equipment Type, and these three items each had a variable tag element defined.

For example, the OP item had a variable tag element that specified the following **Tag Name** syntax:

{equipment.TAGPREFIX}\_OP



In the case of the equipment instance for PMP02, the Tag Prefix was set to "TopMilk\_PMP02\_", resulting in the tag named "TopMilk\_PMP02\_OP".

If you generate your variable tags in this way using the equipment types provided in the Situational Awareness Starter Project, the item bindings required by each library object should fall into place when you use the corresponding Composite Genie. See [Add Equipment Using Equipment Editor](#).

You can change the variable names defined in the equipment type if you want to maintain your own tag naming convention. However, you still need to pair your variables with the relevant equipment items.

---

**Note:** To determine the required bindings for a library object and its default faceplate, search for the object in the [Situational Awareness System Projects](#) section of the help. The description of each library objects lists the `Equipment.Items` it expects.

## See Also

[Insert a Composite Genie](#)

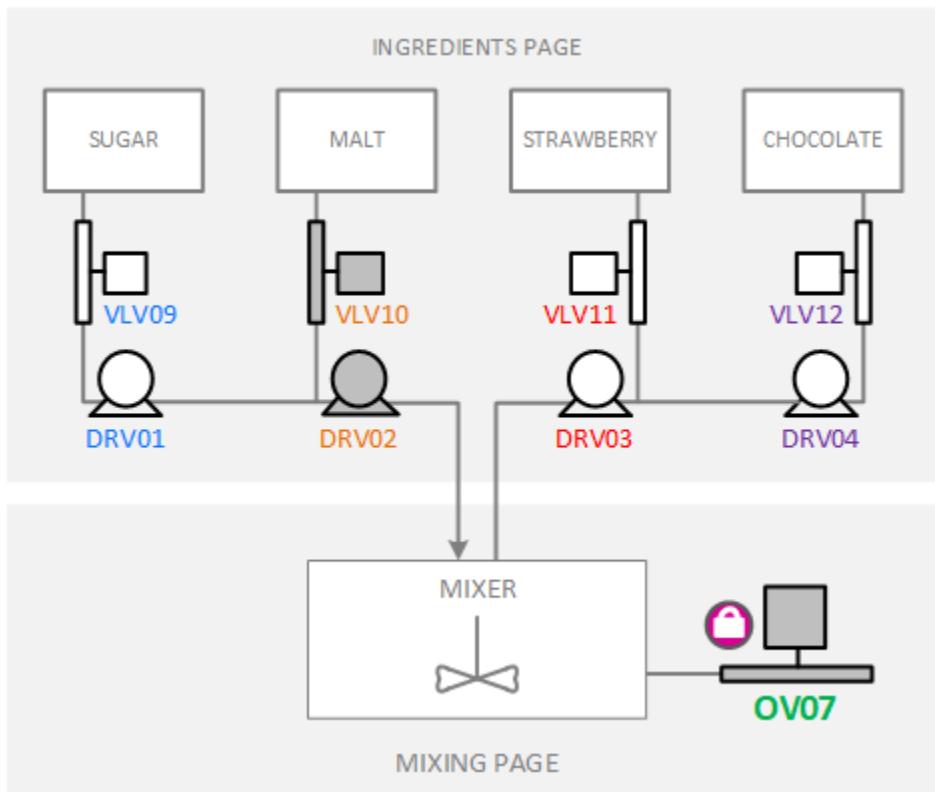
## Interlocks

In a production environment, an interlock is a physical connection between two pieces of equipment that links their operation, so that an event in one piece of equipment will trigger an action in the other. Typically enabled using PLC logic, interlocks are built into a production system to maintain safety, protect equipment and minimize production costs.

Plant SCADA allows you to represent the interlocks that exist in your system within a Situational Awareness project. The activity associated with an interlock can then be presented to an operator at runtime in the

### Information Zone.

For example, the "Mixing" page in the ExampleSA project includes a mixer that receives ingredients from four supply tanks on the "Ingredients" page.



The mixing tank has an outlet valve (OV07). To ensure that this valve is not open while ingredients are being delivered to the mixer, interlocks exist between OV07 and the valves and pumps that deliver the four ingredients for mixing.

This example will be used to demonstrate the role each of following components play when configuring interlocks in a Situational Awareness project.

#### Equipment References

**Equipment References** are used to represent interlocks in a Situational Awareness project, as they allow you to build relationships in an equipment model that reflect those created by the interlocks that exist in the field.

To support the example above, the following equipment references are configured in the ExampleSA project.

	Cluster	Equipment	Referenced Equipment	Referenced Item
1	Cluster1	Company.TopMilk.Mixing.OV07	Company.TopMilk.Ingredients.Sugar.VLV09	Open
2	Cluster1	Company.TopMilk.Mixing.OV07	Company.TopMilk.Ingredients.Sugar.DRV01	Running
3	Cluster1	Company.TopMilk.Mixing.OV07	Company.TopMilk.Ingredients.Strawberry.DRV03	Running
4	Cluster1	Company.TopMilk.Mixing.OV07	Company.TopMilk.Ingredients.Strawberry.VLV11	Open
5	Cluster1	Company.TopMilk.Mixing.OV07	Company.TopMilk.Ingredients.Chocolate.DRV04	Running
6	Cluster1	Company.TopMilk.Mixing.OV07	Company.TopMilk.Ingredients.Chocolate.VLV12	Open
7	Cluster1	Company.TopMilk.Mixing.OV07	Company.TopMilk.Ingredients.Malt.DRV02	Running
8	Cluster1	Company.TopMilk.Mixing.OV07	Company.TopMilk.Ingredients.Malt.VLV10	Open

The **Referenced Item** property specifies the equipment item that will trigger the interlocked state. For example, DRV01 will trigger an interlock when the digital variable associated with its "Running" item equals 1 (TopMilk\_DRV01\_Running = 1).

The **Comment** property can be used to provide a description of the event that caused an interlock to trigger, for example "Sugar DRV01 is Running".

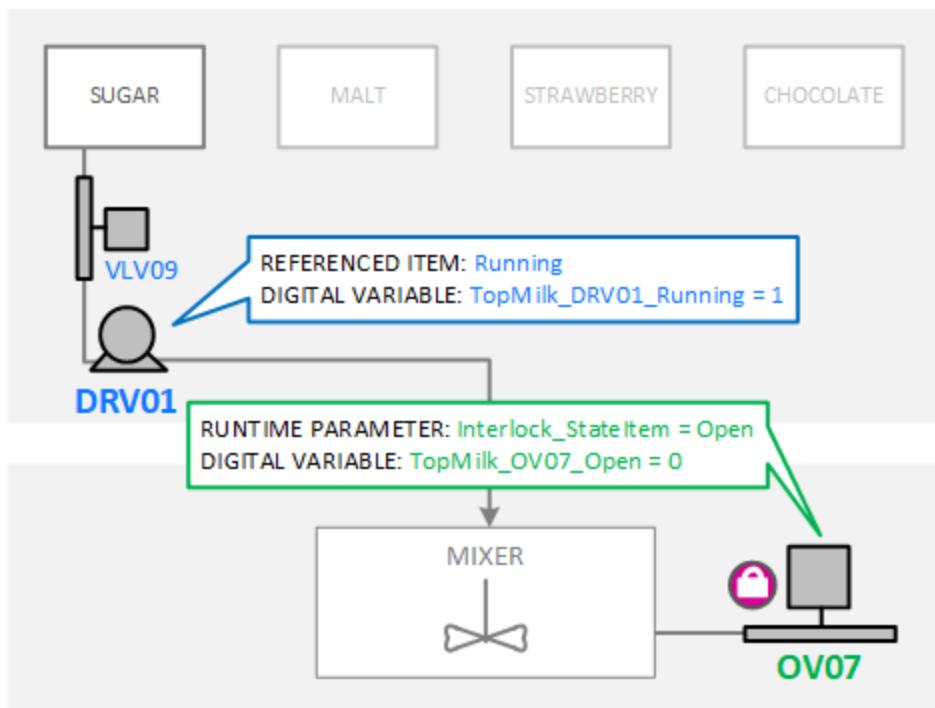
#### "Interlock\_StateItem" Runtime Parameter

When triggering occurs, it causes the "state item" in the interlocked piece of equipment to change. By default, the state item for a piece of equipment is "Running". If this is not the item that should respond to an interlock, you can use the runtime parameter "Interlock\_StateItem" to specify a different state item for a piece of equipment.

For example, OV07 does not have a "Running" item as it is a valve. Interlock\_StateItem is used to specify "Open" as the correct item.

	Cluster	Equipment	Name	Value	Is Tag
1	Cluster1	Company.TopMilk.Mixing.OV07	Interlock_StateItem	Open	FALSE

This enables the interlock behavior demonstrated in the following diagram.



**Note:** The **IsTag** field for a Runtime Parameter defaults to TRUE if no value is specified. If you do not enter FALSE, interlocks will not function correctly.

### Alarms

Alarms are used to generate the events required to track the activity associated with interlocks. They determine how long an interlock has been active, or when it was last active.

In the ExampleSA project, the following digital alarms have been configured for the two pieces of equipment that supply sugar to the mixing tank (DRV01 and VLV09).

Digital Alarms				
	Equipment	Item Name	Alarm Tag	Alarm Name
25	Company.TopMilk.Ingredients.Sugar.DRV01	Running	TopMilk_DRV01_Running	DRV01 Running - Interlock - Start Time
26	Company.TopMilk.Ingredients.Sugar.DRV01	Faulted	TopMilk_DRV01_Faulted	DRV01 Faulted
27	Company.TopMilk.Ingredients.Sugar.LS03	HH	TopMilk_LS03_PV_HH	LS03 Sugar Hopper Level High High
28	Company.TopMilk.Ingredients.Sugar.LS03	H	TopMilk_LS03_PV_H	LS03 Sugar Hopper Level High
29	Company.TopMilk.Ingredients.Sugar.LS03	L	TopMilk_LS03_PV_L	LS03 Sugar Hopper Level Low
30	Company.TopMilk.Ingredients.Sugar.LS03	LL	TopMilk_LS03_PV_LL	LS03 Sugar Hopper Level Low Low
31	Company.TopMilk.Ingredients.Sugar.VLV09	Closed	TopMilk_VLV09_Closed	VLV09 Closed - Interlock - Start Time
32	Company.TopMilk.Ingredients.Sugar.VLV09	Open	TopMilk_VLV09_Open	VLV09 Open - Interlock - Start Time

These alarms monitor the items that are configured to trigger an interlock, providing a record of the start and end times.

Similarly, alarms are configured for the six other pieces of referenced equipment to monitor the items that trigger an interlock.

The following alarm has also been configured to monitor the "Open" item for the outlet valve OV07.

Digital Alarms				
	Equipment	Item Name	Alarm Tag	Alarm Name
1	Company.TopMilk.Mixing.OV07	Open	TopMilk_OV07_Open	OV07 Open - Interlock - Start Time

**Note:** These alarms are intended to be hidden from the operator at runtime. This can be achieved using an alarm category that has **ShowOnActive** and **ShowOnSummary** set to FALSE.

### Interlock Categories

Each equipment reference can also be assigned to a **Category**. In the case of interlocks, the category is used to identify the nature of an interlock. The Situational Awareness Starter Project provides the following default categories.

- **Safety** — conditions that cause equipment to stop running.
- **Process** — conditions that cause equipment within a process to stop running.
- **Permissive** — conditions that stop equipment from starting.

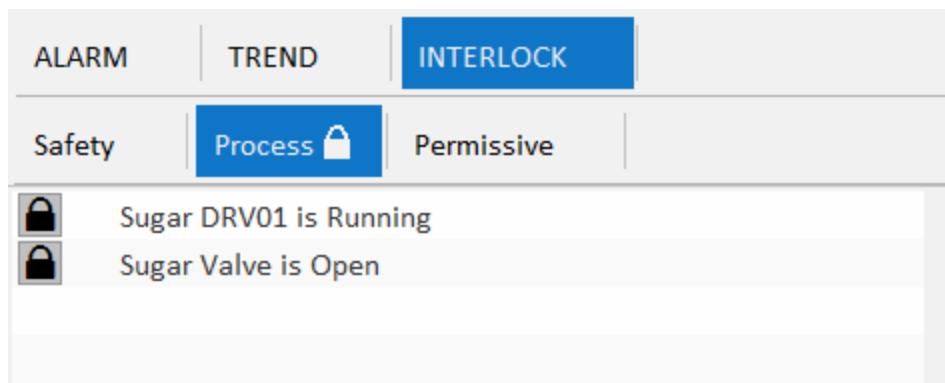
Safety and Process conditions are similar in that both types stop an action when the interlock is triggered. Permissive interlocks are slightly different as they stop an action or process from starting when an interlock is triggered. Process and Permissive interlock conditions may overlap.

In the ExampleSA project, the **Category** specified for each of the equipment references associated with the OV07 output valve is "Interlock.Process".

References				
	Equipment	Referenced Equipment	Referenced Item	Category
1	Company.TopMilk.Mixing.OV07	Company.TopMilk.Ingredients.Sugar.VLV09	Open	Interlock.Process
2	Company.TopMilk.Mixing.OV07	Company.TopMilk.Ingredients.Sugar.DRV01	Running	Interlock.Process
3	Company.TopMilk.Mixing.OV07	Company.TopMilk.Ingredients.Strawberry.DRV03	Running	Interlock.Process
4	Company.TopMilk.Mixing.OV07	Company.TopMilk.Ingredients.Strawberry.VLV11	Open	Interlock.Process
5	Company.TopMilk.Mixing.OV07	Company.TopMilk.Ingredients.Chocolate.DRV04	Running	Interlock.Process
6	Company.TopMilk.Mixing.OV07	Company.TopMilk.Ingredients.Chocolate.VLV12	Open	Interlock.Process
7	Company.TopMilk.Mixing.OV07	Company.TopMilk.Ingredients.Malt.DRV02	Running	Interlock.Process
8	Company.TopMilk.Mixing.OV07	Company.TopMilk.Ingredients.Malt.VLV10	Open	Interlock.Process

When you view the interlocks at runtime, these categories are represented as a set of tabs in the [Information Zone](#). If an interlock is currently active for a selected piece of equipment, a lock symbol is displayed next to the corresponding category tab.

Based on the category specified, all the interlocks associated with OV07 will appear on the "Process" tab.



The interlocks are listed on each tab from oldest (triggered first) to most recent.

Interlocks can be configured to support a bypass. This means an operator can manually override the interlock, forcing the locked process to be available to start (see [Configure a Bypass for an Interlock](#)). When enabled, an interlock can be bypassed by an operator in the [Information Zone](#).

## See Also

[Configure Interlocks](#)

## Getting Started with a Situational Awareness Project

The following topics will guide you through the process required to create and configure a Situational Awareness Project based on a Starter Project.

- [Create a Situational Awareness Project](#)
- [Set Up a Screen Profile for Multiple Monitors](#).

A project created using the Situational Awareness Starter Project can be compiled immediately, however, there are some settings in the Computer Setup Wizard that you will need to adjust to display the correct content at startup. See [Run the Computer Setup Wizard](#).

## See Also

[Key Components of a Situational Awareness Project](#)

[Configure Your Project](#)

## Create a Situational Awareness Project

When you create a Situational Awareness project, it is recommended that you begin with a Starter project. This provides a set of components that are required to support fundamental runtime functionality. You can then modify and extend a project to suit your needs.

### To create a Situational Awareness project:

1. [Create a Project Using a Starter Project](#).

2. In **Project** field, select "SA\_Style\_1\_MultiRes".

The project that is created will include the following components:

- Two master pages defining a default workspace layout - one in HD 1080 and one in UHD 4K resolution.
- Two sets of page templates (one in HD 1080 and one in UHD 4K) for the pre-configured workspace content.
- A cluster named "Cluster1".
- A server of each server types: "IOServer1", "AlarmServer1", "ReportServer1" and "TrendServer1".
- A memory I/O device named "Internal".
- An I/O device for calculated variables named "Cicode".
- Two pre-configured menus named "Headerbar" and "InformationZone".
- Default Content Types for classifying page content.
- Default Alarm Categories and Priorities.
- A default screen profile targeting your primary screen.
- A set of parameters, accessible from the **Parameters** view in Plant SCADA Studio's **Setup** activity.
- A role named "Administrators" which is linked to the "BUILTIN\Administrators" Windows group and have global privilege of 1 to 8. This allows a user to log in Plant SCADA at runtime using a Windows user account who belongs to the Administrators group of the local machine.

The project will also include three system projects:

- **SA\_Include** – hosts content that is used by a Starter Project. It also includes a set of labels.
- **SA\_Library** – includes graphical content you can use to represent equipment such as meters, drives and valves. This includes a set of Composite Genies that provide a template to easily create multiple objects with a consistent look and feel.
- **SA\_Controls** – includes Genies and controls that support the functionality provided by a Situational Awareness project.

A newly created project can be compiled immediately, however, there are some settings in the Computer Setup Wizard that you will need to adjust to display the correct content at startup. See [Run the Computer Setup Wizard](#).

## See Also

[Set Up a Screen Profile for Multiple Monitors](#)

## Set Up a Screen Profile for Multiple Monitors

Plant SCADA allows you to view your runtime project on multiple monitors. To do this:

1. Create a [Screen Profile](#) in the **Setup** activity.
2. Compile the project.
3. Run the Setup Wizard and complete the [Screen Setup](#) page.

If your project was created using the Situational Awareness Starter Project, you can use the **F12** keyboard key at runtime to reset a display client to its default arrangement of screens.

## See Also

[Run the Computer Setup Wizard](#)

[MultiMonitor Parameters](#)

## Run the Computer Setup Wizard

After compiling your Situational Awareness project, you need to configure pages that you want displayed on startup. To do so, follow these steps:

1. [Run the Setup Wizard](#).
2. Click through the pages of the wizard until you reach the **Screen Setup** page.
3. On the Screen Setup page, select a **Startup Page** for each screen (see [Screen Setup](#)).

You will need to select a master page. To just see a list of master pages in the drop-down menu, select the **Show master pages only** check box. The default master pages provided by the starter project are "Master\_PageMenu1\_HD1080" and "Master\_PageMenu1\_UHD4K". Select the page that matches the resolution of the screen.

4. You will also need to select a **Context** for each screen. The context sets the starting point for any navigation that happens on the screen. It can be any Level1 menu item belonging to the Navigation Menu. If you are running a new project created from the Situational Awareness Starter project, "MyPlant" will be available to select.

## See Also

[Set Up a Screen Profile for Multiple Monitors](#)

## Configure Your Project

The following topics describe how to configure your project content so that it engages with the architecture that supports contextual updates at runtime.

- [Add Equipment Using Equipment Editor](#)
- [Generate Variable Tags](#)
- [Create Content Pages](#)
- [Assign a Content Type to a Page](#)
- [Prepare the Navigation Menu](#)
- [Enable Navigation Zone Alarm Counts](#)
- [Configure Panes on a Master Page](#)
- [Set the Context Mode for a Workspace](#)
- [Configure Interlocks](#)

- [Create a New Faceplate](#)
- [Configure Equipment Selection for Group Objects.](#)

## See Also

[Key Components of a Situational Awareness Project](#)

## Add Equipment Using Equipment Editor

The first step towards adding content for a Situational Awareness project is to create the required [Equipment](#) for your project, which involves the following four tasks.

## Create Custom Equipment Templates

The Situational Awareness Starter Project provides a set of sample equipment templates to help with creating equipment. The equipment template is an XML file that uses proprietary tags and attributes to specify the fields of input and output databases, and define filters and transformation rules that create tags from existing database fields. It uses the existing [TagGen XML Template](#) syntax rules.

As part of creating an equipment template, you need to tailor the equipment templates to suit your physical device. You may need to complete one or more of the following tasks to create a custom template:

1. Define [Equipment Types](#).
2. Add an Item to an [Equipment Type](#) to the equipment types.
3. Add [Configuration Parameters](#) to an [Equipment Type](#) for each equipment type. Parameters defined here would typically be the same across every instance of a particular type of equipment. For example, range of values for the equipment or alarm names and their default values.
4. Create custom tags, if required. You may want to define custom tags (parameters) that may be different depending on the type of equipment. For example, the address or engineering units for different types of meters. In this case, you can [Add a Variable Tag](#) in Plant SCADA Studio.

---

**Note:** By default, the sample equipment templates use analog alarming limits. If you wish to use PLC limits with equipment, see [Associate PLC Limits with Equipment](#).

---

## Create Equipment Instances

Create the required [Equipment Instances](#) in [Equipment Editor](#) based on [Equipment Types](#).

## Organize Equipment into a Hierarchy

An [Equipment Hierarchy](#) provides a model of a plant that can be used to reference the machinery or processes being monitored.

In the case of a Situational Awareness project, equipment definitions are used to drive the functionality that enables a runtime system to respond to the current context of a display client. They can also support navigational features and area-based security. For an example see [Equipment Hierarchy Example](#).

When you configure equipment instances for a project based on the Situational Awareness Starter Project, you

should pay particular attention to the following Equipment Properties:

Property	Description
Display Name	This property allows you to create a meaningful variation of an equipment name that can be easily accommodated in the runtime interface.
Type	<p>This property specifies the <a href="#">Equipment Types</a> that is associated with a piece of equipment. In a project created from a Situational Awareness Starter project, a set of default equipment types are available to support the Composite Genies that are in the SA_Library include project.</p> <p>You should select the equipment type associated with the Composite Genie you will use to create the graphic object for the piece of equipment. For example, if the piece of equipment is a pump, you will be able to create the associated graphic object using the Drive Composite Genie. The equipment type you will need is drive.</p>
Page	<p>This property associates a piece of equipment with a page, that is, the primary page on which the associated graphic object appears. If the graphic object appears on more than one page, the page specified here should represent its primary operational context.</p> <p>This enables an operator to navigate directly to the page that hosts a piece of equipment. For example, on an alarm page you can navigate from a selected piece of equipment in an alarms list directly to the page on which it is displayed.</p> <p>This property is also used to calculate alarm counts for a page in the Navigation Zone.</p> <p><b>Note:</b> If a piece of equipment does not have the <a href="#">Page</a> property set, it will not be included in the alarm counts for its host page. For more information, see <a href="#">Enable Navigation Zone Alarm Counts</a>.</p>
Content	<p>This property allows you to associate the piece of equipment with a particular type of graphical content (see ). For example, you can associate a pump with a drive faceplate ("FP_DOL" or "FP_VSD") or an area with an overview page ("Area_L1"). You can specify multiple content types for each equipment definition using a comma-separated list.</p> <p>When the piece of equipment comes into context, any</p>

Property	Description
	panes that are set to autofill will read this property. If a match is made, the pane will display the specified content. For more information, see <a href="#">Autofill</a> .
Tag Prefix	This property is used to apply a prefix to any variable tags associated with the equipment. This is useful for the creation of the variable tags required to support a Situational Awareness Project.
Area	In a Situational Awareness project, Areas are used to manage user access and privileges at runtime. Use a area number (or label) to assign the equipment to a particular area. Only users with access to this area (and any necessary privileges) will be able to perform operations on the equipment.

## Generate Tags

If you are creating a new Situational Awareness project, after you have created your equipment hierarchy, you need to generate variable tags for the equipment (see [Update Equipment](#)).

**Note:** Tags generated this way are read-only, that is, their values cannot be updated in Plant SCADA Studio.

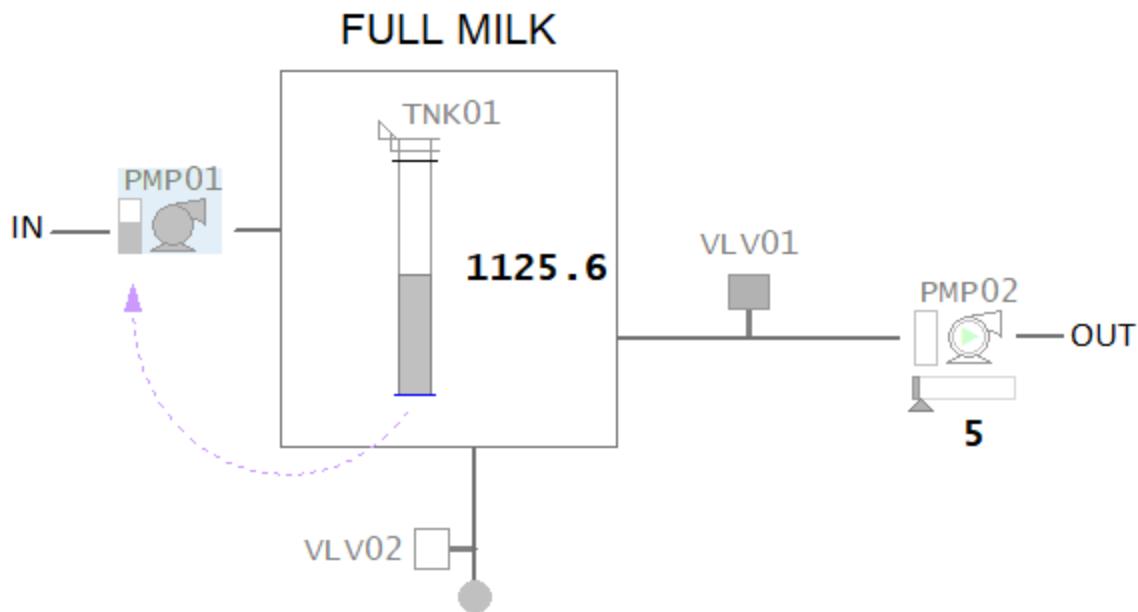
If you are upgrading a previously created project, and would like to use the Situational Awareness Genies with tags in the old project, you will need to migrate the existing tags.

### To migrate existing tags:

1. In the [Equipment Editor](#), build your equipment hierarchy in the right pane. To do this, right click and select **New Equipment**. The Add New Equipment dialog box is displayed.
2. Specify the **Name** and **Project**.
3. Leave the Type as "No type".
4. Click **OK**.
5. In Plant SCADA Studio, navigate to the **System Model** activity. Select **Equipment**.
6. Use the column filtering options to filter the tag names for a piece of equipment.
7. Add the equipment name as specified in the Equipment Editor in step 2.
8. Add the item name for the equipment. Items Names should be the same as the Items displayed in the Equipment Editor, Equipment Types tab.
9. Create the required graphics pages with the equipment instances.
10. [Configure Content Types](#).

## Equipment Hierarchy Example

The example below is a graphics page that represents a tank in a milk production facility.



Composite Genies have been used to represent the following equipment:

- A tank (TNK01)
- An inlet pump for the tank (PMP01)
- An outlet pump for the tank (PMP02)
- A valve for the outlet pump (VLV01)
- A dump valve for the tank (VLV02).

This page would require the following equipment hierarchy to be configured in the Equipment Editor.



Click on an equipment name below for an explanation of the associated equipment property settings.

## TNK01

- **Display Name** — "Supply Tank" has been specified as a name that can be used to identify the tank at runtime, providing a meaningful alternative to the equipment name "MyPlant.FullMilk.TNK01".
- **Type** — the Tank Level will be represented on the page as a 'Meter' Composite Genie. The sample equipment template 'Meter' is used as the equipment type as it contains all the necessary items required for the Composite Genie and its associated faceplate.
- **Page** — TNK01 is represented in its operational capacity on the "FullMilk" page. Therefore, "FullMilk" is specified as the home page for the tank. This means you will be able to directly navigate to the FullMilk page from occurrences of TNK01 in places such as an alarms list.
- **Content** — The content associated with the tank is "FP\_CTRL", which is the name of a faceplate for an Analog Controller. This is one of the default meter faceplates included in a project based on a Situational Awareness Starter Project. When the tank comes into context, the FP\_CTRL faceplate will display in any panes that are set to autofill and have their **Content Type** field set to "FP". See [Content Types](#).

## PMP01

- **Display Name** — "Inlet Pump" has been specified as a name that can be used to identify the pump at runtime, providing a meaningful alternative to the equipment name "MyPlant.FullMilk.PMP01".
- **Type** — the inlet pump will be represented on the page as a 'Drive' Composite Genie. The sample equipment template 'Drive' is used as the equipment type as it contains all the necessary items required for the Composite Genie and its associated faceplate.
- **Page** — PMP01 is represented in its operational capacity on the "FullMilk" page. Therefore, "FullMilk" is specified as the home page for the pump. This means you will be able to directly navigate to the FullMilk page from occurrences of PMP01 in places such as an alarms list.
- **Content** — The content associated with PMP01 is "FP\_DOL", which is the name of a faceplate for a Direct On Line Drive. This is one of the default drive faceplates included in a project based on a Situational Awareness Starter Project. When the pump comes into context, the FP\_DOL faceplate will display in any panes that are set to autofill and have their **Content Type** field set to "FP".

## PMP02

- **Display Name** — "Outlet Pump" has been specified as a name that can be used to identify the pump at runtime, providing a meaningful alternative to the equipment name "MyPlant.FullMilk.PMP02".
- **Type** — the outlet pump will be represented on the page as a 'Drive' Composite Genie. The sample equipment template 'Drive' is used as the equipment type as it contains all the necessary items required for the Composite Genie and its associated faceplate.
- **Page** — TNK01 is represented in its operational capacity on the "FullMilk" page. Therefore, "FullMilk" is specified as the home page for the pump. This means you will be able to directly navigate to the FullMilk page from occurrences of PMP02 in places such as an alarms list.
- **Content** — The content associated with PMP02 is "FP\_VSD", which is the name of a faceplate for a Variable Speed Drive. This is one of the default drive faceplates included in a project based on a Situational Awareness Starter Project. When the pump comes into context, the FP\_VSD faceplate will display in any panes that are set to autofill and have their **Content Type** field set to "FP".

## VLV01

- **Display Name** — "Outlet Valve" has been specified as a name that can be used to identify the valve at runtime, providing a meaningful alternative to the equipment name "MyPlant.FullMilk.VLV01".
- **Type** — the outlet valve will be represented on the page as a 'Valve' Composite Genie. The sample equipment template 'Valve' is used as the equipment type as it contains all the necessary items required for the Composite Genie and its associated faceplate.
- **Page** — VLV01 is represented in its operational capacity on the "FullMilk" page. Therefore, "FullMilk" is specified as the home page for the valve. This means you will be able to directly navigate to the FullMilk page from occurrences of VLV01 in places such as an alarms list.
- **Content** — The content associated with the VLV01 is "FP\_VLV", which is the name of a faceplate for a Simple Valve. This is one of the default meter faceplates included in a project based on a Situational Awareness Starter Project. When the valve comes into context, the FP\_VLV faceplate will display in any panes that are set to autofill and have their **Content Type** field set to "FP".

## VLV02

- **Display Name** — "Dump Valve" has been specified as a name that can be used to identify the valve at runtime, providing a meaningful alternative to the equipment name "MyPlant.FullMilk.VLV02".
- **Type** — the dump valve will be represented on the page as a 'Valve' Composite Genie. The sample equipment template 'Valve' is used as the equipment type as it contains all the necessary items required for the Composite Genie and its associated faceplate.
- **Page** — VLV02 is represented in its operational capacity on the "FullMilk" page. Therefore, "FullMilk" is specified as the home page for the valve. This means you will be able to directly navigate to the FullMilk page from occurrences of VLV02 in places such as an alarms list.
- **Content** — The content associated with the VLV02 is "FP\_VLV", which is the name of a faceplate for a Simple Valve. This is one of the default meter faceplates included in a project based on a Situational Awareness Starter Project. When the valve comes into context, the FP\_VLV faceplate will display in any panes that are set to autofill and have their **Content Type** field set to "FP".

## See Also

[Key Components of a Situational Awareness Project](#)

## Generate Variable Tags

After you have [Add Equipment Using Equipment Editor](#), you need to generate variable tags for the equipment if you are creating a new Situational Awareness project. If you are upgrading a previously created project, and would like to use the Situational Awareness Genies with tags in the old project, you will need to migrate the existing tags.

## Generate Tags for a New Project

Plant SCADA provides sample equipment templates that you can use as guide to create your equipment.

**To generate tags for a new project:**

1. Make a copy of the required equipment templates.
2. Modify the templates to suit your requirements. This is because the templates may not be suitable for certain protocols such as Modbus. In this case, you will need to delete the **Address** property.
3. [Add Equipment Using Equipment Editor](#).
4. Generate the tags (see [Update Equipment](#)).

## Migrating Existing Tags for an Upgraded Project

**To generate tags for an upgraded project:**

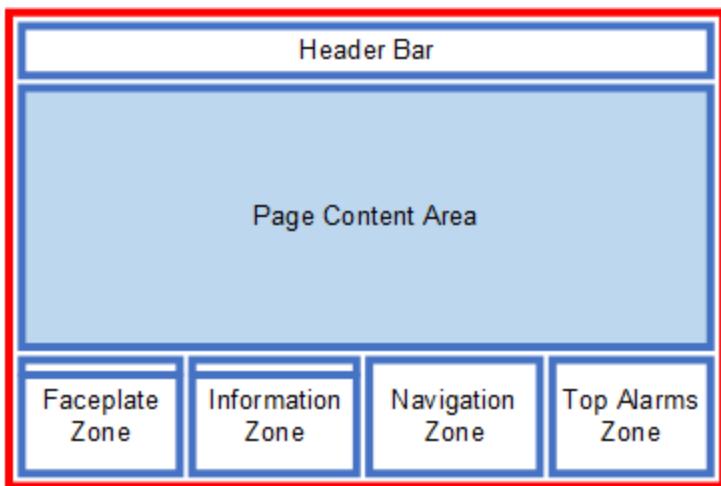
1. In the Equipment Editor, build your equipment hierarchy in the right pane. To do this, right click and select **New Equipment**. The Add New Equipment dialog box is displayed.
2. Specify the **Name** and **Project**.
3. Leave the Type as "No type".
4. Click **OK**.
5. In Plant SCADA Studio, navigate to the **System Model** Activity. Select **Equipment**.
6. Use the column filtering options to filter the tag names for a piece of equipment.
7. Add the equipment name as specified in the Equipment Editor in step 2.
8. Add the item name for the equipment. Items Names should be the same as the Items displayed in the Equipment Editor, Equipment Types tab.
9. Create the required graphics pages with the equipment.
10. Configure the required [Content Types](#).

---

**Note:** If you need only alarm indicators for your upgraded project, create the equipment and map the equipment to alarms in Plant SCADA Studio, **System Model** Activity, **Alarms**.

## Create Content Pages

Content pages display the user-created content that represents a production process or overview. With a Situational Awareness project, it is recommended that you create content pages that are sized according to the pane in which they will appear. In a default Starter Project, this is the Page Content Area.



As the Situational Awareness workspace is available in two screen sizes (HD1080 and UHD4K), the size of Page Content Area will differ according to the resolution of the workspace master page. You need to consider this resolution when you create content pages. A template page named "pagecontent" is available in the SA\_Include project in both resolutions.

If you are building a system that supports multiple resolutions, and you want to adjust the content to suit the increase/decrease in space, then you can apply a suffix to a page name to associate it with a particular resolution. For example, you could create the following pages with duplicated content:

- MyContentPage\_HD1080
- MyContentPage\_UHD4K.

If you use the function `Navigation_ShowTargetPageMultiRes()` to call "MyContentPage", the system will use the suffix to determine which variation of the page to display in the current workspace. For more information, see [Configure a Project that Supports Multiple Screen Resolutions](#).

Your content pages should also adhere to a hierarchy that provides different levels of detail about the processes being monitored. This allows an operator to assess a specific process while maintaining a broad view of the overall status of a plant. For more information, see [Page Levels](#).

#### To create a content page for a Situational Awareness project:

1. Open Graphics Builder.
2. On the **File** menu, select **New**.
3. On the New dialog box, click **Page**. The Use Template dialog box is displayed.
4. In the **Style** field, select "situational\_awareness".
5. In the **Resolution** field, select the resolution appropriate to the host workspace:
  - HD1080 (1920 x 1080, 16:9)
  - UHD4K (3840 x2160, 16:9)
6. In the **Template** section, select "pagecontent".
7. Click **OK**.
8. Save the page with an appropriate name.

If you are creating duplicate pages for a system that uses multiple resolutions, apply the appropriate suffix ("\_HD1080" or "\_UHD4K").

---

**Note:** When you use the PageContent template to create a page, it will include two pre-configured associations that allow the page to engage with a project's equipment hierarchy. Named "\_\_EquipmentName" and "\_\_EquipmentRef", they are accessible via the **Associations** tab of the Page Properties dialog.

---

To activate a page at runtime, you also need to perform the following steps:

- **Assign a Content Type**

It is recommended that you assign a content type to a page to allow it to be correctly handled by the [Autofill](#) process. The content type is used to determine where a page will display within a workspace. For example, if the content type for a page is "L1" (a level one page), it will only display in panes that are configured to display L1 content.

If a page does not have a content type specified, it will display in the pane designated as the "default" pane. By default, this will be the Page Content Area.

For more information, see [Assign a Content Type to a Page](#).

- **Enable navigation**

Runtime navigation for a Situational Awareness project is managed via the [Navigation Zone](#) on the operator dashboard.

To add the page you create to the Navigation Zone, you need to add a menu entry for it using the **Visualization** activity in Plant SCADA Studio. For more information, see [Prepare the Navigation Menu](#).

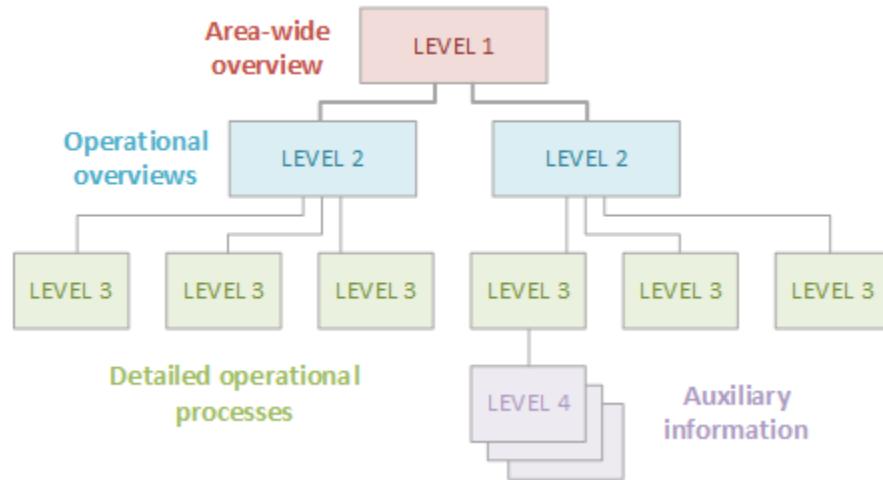
## See Also

[Key Components of a Situational Awareness Project](#)

## Page Levels

The principles of Situational Awareness propose that your content pages should adhere to a hierarchy that provides different levels of detail about the processes being monitored. This allows an operator to assess a specific process while maintaining a broad view of the overall status of a plant.

The following diagram shows the proposed hierarchy of pages.

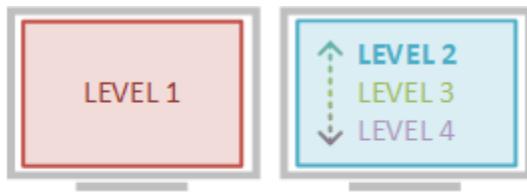


- **Level 1** – Area-wide overview. Pages include KPIs and summary status information.

- **Level 2** – Operational overview. This is the primary operational display. Pages include key operational information related to a particular production process within a facility.
- **Level 3** – Detailed operational processes. Pages includes information about the equipment in a production process and their connections. Level 3 pages are the most similar to traditional SCADA system graphics pages.
- **Level 4** – Auxiliary information. Includes trends, help screens, and so on.

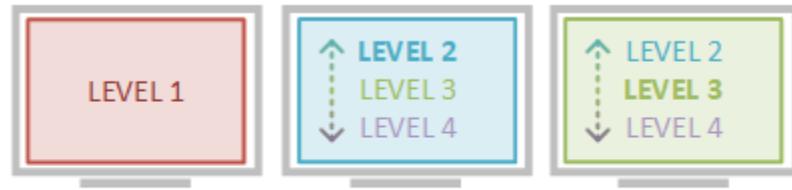
Ideally, a client workstation should have multiple screens with different page levels displayed on each screen. This allows a broader perspective to be maintained while operational details are investigated on lower-level pages.

Typically, a Level 1 page will be on a dedicated screen. An operator would then use a second screen to navigate between Levels 2–4.



Client workstation monitors

In a large system, you might consider having a screen dedicated for each Level 2 area.



Client workstation monitors

You need to consider the impact the page levels will have on your equipment definitions, in particular the **Page** and **Content** properties (see [Add Equipment Using Equipment Editor](#)). If the equipment hierarchy is pre-determined or needs to conform to an industry standard, you should use it as a starting point and plan your page levels accordingly.

If you are using the [Autofill](#) modes "WS\_CONTEXTMODE\_CurrentThenUpThenDown" or "WS\_CONTEXTMODE\_CurrentThenUp", you also need to consider how the page levels align with your equipment hierarchy, as the autofill process will be scanning up and down the hierarchy looking for content to place in your workspace.

## See Also

[Create Content Pages](#)

[Prepare the Navigation Menu](#)

## Assign a Content Type to a Page

When you specify a content type for a page, you determine how the page will be managed by the [Autofill](#)

process and the following workspace page display functions:

- Navigation\_ShowTargetPage
- Workspace\_ShowContent.

For example, if the content type for a page is "L1" (a level one page), it will only display in panes that are configured to display L1 content (see [Page Levels](#)).

Using Content Types, combined with panes that respond to Content Types, will help simplify the maintenance of your system as you will not need to explicitly state where content needs to be shown every time you instruct the system to display a content page. Instead, the system will work this out for itself using your pane configuration and the content type of the page to be displayed.

The content types you can assign to a page are configured in the **Visualization** activity in Plant SCADA Studio (see [Content Types](#)).

#### To assign a content type to a page:

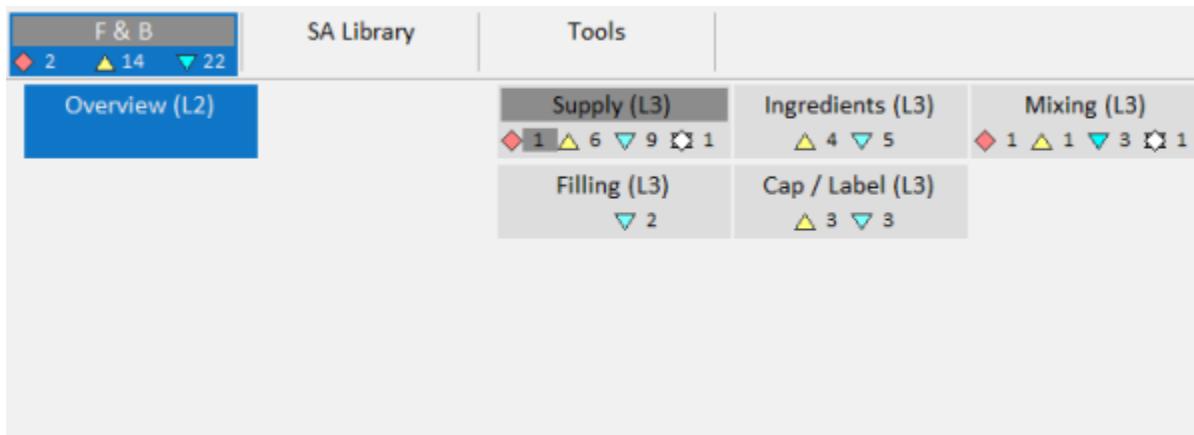
1. In the **Visualization** activity, select **Pages**.
2. Select the required page(s). See [Browse Pages in Plant SCADA Studio](#).
3. Go to the **Content Type** field in the Property Grid.
4. Enter or select a content type. The drop-down list contains the content types that are available to the active project.  
If your project was created using the Situational Awareness Starter project, a default set of content types will be available via the SA\_Include project. For a description, see [Content Types](#).  
If you need to create a new content type, see [Configure Content Types](#).
5. Click **Save**.

#### See Also

[Key Components of a Situational Awareness Project](#)

### Prepare the Navigation Menu

Runtime navigation for a Situational Awareness project is primarily managed via the **Navigation Zone** on the operator dashboard. The image below shows the Navigation Zone from the Example SA project.



The layout of the tabs and buttons in the Navigation Zone is determined by a project's Menu Configuration, which is created using the **Visualization** activity in Plant SCADA Studio (see [Menu Configuration](#)).

The navigation menu for a Situational Awareness project needs to have "Navigation" entered in the **Page** field for each menu entry.

	Page	Level 1	Level 2	Level 3	Menu Command	Target Page	Order
1	Navigation	MyPlant			Navigation_ShowTargetPage()	WhatsNew	
2	Navigation	MyPlant	F & B	Ingredients (L3)	Navigation_ShowTargetPage()	Ingredients_L3	4
3	Navigation	MyPlant	F & B	Supply (L3)	Navigation_ShowTargetPage()	Supply_L3	3
4	Navigation	MyPlant	F & B	Overview (L2)	Navigation_ShowTargetPage()	Overview_L2	1
5	Navigation	MyPlant	F & B				1
6	Navigation	MyPlant	F & B	Cap / Label (L3)	Navigation_ShowTargetPage()	Cap and Label_L3	9
7	Navigation	MyPlant	F & B	Filling (L3)	Navigation_ShowTargetPage()	Filling_L3	8
8	Navigation	MyPlant	F & B	Mixing (L3)	Navigation_ShowTargetPage()	Mixing_L3	5
9	Navigation	MyPlant	SA Library	Valves	Navigation_ShowTargetPage()	Valves	8
10	Navigation	MyPlant	SA Library	Others	Navigation_ShowTargetPage()	Others	10
11	Navigation	MyPlant	SA Library	Drives	Navigation_ShowTargetPage()	Drives	7
12	Navigation	MyPlant	SA Library	Meters	Navigation_ShowTargetPage()	Meters	6
13	Navigation	MyPlant	SA Library				2
14	Navigation	MyPlant	SA Library	Mining	Navigation_ShowTargetPage()	Mining	9
15	Navigation	MyPlant	Tools	Themes	Navigation_ShowTargetPage()	ThemeConfig	4
16	Navigation	MyPlant	Tools				4

To prepare the navigation menu for a Situational Awareness project, follow the instructions below using the Menu Configuration properties as described.

## Set the home page for a workspace

To set the home page for a workspace, use the **Target Page** field for the highest **Level 1** menu entry.

In the case of the ExampleSA project, the home page is the "What's New" page that displays when runtime is launched.

	Page	Level 1	Level 2	Level 3	Menu Command	Target Page	Order
1	Navigation	MyPlant			Navigation_ShowTargetPage()	WhatsNew	
2	Navigation	MyPlant	F & B	Ingredients (L3)	Navigation_ShowTargetPage()	Ingredients_L3	4
3	Navigation	MyPlant	F & B	Supply (L3)	Navigation_ShowTargetPage()	Supply_L3	3
4	Navigation	MyPlant	F & B	Overview (L2)	Navigation_ShowTargetPage()	Overview_L2	1
5	Navigation	MyPlant	F & B				1

You can access the specified home page at runtime from the **Home** button on the header bar.



If your project was created using the Situational Awareness Starter Project, you can also access the home page using the **Home** keyboard key.

**Note:** Alarm counts will not work if the home page specified for your navigation menu differs to the home page specified for your equipment hierarchy (defined via the **Page** field at the hierarchy's root level). See [Enable Navigation Zone Alarm Counts](#).

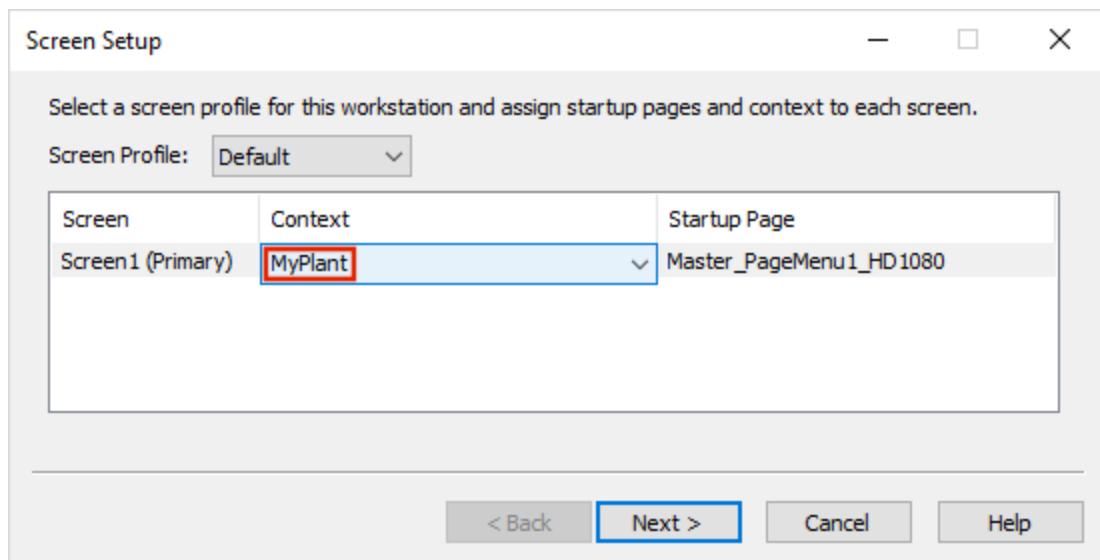
## Configure the startup context for a workspace

The **Level 1** field can be used to specify a startup context for a workspace. This creates a starting point for any navigation that occurs within the workspace.

	Page	Level 1	Level 2	Level 3	Menu Command	Target Page	Order
1	Navigation	MyPlant			Navigation_ShowTargetPage()	WhatsNew	
2	Navigation	MyPlant	F & B	Ingredients (L3)	Navigation_ShowTargetPage()	Ingredients_L3	4
3	Navigation	MyPlant	F & B	Supply (L3)	Navigation_ShowTargetPage()	Supply_L3	3
4	Navigation	MyPlant	F & B	Overview (L2)	Navigation_ShowTargetPage()	Overview_L2	1
5	Navigation	MyPlant	F & B				1
6	Navigation	MyPlant	F & B	Cap / Label (L3)	Navigation_ShowTargetPage()	Cap and Label_L3	9
7	Navigation	MyPlant	F & B	Filling (L3)	Navigation_ShowTargetPage()	Filling_L3	8
8	Navigation	MyPlant	F & B	Mixing (L3)	Navigation_ShowTargetPage()	Mixing_L3	5
9	Navigation	MyPlant	SA Library	Valves	Navigation_ShowTargetPage()	Valves	8
10	Navigation	MyPlant	SA Library	Others	Navigation_ShowTargetPage()	Others	10
11	Navigation	MyPlant	SA Library	Drives	Navigation_ShowTargetPage()	Drives	7
12	Navigation	MyPlant	SA Library	Meters	Navigation_ShowTargetPage()	Meters	6
13	Navigation	MyPlant	SA Library				2
14	Navigation	MyPlant	SA Library	Mining	Navigation_ShowTargetPage()	Mining	9
15	Navigation	MyPlant	Tools	Themes	Navigation_ShowTargetPage()	ThemeConfig	4
16	Navigation	MyPlant	Tools				4

If you have a display client with multiple monitors, you can use assign a different **Level 1** entry to each screen.

To associate a **Level 1** entry with the workspace on a particular screen, use the **Context** column on the **Screen Setup** page of the Computer Setup Wizard.



At runtime, the **Level 1** entry will appear in the menu breadcrumbs on the workspace header bar.



## Create the Navigation Zone tabs

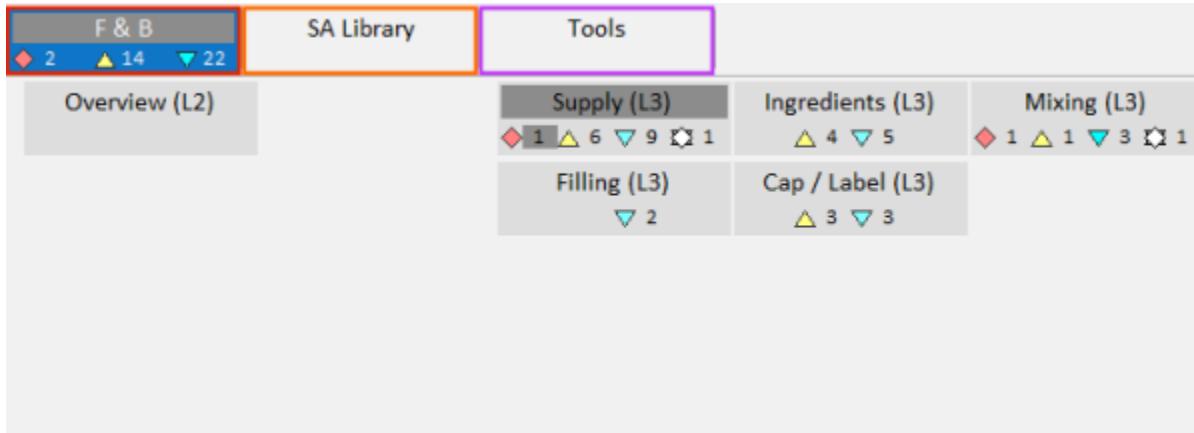
The **Level 2** entries set the tabs along the Navigation Zone header.

Page	Level 1	Level 2	Level 3	Menu Command	Target Page	Order
1	Navigation	MyPlant		Navigation_ShowTargetPage()	WhatsNew	
2	Navigation	MyPlant	F & B	Navigation_ShowTargetPage()	Ingredients_L3	4
3	Navigation	MyPlant	F & B	Navigation_ShowTargetPage()	Supply_L3	3
4	Navigation	MyPlant	F & B	Navigation_ShowTargetPage()	Overview_L2	1
5	Navigation	MyPlant	F & B			1
6	Navigation	MyPlant	F & B	Navigation_ShowTargetPage()	Cap and Label_L3	9
7	Navigation	MyPlant	F & B	Navigation_ShowTargetPage()	Filling_L3	8
8	Navigation	MyPlant	F & B	Navigation_ShowTargetPage()	Mixing_L3	5
9	Navigation	MyPlant	SA Library	Navigation_ShowTargetPage()	Valves	8
10	Navigation	MyPlant	SA Library	Navigation_ShowTargetPage()	Others	10
11	Navigation	MyPlant	SA Library	Navigation_ShowTargetPage()	Drives	7
12	Navigation	MyPlant	SA Library	Navigation_ShowTargetPage()	Meters	6
13	Navigation	MyPlant	SA Library			2
14	Navigation	MyPlant	SA Library	Navigation_ShowTargetPage()	Mining	9
15	Navigation	MyPlant	Tools	Navigation_ShowTargetPage()	ThemeConfig	4
16	Navigation	MyPlant	Tools			4

By default, only five tabs will fit across the Navigation Zone for a project based on the HD1080 master page, and

six will fit for a UHD4K master page. Be aware that these limits will be reduced by one if you customize your navigation tabs to include an additional alarm count (see [Add an Additional Alarm Count to the Navigation Zone](#)). A compile error message will be generated if the number of Level 2 entries exceeds these limits.

In the case of **Level 2** menu entries, the **Order** field is used to determine how the tabs will appear from left to right.



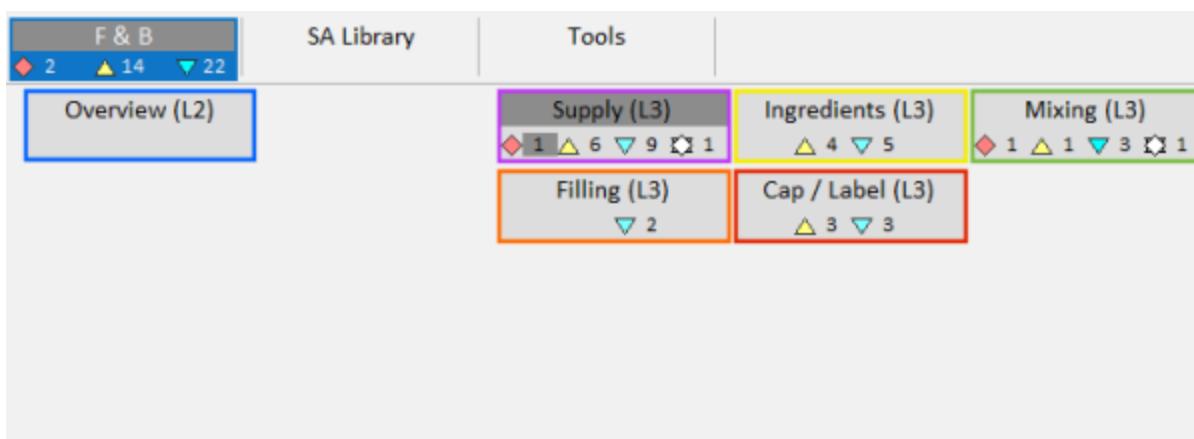
**Note:** It is recommended that you do not specify a **Target Page** for a **Level 2** menu entry, as this will allow an operator to select a tab and assess the associated alarms without navigating away from the page that is currently displayed.

## Configure the page buttons

The **Level 3** fields set the buttons that appear on each tab. Up to 20 menu entries are supported on each tab.

Page	Level 1	Level 2	Level 3	Menu Command	Target Page	Order
1	Navigation	MyPlant	F & B			1
2	Navigation	MyPlant	F & B	Cap / Label (L3)	Navigation_ShowTargetPage()	9
3	Navigation	MyPlant	F & B	Filling (L3)	Navigation_ShowTargetPage()	8
4	Navigation	MyPlant	F & B	Ingredients (L3)	Navigation_ShowTargetPage()	4
5	Navigation	MyPlant	F & B	Mixing (L3)	Navigation_ShowTargetPage()	5
6	Navigation	MyPlant	F & B	Overview (L2)	Navigation_ShowTargetPage()	1
7	Navigation	MyPlant	F & B	Supply (L3)	Navigation_ShowTargetPage()	3

The **Order** field determines the arrangement of buttons on a tab (see *Set the Order field below*).



Notice that the "Overview" button indicates that it links to an "(L2)" (level 2) page. This refers to the location of the page in the hierarchy of pages (see [Page Levels](#)), it does not relate to the fields in your menu configuration. The page buttons you create using **Level 3** of the menu configuration can link to any level in your project's page hierarchy.

## Set the Order field

The **Order** field determines the arrangement of buttons on a tab, as mapped out below.

Level 2				
Order = 1	Order = 2	Order = 3	Order = 4	Order = 5
Order = 6	Order = 7	Order = 8	Order = 9	Order = 10
Order = 11	Order = 12	Order = 13	Order = 14	Order = 15
Order = 16	Order = 17	Order = 18	Order = 19	Order = 20

If no value is specified for the **Order** property, a default value of zero (0) is applied. If two or more items on a common branch of the menu hierarchy have the same value, the layout is determined by the order in which they appear in the in the menu configuration database. The first item in the database will fill the first available gap in the layout of buttons, the next item will fill the next available gap, and so on.

## Set the Target Page field for each button

The **Target Page** field specifies the page to open when the button associated with a menu item is clicked.

The **Menu Command** field specifies the command to use to open the page associated with a menu item. Use the command "Navigation\_ShowTargetPage" in the property.

Where duplicated variations of a page exist to suit HD1080 and UHD4K screen resolutions, the page that is displayed is determined by the resolution of the host workspace. This means it is not necessary to add the suffix "\_HD1080" or "\_UHD4K" to the end of a page name specified in the **Target Page** field. For more information, see [Configure a Project that Supports Multiple Screen Resolutions](#).

---

**Note:** To enable the Navigation Zone alarm counts, the page name you specify in the **Target Page** property needs to be the same as the name used for the **Page** property for the associated equipment definition. This allows the system to count the top three priorities and shelved alarms for all equipment that is 'homed' on that page. See [Enable Navigation Zone Alarm Counts](#).

---

**Note:** To enable alarm counts in the Navigation Zone, there are a number of project settings that need to be configured correctly. For more information, see [Enable Navigation Zone Alarm Counts](#).

---

## See Also

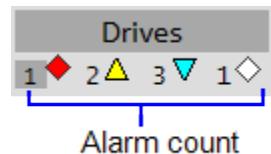
[Prioritize Alarms](#)

[Configure Display Properties for an Alarm Priority](#)

## Enable Navigation Zone Alarm Counts

Every tab and button in the Navigation Zone supports an alarm count that indicates how many alarms are currently active on the associated page (for more details, see [Navigation Zone](#)).

Each value appears to the left of an animated icon that indicates the priority the value represents.



---

**Note:** You can use the parameter [Workspace]NumberOfTopPriorities to display alarm counts for the top four alarm priorities (instead of the top three). For more information see [Add an Additional Alarm Count to the Navigation Zone](#).

---

To enable alarm counts in the Navigation Zone, you should check that the following project settings are correctly configured.

## Confirm that equipment is "homed"

To include a piece of equipment in the alarm count for a page, the piece of equipment needs to be "homed" to the page. This is achieved by setting the **Page** property in an equipment definition. The **Page** property associates a piece of equipment with the page that represents its primary operational context.

## Confirm that alarms have a category

Alarm counts require alarms to be associated with an alarm category. A category then needs to be mapped to one of the three highest priority numbers in your project. The workspace considers 1 as the highest, and 255 as the lowest.

See [Add an Alarm Category](#).

## Configure the priority icons

The priority icons are configured via the **Small Genie Name** property for each alarm priority. For more

information, see [Configure Display Properties for an Alarm Priority](#).

**Note:** Alarm counts will not work if the home page specified for your navigation menu differs to the home page specified for your equipment hierarchy (as defined in the **Page** field for the root level of your equipment hierarchy). See "Set the home page for a workspace" in the topic [Prepare the Navigation Menu](#).

## Configure Panes on a Master Page

Panes are configured while the master page for the host workspace is displayed in Graphics Builder.

You need to define settings such as:

- The page that will display in the pane at startup
- If **Autofill** is enabled for the pane
- The type of content that the pane will display
- How the content of a pane will change as the pane is resized
- If the pane has scroll bars.

### To configure a pane's properties:

1. Open Graphics Builder.
2. Open the master page for the workspace that includes the pane you would like to configure.  
If your project was created using the Situational Awareness Starter Project, the default name for the workspace master page will be "Master\_PageMenu1\_<resolution>".
3. When the master page is open, double click on the pane that you would like to configure. The Pane Properties dialog will display.
4. Make the required changes to the dialog fields. For a description of the properties, see below.
5. Click **OK**.

## Pane Properties

Property	Description
Name	The name used to identify the pane. The name needs to be unique; two panes on the same page cannot have the same name. The syslog.dat file for the client will log a message if it detects duplicate panes.
Default Page	The name of the page that the pane will display at startup. If the <b>FillMode</b> for the pane is set to an Autofill option, this page will also display when a context change results in the pane having no content to display.
Display Mode	Determines how the content of a pane will change as the pane is resized. Use one of the following integers to specify a display mode:

Property	Description
	<ul style="list-style-type: none"> <li>• <b>1 = Stretch</b> — the displayed page will continue to fill the pane as it changes size.</li> <li>• <b>2 = Maintain aspect ratio</b> — the displayed page will enlarge and reduce in size in proportion with the pane, however it will maintain its aspect ratio.</li> <li>• <b>3 = Maintain size</b> — the displayed page will not change in size as the pane is resized. If the page is larger than the pane it will be clipped, or scroll bars will appear if enabled.</li> </ul>
Is Default Pane?	<p>Set this property to TRUE if you want the pane to be the default pane within the workspace. This will make the pane the primary point of reference for the workspace. The default pane is used to display pages that have no content type defined.</p> <p>Each workspace can only contain one default pane. The syslog.dat file for a client will indicate if a default pane is not specified.</p> <p>The workspace Header Bar will refer to the page displayed in the default pane to source the active page title and breadcrumbs. This will also determine the current selection in the Navigation Zone.</p>
Display Scrollbars	<p>Determines if the pane will include scrollbars at runtime.</p> <ul style="list-style-type: none"> <li>• FALSE = No scrollbars.</li> <li>• TRUE = Use scrollbars.</li> </ul> <p>Scrollbars will only appear if the pane's <b>DisplayMode</b> property is set to 3 (maintain size), and the page displayed is larger than the pane.</p>
Fill Mode	<p>Determines how the content of a pane is updated as context changes. Use one of the following integers to specify a fill mode:</p> <ul style="list-style-type: none"> <li>• <b>Static</b> — the content and equipment assignments within the pane remain unchanged when equipment context changes.</li> <li>• <b>StaticContextMustMatch</b> — the content does not change, but any associations are updated to match the new equipment context (see below).</li> </ul>

Property	Description
	<ul style="list-style-type: none"><li>• <b>Autofill</b> — the pane will check the content specified for the equipment that comes into context. If it matches the pane's <b>ContentTypes</b> setting, the pane will display the content. If there is no valid content for the pane linked to the new equipment context, content may be sourced by traversing up/down the equipment hierarchy. See <a href="#">Autofill</a> for configuration examples.</li><li>• <b>AutofillContextMustMatch</b> — The same as Autofill, however content directly associated with the new equipment context will be used to fill the pane. The content will only be updated if the equipment is a direct contextual match (see below).</li></ul> <p>See <a href="#">Autofill</a> for configuration examples.</p>
ContentTypes	<p>Defines the type of content the pane will display. You can specify zero or more types as a comma separated list. Specifying no content type is acceptable for a static pane, but it is recommended that any other type of pane has at least one content type specified.</p> <p>The content types you can use are configured in <b>Visualization</b> activity.</p> <p>At runtime, only pages with a content type that matches with the pane will be displayed. See <a href="#">Content Types</a>.</p>
Excluded Autofill Panes	<p>Allows you to specify panes that will not display updated content when an equipment context change is triggered from this pane. When the context changes in the current pane, the content in the excluded panes will remain static. However, the context in excluded panes will still be updated. Any pane in the workspace or a linked workspace can be excluded.</p> <p>Specify the name(s) of one or more panes or nested panes that you want to exclude from autofill. Use a comma to specify more than one pane name.</p> <p>For more information, see <a href="#">.</a></p>
Tab Header Pane Name	<p>If you want the content of the pane to be controlled via a set of tabs, you can use this property to indicate the pane where the associated tab bar is located.</p> <p>Enter the name of the pane that hosts a page with the tab bar Genie on it.</p>

Property	Description
Tab Control Name	Enter the name of the tab bar Genie that you would like to use to control the content of the pane. The tab control needs to be located on the page of the pane specified in the <b>Tab Header Pane Name</b> property (see above).
Equipment Reference Associations: Categories	This field allows you to associate a pane with an equipment reference category, or a comma-separated list of equipment reference categories (see "Category" in the topic <a href="#">Define Equipment References</a> ). When a piece of equipment comes into context, any equipment references within the specified category will be used to create Super Genie associations for a displayed page. For more information, see <a href="#">AssEquipReferences</a> .

## Excluding Panes

As described above, you can use the Pane Properties dialog box to add a list of panes that you wish to exclude from autofill. Excluded panes will show the same content regardless of the context. If the current pane changes context, content in the panes specified in the **Exclude Autofil Panes** box will not be updated.

**Note:** The context within excluded panes will be updated when the context of the current page changes.

It is also possible to exclude panes at runtime by using a page environment variable. To do this:

1. Open the page that contains the context that will change.
2. Open the Page Properties dialog box.
3. Select the Environment tab.
4. Click **Add**. The Edit Properties dialog box is displayed.
5. In the **Property** box, type the property name `ExcludedAutoFillPanes`.
6. In the **Value** box, specify the name(s) of one or more panes that you want to exclude from autofill.
7. Click **OK** to close the Edit Properties dialog box.
8. Click **OK** to close the Page Properties dialog box.

**Note:** A pane can display a page that includes its own set of panes, creating a multi-level arrangements of nested panes. See [Use Nested Panes](#).

## See also

- [Set the Context Mode for a Workspace](#)  
[Create a Master Page for a Customized Workspace](#)

## Set the Context Mode for a Workspace

Context mode is a workspace setting that determines if the autofill process will respond to equipment on higher and lower levels of the equipment hierarchy when context changes.

There are three context modes you can use:

- CurrentOnly — only the current piece of equipment is considered when panes scan for the type of content they are configured to display.
- CurrentThenUp — the current piece of equipment is considered when panes scan for matching content. When this process is complete, content types on higher levels of the equipment hierarchy are matched with any remaining panes.
- CurrentThenUpThenDown — the current piece of equipment is considered when panes scan for matching content. When this process is complete, equipment on higher levels in the equipment hierarchy are matched with any remaining panes, then equipment on lower levels of the hierarchy.

The Context Mode setting should be consistent across all clients in a system.

See [Autofill](#) for configuration examples.

## Set context mode in the project database

If your project was created using the Situational Awareness Starter Project, the ContextMode parameter will already be included as a database parameter with a default value of 3 (CurrentThenUpThenDown). To configure the parameter:

1. In the **Setup** activity, select **Parameters**.
2. In the **Section Name** column, locate or enter "Workspace".
3. In the **Name** column, locate or enter "ContextMode".
4. Enter the **Value** that represents the context mode you would like to use:
  - 1 = CurrentOnly
  - 2 = CurrentThenUp
  - 3 = CurrentThenUpThenDown.
5. Click **Save** to add the record to the database.

If you set (or change) parameters in the project database, you need to re-compile the project before the new parameter settings are used.

For a description of a workspace, see [Key Components of a Situational Awareness Project](#).

## Configure Interlocks

Plant SCADA allows you to represent the interlocks that exist in a production facility within a Situational Awareness project (see [Interlocks](#)). The activity associated with an interlock can then be presented to an operator at runtime in the [Information Zone](#).

To configure interlocks in your Plant SCADA project you need to complete the following tasks.

## Create the required equipment references

Equipment References allow you to create connections in your equipment model that reflect the interlocks that exist in the field. They are defined in the **Equipment** view of the **System Model** activity (see [Define Equipment References](#) for more information).

When defining an equipment reference to represent an interlock, use the following fields:

Field	Description
Equipment	Enter the name of the piece of equipment that will change state when an interlock is triggered. By default, the "Running" item will change state when an interlock is triggered. If this is not the correct item, use the runtime parameter "Interlock_StateItem" to specify a different state item (see <i>Set the "Interlock_StateItem" Runtime Parameter</i> below).
Referenced Equipment	Enter the name of the piece of equipment where the triggering action will occur.
Referenced Item	Enter the name of the equipment item in the referenced equipment that will trigger the interlock when a change of state occurs.
Category	Identify the type of interlock you are creating. Enter the whole name of the category in the format "Interlock.<category>". The Situational Awareness Starter Project provides the following categories: <ul style="list-style-type: none"><li>• <b>Interlock.Safety</b> — conditions that cause equipment to stop running.</li><li>• <b>Interlock.Process</b> — conditions that cause equipment within a process to stop running.</li><li>• <b>Interlock.Permissive</b> — conditions that stop equipment from starting.</li></ul>
Comment	Enter a description of the interlock in the <b>Comment</b> field. If no comment is entered, the Information Zone will display the 'cluster.equipment.item' name.

You can also use an Equipment Reference to configure a bypass for an interlock. See [Configure a Bypass for an Interlock](#).

## Configure interlock trigger items

By default, the "Running" item will change state when an interlock is triggered by a referenced piece of equipment. If required, you can specify a different state item. To do this, use the runtime parameter

"Interlock\_Stateitem".

Runtime parameters are defined in the **Equipment** view of the **System Model** activity (see [Define Equipment Runtime Parameters](#) for more information).

When defining the runtime parameter, use the following fields:

Property	Description
Cluster	Enter the name of the cluster to which the equipment is assigned.
Equipment	Enter the name of the piece of equipment that will change state when an interlock is triggered.
Name	A name for the parameter, in this case "Interlock_StateItem". This indicates that an alternative item name (entered in the Value column) has been defined.
Value	Enter the name of the equipment item that will change state when an interlock is triggered.
Is Tag	Set this to FALSE. This field defaults to TRUE if no value is specified. If you do not enter FALSE, interlocks will not recognize your state item and not function correctly.

**Note:** You can change the default state item from "Running" for all equipment at a global level using the INI parameter [\[Interlock\]StateItem](#).

## Create the required alarms

Alarms are used to generate the events required to track interlock activity. They are used to determine how long an interlock has been active, or when it was last active.

To support interlocks, you need to configure the following alarms:

- One for the item that changes state when an interlock is triggered.
- One for each of the items that trigger an interlock in any referenced equipment.

**Note:** It is recommended that these alarms be hidden from the operator. To do this, assign the alarms to an alarm category that has the **ShowOnActive** property set to FALSE. If you want the operator's reason for a bypass to appear on the SOE page, set the **ShowOnSummary** to TRUE. See [Add an Alarm Category](#).

## See Also

[Configure a Bypass for an Interlock](#)

## Configure a Bypass for an Interlock

You can configure an [Interlocks](#) so that it can be bypassed. This means an operator can manually override an interlock, making a locked process available to start.

A bypass is implemented at runtime via the **Interlock** tab on the **Information Zone**. For further instructions, see [Information Zone](#).

### To configure a bypass for an interlock

A bypass is configured via the equipment reference that defines the relationship between the two pieces of equipment (see [Configure Interlocks](#)).

To enable a bypass, complete the following **Custom** properties:

- **Custom 1** — BypassStatus - enter the name of the tag that controls the bypass status for the interlock.
- **Custom 2** — BypassCommand - enter the name of the tag that when written to will bypass an interlock.
- **Custom 3** — RemoveBypassCommand - enter the name of the tag that when written to will remove a bypass for an interlock.
- **Custom 4** — EnableBypass/Reset - enter the name of a tag to disable the bypass command. If this tag exists, then the bypass and reset commands will be enabled if the value of this tag is 1.

You need to have variable tags for the bypass states for the referenced equipment. For example, for Motor1 you would configure four variable tags, one for each bypass state:

Variables				
Row	Equipment	Item Name	Tag Name	Cluster
2454	Motor1	BypassStatus	Motor1_BypassStatus	Cluster1
2455	Motor1	BypassCommand	Motor1_BypassCommand	Cluster1
2456	Motor1	RemoveBypassCommand	Motor1_RemoveBypassCommand	Cluster1
2457	Motor1	EnableBypass	Motor1_EnableBypass	Cluster1

**Note:** If custom field 4 is empty, and any of the remaining fields are completed, then the "Reset" and "Bypass" commands will be enabled. Otherwise, if the four custom fields are complete, then the "Reset" and "Bypass" commands will be disabled unless the tag specified in this field is 1.

### Bypass permissions

The ability to initiate a bypassed depends on the privileges defined for the piece of equipment that changes state in response to an interlock trigger.

The function GetPrivEx is used to retrieve the privilege of the interlocked piece of equipment. If the privilege level assigned to the current user does not match the privilege level of the interlocked equipment, the options to bypass and remove a bypass will not be available.

## SOE event journal limitations

Bypassing an interlock adds an event to the sequence of events (SOE) journal. The event message uses the following format:

<Cluster>.<Equipment>.<Item> @(**INTERLOCK BYPASSED:**)<Reason>

The length of the message field in the SOE is 254 characters. If <Cluster>.<Equipment>.<Item> has more than 160 characters, it will be truncated from the beginning.

For example, if the name of a piece of equipment is "Cluster1.Equip1.....Equip10.Item" (which contains 167 characters), the name will be truncated to "1.Equip1.....Equip10.Item".

The truncated equipment name needs to be globally unique, otherwise the bypass information will not display in the remove bypass form. A hardware alarm "Named object already exists" will also be raised.

The length of the reason an operator can specify depends on equipment name and the localized string of "INTERLOCK BYPASSED:". For example, if the equipment name is 80 characters long and the localized string of "INTERLOCK BYPASSED:" is 20 characters long, the operator can type 153 characters as the reason for the bypass. If the operator types more than 153 characters, the text will be truncated from the end.

Similarly, removing an interlock bypass adds an event to the SOE journal. The message of the event is in the following format:

<Cluster>.<Equipment>.<Item> @(**INTERLOCK BYPASS REMOVED:**)<Reason>

Again, if <Cluster>.<Equipment>.<Item> has more than 160 characters, it will be truncated from the beginning.

The length of the reason provided for removing a bypass depends on the equipment name and the localized string of "INTERLOCK BYPASS REMOVED:". For example, if the equipment name is 80 characters long and the localized string of "INTERLOCK BYPASS REMOVED:" is 26 characters long, a user can type 147 characters as the bypass reason. If the

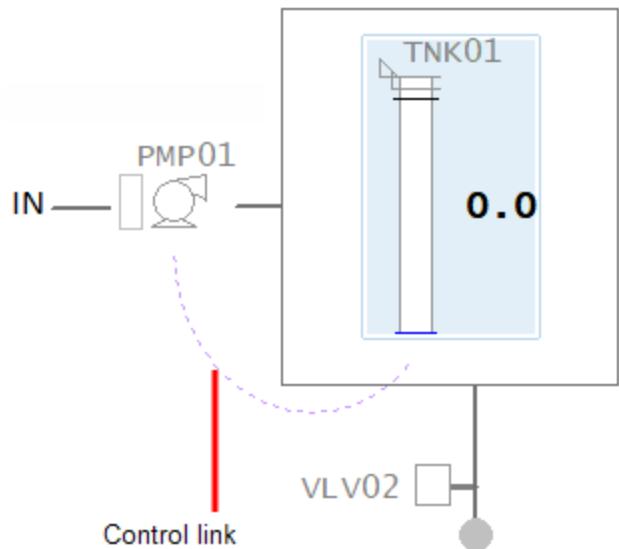
user types more than 148 characters, the text will be truncated from the end.

## See Also

[Equipment References](#)

## Add a Control Link

A control link is a visual link that indicates a relationship between a controller and an associated controller element.



A control link can be drawn in any direction. It becomes visible when either of the objects to which it is connected is selected by the user.

#### To create a content page for a Situational Awareness project:

1. In Graphics Builder, open the required page.
2. Use an ellipse object to draw a line on the page that links the two related items.

To create the required line:

- Go to the **Appearance | General** tab and select **Arc** as the ellipse object type. Use the **Start Angle** and **End Angle** settings to specify the end points for the line.
- Set the line **Color** (pink is recommended).
- Set the **Style** to **Dot**.

3. Go to the **Appearance | Visibility** tab.
4. In the **Hidden when** field, use the Cicode function to control the appearance of the control link.  
For example, to configure the control link in the image above, you would use the following:  
`NavigationLink_IsHidden("MyPlant.FullMilk.TNK01", "MyPlant.FullMilk.PMP01", TRUE)`

5. Click **OK** and save your changes.

It is recommended that you also add an arrow head to the line to indicate which piece of equipment has control. Use a filled polygon for the arrow head and create an object group. You can then perform steps 3 and 4 using the Group Properties dialog.

#### See Also

[Key Components of a Situational Awareness Project](#)

#### Create a New Faceplate

You can use one of the sample faceplates in the required resolution (HD1080 or 4K) to create your own faceplate. Sample faceplates are located in the SA\_Style\_1\_MultiRes project.

**To create a new faceplate, follow these steps:**

1. Modify an existing sample faceplate.

Or:

Create a copy of an existing faceplate.

Or:

Create a new page in Graphics Builder using the "faceplate" template in the required resolution.

---

**Note:** The sample faceplates use a specific set of items to display controls at runtime. You can build your faceplate using these items, which are listed in the topics that describe the faceplates for meters, drives and valves.

---

2. Save your page.
3. Go to Plant SCADA Studio.
4. In the **Visualization activity | Pages**, set the **Content Type** for the above page to "FP". Content of this type will automatically display in the [Faceplate Zone](#) of the workspace.
5. Click **Save**.
6. In Plant SCADA Studio, **System Model activity | Equipment**, locate the equipment for which you are creating the faceplate.
7. Add the name of the page you created earlier to the **Content** field for the equipment. It can include a comma separated list.
8. Click **Save**.

---

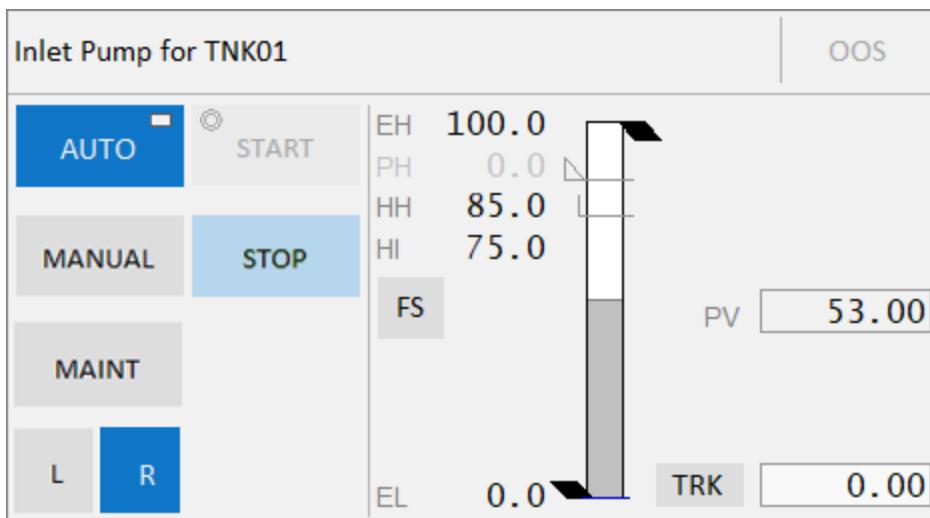
**Note:** User interaction with faceplates is achieved through command tags (items) associated with the various components of a faceplate. Items are listed in the individual faceplate topic. Buttons on a faceplate are designed to support this interaction. However, tags in Plant SCADA need to be mapped to the tags that you have configured on your PLCs. For example, tags for starting a drive (StartCmd) and displaying its status (CtrlMode) available in Plant SCADA. Similarly, tags for starting the drive and displaying its status would exist on your PLC. These two sets of tags need to be mapped to each other for the buttons on the drive faceplate to work. If the mapping is not created, buttons on the faceplates will not function. Refer to the example below to understand how this works.

---

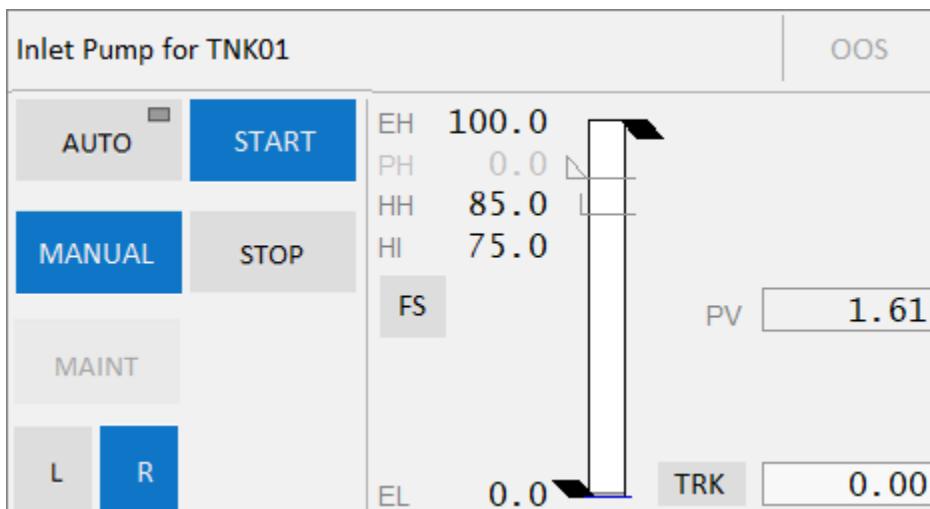
## Example

This example illustrates how tags are used to achieve interaction with faceplates.

The CtrlModeDef tag sets the default mode of operation for a piece of equipment. If you set this tag to 0 for a pump, the default mode of a pump will be Auto. That is indicated on the faceplate by a rectangle in the top-right corner of the **Auto** button on the faceplate as shown below.



The CtrlMode tag represents the current mode for a piece of equipment. If the CtrlModeDef (default) and CtrlMode (current) have different values, the current mode will be displayed in green. For example, the CtrlModeDef = 0 (Auto) and the CtrlMode = 1 (Manual), the **Manual** button will be displayed in green. Clicking the **Manual** button will set the ManCmd tag to 1.

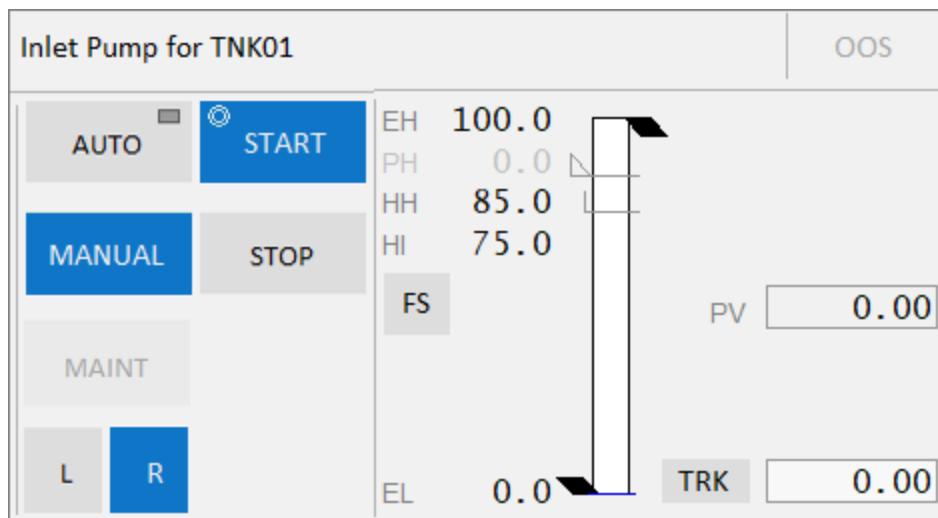


If the pump is not running, the **Start** button will also be displayed in green to allow you to manually start the pump. If the pump is running, the **Stop** button will be green allowing you to manually stop the pump. The **Auto** button will continue to display the rectangle to indicate that it is the default mode. Note that the **Start** and **Stop** buttons are available only when the mode is Manual.

---

**Note:** Clicking any button will set its <name>Cmd tag to 1. For example, clicking the **Manual** button will set ManCmd to 1.

---



When you click one of the buttons, a circle will appear in the top-left corner to indicate that a command (**Start** in the above image) has been sent to the PLC mapped to the corresponding tags. Once the Start command is completed, the circle will disappear as the PLC sets StartCmd to 0 (or StopCmd to 0 in case you clicked the **Stop** button). In addition, the PLC sets the value of the Running tag to 1 (Stopped tag if you clicked the **Stop** button). When you click Stop, the value of the Stopped tag is 1.

## See Also

[Faceplates](#)  
[Faceplate Zone](#)  
[Meter Faceplates](#)  
[Drive Faceplates](#)  
[Valve Faceplates](#)

## Configure Equipment Selection for Group Objects

Some Composite Genie can be configured to represent multiple pieces of equipment.

Examples are the Drive for a Duty/Standby Pump, or a Fan Group. The XY Bar Graph and Polar Star are other examples.

When a Composite Genie is configured to represent multiple pieces of equipment, the operator can still select it on a page. The faceplate that is displayed will be that of the first object in the group.

To allow selection of the other objects in the group, all must share the same faceplate, and references must be setup between the group object and the equipment it is grouping. The faceplate itself must also have 'n' copies of the MEO Selector Genie to allow selection of each piece of equipment.

To configure equipment selection for group objects, you need to perform the following steps.

## Configure the group of Equipment Running State Indicators using Equipment References

The link created between the grouping equipment object and the referenced equipment does not need to be one of parent/child, instead the reference can be made to equipment outside the hierarchy. See for more

information.

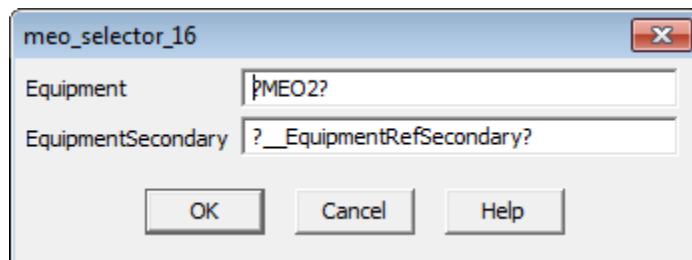
### To configure links between the grouping object and the equipment being grouped:

1. In the **System Model** activity, select **Equipment**.
2. On the menu below the Command Bar, select **References**.

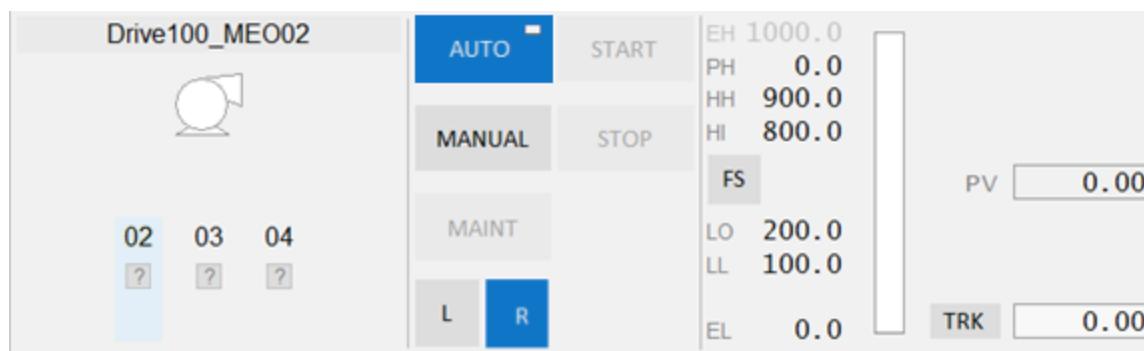
The list of equipment references will display in the Grid Editor.

Row	Cluster	Equipment	Referenced Equipment	Referenced Item	Category	Order	Association
1	Cluster1	Company.TopMilk.Mixin	Company.TopMilk.Ingredients	Open	Interlock.Process	1	
2	Cluster1	Company.TopMilk.Mixin	Company.TopMilk.Ingredients	Running	Interlock.Process	2	
3	Cluster1	Company.TopMilk.Mixin	Company.TopMilk.Ingredients	Running	Interlock.Process	4	
4	Cluster1	Company.TopMilk.Mixin	Company.TopMilk.Ingredients	Open	Interlock.Process	3	
5	Cluster1	Company.TopMilk.Mixin	Company.TopMilk.Ingredients	Running	Interlock.Process	6	
6	Cluster1	Company.TopMilk.Prod	Company.TopMilk.Production		MEO	1	Meter1
7	Cluster1	Company.TopMilk.Prod	Company.TopMilk.Production		MEO	2	Meter2
8	Cluster1	Company.TopMilk.Prod	Company.TopMilk.Production		MEO	3	Meter3

3. Add a row to the Grid Editor.
  4. Type the required information in each column, or in the field select the information from the drop down menu. (See [Define Equipment References](#) for a description of the properties of each column).
  5. In the Category column enter **MEO**.
- MEO stands for Multiple Equipment Object. So MEO1 is the first object in the group (Multiple Equipment Object 1) and so on.
6. In the Association column enter the name of the association for the referenced equipment, for example, MEO1 or MEO2 or so on. Association names are passed to the 'meo\_selector\_xx' Genie instance on faceplates using Super Genie syntax ?MEO2?. A Super Genie association is created for each reference, using the name specified in the Association column.



In the starter project the equipment with "MEO1" will appear in the first position on the faceplate (to the left), MEO2 will be in the second position, and so on. It is not necessary to start at MEO1 if you have three MEOs, then use MEO2, MEO3, MEO4 to use the center positions on the faceplate.

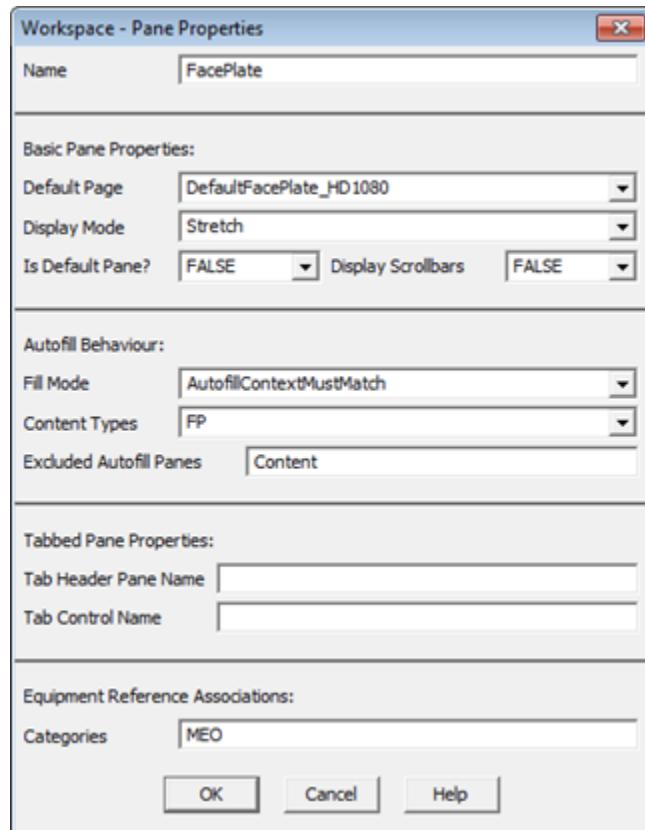


**Note:** The category name "MEO" and association names "MEO1" to "MEO5" are not fixed. They are the values used by the default master page and example faceplates in the starter project. You can create your own pages with different names and, if required, have more than five equipment selectors.

7. Complete the property grid.
8. Click **Save**.

## Verify that the Workspace Pane Properties for the faceplate has the correct category

The category is defined in the pane properties for the Faceplate pane in the Master\_PageMenu1 page:



**Note:** If you have configured your own faceplate, verify the category entered in the Workspace pane properties dialog is the same entered when creating your references to related equipment.

**Note:** In advanced configuration a workspace pane can have zero or more categories configured in the Categories field (comma-separated list). For each category, a super-genie association is created for each

---

reference, using the name specified in the Association column.

---

## See Also

[AssEquipReferences](#)

## Customize a Situational Awareness Project

The following topics outline how you can customize a Plant SCADA Situational Awareness project.

- [Configure a Project that Supports Multiple Screen Resolutions](#)
- [Create a Master Page for a Customized Workspace](#)
- [Configure Display Properties for a Fourth Alarm Priority](#)
- [Add an Additional Alarm Count to the Navigation Zone](#)
- [Add a Fourth Alarm Priority Filter to Alarm Pages](#)
- [Format an Alarm List for a Specific Screen Resolution](#)
- [Apply a Customized Font to Selected Alarms](#)
- [Customize the Logo on the Header Bar](#)
- [Add a Tree View to a Page](#)
- [Add an Alarms List to a Page](#)
- [Add a Generic List to a Page](#)
- [Add an Alarm Filter Button](#)
- [Create a Set of Tabs for a Pane](#)
- [Use Nested Panes](#)
- [Create a Pop-up Window with Panes](#)
- [Use Navigational Genies on Content Pages](#)
- [Apply a Color Theme to a Workspace](#)
- [Use a Scheduler Control in a Situational Awareness Project](#)
- [Defining a Right Click Action on Equipment](#)

## See Also

[Configure Your Project](#)

## Configure a Project that Supports Multiple Screen Resolutions

A Situational Awareness project can be configured to support multiple screen resolutions using two methods:

- By creating content that can be shared across resolutions.
- By explicitly defining content for a particular resolution using a resolution suffix. The appropriate content is then matched to the resolution of the host workspace at runtime.

At runtime, when a request is made to display a page, the workspace will first look for a page that has a resolution extension that matches the resolution of the host workspace master page. If a version of the page with the requested extension does not exist, it will next attempt to display a page without an extension.

Both these methods allows clients to use the same project configuration (including menus, equipment and pages) on different resolution screens.

You can mix and match these methods in your project. For example, you may decide that 90% of your mimics can be shared across resolutions, but 10% need to be optimized for HD and UHD resolutions.

## Create content that can be shared across screen resolutions

In most cases, the content you create will successfully adjust based upon the **Display Mode** of the pane it is displayed in (see [Configure Panes on a Master Page](#)).

If you want to share content across resolutions, do not add a resolution suffix to your faceplate or page names.

## Explicitly define content for a particular resolution

In some cases it may be necessary to explicitly create content to suit a particular resolution. For example, if you mostly have UHD4K monitors in a plant and just a few HD1080 monitors, you will want your project's content to take advantage of the available screen space on the UHD4K screens. However, you may need to create an HD1080 version of some pages for the content to display appropriately on a smaller screen. If this is the case, you could create an HD1080 version of any pages that needs to be adjusted.

Two screen resolutions are supported in this way:

- HD1080 (1920 x 1080, 16:9)
- UHD4K (3840 x 2160, 16:9)

To create content that is specific to a resolution, add one of the following extensions to the end of your faceplate or page names.

- \_HD1080
- \_UHD4K

For example, you could create the following variations of the same content page:

- **MyPage\_HD1080** page would be 1920 x 720 pixels to fit the content pane on an HD1080 workspace.
- **MyPage\_UHD4K** page would be 3840 x 1768 pixels to fit the content pane on an UHD4K workspace.

At runtime, the extension that matches the resolution of the host workspace master page will be used.

The feature also requires that anywhere a page name is called, such as the **Target Page** field of a menu, the extension needs to be *left off* the page name. The workspace will read the page name and automatically look for a version that has a matching resolution suffix.

This feature applies to:

- the **Target Page** field for a Situational Awareness navigation menu.
- the **Content** and **Page** fields for an Equipment definition.
- Workspace\_ShowContent.

- Navigation\_ShowTargetPage.

---

**Note:** When using this feature, you need to be careful not to specify a full page name (with the resolution prefix) when calling a page. This is particularly true when using Plant SCADA Studio, as the interface allows you to select a full page name.

---

## See also

[Create Content Pages](#)

[Create a Master Page for a Customized Workspace](#)

## Create a Master Page for a Customized Workspace

The SA\_Include project has a Pane Genie that you can paste on a master page, allowing you to create a customized framework for a workspace.

For a description of a workspace, see [Key Components of a Situational Awareness Project](#).

---

**Note:** Master pages need to call the function "Workspace\_Init()" in their OnPageShown event.

---

### To customize the master page for a workspace:

1. Open Graphics Builder.
2. Create a page using the "Master" template, in the resolution that will match the display client screen. Ideally, the name you give to a master page should start with "Master" to make it easy to find.

Or:

Use **Save As** to create a copy of a workspace master page that you would like to modify.

3. Verify that the Content Type applied to the page is "Master". See [Assign a Content Type to a Page](#).
4. From the Graphics Builder Edit menu, select Paste Genie.

Or:

From the Drawing Toolbox, select the Genie icon.



The Paste Genie dialog will appear.

5. In the Library field, select "sa\_workspace".
6. In the Genie section, select "pane".

The Pane Genie will appear on the page.

7. Relocate and resize the pane as required.

The Pane Properties dialog will also appear when you paste the Genie on the master page.

The dialog defines setting such as:

- The page that will display in the pane at startup
- If autofill is enabled for the pane
- The type of content that the pane will display
- How the content of a pane will change as the pane is resized

- If the pane has scroll bars.
8. Make the required changes to the dialog fields. For a description of the Pane Properties, see below.
9. Click OK.

---

**Note:** If you are editing an existing master page, be aware that any adjustments to a pane will mean any content designed for that pane should be resized accordingly. Not doing this will result in scaling of the content, which may not be desired. If resizing SA\_Include pages, make your changes on a copy of the original pages. This will mean your changes will not be lost when the content of the SA\_Include is updated in a future release.

---

## Pane Properties

Property	Description
Name	The name used to identify the pane. The name needs to be unique; two panes on the same page cannot have the same name. The syslog.dat file for the client will log a message if it detects duplicate panes.
Default Page	The name of the page that the pane will display at startup. If the <b>FillMode</b> for the pane is set to an Autofill option, this page will also display when a context change results in the pane having no content to display.
Display Mode	<p>Determines how the content of a pane will change as the pane is resized. Use one of the following integers to specify a display mode:</p> <p><b>1 = Stretch</b> — the displayed page will continue to fill the pane as it changes size.</p> <p><b>2 = Maintain aspect ratio</b> — the displayed page will enlarge and reduce in size in proportion with the pane, however it will maintain its aspect ratio.</p> <p><b>3 = Maintain size</b> — the displayed page will not change in size as the pane is resized. If the page is larger than the pane it will be clipped, or scroll bars will appear if enabled.</p>
Is Default Pane?	<p>Set this property to TRUE if you want the pane to be the default pane within the workspace. This will make the pane the primary point of reference for the workspace. The default pane is used to display pages that have no content type defined.</p> <p>Each workspace can only contain one default pane. The syslog.dat file for a client will indicate if a default pane is not specified.</p> <p>The workspace Header Bar will refer to the page</p>

	displayed in the default pane to source the active page title and breadcrumbs. This will also determine the current selection in the Navigation Zone.
Display Scrollbars	Determines if the pane will include scrollbars at runtime.  FALSE = No scrollbars.  TRUE = Use scrollbars.  Scrollbars will only appear if the pane's <b>DisplayMode</b> property is set to 3 (maintain size), and the page displayed is larger than the pane.
Fill Mode	Determines how the content of a pane is updated as context changes. Use one of the following integers to specify a fill mode:  <b>Static</b> — the content and equipment assignments within the pane remain unchanged when equipment context changes. <b>StaticContextMustMatch</b> — the content does not change, but any associations are updated to match the new equipment context (see below). <b>Autofill</b> — the pane will check the content specified for the equipment that comes into context. If it matches the pane's ContentTypes setting, the pane will display the content. If there is no valid content for the pane linked to the new equipment context, content may be sourced by traversing up/down the equipment hierarchy. <b>AutofillContextMustMatch</b> — The same as Autofill, however content directly associated with the new equipment context will be used to fill the pane. The content will only be updated if the equipment is a direct contextual match (see below).  See <a href="#">Autofill</a> for configuration examples.
ContentTypes	Defines the type of content the pane will display. You can specify zero or more types as a comma separated list. Specifying no content type is acceptable for a static pane, but it is recommended that any other type of pane has at least one content type specified.  The content types you can use are configured in <b>Visualization</b> activity.  At runtime, only pages with a content type that matches with the pane will be displayed. See <a href="#">Content Types</a> .

Excluded Autofill Panes	<p>Allows you to specify panes that will not display updated content when an equipment context change is triggered from this pane. When the context changes in the current pane, the content in the excluded panes will remain static. However, the context in excluded panes will still be updated. Any pane in the workspace or a linked workspace can be excluded.</p> <p>Specify the name(s) of one or more panes or nested panes that you want to exclude from autofill. Use a comma to specify more than one pane name.</p> <p>For more information, see <i>Excluding Panes</i> below.</p>
Tab Header Pane Name	<p>If you want the content of the pane to be controlled via a set of tabs, you can use this property to indicate the pane where the associated tab bar is located.</p> <p>Enter the name of the pane that hosts a page with the tab bar Genie on it.</p>
Tab Control Name	<p>Enter the name of the tab bar Genie that you would like to use to control the content of the pane. The tab control needs to be located on the page of the pane specified in the <b>Tab Header Pane Name</b> property (see above).</p>

Equipment Reference Associations: Categories	This field allows you to associate a pane with an equipment reference category, or a comma-separated list of equipment reference categories (see "Category" in the topic <a href="#">Define Equipment References</a> ). When a piece of equipment comes into context, any equipment references within the specified category will be used to create Super Genie associations for a displayed page. For more information, see <a href="#">AssEquipReferences</a> .
--	---

## Excluding Panes

As described above, you can use the Pane Properties dialog box to add a list of panes that you wish to exclude from autofill. Excluded panes will show the same content regardless of the context. If the current pane changes context, content in the panes specified in the Exclude Autofill Panes box will not be updated.

**Note:** The context within excluded panes will be updated when the context of the current page changes.

It is also possible to exclude panes at runtime by using a page environment variable. To do this:

1. Open the page that contains the context that will change.
2. Open the Page Properties dialog box.
3. Select the Environment tab.
4. Click **Add**. The Edit Properties dialog box is displayed.
5. In the **Property** box, type the property name `ExcludedAutoFillPanes`.
6. In the **Value** box, specify the name(s) of one or more panes that you want to exclude from autofill.
7. Click **OK** to close the Edit Properties dialog box.
8. Click **OK** to close the Page Properties dialog box.

**Note:** A pane can display a page that includes its own set of panes, creating a multi-level arrangement of nested panes. See [Use Nested Panes](#).

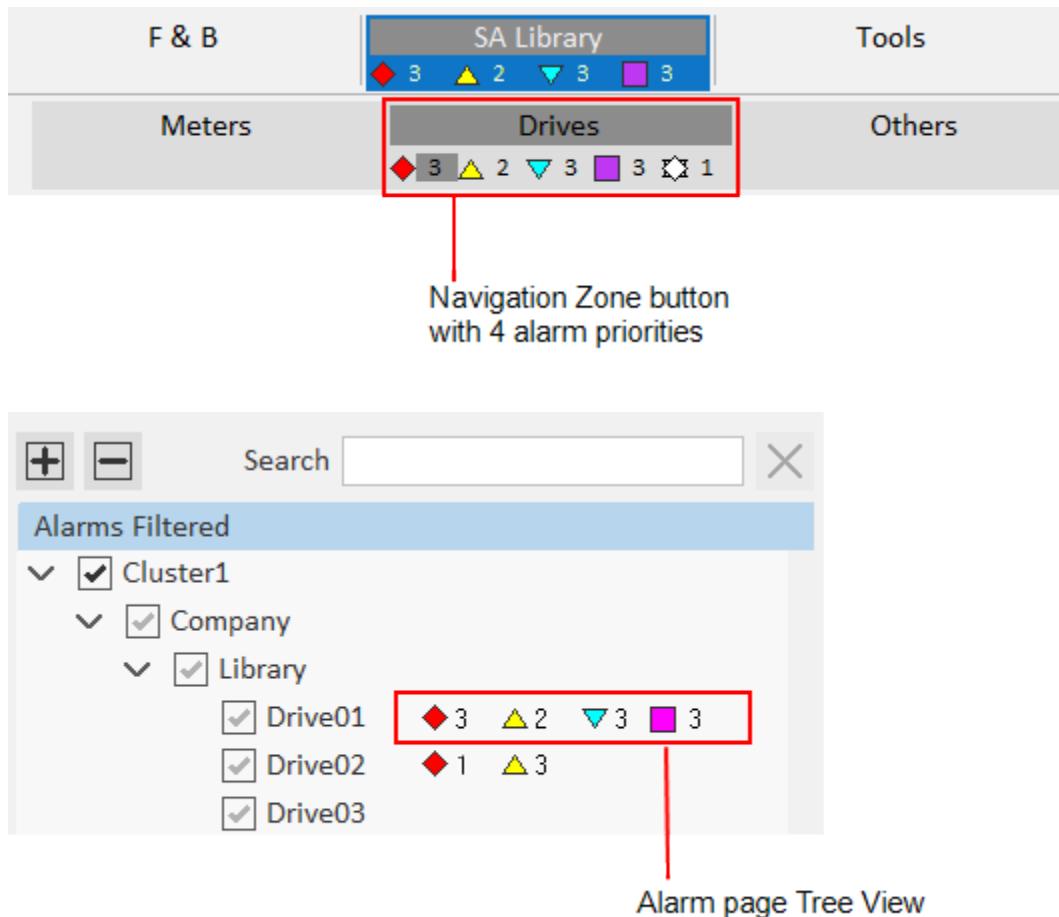
## See Also

[Create Content Pages](#)

[Create a Master Page for a Customized Workspace](#)

## Configure Display Properties for a Fourth Alarm Priority

By default, a Situational Awareness project will only display alarm counts for the top three alarm priorities in the Navigation Zone and Tree Views. However, you can include a fourth highest alarm priority count in these locations if required.



To enable a fourth alarm count, you need to set the parameter [Workspace]NumberOfTopPriorities to 4.

You also need to configure the display properties for the fourth alarm priority.

---

**Note:** The Genie "sa\_p4\_small" is provided in the SA\_Controls project to represent the fourth-highest priority alarms. To associate this Genie with alarm priority 4, use the **Small Genie Name** property. You can use your own Genie if required.

---

### Configure the display properties for the fourth alarm priority

1. In the **Setup** activity, select **Alarming**.
2. On the menu below the Command Bar, select **Alarm Priorities**.

If you have used the starter project, you will see the following default priorities.

Row	Priority	Display Name	Short Name	Comment	Show on Indicator
1	1	Emergency	E	P1 alarm priority	
2	2	High	H	P2 alarm priority	
3	3	Low	L	P3 alarm priority	

3. Add a new row for the fourth alarm priority.
4. Enter the required property values (see the *Alarm Priority Properties* table below).
5. Modify the properties for alarm priority 3 so that it now represents "medium" alarms.
  - Priority = 3
  - Display Name = Medium
  - Short Name = M

**Note:** The Alarm Priority property **Show On Active** must be blank (TRUE) for the alarm counts to show on the Page Navigation zone and Tree View.

6. Save your changes.
7. On the menu below the Command Bar, select **Alarm Categories**
8. Associate an alarm **Category** with the fourth alarm **Priority**.

Row	Category	Priority	Show on Active	Show on Summary	Comment
1	1	1			Emergency alarms
2	2	2			High priority alarms
3	3	3			Medium priority alarms
4	4	4			Low priority alarms

9. Save your change.

## Alarm Priority Properties

Field	Description
Priority	4
Display Name	Low

Field	Description
<b>Short Name</b>	L
<b>Comment</b>	P4 alarm priority
<b>UnAck On Foreground Color</b>	0x00FF00FF
<b>UnAck On Background Color</b>	0x00FFFFFF
<b>Ack On Foreground Color</b>	0x00FFBFFF
<b>Ack On Background Color</b>	0x00FFBFFF
<b>UnAck Off Foreground Color</b>	0x00FFBFFF
<b>UnAck Off Background Color</b>	0x00FFBFFF
<b>Library Name</b>	sa_priorities
<b>Genie Name</b>	sa_p4_normal (This Genie is used for the Alarm Indicator.)
<b>Small Genie Name</b>	sa_p4_small

If you want to add a fourth highest alarm priority count to the Navigation Zone, additional configuration steps are required. For more information, see [Add an Additional Alarm Count to the Navigation Zone](#).

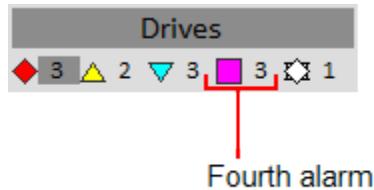
You can also add a fourth-highest alarm priority filter to alarm pages. See [Add a Fourth Alarm Priority Filter to Alarm Pages](#).

---

**Note:** Alarm counts will not work if the home page specified for your navigation menu differs to the home page specified for your equipment hierarchy (as defined in the **Page** field for the root level of your equipment hierarchy). See "Set the home page for a workspace" in the topic [Prepare the Navigation Menu](#).

## Add an Additional Alarm Count to the Navigation Zone

By default, the tabs and buttons in the [Navigation Zone](#) include alarm counts for the top three alarm priorities. If required, you can add an additional alarm count that represents the fourth highest alarm priority.



To add a fourth alarm count to the Navigation Zone's tabs and buttons, you need to perform the following tasks.

### Adjust the parameter [Workspace]NumberOfTopPriorities

Set the parameter [Workspace]NumberOfTopPriorities[Workspace]NumberOfTopPriorities to 4.

## Change the default Navigation Zone page

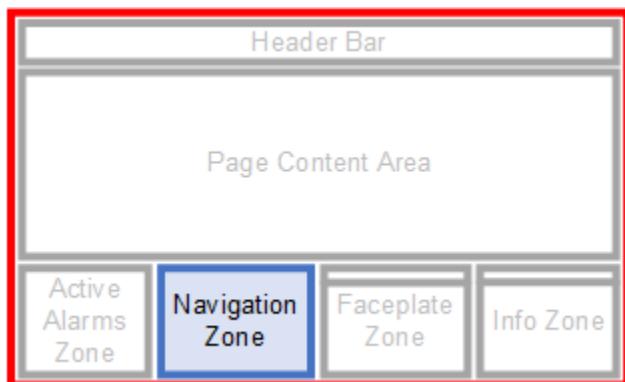
The default page that the Navigation Zone is based on is not able to support the wider tabs and buttons required to display an additional alarm count. To change to the required Navigation Zone page:

1. Go to Graphics Builder.
2. Open the master page for your project. By default, this page will be called "Master\_PageMenu1\_HD1080" or "Master\_PageMenu1\_UHD4K" (depending on the resolution your project uses).
3. Locate the pane that hosts the Navigation Zone.

For **Master\_PageMenu1\_HD1080**, the Navigation Zone pane is located here:



For **Master\_PageMenu1\_UHD4K**, the Navigation Zone pane is located here:



4. Double-click on the pane to open the Pane Properties dialog. If the **Name** field does not display "Navigation", you have double-clicked on the wrong pane.
5. Enter the required page name in the pane's **Default Page** field.
  - For an HD1080 project, enter "PageNav4\_1TabRow\_HD1080".
  - For a UHD4K project, enter "PageNav4\_1TabRow\_UHD4K".
6. Save your changes.

## Configure the display properties for the fourth alarm priority

You should confirm that a fourth alarm priority is appropriately configured in your project. For more information, see [Configure Display Properties for a Fourth Alarm Priority](#).

**Note:** Alarm counts will not work if the home page specified for your navigation menu differs to the home page

---

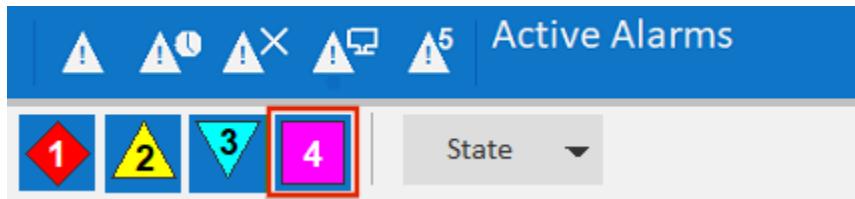
specified for your equipment hierarchy (as defined in the **Page** field for the root level of your equipment hierarchy). See "Set the home page for a workspace" in the topic [Prepare the Navigation Menu](#).

---

## Add a Fourth Alarm Priority Filter to Alarm Pages

You can add a fourth-highest alarm priority filter to the following alarm pages:

- Active Alarms
- Historical Events
- Shelved Alarms.



To achieve this, you need to perform the following steps:

### Adjust the parameter [Workspace]NumberOfTopPriorities

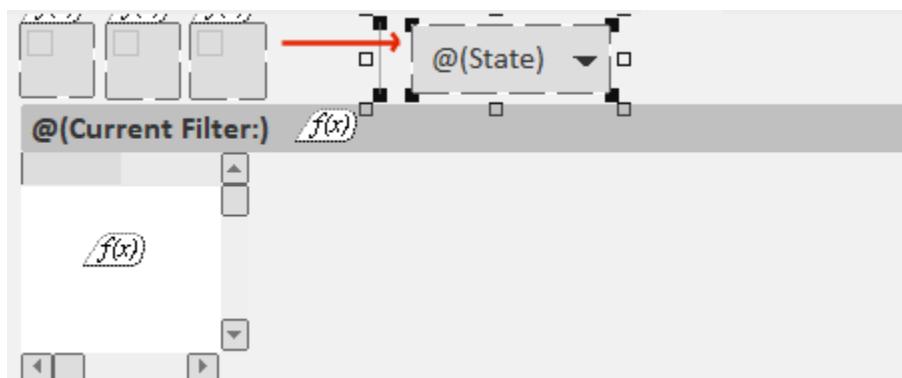
Set the parameter [Workspace]NumberOfTopPriorities to 4.

### Configure the display properties for the fourth alarm priority

You should confirm that a fourth alarm priority is appropriately configured in your project. For more information, see [Configure Display Properties for a Fourth Alarm Priority](#).

### Modify the required alarm page

1. In Graphics Builder, open the alarm page you would like to modify (in the appropriate resolution). The default page names are as follows:
  - Active Alarms - "DefaultAlarm\_HD1080" or "DefaultAlarm\_UHD4K".
  - Historical Events - "DefaultSoE\_HD1080" or "DefaultSoE\_UHD4K".
  - Shelved Alarms - "DefaultShelved\_HD1080" or "DefaultShelved\_UHD4K".
2. On the filter bar, move the separator and "State" button to the right to make room for a fourth priority button.



---

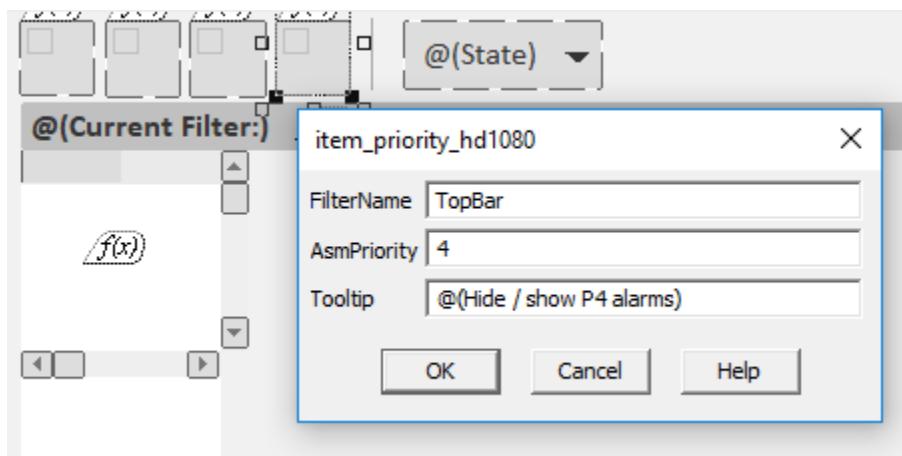
**Note:** If you are modifying the Historical Events page, a fourth alarm button will already exist that is used to filter shelved alarms. In this case, you will need to make room for your fourth alarm priority button between

---

---

the third and fourth buttons.

3. Duplicate one of the existing Item Priority Genies (for priority 1, 2 or 3), and position it in the space you have created.
4. Double-click on the duplicated Genie to display the Genie Parameters dialog.
5. Change the **AsmPriority** field to "4", and adjust the **Tooltip** text.



6. Save your changes.

## See also

[Add an Alarm Filter Button](#)

## Format an Alarm List for a Specific Screen Resolution

You can customize the format used for an alarms list that is displayed on a Situational Awareness alarms page. This allows you to present the list in a way that will suit a particular screen resolution. For example, you can make the columns wider to suit a page that will display on a UHD 4K monitor.

### To customize an alarms list for a specific screen resolution:

1. Use the **[Format]FormatName** parameter to define the alarms list format you want to use for a particular screen resolution.

The following three examples show the default format used for each of the generic alarm pages included in the Situational Awareness Starter Project. You can use one of these as the basis for a new format that is defined to suit a specific screen resolution.

- **[Format]Alarm** = the alarm list format for the generic alarm page.

```
[Format]Alarm =
{PriorityAndState,25}{OnDate,80}{OnTime,90}{Name,250}{State,40}{Cluster,70}{Equipment,220}{I
```

- **[Format]Summary** = the alarm list format for the generic alarm summary page.

```
[Format]Summary =
{PriorityAndState,25}{Name,250}{SumState,40}{OnDate,80}{OnTime,90}{OffDate,80}{OffTime,90}{D
```

- **[Format]SOE** = the alarm list format for the generic sequence of events page.

```
[Format]SOE =
{PriorityAndState,25}{Date,80}{Time,90}{Message,250}{Name,250}{State,40}{Cluster,70}{Equipment,100}
```

For example, you could create [Format]SOE\_UHD4K for a sequence of events page that will display on a 4K monitor. Use the width attribute in each field to specify how wide each column will be (in pixels). For example:

```
[Format]SOE_UHD4K =
{PriorityAndState,50}{Date,160}{Time,180}{Message,500}{Name,500}{State,80}{Cluster,140}{Equipment,100}
```

2. In Graphics Builder, open the alarm page you want to apply the new format to and display the Page Properties.
3. Go to the **Event** tab, and select the **On page entry** event.

By default, Situational Awareness alarm pages will have the following Cicode function configured as the **On page event command**.

```
AlarmPage_Init(AlarmPage_GetDefaultParameters("<res>"), "<res>", <type>)
```

Where:

- <res> = the resolution suffix (for example, HD1080, UHD4K)
- <type> = the display type (for example, 0 = active alarms, 3 = shelved alarms, 15 =SOE).

4. Add the alarm format name defined in step 1 as the fourth argument to the function.

For example, if you configured the format "SOE\_UHD4K", the expression should be as follows:

```
AlarmPage_Init(AlarmPage_GetDefaultParameters("UHD4K"), "UHD4K", 15, "SOE_UHD4K")
```

5. Save your changes, and compile and run the project.

The columns displayed on the alarm page will now be based on the format you have specified.

## See also

[Configure a Project that Supports Multiple Screen Resolutions](#)

[Create Content Pages](#)

## Apply a Customized Font to Selected Alarms

If you have created a project from the Situational Awareness Starter Project, you can customize the font used for selected alarms in an alarms list.

This is achieved by creating the following fonts in your project.

- **SA\_AlmSelRow** - for HD1080 resolution projects.
- **SA\_AlmSelRow4K** - for 4K resolution projects.

### To apply a customized font to selected alarms:

1. In the **Standards** activity, select **FONTS**.
2. On a new row, define the "SA\_AlmSelRow" and/or the "SA\_AlmSelRow4K" font and adjust the properties as required.

---

**Note:** You can refer to the fonts defined in the SA\_Include project to determine the properties used by

---

default in an alarms list.

3. Save your changes and compile the project.

The font you have defined will now be applied to the selected alarm in an alarms list.

## See Also

[Format an Alarm List for a Specific Screen Resolution](#)

## Customize the Logo on the Header Bar

The default header bar for a Situational Awareness project comes with a Plant SCADA logo. This logo can be changed.

---

**Note:** The logo you use needs to be a bitmap.

### To customize the logo:

1. Open Graphics Builder.
2. On the **File** menu, select **New**.

Or:

Select the **New** button.

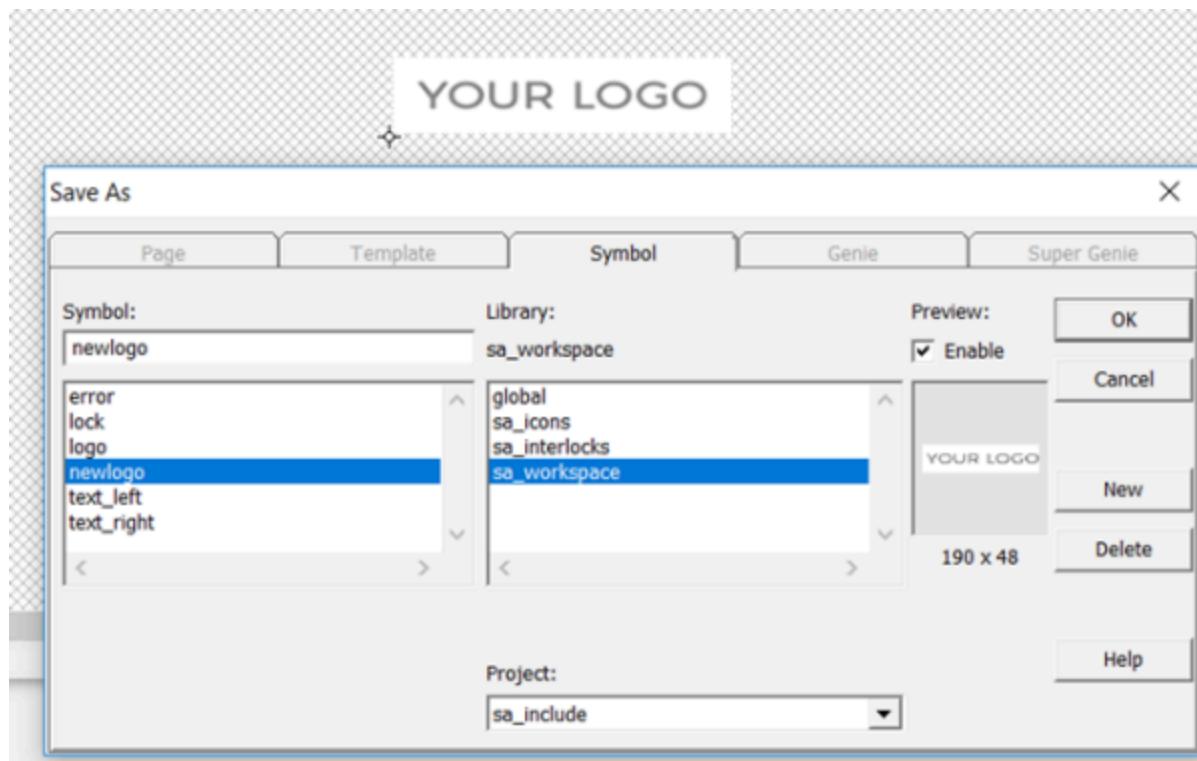


3. The New Dialog box is displayed.
4. On the New dialog box, click **Symbol**.
5. On the **File** menu, select **Import**.
6. From the Import Dialog Box, locate the logo you want to use and click **Open**.
7. The logo is placed onto the symbol page. Position the logo so that the anchor point is at the bottom right corner.

---

**Note:** If you create a new logo using the drawing tools in Graphics Builder, before saving the symbol you need to select all of the objects and select **Convert to Bitmap** from the **Tools** menu. The symbol used by the logo should only contain one bitmap object.

8. Save the symbol, for example, in the "sa\_include" project and the "sa\_workspace" library.



9. Click **OK**.
10. In Plant SCADA Studio, go to the **Project** activity and select **Home**.
11. From the Command Bar, go to the drop-down next to the **Setup Wizard** and select **Setup Editor**.
12. In the **Parameter Details** section of the Setup Editor, enter the following:
  - Section: Page
  - Parameter: Logo
  - Value: <libraryname>.<logoname>  
For example, "sa\_workspace.newlogo".
  - Click **Add**.
13. Save the changes to the INI file.
14. Compile and run the project.



## See Also

[Default Layout](#)

## Add a Tree View to a Page

A tree view is a control that provides a hierarchical view of the information on a page. For example, a tree view is used on the default Active Alarms page in a Situational Awareness Starter Project. You can use it to filter the

displayed list of alarms.

The tree view on the Active Alarms page

Each item in a tree view is a Genie that supports the following functionality:

- Check boxes allow you to select one or more items in the tree.
- Alarms counts indicate the number of associated alarms in the top three alarm priorities, as well as the number of shelved alarms. Alarm and shelve counts will only apply to an equipment data source.

**Note:** You can use the parameter [Workspace]NumberOfTopPriorities to display alarm counts for the top four alarm priorities (instead of the top three). For more setup information, see [Configure Display Properties for a Fourth Alarm Priority](#).

To add a tree view to a page you need to use the Tree View Genie. The Genie's parameters are divided into five main groups:

- Basic Setup parameters — primarily used to specify the data source on which the tree is based.
- Search parameters — identify the object used to input search text at runtime.
- Display parameters — define the size of the space the tree view occupies and the components that will appear at runtime.
- Event parameters — define the functions that are executed when an operator interacts with the Genie at runtime.
- Customization parameters — enable advanced customization of a tree view.

When the Genie is pasted on a page, the Genie Parameters Dialog Box is displayed. See below for a description of each parameter.

## Genie Parameters

### Basic Setup Parameters

Parameter	Description
Datasource	Specifies the menu on which the tree is based. You can enter the name of a menu you have created

Parameter	Description
	<p>using the <b>Menu Configuration</b> view in the <b>Visualization</b> activity (see <a href="#">Prepare the Navigation Menu</a>).</p> <p>Alternatively, you can use the name of a menu that is created at runtime using the <code>MenuGetPageNode</code> Cicode function.</p> <p>You can also enter "<code>__EquipmentModel</code>" to use the equipment hierarchy defined for the current project.</p>
Use Display Names	<p>Determines if display names are used to label equipment in the tree view.</p> <p>A display name is a meaningful name that can be defined for piece of equipment using the <b>Display Name</b> property (see <a href="#">Equipment Properties</a>).</p> <p>If the tree view is displaying the equipment model:</p> <ul style="list-style-type: none"> <li>TRUE = Equipment display names are used</li> <li>FALSE = Equipment names are used.</li> </ul> <p>If this is set to TRUE and no Display Name is configured, the equipment name will be used instead.</p> <p>If the tree view is displaying a generic menu structure:</p> <ul style="list-style-type: none"> <li>TRUE = Menu item comments are used</li> <li>FALSE = Menu item names are used.</li> </ul>

### Search Parameters

Parameter	Description
Search Box Name	<p>This field specifies the name of the <code>ciText.TextBox</code> object that the tree view will use as its input for the search text at runtime. The name to enter is defined in the <b>Text Object Properties</b> dialog on the <b>Access</b> tab.</p> <p>By default, "Search" is used.</p>

### Display Parameters

Parameter	Description
Treeview Height	<p>The height of the area the tree view will occupy (in pixels).</p> <p>If the length of the expanded tree extends beyond the height specified here, a vertical scroll bar will appear.</p>
Treeview Width	<p>The width of the area the tree view will occupy (in pixels).</p> <p>If the width of the expanded tree extends beyond the</p>

Parameter	Description
	left edge of the alarm count list, a horizontal scroll bar will appear.
Row Height	The amount of space that will be available to each row within the tree view.
Display Checkboxes	<p>Determines if a check box is displayed for each item in the tree.</p> <p>TRUE = Check boxes are displayed.</p> <p>FALSE = Check boxes are not displayed.</p>
Auto-check Children	<p>Determines if the check boxes within the lower levels of a branch are automatically checked when a parent item is checked.</p> <p>TRUE = Children are automatically checked.</p> <p>FALSE = Children are not automatically checked.</p>
Display Alarms	<p>Determines if alarm counts for the top three alarm categories are displayed for each item in the tree.</p> <p>TRUE = Alarm counts for the top three alarm categories are displayed.</p> <p>FALSE = Alarm counts for the top three alarm categories are not displayed.</p> <p>If the branch is collapsed, the values will represent the number of active alarms for each category within the current branch and its lower levels.</p> <p>If a node is expanded, the count will only show alarms for that node, as the counts for its children will now be visible on the child nodes.</p> <p><b>Note:</b> You can use the parameter to display alarm counts for the top four alarm priorities (instead of the top three).</p>
Display Shelved	<p>Determines if an alarm count for shelved alarms is displayed for each item in the tree.</p> <p>If the branch is collapsed, the values will represent the number of shelved alarms for each category within the current branch and its lower levels.</p> <p>If a node is expanded, the count will only show shelved alarms for that node, as the counts for its children will now be visible on the child nodes.</p> <p>TRUE = Alarm count for shelved alarms is displayed.</p> <p>FALSE = Alarm count for shelved alarms is not displayed.</p>

**Event Parameters**

<b>Parameter</b>	<b>Description</b>
On Init Complete Function	Represents the name of the function that is invoked when the tree view completes initialization.  <code>OnInitCompleteFunction(INT nTreeviewAN)</code> The argument <i>nTreeviewAN</i> is expected, it is the animation number of the tree view that fires the event.
On Check Function	Represents the name of the function that is invoked upon the checked event.  <code>OnCheckFunction(INT hMenuItem)</code> The argument <i>hMenuItem</i> is expected. It is the handle to the menu item that is represented by the tree node on which the check box has just been clicked.
On Left Click Function	Represents the name of the function that is invoked upon the left click event.  <code>OnLeftClickFunction(INT hMenuItem)</code> The argument <i>hMenuItem</i> is expected. It is the handle to the menu item that is represented by the tree node that has just been left-clicked.
On Right Click Function	Represents the name of the function that is invoked upon the right click event.  <code>OnRightClickFunction(INT hMenuItem)</code> The argument <i>hMenuItem</i> is expected. It is the handle to the menu item that is represented by the tree node that has just been right-clicked.

**Customization Parameters**

<b>Parameter</b>	<b>Description</b>
View Genie	Refers to the Genie that is called to display each item in the tree. By default, the Tree View Item Genie in the "sa_controls" library is used ( <code>sa_controls.treeviewitem</code> ).  You can use this parameter to substitute the default Genie with your own. This will allow you to customize the appearance of the tree and the features it supports.
View Init Function	The Cicode function that is called to initialize the <b>View Genie</b> (see above) for each row.  By default, a system function called

Parameter	Description
	" <code>_Treeview_InitViewGenie_&lt;resolution&gt;</code> " is used (for example, " <code>_Treeview_InitViewGenie_HD1080</code> ").
Fill Function	<p>This function is called on each page scan to manage data updates in the tree view when the tree view state changes (for example, when scrolling, expanding, or collapsing occurs).</p> <p>By default, a system function called "<code>_Treeview_Update</code>" is used.</p>
View Model Data Size	<p>Specifies the number of custom fields per tree view item to be allocated for user data. The data in these fields will be updated by the function specified in the parameter <b>Datasource Item Retrieve Function</b> (see below).</p> <p>This parameter is set to zero (0) by default.</p>
View Model Init Function	<p>Custom function to configure the additional columns specified by the parameter 'View Model Data Size'. This function is called once when the tree view is initialized.</p>
Datasource Item Retrieve Function	<p>A custom function that loads additional user data for the tree view. This function is called whenever the tree view data is refreshed (for example, when scrolling, expanding, or collapsing occurs). The number of custom fields to be updated by this function is defined by the parameter <b>View Model Data Size</b> (see above).</p>
Is Source Ready Function	<p>Enter the name a function that is used to determine if the data source is ready.</p> <p>The tree view initialization code will run the function (in addition to the checks for navigation initialized and workspace system initialized) before allowing the tree view to be displayed.</p> <p>If this function does not return TRUE within five seconds of displaying the page, the tree view initialization will time out. This function needs to return an INT (TRUE or FALSE) and takes a single argument, INT nTreeViewAN.</p>
Datasource Watcher Function	<p>Enter a Cicode function that will be used to monitor the tree view's associated data source for any updates. The function will be called when the Tree View Genie is initialized, and needs to be responsible</p>

Parameter	Description
	<p>for triggering a reload if required.</p> <p>The function will run continuously, so it should not return any values. It also needs to accept the arguments <i>sDatasource</i> and <i>hTreeviewAN</i>.</p> <p>If you are using the Tree View Genie to display the entire equipment model (for example, if the <b>Datasource</b> parameter is set to "<code>__EquipmentModel</code>"), you can enter the following function name to trigger a reload of the tree whenever the equipment model changes:</p> <p><code>_Treeview_MonitorEquipmentModel</code></p>
Can Check Items Function	<p>A custom function that is used to determine if tree view item checking is allowed.</p> <p>By default, toggling the checked state of a tree view item is allowed at runtime. This function provides a way to disable the check boxes on all items if required.</p> <p>For example, you may want to disable checking if the data that the tree view will interact with is not ready, or if the current user does not have sufficient privileges.</p> <p>The function needs to return a value of TRUE or FALSE.</p>

#### AN Parameter

Parameter	Description
AN	A unique ID for the Genie object.

**Note:** You can use the **View Genie** parameter to substitute the default Genie with your own. This will allow you to customize the appearance of the tree and the features it supports.

#### See Also

[Key Components of a Situational Awareness Project](#)

[Configure Your Project](#)

#### Add an Alarms List to a Page

You can add an alarms list to a page using the following Genies:

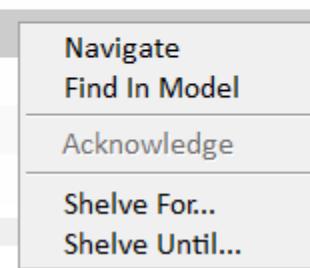
- Alarm List Genie — creates an alarm list with a header row and scroll bars.
- Alarm List Vertical Genie — creates an alarms list with no header row and just a vertical scroll bar.

These Genies use the `AlarmListCreate` Cicode function to create a basic alarms list.

...	Date	Time	Name
▲	26/03/2018	10:35:30 AM	Drive1_OP_TRK_H_P2
▼	26/03/2018	10:35:27 AM	Drive1_V_Feedback_H_P3
◆	26/03/2018	10:34:50 AM	Drive1_V_OP_HH_P1
◆	15/03/2018	10:13:03 AM	Drive1_Running_False_P1

To apply formatting to the list rows and enable on-click commands, you can also add the List Row Genie.

...	Date	Time	Name
▲	26/03/2018	02:55:15 PM	Drive1_OP_TRK_H_P2
▼	26/03/2018	02:55:15 PM	Drive1_V_Feedback_H_P3
◆	26/03/2018	02:55:15 PM	Drive1_V_OP_HH_P1
◆	20/03/2018	10:10:29 AM	Drive1_Running_False_P1



- Navigate
- Find In Model
- 
- Acknowledge
- 
- Shelve For...
- Shelve Until...

## Add an alarm list Genie

To add an alarm list Genie ("alarmlist" or "alarmlist\_v") to a page:

1. Open a page in Graphics Builder.
2. Click the **Paste Genie** button in the objects toolbox.



Or, select **Paste Genie** from the **Edit** menu.

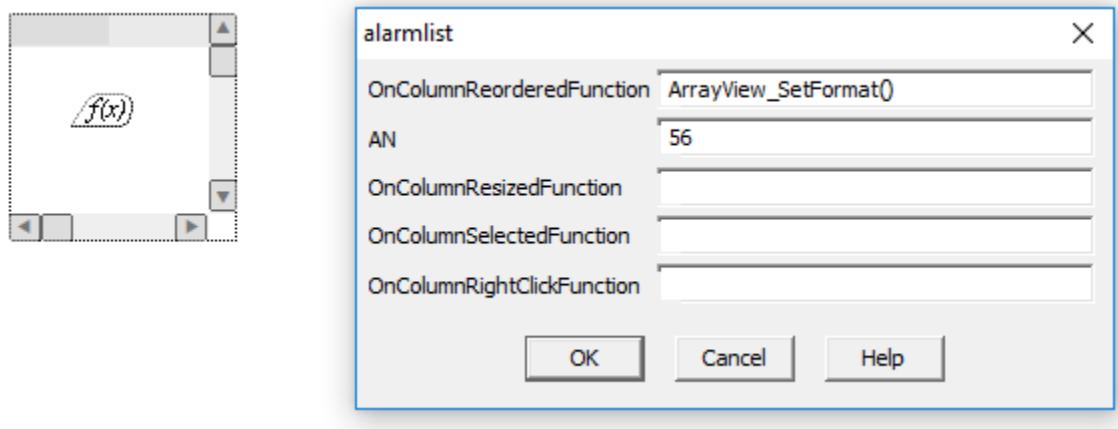
3. In the Paste Genie dialog, select the **Library** "sa\_controls".
4. Select one of the following from the **Genie** list and click **OK**.
  - alarmlist = Alarm List Genie
  - alarmlist\_v = Alarm List - Vertical Genie

The Genie is pasted onto the graphics page. A dialog box will open, prompting you to configure the parameters for the Genie.

The **Alarm List Vertical Genie** dialog box will show the **AN** parameter. The number that is displayed is the **AN** value that was automatically applied to the Genie when it was pasted on the page.



The **Alarm List Genie** dialog box will include some additional parameters that enable runtime interaction with the header row. By default, the **OnColumnReorderedFunction** will display "ArrayView\_SetFormat()". This is a system function that allows an operator to rearrange the header row columns at runtime. The other parameters can be left blank.



5. Click **OK** to close the parameters dialog box.

To load alarm data into the list at runtime, you need to call the [AlarmListCreate](#) Cicode function as an On Page Entry command in the Page Properties dialog.

6. In Graphics Builder, right-click on the page and select **Page Properties** from the menu that appears.

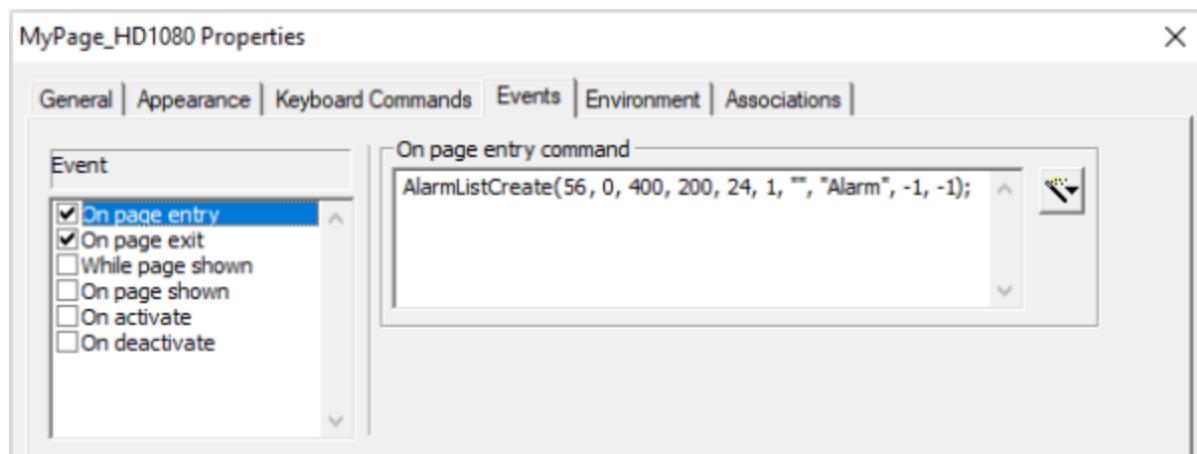
Or:

From the **File** menu, select **Properties**.

The Page Properties dialog box will appear.

7. On the **Events** tab, select the **On page entry** event.

8. In the **On page entry command** field, configure the AlarmListCreate Cicode function as required.



The value specified for the *nAN* argument needs to match the AN number applied to the alarm list Genie.

9. On the **Events** tab, select the **On page exit** event.
10. In the **On page exit command** field, configure the [AlarmListDestroy](#) Cicode function. For example,  
`AlarmListDestroy(56);`
11. Click **OK**, and **Save** your page.

You can now run your project and view the alarms list.

## Add the List Row Genie

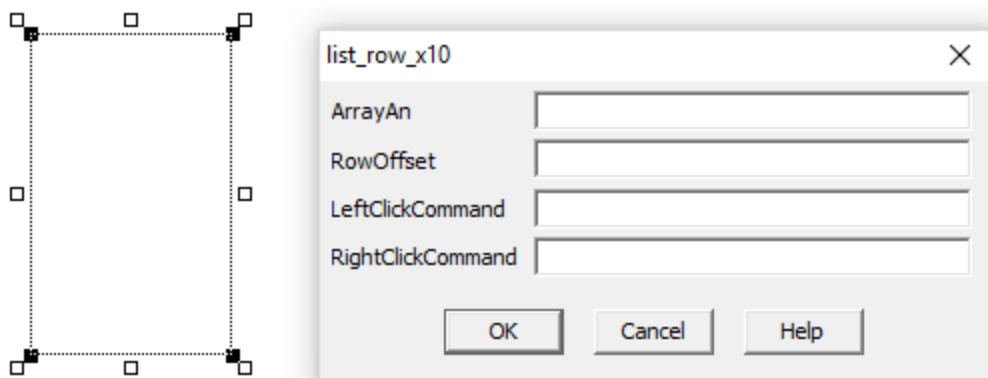
1. In Graphics Builder, open the page that includes the list to which you want to add a List Row Genie.
2. Click the **Paste Genie** button in the objects toolbox.



Or, select **Paste Genie** from the **Edit** menu.

3. In the Paste Genie dialog, select the **Library** "sa\_controls".
4. Select one of the following from the **Genie** list, based on the approximate size of the list you require.
  - list\_row\_x5 – use this to create a list with up to 5 rows.
  - list\_row\_x10 – use this to create a list with up to 10 rows.
  - list\_row\_x40 – use this to create a list with up to 40 rows.
  - list\_row\_x80 – use this to create a list with up to 80 rows.

The Genie is pasted onto the graphics page. A dialog box will open, prompting you to configure the parameters for the Genie.



**Note:** There is no need to adjust the size of the List Row Genie on a graphics page to fill an area. The required space will be determined at runtime by the size of the list with which it is associated.

5. In the **ArrayAN** field, enter the AN value for the associated list.

You can use the Cicode function [DspGetAnFromName](#) to retrieve the AN. For example:

```
DspGetAnFromName("AlarmList").
```

6. To enable the default set of right-click commands that are available on the default alarm page, go to the **RightClickCommand** field and enter the following:

```
AlarmPage_AlarmListOnRightClick()
```

If you want to use a custom set of commands, you can add your own Cicode.

7. Click **OK**, and **Save** your page.

You can now run your project and view the alarms list.

## See Also

[Create Priority and State Symbols for an Alarms List](#)

[Key Components of a Situational Awareness Project](#)

[Configure Your Project](#)

## Add a Generic List to a Page

You can add a generic list to a page using the following Genies:

- Generic List Genie — creates a list with a header row and scroll bars.
- Generic List Vertical Genie — creates a list with no header row and just a vertical scroll bar.

To apply formatting to the list rows and enable on-click commands, you can also add the .

## Add a list Genie

To add a list Genie ("genericlist" or "genericlist\_v") to a page:

1. Open a page in Graphics Builder.

2. Click the **Paste Genie** button in the objects toolbox.



Or, select **Paste Genie** from the **Edit** menu.

3. In the Paste Genie dialog, select the **Library** "sa\_controls".
4. Select one of the following from the **Genie** list and click **OK**.

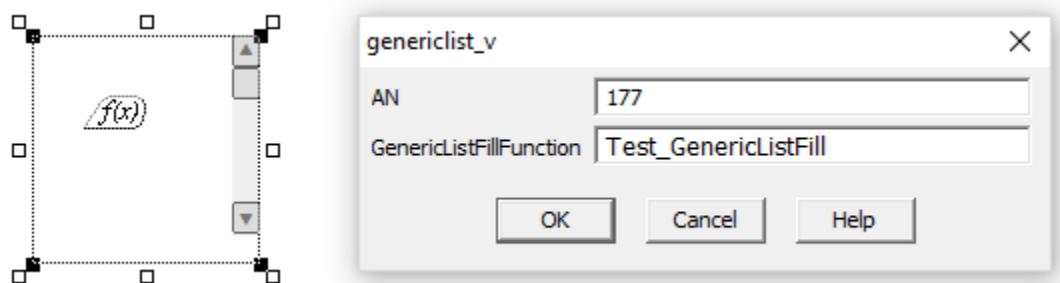
- genericlist = Generic List Genie
- genericlist\_v = Generic List Vertical Genie

The Genie is pasted onto the graphics page. A dialog box will open, prompting you to configure the parameters for the Genie.

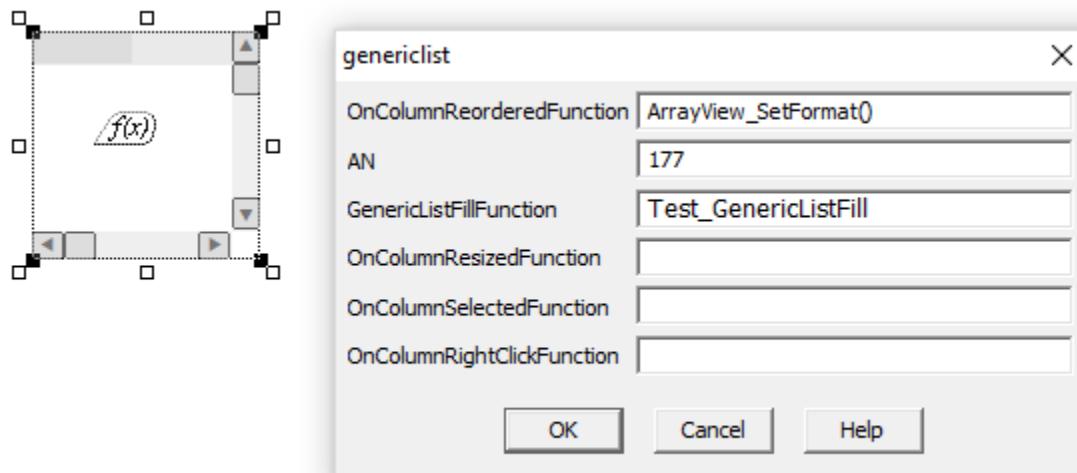
The **Generic List Vertical Genie** dialog box will show the following parameters:

- **AN** — the number that is displayed is the ANvalue that was automatically applied to the Genie when it was pasted on the page.
- **GenericListFillFunction** — this field specifies the user Cicode function to be called at runtime to populate the list. The function needs to retrieve the data to be displayed.

In the example below, "Test\_GenericListFill" has been entered. To view the custom Cicode that this function calls, see the Cicode examples below.



The **Generic List Genie** dialog box will include some additional parameters that enable runtime interaction with the header row. By default, the **OnColumnReorderedFunction** will display "ArrayView\_SetFormat()". This is a system function that allows an operator to rearrange the header row columns at runtime.



5. Click **OK** to close the parameters dialog box.

To load data into the list at runtime, you need to call the required Cicode function as an On Page Entry command in the Page Properties dialog.

6. In Graphics Builder, right-click on the page and select **Page Properties** from the menu that appears.

Or:

From the **File** menu, select **Properties**.

The Page Properties dialog box will appear.

7. On the **Events** tab, select the **On page entry** event.

8. In the **On page entry command** field, enter the required expression.

For this example, the "Test\_GenericListCreate" function is entered. To view the custom Cicode that this function calls, see the Cicode examples below.

9. On the **Events** tab, select the **On page exit** event.

10. In the **On page exit command** field, enter an expression to destroy the list that has been created. For an example, search locate "Test\_GenericListDestroy" in the example Cicode below.

11. Click **OK**, and **Save** your page.

## Add the List Row Genie

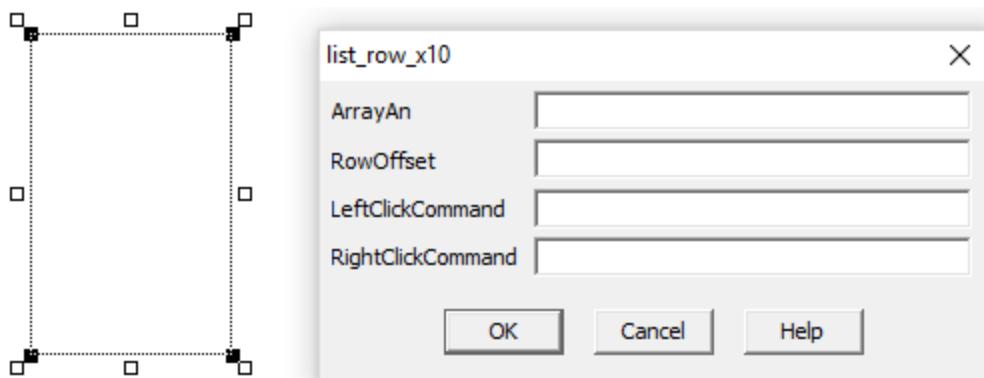
1. In Graphics Builder, open the page that includes the list to which you want to add a List Row Genie.
2. Click the **Paste Genie** button in the objects toolbox.



Or, select **Paste Genie** from the **Edit** menu.

3. In the Paste Genie dialog, select the **Library** "sa\_controls".
4. Select one of the following from the **Genie** list, based on the approximate size of the list you require.
  - list\_row\_x5 – use this to create a list with up to 5 rows.
  - list\_row\_x10 – use this to create a list with up to 10 rows.
  - list\_row\_x40 – use this to create a list with up to 40 rows.
  - list\_row\_x80 – use this to create a list with up to 80 rows.

The Genie is pasted onto the graphics page. A dialog box will open, prompting you to configure the parameters for the Genie.



**Note:** There is no need to adjust the size of the List Row Genie on a graphics page to fill an area. The required space will be determined at runtime by the size of the list with which it is associated.

5. In the **ArrayAN** field, enter the AN value for the associated list.

You can use the Cicode function [DspGetAnFromName](#) to retrieve the AN. For example:

`DspGetAnFromName("GenericList").`

6. To enable on-click commands, go to the **LeftClickCommand** and/or **RightClickCommand** fields and enter the required Cicode.

See the Cicode example [Test\\_GenericListRightClick](#) below.

7. Click **OK**, and **Save** your page.

You can now run your project and view the alarms list.

## Cicode Examples

The following Cicode provides an example of how to create the following generic list using Plant SCADA's Array Functions. The list displays random numbers with column and row identifiers.

Column1	Column2	Column3	Column4
C1R1>95	C2R1>80	C3R1>94	C4R1>65
C1R2>40	C2R2>35	C3R2>51	C4R2>58
C1R3>9	C2R3>81	C3R3>71	C4R3>54
C1R4>79	C2R4>80	C3R4>44	C4R4>39
C1R5>20	C2R5>86	C3R5>14	C4R5>58
C1R6>46	C2R6>44	C3R6>13	C4R6>86
C1R7>41	C2R7>28	C3R7>51	C4R7>32
C1R8>5	C2R8>94	C3R8>20	C4R8>72
C1R9>58	C2R9>63	C3R9>22	C4R9>75
C1R10>8	C2R10>71	C3R10>13	C4R10>24

### [Test\\_GenericListCreate](#)

```
FUNCTION Test_GenericListCreate(INT hAn)
    INT bDrawHeader = 1; // Draw the header row
    INT hFontRow = DspFontHnd("SA_AlmRow");
    INT hFontHeader = DspFontHnd("SA_AlmRow");
    INT nColumn = 1;
    INT nColumns = 5;
    INT nRows = 10;
    INT nRowHeight = 20; // pixels
    INT nHeight = (nRows + 1) * nRowheight;
    INT nWidth = 280; // pixels
    INT nWidthTotal = 0; // Pixels, sum of column widths
    INT nDataColumns = 10; // Maximum possible number of data columns
    INT nDataRows = 40; // Maximum possible number of data rows
    INT hArray = ArrayCreateByAn(hAn , nColumns + 1, nRows + 1, ARRAY_ZINDEX_MAX);
    // +1 for Column0 (hidden), +1 for Row0 (hidden)
    STRING sMapName = ArrayGetMapNameByAn(hAn);

    .
    // Set up array attributes
    .
    ArraySetIntByAn(hAn, nWidth, ARRAY_INFORMATION_COLUMN, ARRAY_INFORMATION_ROW,
    ARRAY_ZINDEX_ARRAY_XMAX);
    ArraySetIntByAn(hAn, nHeight, ARRAY_INFORMATION_COLUMN, ARRAY_INFORMATION_ROW,
    ARRAY_ZINDEX_ARRAY_YMAX);
    ArraySetIntByAn(hAn, nRowHeight, ARRAY_INFORMATION_COLUMN, ARRAY_INFORMATION_ROW,
    ARRAY_ZINDEX_ARRAY_ROWHEIGHT);
    ArraySetIntByAn(hAn, bDrawHeader, ARRAY_INFORMATION_COLUMN, ARRAY_INFORMATION_ROW,
    ARRAY_ZINDEX_ARRAY_DRAWHEADER);
    ArraySetIntByAn(hAn, hFontRow, ARRAY_INFORMATION_COLUMN, ARRAY_INFORMATION_ROW,
    ARRAY_ZINDEX_ARRAY_FONTROW);
    ArraySetIntByAn(hAn, hFontHeader, ARRAY_INFORMATION_COLUMN, ARRAY_INFORMATION_ROW,
    ARRAY_ZINDEX_ARRAY_FONTHDR);
    // Set up column headers
    FOR nColumn = 1 TO nColumns DO
        ArraySetStringByAn(hAn, "Column" + nColumn:#, nColumn, ARRAY_INFORMATION_ROW,
        ARRAY_ZINDEX_COLUMN_VALUE);
        ArraySetIntByAn(hAn, 70, nColumn, ARRAY_INFORMATION_ROW, ARRAY_ZINDEX_COLUMN_WIDTH);
        ArraySetIntByAn(hAn, ARRAY_COLUMN_TYPE_STRING, nColumn, ARRAY_INFORMATION_ROW,
        ARRAY_ZINDEX_COLUMN_TYPE);
        nWidthTotal = nWidthTotal + 70;
    END
    .
    ArraySetIntByAn(hAn, nWidthTotal, ARRAY_INFORMATION_COLUMN, ARRAY_INFORMATION_ROW,
    ARRAY_ZINDEX_ARRAY_XTOTAL);
    ArraySetIntByAn(hAn, 0, ARRAY_INFORMATION_COLUMN, ARRAY_INFORMATION_ROW,
    ARRAY_ZINDEX_ARRAY_YTOTAL);
    INT hdataArray = ArrayCreate("Test_Data", nDataColumns, nDataRows, 1);
    // Data Array; 10 fields x 40 rows
    MapValueSet(sMapName, "dataArray", hdataArray);
    MapValueSet(sMapName, "DataColumns", nDataColumns);
    MapValueSet(sMapName, "DataRows", nDataRows);
    MapValueSet(sMapName, "Offset", -1);
    .
    TaskNew("Test_Task_GetData", hAn:#, 1 + 8);
END
FUNCTION Test_GenericListDestroy(INT hAn
    INT hdataArray = -1;
```

```

STRING sMapName = "";
IF (ArrayExistsByAn(hAn)) THEN
    sMapName = ArrayGetMapNameByAn(hAn);
    hdataArray = MapValueGet(sMapName, "dataArray");
    ArrayDestroy(hdataArray);
    GenericListDestroy(hAn);
END
END
.

FUNCTION Test_Task_GetData(STRING sArg);
    INT hAn = StrToInt(sArg);
    INT nDataColumn = 0;
    INT nDataRow = 0;
    INT nDataColumns = 0;
    INT nDataRows = 0;
    INT hdataArray = -1;
    STRING sMapName = "";
    STRING nRandom = 0;
    STRING sValue = "";
WHILE (TRUE) DO
    IF (ArrayExistsByAn(hAn)) THEN
        sMapName = ArrayGetMapNameByAn(hAn);
        hdataArray = MapValueGet(sMapName, "dataArray");
        nDataColumns = MapValueGet(sMapName, "DataColumns");
        nDataRows = MapValueGet(sMapName, "DataRows");
        ArraySetIntByAn(hAn, nDataRows, ARRAY_INFORMATION_COLUMN, ARRAY_INFORMATION_ROW,
        ARRAY_ZINDEX_ARRAY_YTOTAL);
        FOR nDataRow = 1 TO nDataRows DO
            FOR nDataColumn = 1 TO nDataColumns DO
                nRandom = Rand(99);
                sValue = "C" + nDataColumn:# + "R" + nDataRow:# + ">" + nRandom:#;
                ArraySetString(hdataArray, sValue, nDataColumn - 1, nDataRow - 1, 0);
            END
        END
    END
    SleepMS(30000);
END
END

```

**Test\_GenericListFill**

```

FUNCTION Test_GenericListFill(INT hAn)
    INT nRow = 0;
    INT nRows = 10;
    INT nColumn = 0;
    INT nColumns = 5;
    INT nOffset = 0;
    INT nOffset_Previous = 0;
    INT nDataColumns = 0;
    INT nDataRows = 0;
    INT hdataArray = -1;
    STRING sMapName = "";
    STRING sValue = "";
    IF (ArrayExistsByAn(hAn)) THEN
        _ArrayView_Header_Run(hAn);
        _ArrayView_VScroll_Run(hAn);
        _ArrayView_HScroll_Run(hAn);

```

```
sMapName = ArrayGetMapNameByAn(hAn);
hdataArray = MapValueGet(sMapName, "dataArray");
nOffset_Previous = MapValueGet(sMapName, "Offset");
nOffset = ArrayGetIntByAn(hAn, ARRAY_INFORMATION_COLUMN, ARRAY_INFORMATION_ROW,
ARRAY_ZINDEX_ARRAY_YOFF);
hdataArray = MapValueGet(sMapName, "dataArray");
IF ((ArrayIsDirty(hdataArray)) OR (nOffset <> nOffset_Previous)) THEN
    nDataColumns = MapValueGet(sMapName, "DataColumns");
    nDataRows = MapValueGet(sMapName, "DataRows");

    FOR nRow = 1 TO nRows DO
        FOR nColumn = 1 TO nColumns DO
            sValue = ArrayGetString(hdataArray, nColumn - 1, (nOffset + nRow) - 1, 0);
            ArraySetStringByAn(hAn, sValue, nColumn, nRow, ARRAY_ZINDEX_CELL_VALUE);
        END
    END

    MapValueSet(sMapName, "Offset", nOffset);
    ArraySetIsDirty(hdataArray, FALSE);
END
END
FUNCTION Test_GenericListRightClick(INT hAn)
    Message("TestGenericListRightClick", "Add context menu code here", 0);
END
```

### Test\_GenericListRightClick

```
FUNCTION Test_GenericListRightClick(INT hAn)
    Message("TestGenericListRightClick", "Add context menu code here", 0);
END
```

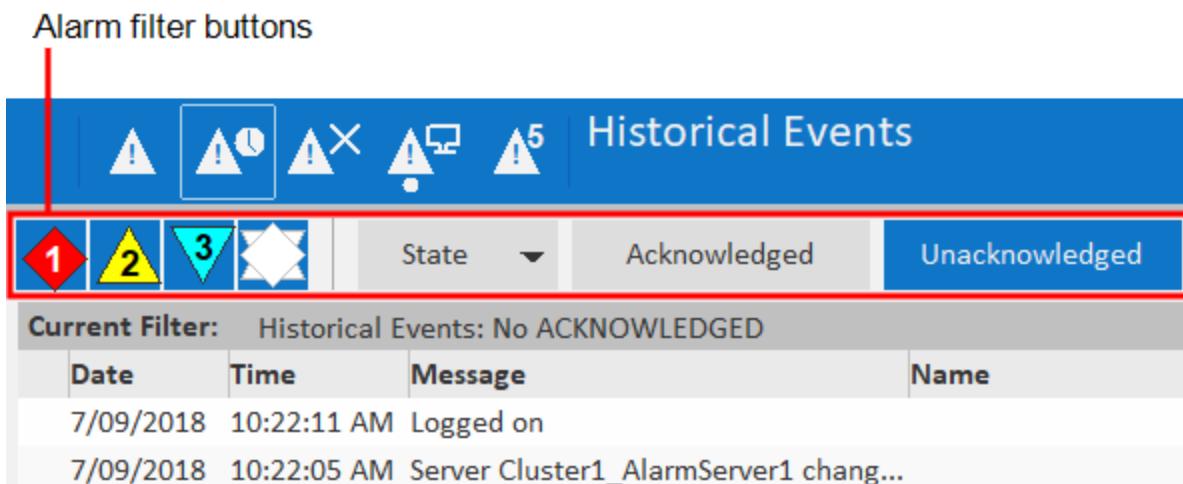
## See Also

[Key Components of a Situational Awareness Project](#)

[Configure Your Project](#)

## Add an Alarm Filter Button

You can add your own alarm filter buttons to one of the [Default Alarm Pages](#) in a Situational Awareness project.



The following Alarm Filter Genies allow you to add buttons that provide basic filter functionality to an alarm page:

- Item Acknowledged Genie (HD1080)
- Item Acknowledged Genie (UHD4K)
- Item Priority Genie (HD1080)
- Item Priority Genie (UHD4K)
- Item Shelved Genie (HD1080)
- Item Shelved Genie (UHD4K)
- Item State Genie (HD1080)
- Item State Genie (UHD4K)
- Item Unacknowledged Genie (HD1080)
- Item Unacknowledged Genie (UHD4K).

You can also use the Item Base Genie to create a button that implements a custom filter.

---

**Note:** The default hardware alarms page does not support filter buttons.

---

## Add an alarm filter button using a Genie

To add an alarm filter button to a default alarm page:

1. In Graphics Builder, open the alarm page to which you want to add a filter button.

The default alarm pages have the following names:

- DefaultAlarm\_HD1080 (Active Alarms Page HD1080)
- DefaultAlarm\_UHD4K (Active Alarms Page UHD4K)
- DefaultSOE\_HD1080 (Historical Events Page HD1080)
- DefaultSOE\_UHD4K (Historical Events Page UHD4K)
- DefaultShelved\_HD1080 (Shelved Alarms Page HD1080)
- DefaultShelved\_UHD4K (Shelved Alarms Page UHD4K)

2. Locate the filters section at the top of the page. The filters section will contain any Alarm Filter Genies that already appear on the page.
3. Click the **Paste Genie** button in the objects toolbox, or select **Paste Genie** from the **Edit** menu.
4. In the Paste Genie dialog, select the "sa\_filter" library.
5. Select the required Alarm Filter Genie from those listed above. The name of the Genie will reflect the type of filtering it supports.  
A dialog will open prompting you to configure the parameters for the Genie. For a description of each parameters, click on appropriate link in the list of Alarm Filter Genies above.
6. Enter the required values, and click **OK** to close the dialog.

## Add a custom alarm filter button using the Item Base Genie

To add a custom alarm filter button to a default alarm page:

1. In Graphics Builder, open the alarm page to which you want to add a custom filter button.  
The default alarm pages have the following names:
  - DefaultAlarm\_HD1080 (Active Alarms Page HD1080)
  - DefaultAlarm\_UHD4K (Active Alarms Page UHD4K)
  - DefaultSOE\_HD1080 (Historical Events Page HD1080)
  - DefaultSOE\_UHD4K (Historical Events Page UHD4K)
  - DefaultShelved\_HD1080 (Shelved Alarms Page HD1080)
  - DefaultShelved\_UHD4K (Shelved Alarms Page UHD4K)
2. Locate the filters section at the top of the page. The filters section will contain any Alarm Filter Genies that already appear on the page.
3. Click the **Paste Genie** button in the objects toolbox, or select **Paste Genie** from the **Edit** menu.
4. In the Paste Genie dialog, select the "sa\_filter" library.
5. Select the "Item Base Genie" and click **OK**.  
A dialog will open prompting you to configure the parameters for the Genie. For a description of each parameters, see Item Base Genie.
6. Enter the required values, and click **OK** to close the dialog.

---

**Note:** The Default Alarm Pages incorporate an alarm filter named "TopBar". To implement the expected filtering functionality, you can enter "TopBar" as the **Filter Name** in the Genie parameters. If you need to create your own custom filter, use the Item Base Genie. You can then use the **FilterExpr** parameter to enter a Cicode expression that returns the name of a custom filter.

## See Also

[Add a Fourth Alarm Priority Filter to Alarm Pages](#)

## Create a Set of Tabs for a Pane

You can use the Tab Bar Genie to create a set of tabs for a pane, just like those displayed in the Information Zone

on the Situational Awareness dashboard.

The screenshot shows the AVEVA Plant SCADA Graphics interface. At the top is a tab bar with three tabs: 'ALARM' (selected and highlighted in blue), 'TREND', and 'INTERLOCKS'. Below the tabs is a list of alarms:

Time	Category	Description
03:49:26 PM	V_OP	Drive1_V_OP_HH_P1
10:10:46 AM	S_Running	Drive1_Running_False_P1
03:49:26 PM	OP_TRK	Drive1_OP_TRK_H_P2
03:49:26 PM	V_Feedback	Drive1_V_Feedback_H_P3

Below the alarm list is a detailed view of the first alarm:

Cause	Drive is overheating.
Response	Check the thermostat.
Response Time	00:05:00
Consequence	Drive may fail.

If the available space cannot accommodate the configured tabs, you can access those that are not displayed via the drop-down menu to the right of the tabs. You can also enable "pinning", which allows an operator to specify that a particular tab remains displayed.

To configure a set of tabs, you need to perform the tasks described below.

## Create a menu

The Tab Bar Genie calls a menu defined in a project's Menu Configuration database. You will need to create an appropriate menu in Plant SCADA Studio's **Visualization** activity (see [Menu Configuration](#)).

In the case of a menu created for a set of tabs, use the following fields:

- The **Page** property is used to give the tabbed menu a name.
- The **Level 1** entries define the tabs that appear.
- The **Target Page** property determines what will be displayed in the content pane when a tab is selected.

You can also customize the tabs using the following properties:

- The **Symbol** property allows you to display a lock icon on the tab. Enter any value if you want the lock to display. You can use the **Custom 2** property (see below) to hide the icon if required.
- The **Checked** property allows you to specify if a tab is pinned by default. Leaving the field blank will mean the tab is not pinned by default. Any other value will mean the tab is pinned by default. This setting will only work if pinning is enabled for the tab bar. This is determined by the **Can Tabs Be Pinned?** parameter in the Tab Bar Genie (see [Configure the tab bar genie](#) below).
- The **Menu Command** property can be used to specify a Cicode expression that is executed when the associated tab is selected.
- The **Custom 1** property determines if the **Close** button will appear on the tab. If you leave this field blank, the Close button will appear. Any other value will hide the Close button.

- The **Custom 2** property determines if the icon identified in the **Symbol** field will appear on the tab. If you leave this field blank, the icon will appear. Any other value will hide the icon.
- The **Custom 3** property determines if a label will appear on the tab. If you leave this field blank, the label will appear. Any other value will hide the label.

**Note:** The **Custom 4 – Custom 8** fields are reserved for system use.

## Create the host panes

Two adjacent panes are required to support a set of tabs:

- Header pane - hosts a page with the Tab Bar Genie on it
- Content pane - displays the target page that is associated with a selected tab.

The two are linked with each other via the Pane Properties for the content pane. The following Tabbed Pane Properties are used:

- **Header Pane Name** — identifies the associated header pane
- **Tab Control Name** — identifies the Tab Bar Genie in the header pane that determines what the content pane will display.

## Create the required pages

You need to create the following pages:

- Header page — you need an appropriately sized page that will occupy the header pane and host the Tab Bar Genie.
- Content pages — you need appropriately sized pages for each item specified in the menu you created. The page names need to match those defined in the **Target Page** field for each menu entry.

## Configure the Tab Bar Genie

Lastly, you need to configure the Tab Bar Genie parameters. To display the Genie Parameters dialog box, double-click on the Genie, or select **Genie Parameters** from the right-click menu. Make the required adjustments to the following fields.

Parameter	Description
Default Tabs Menu Name	Enter the name of the menu you configured to define the tabs that will display.
OnInit Function Name	If required, specify a Cicode function that is executed when the tab bar initialization has completed.
Can Tabs be Pinned?	Specify if an operator can pin a tab so that it remains displayed. TRUE = Pinning is enabled FALSE = Pinning is not enabled.

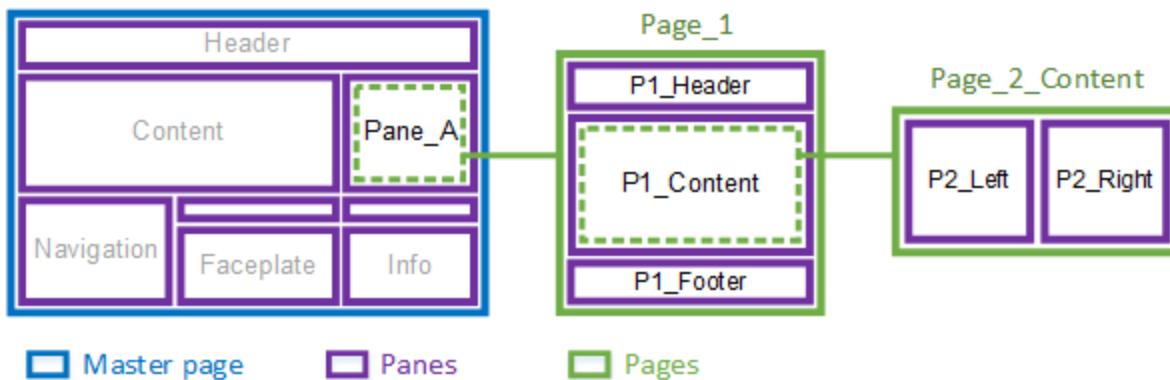
Parameter	Description
Max Open Items	If required, sets a limit on the number of menu items that are included on the tab bar at runtime.  The menu items are selected according to the value set for <b>Order</b> property. If the Order property is not configured, items are selected according to the order in which they were added to the Menu Configuration database.
Tab Bar Width	Specify the width of the tab bar in pixels.
Max Tab Width	Specify the maximum width for each tab in pixels.
Use for 4K Page?	Specify if the tab bar will appear on a UHD4K master page.  TRUE = The tab bar will be used in a UHD4K project. FALSE = The tab bar will not be used in a UHD4K project.
Allow Duplicate Content?	Determines if multiple tabs can display the same content.  TRUE = Multiple tabs can display the same page. FALSE = Each tab displays a different page (default).  For example, you could set this to TRUE if you need two tabs to display the same page, but with a different filter applied.

## See Also

[Key Components of a Situational Awareness Project](#)

## Use Nested Panes

A pane can display a page that includes its own set of panes, creating a multi-level arrangement of nested panes.



In the example above, Pane\_A on the master page is configured to display a page called "Page\_1". Page\_1 includes its own sets of panes:

- P1\_Header
- P1\_Content
- P1\_Footer.

The P1\_Content pane is configured to display a page called "Page\_2\_Content", which in turn has two panes:

- P2\_Left
- P2\_Right.

When displaying content, you can address nested panes using dot (.) notation. For example, you could use the following to address the P2\_Left pane:

`Pane_A.P1_Content.P2_Left`

To use nested panes, you need to perform the following tasks:

## Create a page that includes nested panes

1. Open Graphics Builder.
2. Create a page using the **Style** "situational Awareness" and the **Template** "blank".
3. Display the Page Properties.

From the **File** menu, select **Properties**.

Or:

Right-click on the page and select **Properties**.

4. On the **Environment** tab, **Add** a variable.
5. Enter the following values in the Edit Property dialog box:

**Property:** HasPanes

**Value:** 1

This setting instructs the workspace to initialize the panes, display their default content, and consider them when contextual updates occur.

6. On the **Appearance** tab, use the **View Area** settings to define a size for the page that is appropriate to the

pane that will host it.

---

**Note:** If you need to create a large number of pages for a custom-sized pane, consider creating a page template that matches the pane's size.

7. Configure the remaining properties for the page as required.
8. Click **OK**.
9. **Save** the page.

You can then add panes to the page using the procedure described in [Create a Master Page for a Customized Workspace](#). Use the page you have just created, and start at step 3.

## Configure the pane that will host the page

1. In Graphics Builder, open the page that will host the nested panes.
2. Double click on the pane that will display the nested panes. The Pane Properties dialog will appear.
3. In the **Default Page** field, enter the name of the page you created.
4. Configure the remaining properties as required.
5. Click **OK**.
6. **Save** the page.

## Enable autofilling

To enable [Autofill](#) in nested panes you use [Content Types](#). For example, you could use **Pane\_A** in the diagram above to display a report for a piece of equipment that is currently in context. To do this, you would use the following settings:

- The pane **Pane\_A** will have its Content Type property set to 'Report'.
- The page **Page\_1** will have its Content Type set to 'Report'.
- The pane **P2\_Left** will have its Content Type property set to 'Report\_Text'.
- The pane **P2\_Right** will have its Content Type property set to 'Report\_Chart'.
- Pages that appear in **P2\_Left** (for example, "Report\_1"), will have their Content Type set to 'Report\_Text'.
- Pages that appear in **P2\_Right** (for example, "Chart\_1"), will have their Content Type set to 'Report\_Chart'.

If you then want to display a report for a piece of equipment, you need to enter the names of the associated pages as a comma-separated list in the **Content** property for the equipment. In the example above, this would be "Page\_1, Report\_1, Chart\_1".

For more information, see the following:

- [Configure Panes on a Master Page](#)
- [Configure Content Types](#)
- [Assign a Content Type to a Page](#).

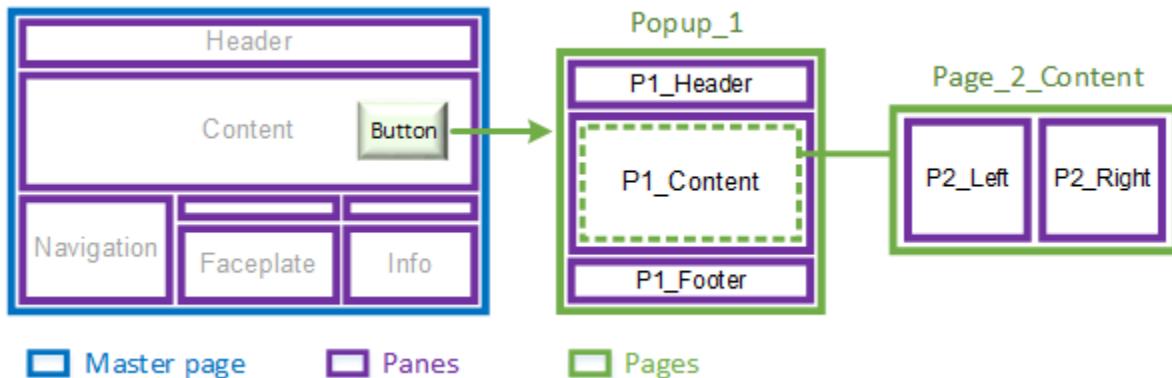
## See Also

[Create a Pop-up Window with Panes](#)

[Workspace\\_ShowContent](#)

## Create a Pop-up Window with Panes

You can create a pop-up window that includes a set of panes.



In the example above, a button in the main content area launches a pop-up window called "Popup\_1".

Popup\_1 includes its own sets of panes:

- P1\_Header
- P1\_Content
- P1\_Footer.

The P1\_Content pane is configured to display a page called "Page\_2\_Content", which in turn has two nested panes:

- P2\_Left
- P2\_Right.

When displaying content, you can address the panes in a pop-up window using the following dot (.) notation. For example, you could use the following to address the P2\_Left pane:

`Popup_1.P1_Content.P2_Left`

To launch a pop-up window, use the Cicode function WinNewAt.

To create the required content for the window, perform the following tasks:

## Create a page that will display in the pop-up window

1. Open Graphics Builder.
2. Create a page using the **Style** "situational Awareness" and the **Template** "blank".
3. Display the Page Properties.  
From the **File** menu, select **Properties**.

Or:

Right-click on the page and select **Properties**.

4. On the **Appearance** tab, use the **View Area** settings to define a size for the page that is appropriate to the pop-up window that will host it.
5. Configure the remaining properties for the page as required.
6. Click **OK**.
7. Save the page.

If you want to include panes on the page, you will also need to perform the following steps.

1. Display the Page Properties.
2. On the **Environment** tab, Add a variable.
3. Enter the following values in the Edit Property dialog box:

**Property:** HasPanes

**Value:** 1

This setting instructs the workspace to initialize the panes, display their default content, and consider them when contextual updates occur.

4. Click **OK**.
5. Save the page.

You can then add panes to the page using the procedure described in [Create a Master Page for a Customized Workspace](#). Use the page you have just created, and start at step 3.

## Enable autofilling

You can enable [Autofill](#) in a pop-up window. For example, you could use Popup\_1 in the diagram above to display a report for a piece of equipment that is currently in context. To do this, you would use the following settings:

- The pane **P1\_Content** will have its Content Type property set to 'Report'.
- The page **Page\_2\_Content** will have its Content Type set to 'Report'.
- The pane **P2\_Left** will have its Content Type property set to 'Report\_Text'.
- The pane **P2\_Right** will have its Content Type property set to 'Report\_Chart'.
- Pages that appear in **P2\_Left** (for example, "Report\_1"), will have their Content Type set to 'Report\_Text'.
- Pages that appear in **P2\_Right** (for example, "Chart\_1"), will have their Content Type set to 'Report\_Chart'.

If you then want to display a report for a piece of equipment, you need to enter the names of the associated pages as a comma-separated list in the **Content** property for the equipment. In the example above, this would be "Page\_2\_Content, Report\_1, Chart\_1".

For more information, see the following:

- [Configure Panes on a Master Page](#)
- [Configure Content Types](#)
- [Assign a Content Type to a Page](#).

## See Also

[Workspace\\_ShowContent](#)

## Use Navigational Genies on Content Pages

The SA\_Include project contains a set of navigational Genies that you can use to direct an operator through a production process that spans multiple graphics pages. The Genies indicate the direction of a link, and include a label that describes the consecutive location. Clicking on the Genie at runtime will take you directly to the linked page (except where a Static Link Genie has been used).

The following navigational Genies are included in the "sa\_navigation" library:

### Link Left/Right

The left and right navigational Genies can be used to link pages that are a part of a linear process.



The link is defined using the path to a menu item in the project's Navigation menu. For example, "Plant1.Area1.MyPage".

You can also indicate that a left/right Genie points to a location that is on a different "console". This means the target is part of a process that is not usually monitored by the current client. For example, a link in a production area of a plant may point to a destination in the warehouse.

To indicate that it points to a different console, the Genie will appear with a gray background.



You can also use a left/right Genie as a control link. This allows you to associate a piece of equipment with the Genie; the Genie will only appear when their associated object is in context.

The navigational Genies will display a purple drop-shadow when configured as a control link.



---

**Note:** Control Links can be enabled or disabled by selecting **Options | Show/Hide Control Links** on the Header Bar.

### Link Up/Down

The up and down navigational Genies can be used to link to pages that are on a different level of a project's menu.



The link is defined using the path to a menu item in the project's navigation menu. For example, an up Genie could point to "Plant1", and a down Genie "Plant1.Area1.MyPage".

## Static Links

The static link Genies function primarily as a label; they are not designed to provide a direct link to a destination.



You can also specify that a static link identifies a location that is on a different console.



### To add a navigational Genie to a page:

1. Open a page in Graphics Builder.
2. Click the **Paste Genie** button in the objects toolbox.



Or, select **Paste Genie** from the **Edit** menu.

3. In the Paste Genie dialog, select the **Library** "sa\_navigation".
4. Select one of the following from the **Genie** list and click **OK**.
  - link\_d = link down
  - link\_l = link left
  - link\_r = link right
  - link\_u = link up
  - slink\_l = static link left
  - slink\_r = static link right.

The Genie is pasted onto the graphics page. A dialog will open, prompting you to configure the properties of the Genie. For a description of the properties, see below.

5. Click **Save**.

## Link Left & Link Right

Field	Description
Menu Item	<p>The path to the linked item in the project's Menu Configuration. This is specified using the following notation:</p> <p>&lt;Level1&gt;.&lt;Level2&gt;.&lt;Level3&gt;.&lt;Level4&gt;.&lt;Level5&gt;.&lt;Level6&gt;</p> <p>For example, "Plant1.Area1.MyPage"</p> <p>The menu item needs to be included under the "Navigation" menu (defined in the <b>Page</b> field). As "Navigation" is assumed as the top level of the path, there is no need to enter it at the start of the menu</p>

Field	Description
	item path.
Tooltip	An optional description of the Genie that appears when the mouse points at it.
Use for 4K Page?	<p>Determines if the Genie will appear on a page that displays within a UHD4K workspace.</p> <p>TRUE = The Genie will appear in a UHD4K workspace.</p> <p>FALSE = The Genie will not appear in a UHD4K workspace.</p>
Links to another console?	<p>Allows you to indicate that a Genie points to a destination on a different console, which is a destination that is not usually monitored by the current client.</p> <p>TRUE = The Genie points to a different console.</p> <p>FALSE = The Genie does not point to a different console.</p> <p>If set to TRUE, the Genie will display a gray background.</p>
Use as Control Link?	<p>Determines if the Genie will function as a control link.</p> <p>TRUE = The Genie will use a control link.</p> <p>FALSE = The Genie will not use a control link.</p> <p>If set to TRUE, the Genie will display a purple drop-shadow.</p>
Control Signal Equipment	The piece of equipment that is associated with the left/right Genie. The Genie will only appear when their associated object is selected.
Controlled Equipment	The piece of equipment that the Control Link points to. When you click on the Genie, the page associated with the piece of equipment will display.

## Link Up & Link Down

Field	Description
Menu Item	<p>The path to the linked item in the project's Menu Configuration. This is specified using the following notation:</p> <p>&lt;Level1&gt;. &lt;Level2&gt;. &lt;Level3&gt;. &lt;Level4&gt;. &lt;Level5&gt;. &lt;Level6&gt;</p>

Field	Description
	For example, "Plant1.Area1.MyPage" The menu item needs to be included under the "Navigation" menu (defined in the <b>Page</b> field). As "Navigation" is assumed as the top level of the path, there is no need to enter it at the start of the menu item path.
Tooltip	An optional description of the Genie that appears when the mouse points at it.
Use for 4K Page?	Determines if the Genie will appear on a page that displays within a UHD_4K workspace. TRUE = The Genie will appear in a UHD4K workspace. FALSE = The Genie will not appear in a UHD4K workspace.

## Static Link Left & Static Link Right

Field	Description
Label	The label that will appear on the Genie.
Tooltip	An optional description of the Genie that appears when the mouse points at it.
Use for 4K Page?	Determines if the Genie will appear on a page that displays within a UHD_4K workspace. TRUE = The Genie will appear in a UHD4K workspace. FALSE = The Genie will not appear in a UHD4K workspace.
Links to another console?	Allows you to indicate that a Genie points to a destination on a different console, which is a destination that is not usually monitored by the current client. TRUE = The Genie points to a different console. FALSE = The Genie does not point to a different console. If set to TRUE, the Genie will display a gray background.

## See Also

[Prepare the Navigation Menu](#)

SA\_Include Navigational Genie Library

## Apply a Color Theme to a Workspace

If you have created a project from the Situational Awareness Starter Project, you can change the theme color used for the workspace(s). This will change the color of the following items:

- Header bar
- Navigation Zone buttons and tabs
- Information Zone buttons and tabs
- Display page buttons
- Selection rectangle.

There are five default color themes you can use:

- Blue (default) — blue header with blue selection/control highlighting
- Grey — grey header with green selection/control highlighting
- Green — green header with green selection/control highlighting
- Indigo — indigo header with indigo selection/control highlighting
- Orange — orange header with orange selection/control highlighting
- Dark — black header with green selection/control highlighting.

For the Grey and Dark themes, green is used for the selection rectangle, buttons and tabs.

### To apply a color theme to a project's workspaces:

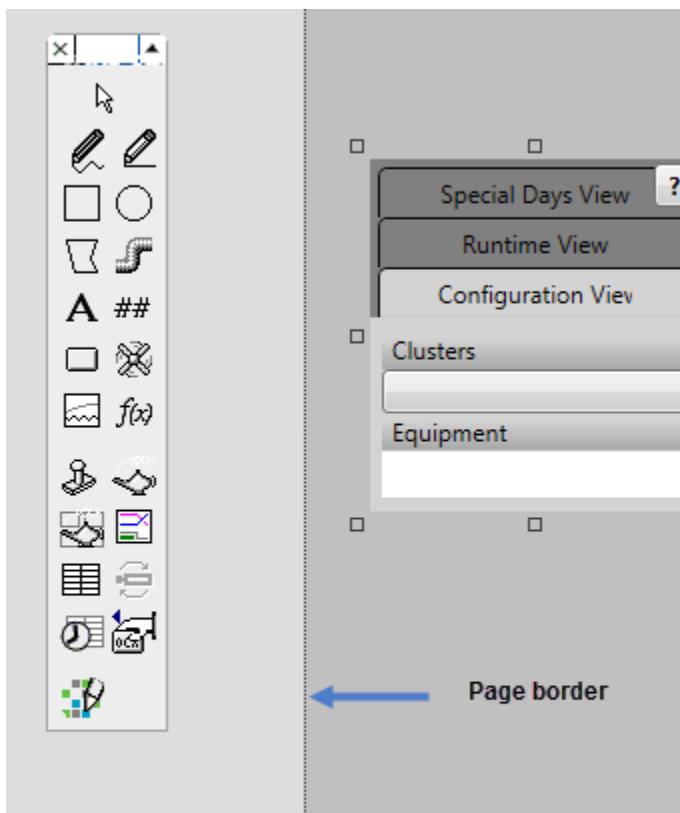
1. In the **Standards** activity, select **Labels**.
2. In the Grid Editor, locate the following label:
  - **Label Name:** \_THEME\_Default
3. In the **Expression** field, specify the required theme color using one of the following labels:
  - THEME\_Blue
  - THEME\_Grey
  - THEME\_SEGreen
  - THEME\_Dark
  - THEME\_Indigo
  - THEME\_Orange
4. Click **Save**.

For a description of a workspace, see [Key Components of a Situational Awareness Project](#).

## Use a Scheduler Control in a Situational Awareness Project

When using a Scheduler control in your Situational Awareness project, it is recommended you add a Scheduler

instance to your project's master pages outside of the page bounds to avoid possible delays when loading Scheduler pages.



Scheduler Control added to master page outside of page border.

To add a Scheduler instance to a master page, follow the steps outlined in [Add the Scheduler ActiveX Object to a Page](#).

Once pasted on the master page, move and position the Scheduler control outside the page border.

## See also

[Scheduler](#)

## Defining a Right Click Action on Equipment

In a Situational Awareness Project, you can define a right-click action/command for a piece of equipment using a callback. This is achieved by specifying a custom function for the label "\_CALLBACK\_EquipmentRightClick".

In a project created from the Situational Awareness Starter Project, the default function specified for the label is "DefaultEquipmentRightClickCallback". To change this:

1. Go to the **Standards** activity and select **Labels**.
2. Locate the "\_CALLBACK\_EquipmentRightClick" label.

	Label Name	Expression
1	_CALLBACK_EquipmentRightClick(sEquip, sEquipSecondary)	DefaultEquipmentRightClickCallback(sEquip, sEquipSecondary)
2	_CALLBACK_FaceplateHeaderDisabled(sEquip, sEquipSecondary)	DefaultFaceplateHeaderDisabledCallback(sEquip, sEquipSecondary)
3	_CALLBACK_OutOfService(sEquip, sEquipSecondary)	DefaultOutOfServiceCallback(sEquip, sEquipSecondary)
4	_THEME_Default	THEME_Blue

3. In the **Expression** field, enter a custom function.

The function you create needs to take the following two arguments:

- sEquip - the current primary equipment context
- sEquipSecondary - the current secondary equipment context.

4. **Save** your changes.

If you do not define a label specifying a custom function (or the default function), a callback error message will be generated when you compile your project.

## See also

[Define a Label](#)

# System Projects

This section of the help provides reference information about the system projects that are included with a Plant SCADA installation. This includes a description of the content of each project and how it can be implemented in a user project.

The system projects described here include:

- **Situational Awareness System Projects** — see [Situational Awareness System Projects](#).
  - **SA\_Include** — see [Situational Awareness Include Project](#)
  - **SA\_Controls** — see [Situational Awareness Controls Project](#)
  - **SA\_Library** — see [Situational Awareness Library Project](#)
- **Sxw\_Style\_Include** — see [StruxureWare \(SxW\) Templates Project](#)
- **Tab\_Style\_Include** — see [Tab Style Templates Project](#)
- **Library\_Controls** — see [Library Controls Include Project](#)
- **Include Project** — see [Include Project Specifications](#).

## See Also

[Project Types](#)

## Situational Awareness System Projects

When you create a project using the Situational Awareness Starter project, the following three system projects are included:

- **Situational Awareness Include Project**: Contains public functions and Genie libraries.
- **Situational Awareness Library Project**: Contains the controls, common controls and priorities Genie libraries.
- **Situational Awareness Controls Project**: Contains a set of Genies for common user interface controls for meter, drive and valve objects.

↳ ExampleSA	<b>Active Project</b>	8.10.0000	May 28th 2018	Workspace and Situational Awareness Example Project
↳ SA_Controls		8.10.0000	May 24th 2018	Situational Awareness Common GUI Controls
↳ SA_Include		8.10.0000	May 24th 2018	Situational Awareness Workspace
↳ SA_Library		8.10.0000	May 24th 2018	Situational Awareness Library Objects

For more information, see [Create a Project Using a Starter Project](#).

When you create a Situational Awareness project that is not based on the starter project but created using the Situational Awareness project template, only the SA\_Library project is included, which in turn includes the SA\_Controls project. For more information, see [Create a Standard Project](#).

**Note:** The library project (SA\_Library) is not included with such a project.

↳ ExampleSA_1	<b>Active Project</b>	1.0	14/08/2018	
↳ SA_Include		8.10.0000	May 24th 2018	Situational Awareness Workspace
Include		8.10.0000	May 24th 2018	Standard Citect include project.
↳ SA_Controls		8.10.0000	May 24th 2018	Situational Awareness Common GUI Controls

**Note:** Avoid making changes to the included system projects for use as a runtime project. A Plant SCADA upgrade will install a new version of the projects, which will overwrite any changes you make.

## See Also

[Situational Awareness Projects](#)

## Situational Awareness Include Project

The Situational Awareness Include Project (named "SA\_Include") is an include project contains the following:

- [Public Functions](#)
- [SA\\_Include Genie Libraries](#).

This project is included when you create a project using the Situational Awareness Starter project.

**Note:** The SA\_Include project is a system include project, and only appears in the Project list tree if the **System Projects** filter is applied in the Projects activity.

## Public Functions

The Situational Awareness Include project, sa-include, contains public Cicode functions that can be used to customize your installation of Plant SCADA Studio, navigation, themes and workspaces. The following categories of functions are available:

- **Debugging functions:** Allow you to trace the user's actions for a workspace and its components. Traced output is written to syslog.dat, which you can review to address any issues encountered.
- **Equipment functions:** Allow you to pass or retrieve equipment context and retrieve workspace content using the equipment context.
- **Help functions:** Allow you customize the display of help in a floating window or a pane within a workspace.
- **Navigation functions:** Allow you to display a specific page in the workspace and add the transition to the navigation stack.
- **NavLink functions:** Allow you to hide the navigation link genies on a page.
- **Security functions:** Allow for security-related operations within workspaces and faceplates.
- **TabBar functions:** Allow you to add tabs to a workspace and control the display if open items do not fit on the display.
- **Theme functions:** Allow for set the basic theme of a workspace and its components, or add custom themes.
- **Workspace functions:** Allow you to:
  - display customized content in any pane
  - set equipment-based context for a workspace

- set the content type for a workspace pane

Below is a list of available functions:

<b>Debugging Functions</b>	<a href="#">Debug_SetTraceLevel</a> <a href="#">Debug_Trace</a>
<b>Equipment Functions</b>	<a href="#">Equipment_GetContent</a> <a href="#">Equipment_GetContext</a>
<b>Help Functions</b>	<a href="#">Help_DisplayFloating</a> <a href="#">Help_DisplayInPane</a>
<b>Navigation Functions</b>	<a href="#">Navigation_ShowHomePage</a> <a href="#">Navigation_ShowTargetPage</a> <a href="#">NavLink_IsHidden</a>
<b>Security Functions</b>	<a href="#">GetPrivEx</a>
<b>TabBar Functions</b>	<a href="#">TabBar_AddTab</a> <a href="#">TabBar_GetTabCount</a> <a href="#">TabBar_GetOpenItemsCount</a>
<b>Theme Functions</b>	<a href="#">Theme_Header_GetBackgroundColorIndex</a> <a href="#">Theme_Header_GetColorIndex</a> <a href="#">Theme_Header_GetMidColorIndex</a> <a href="#">Theme_Header_GetForegroundColorIndex</a> <a href="#">Theme_Control_GetColorIndex</a> <a href="#">Theme_Control_GetForegroundColorIndex</a> <a href="#">Theme_SetTheme</a> <a href="#">Theme_ToggleButton_GetForegroundColorIndex</a>
<b>Workspace Functions</b>	<a href="#">Workspace_ShowContent</a> <a href="#">Workspace_IsSelContext</a> <a href="#">Workspace_GetSelContext</a> <a href="#">Workspace_GetSecondaryContext</a> <a href="#">Workspace_GetSelSecondaryContext</a> <a href="#">Workspace_SetSelContext</a> <a href="#">Workspace_SelectEquipment</a> <a href="#">Workspace_GetEquipmentCluster</a> <a href="#">Workspace_GetWindow</a> <a href="#">Workspace_GetWorkspaceFromWindow</a> <a href="#">Workspace_GetWorkspaceFromName</a>

	<a href="#">Workspace_GetContext</a> <a href="#">Workspace_SetContext</a> <a href="#">Workspace_GetResolution</a> <a href="#">Workspace_GetPageNameWithRes</a> <a href="#">Workspace_IsSystemInitialised</a> <a href="#">Workspace_Init</a> <a href="#">Workspace_RightClickEquipment</a>
--	---

## Debug\_SetTraceLevel

Sets the level of tracing. Setting the trace level to Information will include Warning and Error. Setting it to Warning will include Error. The tracing information is stored in the syslog.dat file.

---

**Note:** Setting the trace level to Information can log a lot of information to the syslog.dat file and may affect performance.

---

You can also configure the default for your project by setting the parameter [\[Debug\]UserTraceMode](#). When setting the trace level through the parameter, you need to set it to the numeric value of the trace level:

- TRACE\_Error: 1
- TRACE.Warning: 2
- TRACE.Information: 3
- TRACE\_Suspend: 0

Setting the trace level to 0 (TRACE\_Suspend) will stop the logging.

## Syntax

**Debug\_SetTraceLevel(INT *nTraceLevel*)**

*nTraceLevel*

The trace level labels defined in the sa\_include project. The default trace level is 2.

## Return Value

Returns -1 if the specified trace level does not exist. Returns 1 if setting the level was successful.

## Example

```
Debug_SetTraceLevel(TRACE_Error);
```

## Related Functions

[Debug\\_Trace](#)

## See Also

[Public Functions](#)

### Debug\_Trace

Adds a trace message to the syslog.dat file. The message is classified by trace level, module and function.

**Note:** It is recommended that this function be used sparingly because it is called even when tracing is turned off.

## Syntax

`Debug_Trace(STRING sModule, STRING sFunction, STRING sMessage, INT nLevel=TRACE_Information)`

*sModule*

The module of the function that generated the trace message.

*sFunction*

The function which generated the trace message.

*sMessage*

The trace message to output.

*nLevel*

The TRACE\_<level> Labels defined in sa\_include. See [Debug\\_SetTraceLevel](#) for the list of values.

## Return Value

N/A.

## Example

```
Debug_Trace("Workspace", "_Workspace_FindPane", "Pane '" + sSearchTerm + "' not found.",  
TRACE_Error);
```

## Related Functions

[Debug\\_SetTraceLevel](#)

## See Also

[Public Functions](#)

### Equipment\_GetContent

Returns the content of the equipment instance referenced by the specified menu item. This function should be used with the built equipment models which are loaded into menus.

## Syntax

Equipment\_GetContent(INT *hMenuItem*)

*hMenuItem*

Menu item from which to retrieve the equipment content.

## Return Value

A comma separated list of page names belonging to the equipment. The content returned will be the combination of both the equipment's 'Page', and its 'Content' fields.

## Related Functions

[Equipment\\_GetContext](#)

## See Also

[Public Functions](#)

## Equipment\_GetContext

Returns the fully qualified context of the equipment referenced by the specified menu. This function should be used with the built equipment models which are loaded into menus.

## Syntax

Equipment\_GetContext(INT *hMenuItem*)

*hMenuItem*

Menu item from which to retrieve the equipment context.

## Return Value

<cluster name>.<equipment path>

## Related Functions

[Equipment\\_GetContent](#)

## See Also

[Public Functions](#)

## EquipmentStatus\_Drive\_GetValue

Maps the various item names for the status items within the PLC into the states required by the status indicator for a drive object.

### Syntax

EquipmentStatus\_Drive\_GetValue(STRING *sEquipment*, STRING *sEquipmentValueItem*)

*sEquipment*

The name of the equipment that has been selected.

*sEquipmentValueItem*

The name of the item to use for the equipment "value".

### Return Value

The status as calculated by \_EquipmentStatus\_UpdateValue for the supplied arguments.

### Related Functions

[Equipment\\_GetContext](#)

### See Also

[Public Functions](#)

## EquipmentStatus\_Meter\_GetValue

Maps the various item names for the status items within the PLC into the states required by the status indicator for a meter object.

### Syntax

EquipmentStatus\_Meter\_GetValue(STRING *sEquipment*, STRING *sEquipmentValueItem*)

*sEquipment*

The name of the equipment that has been selected.

*sEquipmentValueItem*

The name of the item to use for the equipment "value".

### Return Value

The status as calculated by \_EquipmentStatus\_UpdateValue for the supplied arguments.

## Related Functions

[Equipment\\_Context](#)

## See Also

[Public Functions](#)

### EquipmentStatus\_Valve\_GetValue

Maps the various item names for the status items within the PLC into the states required by the status indicator for a valve object.

## Syntax

EquipmentStatus\_Valve\_GetValue(STRING *sEquipment*, STRING *sEquipmentValueItem*)

*sEquipment*

The name of the equipment that has been selected.

*sEquipmentValueItem*

The name of the item to use for the equipment "value".

## Return Value

The status as calculated by \_EquipmentStatus\_UpdateValue for the supplied arguments.

## Related Functions

[Equipment\\_Context](#)

## See Also

[Public Functions](#)

### GetPrivEx

Checks if the current user has the required privilege for a specified area and a specified piece of equipment.

If you have specific additional security requirements per equipment, you can set the default

'\_CALLBACK\_EquipmentSecurity' label to a custom function of your own so that it will be called in addition to the default security check. The current default function is "DefaultEquipmentSecurityCallback".

The '\_CALLBACK\_EquipmentSecurity' label is set by default in the SA\_Style1\_MultiRes starter project. To view go to Standards | Labels.

	Label Name	Expression
1	_CALLBACK_EquipmentRightClick(sEquip, sEquipSecondary)	DefaultEquipmentRightClickCallback(sEquip, sEquipSecondary)
2	_CALLBACK_FaceplateHeaderDisabled(sEquip, sEquipSecond	DefaultFaceplateHeaderDisabledCallback(sEquip, sEquipSecond
3	_CALLBACK_OutOfService(sEquip, sEquipSecondary)	DefaultOutOfServiceCallback(sEquip, sEquipSecondary)
4	_THEME_Default	THEME_Blue

**Note:** The default callback function does not use the *sOperation* or *sArg* fields.

## Syntax

GetPrivEx(INT *nPriv*, INT *nArea* [, STRING *sEquipment* [, STRING *sOperation* [, STRING *sArg*]]])

*nPriv*

The privilege level - 0 to 8.

*nArea*

The area of privilege - 0 to 255.

*sEquipment*

Equipment reference in which you are interested. It does not need to be in the same area. This is an optional parameter.

*sOperation*

The reason for privilege request. This is an optional parameter.

*sArg*

Custom argument to be passed to callback function, which is defined by its caller. This is an optional parameter.

## Return Value

Returns 1 if the user has the specified privilege in the area, or 0 (zero) if the user does not have the privilege.

## Example

Example below outlines using a custom function to determine if the user can bypass or removebypass when working with interlocks.

```
// Extended callback that supports interlocks context
GLOBAL STRING g_sInterlockBypassOperation = "Interlock.Bypass"; // Interlock Bypass
Operation
GLOBAL STRING g_sInterlockRemoveBypassOperation = "Interlock.RemoveBypass"; // Interlock
Remove Bypass Operation
PUBLIC
INT
FUNCTION CustomSecurityCallback(INT nPriv, INT nArea, STRING sEquipment, STRING sOperation,
STRING sArg)
    STRING sCategory;
```

```
STRING sInterlockingItem;
INT nResult = 1;
sCategory = Interlocks_GetSecurityCallbackCategory(sArg);
SinterlockingItem = Interlocks_GetSecurityCallbackReferencedItem(sArg);
TraceMsg("Interlocks CustomSecurityCallback: " + sEquipment + "/" + sArg + "/" +
sOperation + "/" + sCategory + "/" + sInterlockingItem);
IF (sOperation = g_sInterlockBypassOperation) THEN
    TraceMsg(sInterlockingItem + " in interlock category " + sCategory + " is being
bypassed for " + sEquipment);
    // Don't allow bypass for 'GuestEngineer'
    IF Name() = "GuestEngineer" THEN
        nResult = 0;
    END
ELSE
    IF (sOperation = g_sInterlockRemoveBypassOperation) THEN
        TraceMsg(sInterlockingItem + " in interlock category " + sCategory + " is
removing bypass for " + sEquipment);
        // Don't allow removebypass for 'GuestManager'
        IF Name() = "GuestManager" THEN
            nResult = 0;
        END
    END
END
RETURN nResult;
END
```

## See Also

[Public Functions](#)

## Help\_DisplayFloating

Displays the help page in a floating window.

## Syntax

Help\_DisplayFloating(INT *bAutoPosition*, INT *nPositionMode*, INT *x*, INT *y*)

*bAutoPosition*

TRUE window will be placed near the focused object. FALSE will use the x/y coordinates

*nPositionMode*

Determines the position where the window will be placed. This can take the following values:

0 - use the passed x and y co-ordinates

1 - place below the focused object

2 - place below the Header Bar Help button

*x*

*x* position of the left of the window.

*y*

*y* position of the top of the window.

## Return Value

N/A

## Example

```
Help_DisplayFloating(FALSE, WndInfo(0), 0)
```

## Related Functions

[Help\\_DisplayInPane](#)

## See Also

[Public Functions](#)

## Help\_DisplayInPane

Displays the help page in a pane.

## Syntax

```
Help_DisplayInPane(STRING sPane, STRING sWorkspaceName)
```

*sPane*

The name of the pane in which to display the help page. To ensure the help displays without scaling, ensure your pane is sized to match the desired help page size.

*sWorkspaceName*

The name of the workspace in which to display the pane. Specify "" to display in *sPane* of the workspace of the current window (default). This is an optional parameter.

## Return Value

N/A

## Example

```
Help_DisplayInPane("MyPane")
```

## Related Functions

[Help\\_DisplayFloating](#)

## See Also

[Public Functions](#)

### Navigation\_ShowHomePage

Shows the home page of the current workspace's context in the specified workspace.

**Note:** If your project contains "Home\_HD1080" and "Home\_UHD4K" variations of the home page to support multiple screen resolutions, passing "Home" will automatically display Home\_HD1080 if the workspace master page also has \_HD1080 as its extension. For more information, see [Configure a Project that Supports Multiple Screen Resolutions](#).

## Syntax

`Navigation_ShowHomePage([STRING sWorkspaceName])`

*sWorkspaceName*

The name of the workspace in which the home page will be displayed. This is an optional parameter. The default value for this parameter is "", which uses the workspace of the current window.

## Return Value

N/A

## Related Functions

[Navigation\\_ShowTargetPage](#), [NavigationLink\\_IsHidden](#)

## See Also

[Public Functions](#)

### Navigation\_ShowTargetPage

Displays the page specified in the **Target Page** field configured for the current Navigation Menu Item.

This function should only be called as a menu command of a menu item. If your project is using multiple resolutions, you only need to specify the first part of your page name and the workspace will choose the best version to suit the required screen resolution. For example, if you have "MyPage\_HD1080" and "MyPage\_UHD4K", just specify "MyPage". For more information, see [Configure a Project that Supports Multiple Screen Resolutions](#).

## Syntax

`Navigation_ShowTargetPage([STRING sPaneRef [, STRING sEquipmentContext [, STRING sEquipmentSecondaryContext [, STRING sWorkspaceName]]]])`

***sPaneRef***

Name of the pane in which to display the target page. This is an optional parameter. This is the format of "<panename>.<panename>" if needing to display in a nested pane. If not specified, a pane that hosts the content type of the target page will be used. The default is "", which displays the page in the default pane.

***sEquipmentContext***

Equipment context to set for the page. This is an optional parameter. If not specified, the current context of the workspace will be cleared. The default value is label WS\_CONTEXT\_Clear, which clears the context. If set to "", it retains the existing context.

***sEquipmentSecondaryContext***

The value of the equipment name that is set as the secondary context for related equipment. This is an optional parameter. For example, for a Polar Star Composite Genie, this will be the value specified in the **Equipment Name** option in the Presentation Options dialog box. This is used to determine the number of symbols to be displayed for MEOs. The default value is "", which retains the existing context.

***sWorkspaceName***

The name of the workspace in which to display the page. This is an optional parameter. If not specified, the workspace of the current window will be used. The default is "", which uses the workspace of the current window.

## Return Value

N/A

## Related Functions

[Navigation\\_ShowHomePage](#), [NavLink\\_IsHidden](#)

## See Also

[Public Functions](#)

## NavLink\_IsHidden

Checks if link object should be hidden based on whether the selected equipment is a member of the control group. See Use Navigational Genies on Content Pages.

## Syntax

NavLink\_IsHidden(STRING *sEquipFrom*, STRING *sEquipTo* [, INT *bIsControlLink*])

***sEquipFrom***

The name (path) of the equipment that the control link is from. Set this to empty if have none.

***sEquipTo***

The name (path) of equipment that the control link links to. Set this to empty if have none.

***bIsControlLink***

Indicates whether the link is a control link. Default value is TRUE. This is an optional parameter.

#### Return Value

Returns FALSE (not hidden) if the selected equipment belongs to the control group; otherwise it returns TRUE.

#### Example

```
NavigationLink_IsHidden("AlarmSite.Area2.Equipment1","AlarmSite.Area2.Equipment11",TRUE)
```

## Related Functions

[Navigation\\_ShowHomePage](#), [Navigation\\_ShowTargetPage](#)

## See Also

[Public Functions](#)

### **TabBar\_AddTab**

Adds a new tab to a given tab control.

## Syntax

```
TabBar_AddTab(INT nTabBarAN, INT bActivate, STRING sTabLabel, STRING sContent [, STRING sEquipmentContext [, STRING sEquipmentSecondaryContext [, STRING sIcon [, INT bCanClose [, INT bIsPinned [, STRING sOnActivateCommand]]]]]])
```

*nTabBarAN*

The AN of the tab bar to which the tab will be added.

*bActivate*

Activate the tab after adding it.

*sTabLabel*

Display label of the tab.

*sContent*

Page Name to display.

*sEquipmentContext*

Equipment to which the content belongs. The default value is "".

*sEquipmentSecondaryContext*

Secondary equipment to which the content belongs. The default value is "".

*sIcon*

Not used.

*bCanClose*

Indicates whether the tab can be closed by the operator. The default value is TRUE.

*bIsPinned*

Indicates whether the tab is pinned (TRUE) or unpinned (FALSE). The default value is FALSE.

**sOnActivateCommand**

Command to run after the tab is activated. This is an optional parameter. The default value is "", which means no command will be run. This parameter can take the following values:

h:<menu handle>  
<function>(<arguments>)

**Return Value**

The ID of the tab (opened item number)

**Related Functions**

[TabBar\\_GetOpenItemCount](#), [TabBar\\_GetTabCount](#)

**See Also**

[Public Functions](#)

**TabBar\_GetTabCount**

Gets the number of tabs currently displayed.

**Syntax**

TabBar\_GetTabCount(INT *nTabBarAN*)

*nTabBarAN*

The AN of the tab bar.

**Return Value**

The number of displayed tabs.

**Related Functions**

[TabBar\\_AddTab](#)

**See Also**

[Public Functions](#)

**TabBar\_GetOpenItemCount**

Gets the number of open items.

## Syntax

```
TabBar_GetOpenItemsCount(INT nTabBarAN)
```

*nTabBarAN*

The AN of the tab bar.

## Return Value

The number of open items

## Example

```
// Add a Tab Bar control "Tabbar.h" to a page
// The genie is named "Tabbar"
INT hTabBarAN = DspGetANFromName("Tabbar")
INT nCount = TabBar_GetOpenItemsCount(hTabBarAN)
```

## Related Functions

[TabBar\\_AddTab](#), [TabBar\\_GetTabCount](#)

## See Also

[Public Functions](#)

## TabBar\_ShowIcon

Shows/hides the icon of an open item of the tab bar.

## Syntax

```
TabBar_ShowIcon(INT nTabBarAN, INT nTabId [, INT bShow])
```

*nTabBarAN*

The AN of the tab bar.

*nTabId*

The opened item number starting from 1.

*bShow*

Whether to show the icon configured for the tab. Set this to TRUE to show or FALSE to hide. Default value is TRUE.

## Return Value

None.

## See Also

[Public Functions](#)

### **Theme\_Control\_GetColorIndex**

Returns the common control (button) color index for the specific button state. This is different from the header button. The normal state color is always light grey regardless of the theme selected.

## Syntax

`Theme_Control_GetColorIndex(INT nState)`

*nState*

Indicates which state the UI element is in such that correct shade of the color is returned. This can be one of the following values:

CTRL_ITEM_STATE_Normal	0	Normal state - no mouse over
CTRL_ITEM_STATE_NormalHover	2	Mouse is hovering over control
CTRL_ITEM_STATE_NormalPressed	3	Mouse button pressed on control
CTRL_ITEM_STATE_Active	4	Item is in selected state
CTRL_ITEM_STATE_ActiveHover	5	Item is in selected and mouse is hovering
CTRL_ITEM_STATE_ActivePressed	6	Item is selected and mouse button is down
CTRL_ITEM_STATE_Disabled	1	Item is disabled
CTRL_ITEM_STATE_ActiveDisabled	7	Item is in selected AND disabled state\

Typically, this state will be returned by a function such as "Button\_GetState".

## Return Value

Returns an index into the color array configured on the graphics object that called this function.

## Example

```
Theme_Control_GetColorIndex(Button_GetState(DspGetAnCur())))
```

## Related Functions

[Theme\\_Control\\_GetForegroundColorIndex](#), [Theme\\_Header\\_GetBackgroundColorIndex](#),

[Theme\\_Header\\_GetColorIndex](#), [Theme\\_Header\\_GetForegroundColorIndex](#), [Theme\\_Header\\_GetMidColorIndex](#),  
[Theme\\_SetTheme](#), [Theme\\_ToggleButton\\_GetForegroundColorIndex](#)

## See Also

[Public Functions](#)

### Theme\_Control\_GetForegroundColorIndex

Returns the control highlight foreground color index used for icons and text.

## Syntax

Theme\_Control\_GetForegroundColorIndex(INT *nState*)

*nState*

Indicates which state the UI element is in such that correct shade of the color is returned. This can be one of the following values:

CTRL_ITEM_STATE_Normal	0	Normal state - no mouse over
CTRL_ITEM_STATE_NormalHover	2	Mouse is hovering over control
CTRL_ITEM_STATE_NormalPressed	3	Mouse button pressed on control
CTRL_ITEM_STATE_Active	4	Item is in selected state
CTRL_ITEM_STATE_ActiveHover	5	Item is in selected and mouse is hovering
CTRL_ITEM_STATE_ActivePressed	6	Item is selected and mouse button is down
CTRL_ITEM_STATE_Disabled	1	Item is disabled
CTRL_ITEM_STATE_ActiveDisabled	7	Item is in selected AND disabled state

Typically, this state will be returned by a function such as "Button\_GetState".

## Return Value

Returns an index into the color array configured on the graphics object that called this function.

## Example

```
Theme_Control_GetForegroundColorIndex(Button_GetState(DspGetAnCur())))
```

## Related Functions

[Theme\\_Control\\_GetColorIndex](#), [Theme\\_Header\\_GetBackgroundColorIndex](#), [Theme\\_Header\\_GetColorIndex](#),  
[Theme\\_Header\\_GetMidColorIndex](#), [Theme\\_SetTheme](#), [Theme\\_ToggleButton\\_GetForegroundColorIndex](#)

## See Also

[Public Functions](#)

### Theme\_Header\_GetBackgroundColorIndex

Returns the background color index for the Header Bar based on the active theme.

## Syntax

```
Theme_Header_GetBackgroundColorIndex()
```

## Return Value

Returns an index into the color array configured on the graphics object that called this function.

## Example

```
Theme_Header_GetBackgroundColorIndex()
```

## Related Functions

[Theme\\_Control\\_GetColorIndex](#), [Theme\\_Control\\_GetForegroundColorIndex](#), [Theme\\_Header\\_GetColorIndex](#),  
[Theme\\_Header\\_GetForegroundColorIndex](#), [Theme\\_Header\\_GetMidColorIndex](#), [Theme\\_SetTheme](#),  
[Theme\\_ToggleButton\\_GetForegroundColorIndex](#)

## See Also

[Public Functions](#)

### Theme\_Header\_GetColorIndex

Returns the Header toolbar button color index for the specific button state.

## Syntax

```
Theme_Header_GetColorIndex(INT nState)
```

*nState*

Indicates which state the UI element is in such that correct shade of the color is returned. This can be one of the

following values:

CTRL_ITEM_STATE_Normal	0	Normal state - no mouse over
CTRL_ITEM_STATE_NormalHover	2	Mouse is hovering over control
CTRL_ITEM_STATE_NormalPressed	3	Mouse button pressed on control
CTRL_ITEM_STATE_Active	4	Item is in selected state
CTRL_ITEM_STATE_ActiveHover	5	Item is in selected and mouse is hovering
CTRL_ITEM_STATE_ActivePressed	6	Item is selected and mouse button is down
CTRL_ITEM_STATE_Disabled	1	Item is disabled
CTRL_ITEM_STATE_ActiveDisabled	7	Item is in selected AND disabled state

Typically, this state will be returned by a function such as "Button\_GetState".

## Return Value

Returns an index into the color array configured on the graphics object that called this function.

## Example

```
Theme_Header_GetColorIndex(Button_GetState(dspGetAnCur())))
```

## Related Functions

[Theme\\_Control\\_GetColorIndex](#), [Theme\\_Control\\_GetForegroundColorIndex](#),  
[Theme\\_Header\\_GetForegroundColorIndex](#), [Theme\\_Header\\_GetMidColorIndex](#), [Theme\\_SetTheme](#),  
[Theme\\_ToggleButton\\_GetForegroundColorIndex](#)

## See Also

[Public Functions](#)

### Theme\_Header\_GetForegroundColorIndex

Returns the header foreground color index used for icons and text.

## Syntax

```
Theme_Header_GetForegroundColorIndex(INT nState)
```

*nState*

Indicates which state the UI element is in such that correct shade of the color is returned. This can be one of the following values:

CTRL_ITEM_STATE_Normal	0	Normal state - no mouse over
CTRL_ITEM_STATE_NormalHover	2	Mouse is hovering over control
CTRL_ITEM_STATE_NormalPressed	3	Mouse button pressed on control
CTRL_ITEM_STATE_Active	4	Item is in selected state
CTRL_ITEM_STATE_ActiveHover	5	Item is in selected and mouse is hovering
CTRL_ITEM_STATE_ActivePressed	6	Item is selected and mouse button is down
CTRL_ITEM_STATE_Disabled	1	Item is disabled
CTRL_ITEM_STATE_ActiveDisabled	7	Item is in selected AND disabled state

Typically, this state will be returned by a function such as "Button\_GetState".

**Return Value**

Returns an index into the color array configured on the graphics object which called this function.

**Example**

```
Theme_Header_GetForegroundColorIndex(Button_GetState(DspGetAnCur())))
```

**Related Functions**

[Theme\\_Control\\_GetColorIndex](#), [Theme\\_Control\\_GetForegroundColorIndex](#), [Theme\\_Header\\_GetColorIndex](#),  
[Theme\\_Header\\_GetMidColorIndex](#), [Theme\\_SetTheme](#), [Theme\\_ToggleButton\\_GetForegroundColorIndex](#)

**See Also**

[Public Functions](#)

**Theme\_Header\_GetMidColorIndex**

Returns the header mid color index. This color is used for separators and other decorative UI elements in the Header Bar.

## Syntax

```
Theme_Header_GetMidColorIndex()
```

## Return Value

Returns an index into the color array configured on the graphics object that called this function.

## Example

```
Theme_Header_GetMidColorIndex()
```

## Related Functions

[Theme\\_Control\\_GetColorIndex](#), [Theme\\_Control\\_GetForegroundColorIndex](#),  
[Theme\\_Header\\_GetBackgroundColorIndex](#), [Theme\\_Header\\_GetColorIndex](#),  
[Theme\\_Header\\_GetForegroundColorIndex](#), [Theme\\_SetTheme](#), [Theme\\_ToggleButton\\_GetForegroundColorIndex](#)

## See Also

[Public Functions](#)

## Theme\_SetTheme

Selects the theme to apply to the user interface. This will update dynamically at runtime.

## Syntax

```
Theme_SetTheme(INT nTheme)
```

*nTheme*

The ID of the theme. This can be one of the default themes:

THEME\_Grey - Header is Grey, selection color is SE\_Green

THEME\_Blue - Header is Blue, selection color is Blue

THEME\_SEGreen - Header is SE\_Green, selection color is SE\_Green

THEME\_Indigo - Header is Indigo, selection color is Indigo

THEME\_Orange - Header is Orange, selection color is Orange

THEME\_Dark - Header is Black, selection color is SE\_Green

Or it can be value 5 or 6, which are custom themes which you can configure with [Theme\\_SetHeaderColor](#) and [Theme\\_SetSelectionColor](#).

## Return Value

N/A

## Example

```
Theme_SetTheme(THEME_SEGreen)
```

## Related Functions

[Theme\\_Control\\_GetColorIndex](#), [Theme\\_Control\\_GetForegroundColorIndex](#),  
[Theme\\_Header\\_GetBackgroundColorIndex](#), [Theme\\_Header\\_GetColorIndex](#),  
[Theme\\_Header\\_GetForegroundColorIndex](#), [Theme\\_Header\\_GetMidColorIndex](#),  
[Theme\\_ToggleButton\\_GetForegroundColorIndex](#)

## See Also

[Public Functions](#)

### Theme\_ToggleButton\_GetForegroundColorIndex

Returns the control highlight foreground color for a toggle button used for icons and text.

## Syntax

```
Theme_ToggleButton_GetForegroundColorIndex(INT nState)
```

*nState*

Indicates which state the UI element is in such that correct shade of the color is returned.

## Return Value

Returns an index into the color array configured on the graphics object that called this function.

## Example

```
Theme_ToggleButton_GetForegroundColorIndex(Button_GetState(DspGetAnFromName("../Button3.Background"))))
```

## Related Functions

[Theme\\_Control\\_GetColorIndex](#), [Theme\\_Control\\_GetForegroundColorIndex](#),  
[Theme\\_Header\\_GetBackgroundColorIndex](#), [Theme\\_Header\\_GetColorIndex](#),  
[Theme\\_Header\\_GetForegroundColorIndex](#), [Theme\\_Header\\_GetMidColorIndex](#), [Theme\\_SetTheme](#)

## See Also

[Public Functions](#)

## **Workspace\_GetContext**

Returns the equipment context of the current window's workspace.

### **Syntax**

```
Workspace_GetContext()
```

### **Return Value**

Returns the fully qualified equipment name. For example, Site1.Level1.Level12.FT001.

### **Related Functions**

[Workspace\\_GetEquipmentCluster](#), [Workspace\\_GetPageNameWithRes](#), [Workspace\\_GetResolution](#),  
[Workspace\\_GetSelContext](#), [Workspace\\_GetWindow](#), [Workspace\\_GetWorkspaceFromName](#),  
[Workspace\\_GetWorkspaceFromWindow](#), [Workspace\\_Init](#), [Workspace\\_IsSelContext](#),  
[Workspace\\_IsSystem\\_Initialised](#), [Workspace\\_SelectEquipment](#), [Workspace\\_SetSelContext](#),  
[Workspace\\_ShowContent](#)

### **See Also**

[Public Functions](#)

## **Workspace\_GetEquipmentCluster**

Returns the given equipment string with its cluster prefix and caching it in the process. The function will attempt to automatically detect the cluster of the equipment in the following order:

1. In a single cluster system, the cluster may be automatically resolved and require no further custom code.
2. Get the cluster from the page cluster context.
3. Check each configured cluster to see if the passed equipment exists.

If the cluster cannot be automatically detected, the original equipment string will be returned.

### **Syntax**

```
Workspace_GetEquipmentCluster(STRING sEquipmentContext [, INT bRefreshCache])
```

*sEquipmentContext*

The equipment context to verify.

*bRefreshCache*

Force the function to ignore any cached value. This is an optional parameter. The default value is FALSE.

## Return Value

The context will be returned with the cluster prefix. If the cluster cannot be automatically detected, the original equipment string will be returned.

## Example

```
STRING sEquipmentNameWithCluster = WWorkspace_GetEquipmentCluster("MyPlant.Mixing.TNK01");  
Result: Cluster1.MyPlant.Mixing.TNK01
```

## Related Functions

[Workspace\\_GetContext](#), [Workspace\\_GetPageNameWithRes](#), [Workspace\\_GetResolution](#),  
[Workspace\\_GetSelContext](#), [Workspace\\_GetWindow](#), [Workspace\\_GetWorkspaceFromName](#),  
[Workspace\\_GetWorkspaceFromWindow](#), [Workspace\\_Init](#), [Workspace\\_IsSelContext](#),  
[Workspace\\_IsSystem\\_Initialised](#), [Workspace\\_SelectEquipment](#), [Workspace\\_SetSelContext](#),  
[Workspace\\_ShowContent](#)

## See Also

[Public Functions](#)

## Workspace\_GetPageNameWithRes

If you are using multiple resolutions in your project, this function can be used to return an adjusted page name matching your workspace resolution. If a page does exist in your project matching the resolution, then the original name will be returned.

Your pages need to be named "<PageName>\_HD1080" or "<PageName>\_UHD4K".

For example if your project supports HD1080 and UHD4K, and you have designed variations of your page for each resolution: "Test\_HD1080" and "Test\_UHD4K", then call this function with the page name "Test" . .

This is BLOCKING function.

## Syntax

```
Workspace_GetPageNameWithRes(STRING sPage [, STRING sWorkspaceName])
```

*sPage*

Name of the page.

*sWorkspaceName*

Name of the workspace to which the page belongs. This is an optional parameter. The default value is "", which uses the workspace of the current window.

## Return Value

Returns the adjusted page name or the original passed in name if a page is not found matching the workspace

resolution.

## Related Functions

[Workspace\\_GetContext](#), [Workspace\\_GetEquipmentCluster](#), [Workspace\\_GetResolution](#),  
[Workspace\\_GetSelContext](#), [Workspace\\_GetWindow](#), [Workspace\\_GetWorkspaceFromName](#),  
[Workspace\\_GetWorkspaceFromWindow](#), [Workspace\\_Init](#), [Workspace\\_IsSelContext](#),  
[Workspace\\_IsSystem\\_Initialised](#), [Workspace\\_SelectEquipment](#), [Workspace\\_SetSelContext](#),  
[Workspace\\_ShowContent](#)

## See Also

[Public Functions](#)

### Workspace\_GetResolution

Returns the resolution of a workspace.

## Syntax

`Workspace_GetResolution(STRING sWorkspaceName)`

*sWorkspaceName*

Name of the workspace for which to get the resolution.

## Return Value

Returns the resolution name for the workspace.

## Related Functions

[Workspace\\_GetContext](#), [Workspace\\_GetEquipmentCluster](#), [Workspace\\_GetPageNameWithRes](#),  
[Workspace\\_GetSelContext](#), [Workspace\\_GetWindow](#), [Workspace\\_GetWorkspaceFromName](#),  
[Workspace\\_GetWorkspaceFromWindow](#), [Workspace\\_Init](#), [Workspace\\_IsSelContext](#),  
[Workspace\\_IsSystem\\_Initialised](#), [Workspace\\_SelectEquipment](#), [Workspace\\_SetSelContext](#),  
[Workspace\\_ShowContent](#)

## See Also

[Public Functions](#)

### Workspace\_GetSecondaryContext

Returns the secondary equipment context of the workspace to which the current window belongs.

## Syntax

```
Workspace_GetSecondaryContext()
```

## Return Value

Returns the fully qualified name of the secondary equipment. For example, Site1.Level1.Level2.PMP01.

## Example

```
STRING sCurrentSecondaryContext = Workspace_GetSecondaryContext()
```

## Related Functions

[Workspace\\_GetContext](#), [Workspace\\_GetSelContext](#), [Workspace\\_GetWindow](#),  
[Workspace\\_GetWorkspaceFromName](#), [Workspace\\_GetWorkspaceFromWindow](#), [Workspace\\_SetSelContext](#)

## See Also

[Public Functions](#)

## Workspace\_GetSelContext

Gets the selected equipment context of the workspace on the current page.

## Syntax

```
Workspace_GetSelContext()
```

## Return Value

Returns the selected context.

## Example

```
STRING sCurrentContext = Workspace_GetSelContext()
```

## Related Functions

[Workspace\\_GetContext](#), [Workspace\\_GetEquipmentCluster](#), [Workspace\\_GetPageNameWithRes](#),  
[Workspace\\_GetResolution](#), [Workspace\\_GetWindow](#), [Workspace\\_GetWorkspaceFromName](#),  
[Workspace\\_GetWorkspaceFromWindow](#), [Workspace\\_Init](#), [Workspace\\_IsSelContext](#),  
[Workspace\\_IsSystem\\_Initialised](#), [Workspace\\_SelectEquipment](#), [Workspace\\_SetSelContext](#),  
[Workspace\\_ShowContent](#)

## See Also

[Public Functions](#)

### Workspace\_GetSelSecondaryContext

Gets the selected secondary equipment context of the workspace on the current page.

## Syntax

```
Workspace_GetSelSecondaryContext()
```

## Return Value

Returns the selected secondary context for the current workspace.

## Example

```
STRING sCurrentSecondaryContext = Workspace_GetSelSecondaryContext()
```

## Related Functions

[Workspace\\_GetContext](#), [Workspace\\_GetEquipmentCluster](#), [Workspace\\_GetPageNameWithRes](#),  
[Workspace\\_GetResolution](#), [Workspace\\_GetSelContext](#), [Workspace\\_GetWindow](#),  
[Workspace\\_GetWorkspaceFromName](#), [Workspace\\_GetWorkspaceFromWindow](#), [Workspace\\_Init](#),  
[Workspace\\_IsSelContext](#), [Workspace\\_IsSystem\\_Initialised](#), [Workspace\\_SelectEquipment](#),  
[Workspace\\_SetSelContext](#), [Workspace\\_ShowContent](#)

## See Also

[Public Functions](#)

### Workspace\_GetWindow

Returns the workspace window for the specified workspace.

## Syntax

```
Workspace_GetWindow([STRING sWorkspaceName])
```

*sWorkspaceName*

The workspace name of the window, OR "" to get the window number of the current pinned window. This is an optional parameter.

## Return Value

A Plant SCADA window number OR BAD\_HANDLE if the specified workspace does not exist.

## Example

```
INT nCitectWindow = Workspace_GetWindow("PrimaryScreen");
```

## Related Functions

[Workspace\\_GetContext](#), [Workspace\\_GetEquipmentCluster](#), [Workspace\\_GetPageNameWithRes](#),  
[Workspace\\_GetResolution](#), [Workspace\\_GetSelContext](#), [Workspace\\_GetWorkspaceFromName](#),  
[Workspace\\_GetWorkspaceFromWindow](#), [Workspace\\_Init](#), [Workspace\\_IsSelContext](#),  
[Workspace\\_IsSystem\\_Initialised](#), [Workspace\\_SelectEquipment](#), [Workspace\\_SetSelContext](#),  
[Workspace\\_ShowContent](#)

## See Also

[Public Functions](#)

## Workspace\_GetWorkspaceFromName

Returns the workspace ID of the specified workspace name.

## Syntax

Workspace\_GetWorkspaceFromName(STRING *sWorkspaceName* [, INT *nTrace*])

*sWorkspaceName*

Name of the workspace for which you want to get the ID.

*nTrace*

Indicates the trace level to record if an error results. The default value is TRACE\_Error. This is an optional parameter.

## Return Value

Returns the Workspace ID.

## Related Functions

[Workspace\\_GetContext](#), [Workspace\\_GetEquipmentCluster](#), [Workspace\\_GetPageNameWithRes](#),  
[Workspace\\_GetResolution](#), [Workspace\\_GetSelContext](#), [Workspace\\_GetWindow](#),  
[Workspace\\_GetWorkspaceFromWindow](#), [Workspace\\_Init](#), [Workspace\\_IsSelContext](#),  
[Workspace\\_IsSystem\\_Initialised](#), [Workspace\\_SelectEquipment](#), [Workspace\\_SetSelContext](#),  
[Workspace\\_ShowContent](#)

## See Also

[Public Functions](#)

### Workspace\_GetWorkspaceFromWindow

Returns the workspace ID to which the current window belongs.

## Syntax

`Workspace_GetWorkspaceFromWindow(INT nWindow [, INT nTrace])`

*nWindow*

Handle of the window.

*nTrace*

Indicates the trace level to record if an error results. The default value is TRACE\_Error. This is an optional parameter.

## Return Value

Returns the workspace ID

## Example

```
Workspace_GetWorkspaceFromWindow(WinNumber())
```

## Related Functions

[Workspace\\_GetContext](#), [Workspace\\_GetEquipmentCluster](#), [Workspace\\_GetPageNameWithRes](#),  
[Workspace\\_GetResolution](#), [Workspace\\_GetSelContext](#), [Workspace\\_GetWindow](#),  
[Workspace\\_GetWorkspaceFromName](#), [Workspace\\_Init](#), [Workspace\\_IsSelContext](#),  
[Workspace\\_IsSystem\\_Initialised](#), [Workspace\\_SelectEquipment](#), [Workspace\\_SetSelContext](#),  
[Workspace\\_ShowContent](#)

## See Also

[Public Functions](#)

### Workspace\_Init

Initializes a top level window as a Workspace.

**Note:** This function must only be called from OnPageShown event. Only top level windows are considered to be a Workspace.

## Syntax

```
Workspace_Init()
```

## Return Value

N/A

## Example

```
Workspace_Init()
```

## Related Functions

[Workspace\\_GetContext](#), [Workspace\\_GetEquipmentCluster](#), [Workspace\\_GetPageNameWithRes](#),  
[Workspace\\_GetResolution](#), [Workspace\\_GetSelContext](#), [Workspace\\_GetWindow](#),  
[Workspace\\_GetWorkspaceFromName](#), [Workspace\\_GetWorkspaceFromWindow](#), [Workspace\\_IsSelContext](#),  
[Workspace\\_IsSystem\\_Initialised](#), [Workspace\\_SelectEquipment](#), [Workspace\\_SetSelContext](#),  
[Workspace\\_ShowContent](#)

## See Also

[Public Functions](#)

## Workspace\_IsSelContext

Checks if an equipment name is the selected context of the workspace on the current page.

## Syntax

```
Workspace_IsSelContext(STRING sEquipmentName, STRING sSecondaryEquipmentName)
```

*sEquipmentName*

Name of a piece of equipment.

*sSecondaryEquipmentName*

The value of the equipment name that is set as the secondary context for related equipment. For example, for a Polar Star Composite Genie, this will be the value specified in the **Equipment Name** option in the Presentation Options dialog box. This is used to determine the number of symbols to be displayed for MEOs.

## Return Value

Returns TRUE if equipment is the selected context. Otherwise returns FALSE.

## Example

```
IF Workspace_IsSelContext(DspAnGetMetadata(KeyGetCursor(),"EquipmentName")) THEN
    Workspace_RightClickEquipment(DspAnGetMetadata(KeyGetCursor(),"EquipmentName"))
END
```

## Related Functions

[Workspace\\_GetContext](#), [Workspace\\_GetEquipmentCluster](#), [Workspace\\_GetPageNameWithRes](#),  
[Workspace\\_GetResolution](#), [Workspace\\_GetSelContext](#), [Workspace\\_GetWindow](#),  
[Workspace\\_GetWorkspaceFromName](#), [Workspace\\_GetWorkspaceFromWindow](#), [Workspace\\_Init](#),  
[Workspace\\_IsSystem\\_Initialised](#), [Workspace\\_SelectEquipment](#), [Workspace\\_SetSelContext](#),  
[Workspace\\_ShowContent](#)

## See Also

[Public Functions](#)

### Workspace\_IsSystemInitialised

Returns whether the workspace system has successfully initialized and it is ready to operate.

## Syntax

```
Workspace_IsSystemInitialised()
```

## Return Value

Returns TRUE when system is initialized and FALSE when system is not initialized.

## Related Functions

[Workspace\\_GetContext](#), [Workspace\\_GetEquipmentCluster](#), [Workspace\\_GetPageNameWithRes](#),  
[Workspace\\_GetResolution](#), [Workspace\\_GetSelContext](#), [Workspace\\_GetWindow](#),  
[Workspace\\_GetWorkspaceFromName](#), [Workspace\\_GetWorkspaceFromWindow](#), [Workspace\\_Init](#),  
[Workspace\\_IsSelContext](#), [Workspace\\_SelectEquipment](#), [Workspace\\_SetSelContext](#), [Workspace\\_ShowContent](#)

## See Also

[Public Functions](#)

### Workspace\_RightClickEquipment

If you have a specific function that needs to be run when the user right-clicks on the selection adorner, you can set the default '\_CALLBACK\_EquipmentRightClick' label to a custom function of your own. The current default function is "DefaultEquipmentRightClickCallback".

The '\_CALLBACK\_EquipmentRightClick' label is set by default in the SA\_Style\_1\_MultiRes starter project. To view labels, navigate to the **Standards** activity and select **Labels**.

	Label Name	Expression
1	_CALLBACK_EquipmentRightClick(sEquip, sEquipSecondary)	DefaultEquipmentRightClickCallback(sEquip, sEquipSecondary)
2	_CALLBACK_FaceplateHeaderDisabled(sEquip, sEquipSecondary)	DefaultFaceplateHeaderDisabledCallback(sEquip, sEquipSecondary)
3	_CALLBACK_OutOfService(sEquip, sEquipSecondary)	DefaultOutOfServiceCallback(sEquip, sEquipSecondary)
4	_THEME_Default	THEME_Blue

## Syntax

Workspace\_RightClickEquipment()

### Returns

None

### Example

```
IF Workspace_IsSelContext("MyPlant.SkimMilk.TNK02", "", TRUE) THEN
    Workspace_RightClickEquipment()
END
```

## See Also

[Public Functions](#)

## Workspace\_SelectEquipment

Changes the context of the workspace to the piece of equipment selected on the current page. This will trigger the automatic filling of content in your workspace panes according to the [Workspace]contextmode that has been configured. See also .

Any panes in your workspace (or linked workspaces) that have been configured as PANE\_FILLMODE\_Autofill, PANE\_FILLMODE\_AutofillContextMustMatch or PANE\_FILLMODE\_StaticContextMustMatch will be updated with the new context and this is available in your page via two super genie associations \_\_EquipmentName and \_\_EquipmentRef. By default, the page from which this function is called will not be updated. If you need the page to be updated, set the *bIncludeCurrentPane* parameter to TRUE when calling this function.

## Syntax

Workspace\_SelectEquipment([INT *bIncludeCurrentPane*])

*bIncludeCurrentPane*

Allow the current page to be automatically filled and context updated. This may result in the page that you clicked on changing. Default = FALSE. This is an optional parameter.

## Return value

N/A

## Example

```
Workspace_SetSelContext("Cluster1.MyPlant.Mixing.TNK01");
Workspace_SelectEquipment();
```

## Related Functions

[Workspace\\_GetContext](#), [Workspace\\_GetEquipmentCluster](#), [Workspace\\_GetPageNameWithRes](#),  
[Workspace\\_GetResolution](#), [Workspace\\_GetSelContext](#), [Workspace\\_GetWindow](#),  
[Workspace\\_GetWorkspaceFromName](#), [Workspace\\_GetWorkspaceFromWindow](#), [Workspace\\_Init](#),  
[Workspace\\_IsSelContext](#), [Workspace\\_IsSystem\\_Initialised](#), [Workspace\\_SetSelContext](#), [Workspace\\_ShowContent](#)

## See Also

[Public Functions](#)

### Workspace\_SetContext

Sets the context of the current workspace to the specified piece of equipment.

## Syntax

```
Workspace_SetContext(STRING sEquipmentName, INT bIncludeCurrentPane, STRING
sEquipmentSecondaryContext, STRING sWorkspaceName)
```

*sEquipmentName*

Name of a piece of equipment.

*bIncludeCurrentPane*

Allow the current page to be automatically filled and context updated. This may result in the page that you clicked on changing. Default = FALSE. This is an optional parameter.

*sWorkspaceName*

Name of the workspace for which to set the context.

*sEquipmentSecondaryContext*

The value of the equipment name that is set as the secondary context for related equipment. For example, for a Polar Star Composite Genie, this will be the value specified in the **Equipment Name** option in the Presentation Options dialog box. This is used to determine the number of symbols to be displayed for MEOs.

## Return Value

N/A

## Related Functions

[Workspace\\_GetContext](#), [Workspace\\_GetEquipmentCluster](#), [Workspace\\_GetPageNameWithRes](#),  
[Workspace\\_GetResolution](#), [Workspace\\_GetSelContext](#), [Workspace\\_GetWindow](#),  
[Workspace\\_GetWorkspaceFromName](#), [Workspace\\_GetWorkspaceFromWindow](#), [Workspace\\_Init](#),  
[Workspace\\_IsSelContext](#), [Workspace\\_IsSystem\\_Initialised](#), [Workspace\\_SelectEquipment](#),  
[Workspace\\_SetSelContext](#), [Workspace\\_ShowContent](#)

## See Also

[Public Functions](#)

### Workspace\_SetSelContext

Sets the selected context of the workspace on the current page to the specified equipment name. This function is only necessary to use when you have excluded a pane configured as PANE\_FILLMODE\_Autofill, PANE\_FILLMODE\_AutofillContextMustMatch or PANE\_FILLMODE\_StaticContextMustMatch from the autofill system.

Typically, this function would be used when building your own custom selection adorner and paired with the [Workspace\\_SelectEquipment\(\)](#) function.

## Syntax

`Workspace_SetSelContext(STRING sEquipmentName, STRING sEquipmentSecondaryContext)`

*sEquipmentName*

Name of a piece of equipment.

*sEquipmentSecondaryContext*

The value of the equipment name that is set as the secondary context for related equipment. For example, for a Polar Star Composite Genie, this will be the value specified in the **Equipment Name** option in the Presentation Options dialog box. This is used to determine the number of symbols to be displayed for MEOs.

## Return Value

N/A

## Example

```
Workspace_SetSelContext(Cluster1.MyPlant.Mixing.TNK01");
Workspace_SelectEquipment();
```

## Related Functions

[Workspace\\_GetContext](#), [Workspace\\_GetEquipmentCluster](#), [Workspace\\_GetPageNameWithRes](#),  
[Workspace\\_GetResolution](#), [Workspace\\_GetSelContext](#), [Workspace\\_GetWindow](#),  
[Workspace\\_GetWorkspaceFromName](#), [Workspace\\_GetWorkspaceFromWindow](#), [Workspace\\_Init](#),

[Workspace\\_IsSelContext](#), [Workspace\\_IsSystem\\_Initialised](#), [Workspace\\_SelectEquipment](#),  
[Workspace\\_ShowContent](#)

## See Also

[Public Functions](#)

### Workspace\_ShowContent

Shows a piece of content (page) in a particular pane of a particular workspace with a specific equipment context.

**Note:** When calling this function in the workspace, any tabs that are added to the tab bar will have the name of the page title and the name of the page itself. For example, If the shelfe form was shown and displayed as content for a tab, the name of the tab would be "Shelve Alarm Form" and not "DefaultShelveForm\_HD1080".

## Syntax

`Workspace_ShowContent(STRING sContentItem [, STRING sPaneRef [, STRING sEquipmentContext [, STRING sEquipmentSecondaryContext [, STRING sWorkspaceName]]]])`

*sContentItem*

The Plant SCADA page to be displayed.

*sPaneRef*

The name of the pane or a pane path reference, in which to display the content. Default is "", which means the pane will be automatically detected based upon the content type of the content and the content types defined on the panes in the workspace(s).

As a Pane may host a page which also has panes, you can use '.' (dot) syntax to refer to them as a pane path reference. For example, Pane.Pane.Pane.

This is an optional parameter.

*sEquipmentContext*

The context to set on the content when it is displayed. Default is "", which will get the context of the workspace. This is an optional parameter.

*sEquipmentSecondaryContext*

The value of the equipment name that is set as the secondary context for related equipment. For example, for a Polar Star Composite Genie, this will be the value specified in the **Equipment Name** option in the Presentation Options dialog box. This is used to determine the number of symbols to be displayed for MEOs. The default value is "".

*sWorkspaceName*

The name of the workspace (a screen name in the active screen profile) in which you want to display your content. Default is "", which is the workspace of the current window and its linked workspaces. This is an optional parameter.

## Return Value

Returns the Plant SCADA Window number (if only one window is updated). Otherwise returns -1.

## Example

```
Workspace_ShowContent("MyPage_HD1080")
```

## Related Functions

[Workspace\\_GetContext](#), [Workspace\\_GetEquipmentCluster](#), [Workspace\\_GetPageNameWithRes](#),  
[Workspace\\_GetResolution](#), [Workspace\\_GetSelContext](#), [Workspace\\_GetWindow](#),  
[Workspace\\_GetWorkspaceFromName](#), [Workspace\\_GetWorkspaceFromWindow](#), [Workspace\\_Init](#),  
[Workspace\\_IsSelContext](#), [Workspace\\_IsSystem\\_Initialised](#), [Workspace\\_SelectEquipment](#),  
[Workspace\\_SetSelContext](#)

## See Also

[Public Functions](#)

## SA\_Include Genie Libraries

The Situational Awareness Include Project (named "SA\_Include") contains the following Genie libraries:

- [Alarm Filter Genie Library](#)
- [Controls Genie Library](#)
- [Common Controls Genie Library](#)
- [Navigation Genie Library](#)
- [Workspace Genie Library](#).

## See Also

[Situational Awareness System Projects](#)

## Alarm Filter Genie Library

The SA\_Include system project includes the "sa\_filter" library. It includes the following Genies:

- [Item Acknowledged Genie \(HD1080\)](#)
- [Item Acknowledged Genie \(UHD4K\)](#)
- [Item Base Genie](#)
- [Item Priority Genie \(HD1080\)](#)
- [Item Priority Genie \(UHD4K\)](#)
- [Item Shelved Genie \(HD1080\)](#)
- [Item Shelved Genie \(UHD4K\)](#)
- [Item State Genie \(HD1080\)](#)
- [Item State Genie \(UHD4K\)](#)

- Item Unacknowledged Genie (HD1080)
- Item Unacknowledged Genie (UHD4K).

These Genie are designed to provide filtering functionality for the Situational Awareness Default Alarm Pages. For more information, see [Add an Alarm Filter Button](#).

## See Also

[Situational Awareness System Projects](#)

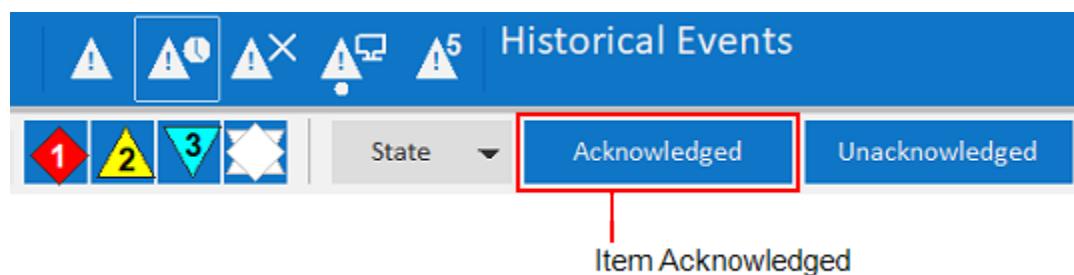
### Item Acknowledged Genie (HD1080)

**Genie Name:** item\_ack\_hd1080

**Genie Library:** sa\_filter

**System Project:** SA\_Include

The HD1080 version of the Item Acknowledged Genie (named "item\_ack\_hd1080") is used to create a button that filters an alarms list so that only acknowledged alarms will appear.



This Genie is designed for use with the Situational Awareness [Default Alarm Pages](#).

For more information, see [Add an Alarm Filter Button](#).

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
FilterName	The name "TopBar" is entered by default. This is the name of the filter used by the Situational Awareness default alarm pages.
Tooltip	An optional description of the Genie that appears when the mouse points at it.

## See also

[Configure a Genie](#)

[Item Acknowledged Genie \(UHD4K\)](#)

[Alarm Filter Genie Library](#)

### Item Acknowledged Genie (UHD4K)

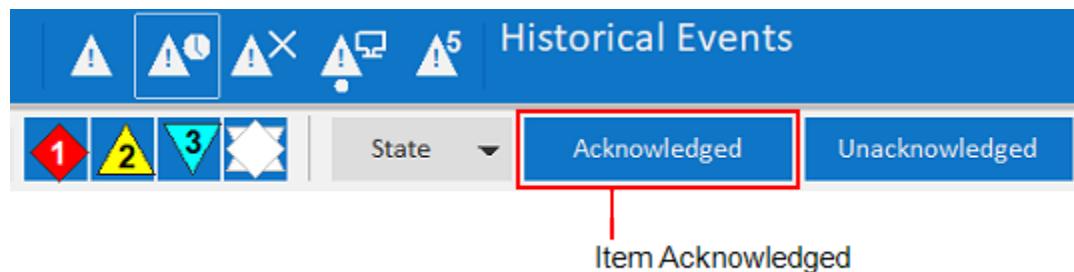
**Genie Name:** item\_ack\_uhd4k

**Genie Library:** sa\_filter

**System Project:** SA\_Include

The UHD4K version of the Item Acknowledged Genie (named "item\_ack\_uhd4k") is used to create a button that filters an alarms list so that only acknowledged alarms will appear.

This Genie is designed for use with the Situational Awareness [Default Alarm Pages](#).



For more information, see [Add an Alarm Filter Button](#).

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
FilterName	The name "TopBar" is entered by default. This is the name of the filter used by the Situational Awareness default alarm pages.
Tooltip	An optional description of the Genie that appears when the mouse points at it.

## See Also

[Configure a Genie](#)

[Item Acknowledged Genie \(HD1080\)](#)

[Alarm Filter Genie Library](#)

## Item Base Genie

**Genie Name:** item\_base

**Genie Library:** sa\_filter

**System Project:** SA\_Include

The Item Base Genie (named "item\_base") is used to create a customized alarm filter button. This Genie is designed for use with the Situational Awareness [Default Alarm Pages](#).

For more information, see [Add an Alarm Filter Button](#).

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
IsExclusion	Determines if the results of an applied filter are excluded or included in the list that is displayed.  TRUE = Filter results are excluded. For example, if the value of the filter is "CATEGORY <> 3", when the filter is applied category 3 alarms will be excluded.  FALSE = Filter results are included. For example, if the value of the filter is "CATEGORY=3", only category 3

Parameter	Description
	alarms will be included in the list.
FilterExpr	<p>Enter the Cicode expression that returns the name of a filter. To use the default filter name that is applied to Situational Awareness alarm pages, enter the following:</p> <pre>AlarmPage_GetFilter("TopBar")</pre> <p>To use a custom filter, you need to create the required Cicode as on OnPageEntry event. You can then call the filter name.</p>
ValueExpr	Enter the Cicode expression that returns the value of the filter string. For example, "CATEGORY<>3".
Tooltip	An optional description of the Genie that appears when the mouse points at it.
Label	Enter the text that will appear in the header row of the alarms list when the filter is applied. For example, "No Cat 3 Alarms".

## See Also

[Configure a Genie](#)

[Alarm Filter Genie Library](#)

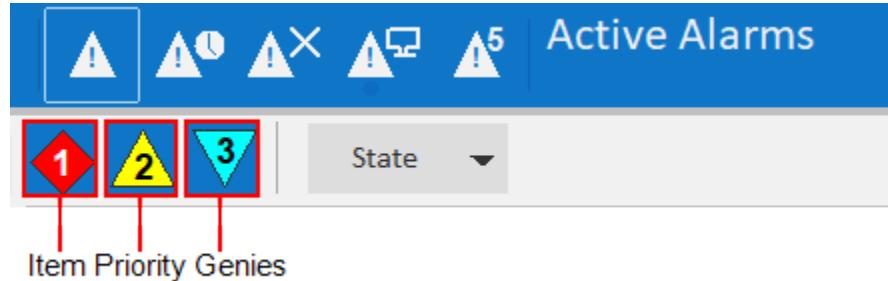
## Item Priority Genie (HD1080)

**Genie Name:** item\_priority\_hd1080

**Genie Library:** sa\_filter

**System Project:** SA\_Include

The HD1080 version of the Item Priority Genie (named "item\_priority\_hd1080") is used to create a button that filters an alarms list so that only alarms of a specified priority will appear.



This Genie is designed for use with the Situational Awareness Default Alarm Pages.

For more information, see [Add an Alarm Filter Button](#).

**Note:** You can use this Genie to add a fourth-highest alarm priority filter to alarm pages. See [Add a Fourth Alarm Priority Filter to Alarm Pages.](#)

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
FilterName	The name "TopBar" is entered by default. This is the name of the filter used by the Situational Awareness default alarm pages.
AsmPriority	Enter 1, 2, 3 or 4 to filter an alarms list using one of the four highest Situational Awareness alarm priorities (that is, the alarm priorities with the lowest values).
Tooltip	An optional description of the Genie that appears when the mouse points at it.

## See Also

[Configure a Genie](#)

[Item Priority Genie \(UHD4K\)](#)

[Alarm Filter Genie Library](#)

## Item Priority Genie (UHD4K)

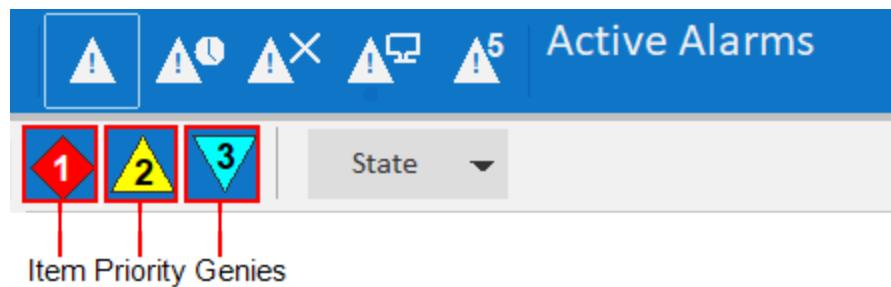
**Genie Name:** item\_priority\_uhd4k

**Genie Library:** sa\_filter

**System Project:** SA\_Include

The UHD4K version of the Item Priority Genie (named "item\_priority\_uhd4k") is used to create a button that filters an alarms list so that only alarms of a specified priority will appear.

This Genie is designed for use with the Situational Awareness [Default Alarm Pages](#).



For more information, see [Add an Alarm Filter Button](#).

**Note:** You can use this Genie to add a fourth-highest alarm priority filter to alarm pages. See [Add a Fourth Alarm Priority Filter to Alarm Pages](#).

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
FilterName	The name "TopBar" is entered by default. This is the name of the filter used by the Situational Awareness default alarm pages.
AsmPriority	Enter 1, 2, 3 or 4 to filter an alarms list using one of the four highest Situational Awareness alarm priorities (that is, the alarm priorities with the lowest values).
Tooltip	An optional description of the Genie that appears when the mouse points at it.

## See Also

[Configure a Genie](#)

[Item Priority Genie \(HD1080\)](#)

[Alarm Filter Genie Library](#)

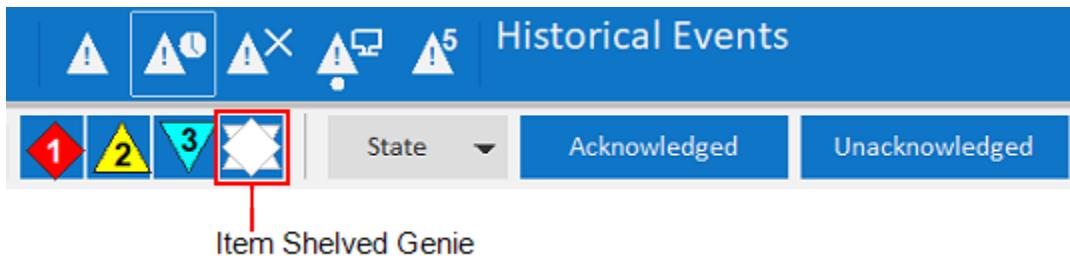
## Item Shelved Genie (HD1080)

**Genie Name:** item\_shelved\_hd1080

**Genie Library:** sa\_filter

**System Project:** SA\_Include

The HD1080 version of the Item Shelved Genie (named "item\_shelved\_hd1080") is used to create a button that filters an alarms list so that only shelved alarms will appear.



This Genie is designed for use with the Situational Awareness [Default Alarm Pages](#).

For more information, see [Add an Alarm Filter Button](#).

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- AN:** This is an automatically-generated unique ID for the Genie object.
- Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
FilterName	The name "TopBar" is entered by default. This is the name of the filter used by the Situational Awareness default alarm pages.
Tooltip	An optional description of the Genie that appears when the mouse points at it.

## See Also

[Configure a Genie](#)

[Item Shelved Genie \(UHD4K\)](#)

[Alarm Filter Genie Library](#)

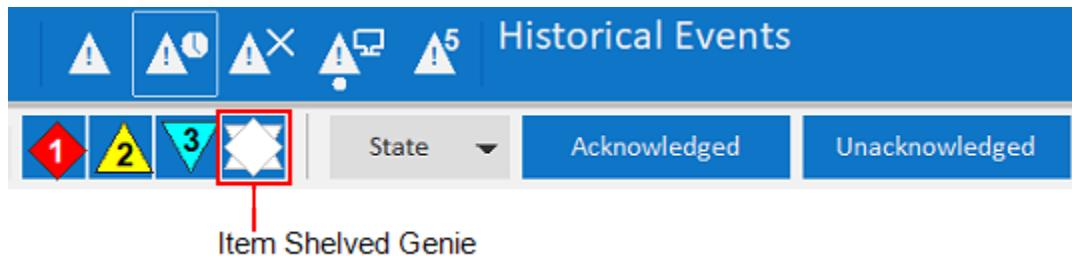
## Item Shelved Genie (UHD4K)

**Genie Name:** item\_shelved\_uhd4k

**Genie Library:** sa\_filter

**System Project:** SA\_Include

The UHD4K version of the Item Shelved Genie (named "item\_shelved\_uhd4k") is used to create a button that filters an alarms list so that only shelved alarms will appear.



This Genie is designed for use with the Situational Awareness [Default Alarm Pages](#).

For more information, see [Add an Alarm Filter Button](#).

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
FilterName	The name "TopBar" is entered by default. This is the name of the filter used by the Situational Awareness default alarm pages.
Tooltip	An optional description of the Genie that appears when the mouse points at it.

## See Also

[Configure a Genie](#)

[Item Shelved Genie \(HD1080\)](#)

[Alarm Filter Genie Library](#)

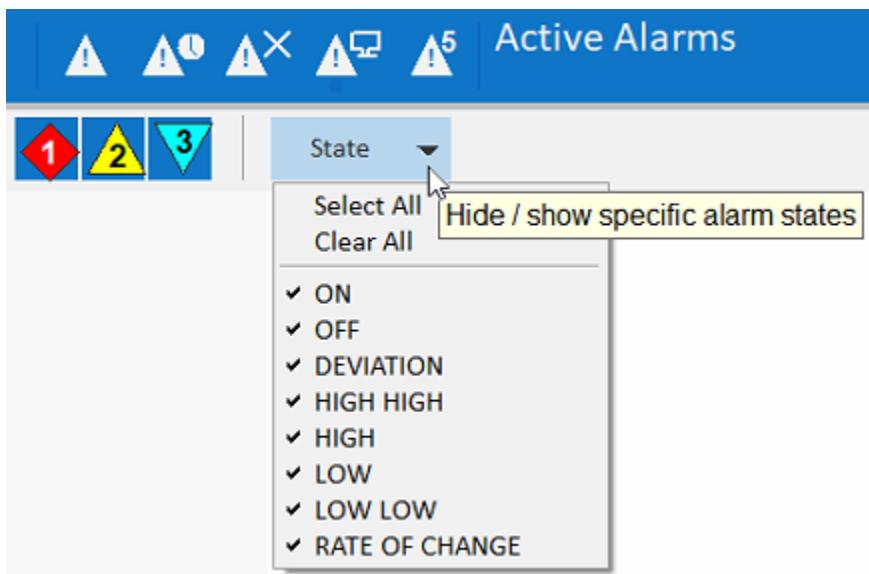
## Item State Genie (HD1080)

**Genie Name:** item\_state\_hd1080

**Genie Library:** sa\_filter

**System Project:** SA\_Include

The HD1080 version of the Item State Genie (named "item\_state\_hd1080") is used to create a button that filters an alarms list so that only alarms in a selected state will appear. When you click on the button, a check list of alarm states appears.



This Genie is designed for use with the Situational Awareness [Default Alarm Pages](#).

For more information, see [Add an Alarm Filter Button](#).

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
FilterName	The name "TopBar" is entered by default. This is the name of the filter used by the Situational Awareness default alarm pages.
Tooltip	An optional description of the Genie that appears when the mouse points at it.

## See Also

[Configure a Genie](#)

[Item Shelved Genie \(UHD4K\)](#)

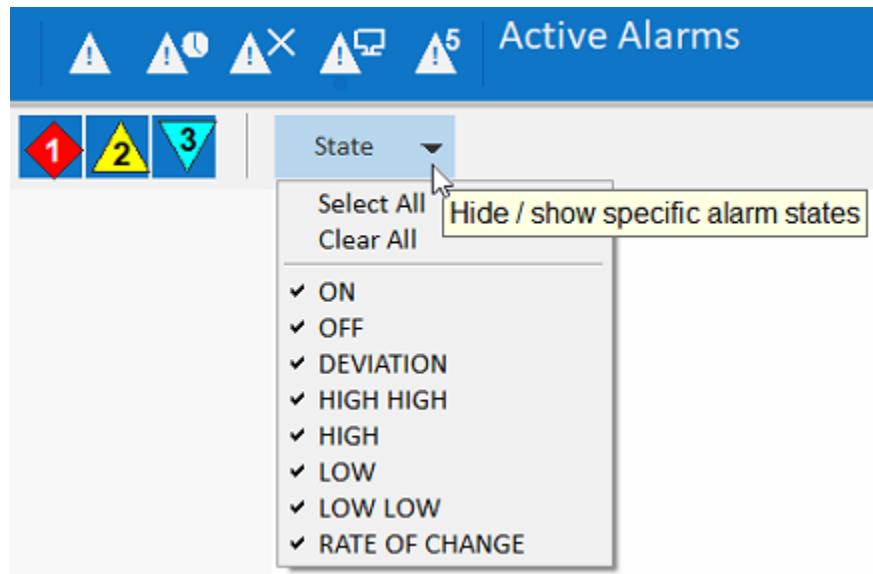
### Item State Genie (UHD4K)

**Genie Name:** item\_state\_uhd4k

**Genie Library:** sa\_filter

**System Project:** SA\_Include

The UHD4K version of the Item State Genie (named "item\_state\_uhd4k") is used to create a button that filters an alarms list so that only alarms in a selected state will appear. When you click on the button, a check list of alarm states appears.



This Genie is designed for use with the Situational Awareness [Default Alarm Pages](#).

For more information, see [Add an Alarm Filter Button](#).

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
FilterName	The name "TopBar" is entered by default. This is the name of the filter used by the Situational Awareness default alarm pages.
Tooltip	An optional description of the Genie that appears when the mouse points at it.

## See Also

[Configure a Genie](#)

[Item Shelved Genie \(HD1080\)](#)

[Alarm Filter Genie Library](#)

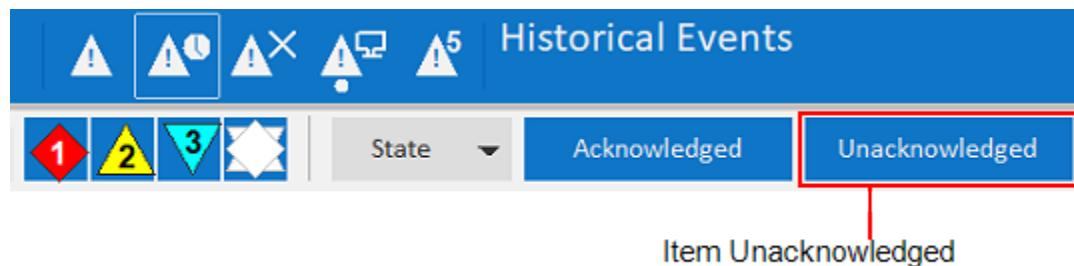
## Item Unacknowledged Genie (HD1080)

**Genie Name:** item\_unack\_hd1080

**Genie Library:** sa\_filter

**System Project:** SA\_Include

The HD1080 version of the Item Unacknowledged Genie (named "item\_unack\_hd1080") is used to create a button that filters an alarms list so that only unacknowledged alarms will appear.



This Genie is designed for use with the Situational Awareness [Default Alarm Pages](#).

For more information, see [Add an Alarm Filter Button](#).

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
FilterName	The name "TopBar" is entered by default. This is the name of the filter used by the Situational Awareness default alarm pages.
Tooltip	An optional description of the Genie that appears when the mouse points at it.

## See Also

[Configure a Genie](#)

[Item Unacknowledged Genie \(UHD4K\)](#)

[Navigation Genie Library](#)

### Item Unacknowledged Genie (UHD4K)

**Genie Name:** item\_unack\_uhd4k

**Genie Library:** sa\_filter

**System Project:** SA\_Include

The UHD4K version of the Item Unacknowledged Genie (named "item\_unack\_uhd4k") is used to create a button that filters an alarms list so that only unacknowledged alarms will appear.



This Genie is designed for use with the Situational Awareness [Default Alarm Pages](#).

For more information, see [Add an Alarm Filter Button](#).

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
FilterName	The name "TopBar" is entered by default. This is the name of the filter used by the Situational Awareness default alarm pages.
Tooltip	An optional description of the Genie that appears when the mouse points at it.

## See Also

[Configure a Genie](#)

[Item Unacknowledged Genie \(HD1080\)](#)

[Navigation Genie Library](#)

## Controls Genie Library

The SA\_Include system project includes the "sa\_controls" library. It includes the following Genies:

- [Navigation Zone Button Genie \(HD1080\)](#)
- [Navigation Zone Button Genie \(UHD4K\)](#)
- [Navigation Zone - 4 Priority Button Genie \(HD1080\)](#)
- [Navigation Zone - 4 Priority Button Genie \(UHD4K\)](#)
- [Navigation Zone Tab Genie \(HD1080\)](#)
- [Navigation Zone Tab Genie \(UHD4K\)](#)
- [Navigation Zone - 4 Priority Tab Genie \(HD1080\)](#)
- [Navigation Zone - 4 Priority Tab Genie \(UHD4K\)](#)
- [Tab Bar Genie](#)
- [Tree View Genie](#)
- [Tree View Item Genie.](#)

---

**Note:** The **SA\_Controls** system project also contains a library named "sa\_controls". For a list of the Genies it includes, see [Controls Genie Library](#).

---

## See Also

[Situational Awareness System Projects](#)

## Navigation Zone Button Genie (HD1080)

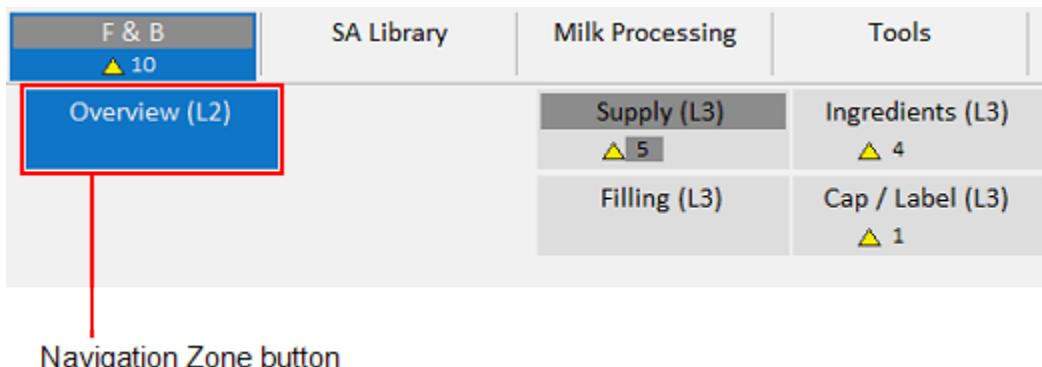
**Genie Name:** button\_navigation\_hd1080

**Genie Library:** sa\_controls

**System Project:** SA\_Include

The HD1080 version of the Navigation Zone Button Genie (named "button\_navigation\_hd1080") is used to create

the buttons that appear in the [Navigation Zone](#) on the Situational Awareness Operator Dashboard.



## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- AN:** This is an automatically-generated unique ID for the Genie object.
- Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
Left Click Function	Enter a function (no argument requirements) that is called when the operator clicks on the button.
Disabled When	You can enter an expression that provides logic to disable the button.

## See Also

[Configure a Genie](#)

[Navigation Zone Button Genie \(UHD4K\)](#)

[Navigation Genie Library](#)

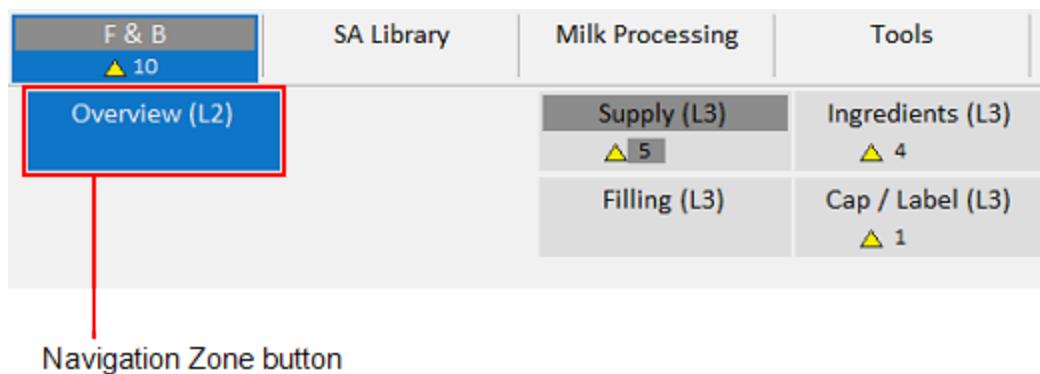
## Navigation Zone Button Genie (UHD4K)

**Genie Name:** button\_navigation\_uhd4k

**Genie Library:** sa\_controls

**System Project:** SA\_Include

The UHD4K version of the Navigation Zone Button Genie (named "button\_navigation\_uhd4k") is used to create the buttons that appear in the [Navigation Zone](#) on the Situational Awareness Operator Dashboard.



## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- AN:** This is an automatically-generated unique ID for the Genie object.
- Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
Left Click Function	Enter a function (no argument requirements) that is called when the operator clicks on the button.
Disabled When	Enter an expression that provides logic to disable the button.

## See Also

[Configure a Genie](#)

[Navigation Zone Button Genie \(HD1080\)](#)

[Navigation Genie Library](#)

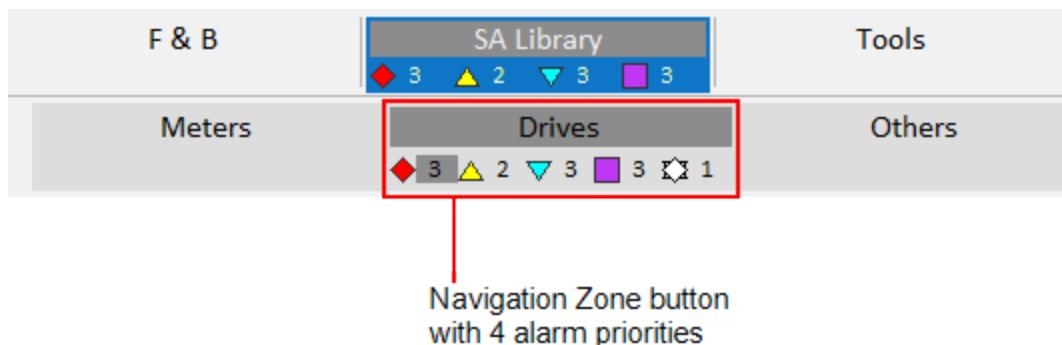
## Navigation Zone - 4 Priority Button Genie (HD1080)

**Genie Name:** button\_nav4\_hd1080

**Genie Library:** sa\_controls

**System Project:** SA\_Include

The HD1080 version of the Navigation Zone - 4 Priority Button Genie (named "button\_nav4\_hd1080") is used to create the buttons that appear in the [Navigation Zone](#) on the Situational Awareness Operator Dashboard. It is specifically used when four alarm priorities are displayed on the button.



For more information, see [Add an Additional Alarm Count to the Navigation Zone](#).

## See Also

[Configure a Genie](#)

[Navigation Zone - 4 Priority Button Genie \(UHD4K\)](#)

[Navigation Zone Button Genie \(HD1080\)](#)

[Navigation Genie Library](#)

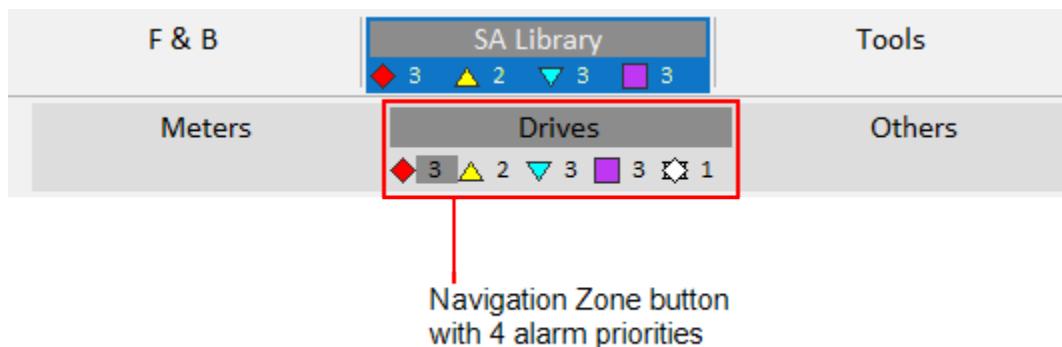
### Navigation Zone - 4 Priority Button Genie (UHD4K)

**Genie Name:** button\_nav4\_uhd4k

**Genie Library:** sa\_controls

**System Project:** SA\_Include

The UHD4K version of the Navigation Zone - 4 Priority Button Genie (named "button\_nav4\_uhd4k") is used to create the buttons that appear in the Navigation Zone on the Situational Awareness Operator Dashboard. It is specifically used when four alarm priorities are displayed on the button.



For more information, see [Add an Additional Alarm Count to the Navigation Zone](#).

## See Also

[Configure a Genie](#)

[Navigation Zone - 4 Priority Button Genie \(HD1080\)](#)

[Navigation Zone Button Genie \(UHD4K\)](#)

[Navigation Genie Library](#)

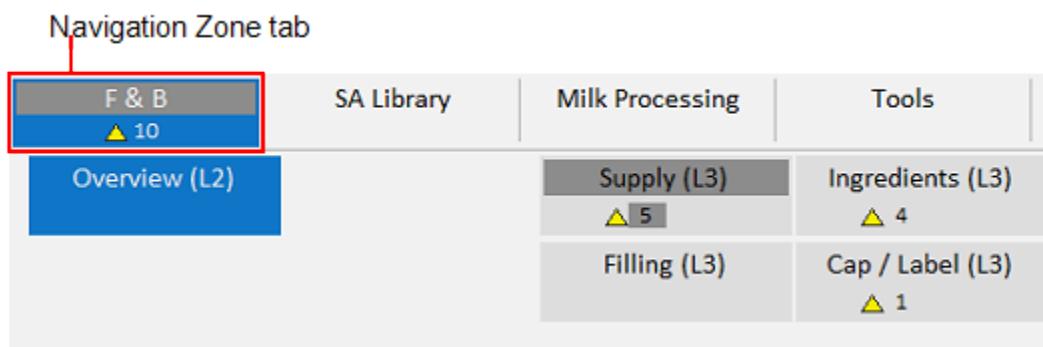
## Navigation Zone Tab Genie (HD1080)

**Genie Name:** tab\_navigation\_hd1080

**Genie Library:** sa\_controls

**System Project:** SA\_Include

The HD1080 version of the Navigation Zone Tab Genie (named "tab\_navigation\_hd1080") is used to create the tabs that appear in the Navigation Zone on the Situational Awareness Operator Dashboard.



## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
Left Click Function	Enter a function (no arguments required) that is called when the operator clicks on the tab.
Disabled When	Enter an expression that provides logic to disable the button.

## See also

[Configure a Genie](#)

[Navigation Zone Tab Genie \(UHD4K\)](#)

## Navigation Genie Library

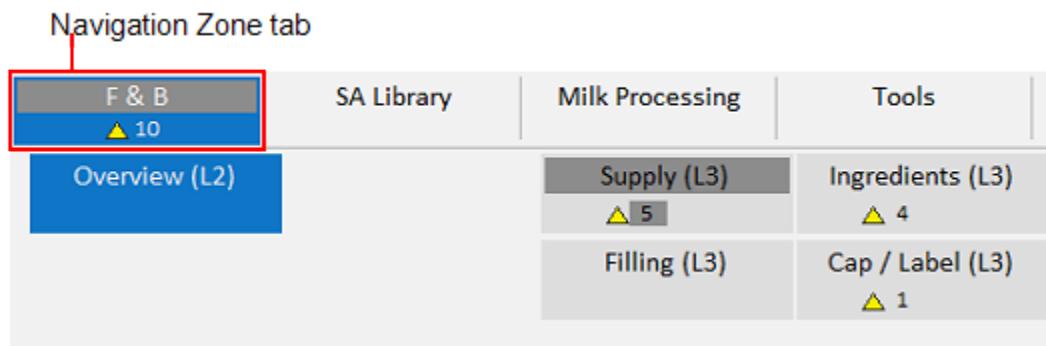
### Navigation Zone Tab Genie (UHD4K)

**Genie Name:** tab\_navigation\_uhd4k

**Genie Library:** sa\_controls

**System Project:** SA\_Include

The UHD4K version of the Navigation Zone Tab Genie (named "tab\_navigation\_uhd4k") is used to create the tabs that appear in the [Navigation Zone](#) on the Situational Awareness Operator Dashboard.



### Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- AN:** This is an automatically-generated unique ID for the Genie object.
- Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

### Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
Left Click Function	Enter a function (no arguments required) that is called when the operator clicks on the tab.
Disabled When	Enter an expression that provides logic to disable the button.

### See Also

[Configure a Genie](#)

[Navigation Zone Tab Genie \(HD1080\)](#)

[Navigation Genie Library](#)

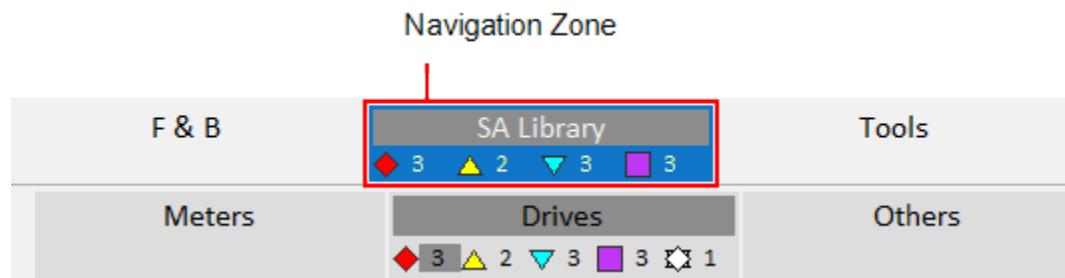
## Navigation Zone - 4 Priority Tab Genie (HD1080)

**Genie Name:** tab\_nav4\_hd1080

**Genie Library:** sa\_controls

**System Project:** SA\_Include

The HD1080 version of the Navigation Zone - 4 Priority Tab Genie (named "tab\_nav4\_hd1080") is used to create the tabs that appear in the [Navigation Zone](#) on the Situational Awareness Operator Dashboard. It is specifically used when four alarm priorities are displayed on the tab.



For more information, see [Add an Additional Alarm Count to the Navigation Zone](#).

## See Also

[Configure a Genie](#)

[Navigation Zone Tab Genie \(UHD4K\)](#)

[Navigation Genie Library](#)

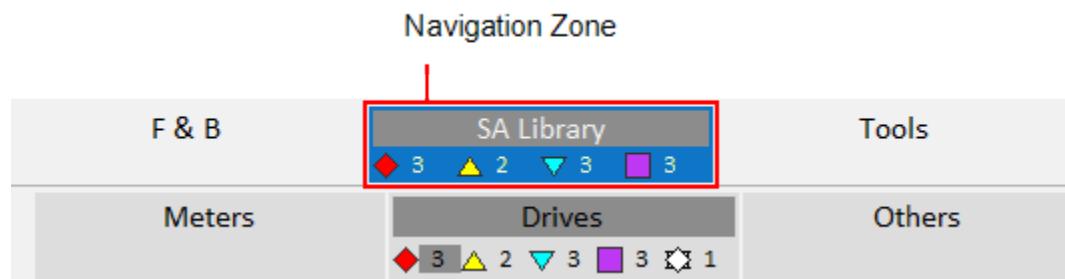
## Navigation Zone - 4 Priority Tab Genie (UHD4K)

**Genie Name:** tab\_nav4\_uhd4k

**Genie Library:** sa\_controls

**System Project:** SA\_Include

The UHD4K version of the Navigation Zone - 4 Priority Tab Genie (named "tab\_nav4\_uhd4k") is used to create the tabs that appear in the [Navigation Zone](#) on the Situational Awareness Operator Dashboard. It is specifically used when four alarm priorities are displayed on the tab.



For more information, see [Add an Additional Alarm Count to the Navigation Zone](#).

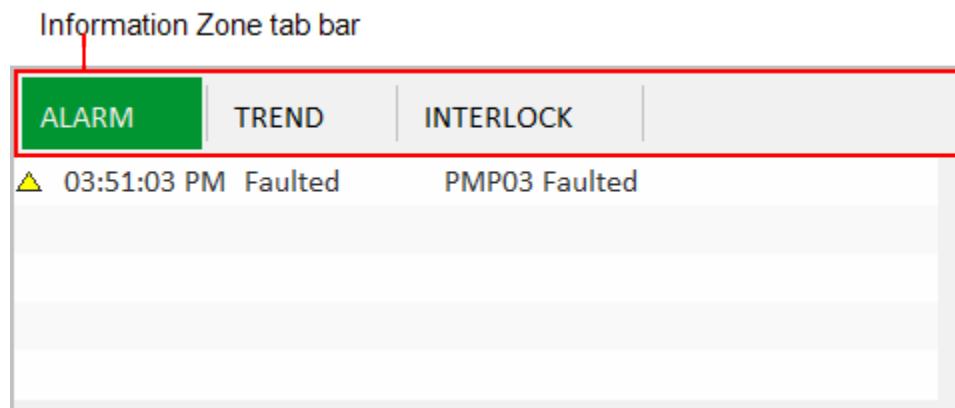
## See Also

- [Configure a Genie](#)
- [Navigation Zone Tab Genie \(HD1080\)](#)
- [Navigation Genie Library](#)

### Tab Bar Genie

**Genie Name:** tabbar\_h  
**Genie Library:** sa\_controls  
**System Project:** SA\_Include

The Tab Bar Genie (named "tabbar\_h") forms the basis for the tabs that appear in the [Information Zone](#) on the Situational Awareness Operator Dashboard.



You can also use it to build a set of tabs for a pane on a customized master page (see [Create a Set of Tabs for a Pane](#)).

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
Default Tabs Menu Name	Enter the name of the menu you configured to define the tabs that will display.
OnInit Function Name	If required, specify a Cicode function that is executed

Parameter	Description
	when the tab bar initialization has completed.
Can Tabs be Pinned?	Specify if an operator can pin a tab so that it remains displayed. TRUE = Pinning is enabled FALSE = Pinning is not enabled.
Max Open Items	If required, sets a limit on the number of menu items that are included on the tab bar at runtime.  The menu items are selected according to the value set for <b>Order</b> property. If the Order property is not configured, items are selected according to the order in which they were added to the Menu Configuration database.
Tab Bar Width	Specify the width of the tab bar in pixels.
Max Tab Width	Specify the maximum width for each tab in pixels.
Use for 4K Page?	Specify if the tab bar will appear on a UHD4K master page.  TRUE = The tab bar will be used in a UHD4K project. FALSE = The tab bar will not be used in a UHD4K project.
Allow Duplicate Content?	Determines if multiple tabs can display the same content.  TRUE = Multiple tabs can display the same page. FALSE = Each tab displays a different page (default).  For example, you could set this to TRUE if you need two tabs to display the same page, but with a different filter applied.

## See Also

[Configure a Genie](#)

[Navigation Zone Tab Genie \(HD1080\)](#)

[Navigation Zone Tab Genie \(UHD4K\)](#)

[Navigation Genie Library](#)

## Tree View Genie

**Genie Name:** treeview

**Genie Library:** sa\_controls

**System Project:** SA\_Include

The Tree View Genie (named "treeview") can be used to display a menu structure on a page. For example, you can use it to:

- Display an equipment hierarchy
- Filter an alarm page
- Navigate pages
- Display alarm counts.

The screenshot shows the 'Active Alarms' page interface. At the top, there are navigation icons (New, Open, Search, Close) and a status bar with 'State'. Below the header, a table titled 'Current Filter: Active Alarms of selected equipment' lists alarms by Date, Time, and Name. On the left, a tree view titled 'Alarms Filtered' shows clusters: Cluster1 (selected, checked), Cluster2, and Cluster3. Each cluster has a checkbox and alarm counts (red diamond for severity 2, yellow triangle for severity 1, blue inverted triangle for severity 1). A red box highlights the tree view area. A red arrow points from the caption 'The tree view on the Active Alarms page' to the tree view section.

Date	Time	Name
25/01/2018	10:56:58 AM	Drive1_V_Feedback_H_P3
25/01/2018	10:56:55 AM	Drive1_OP_TRK_H_P2
25/01/2018	10:56:48 AM	Drive1_V_OP_HH_P1
25/01/2018	10:56:52 AM	Drive1_Running_False_P1

The tree view on the Active Alarms page

For an example of how to use the Tree View Genie, see [Add a Tree View to a Page](#).

**Note:** You can use the parameter [\[Workspace\]NumberOfTopPriorities](#) to display alarm counts for the top four alarm priorities (instead of the top three). The Genie "Priority 4 Small Genie" is provided in the SA\_Controls project to represent the fourth-highest priority. To associate this Genie with an alarm priority, see [Configure Display Properties for an Alarm Priority](#).

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

### Basic Setup Parameters

Parameter	Description
Datasource	Specifies the menu on which the tree is based. You can enter the name of a menu you have created using the <b>Menu Configuration</b> view in the <b>Visualization</b> activity (see <a href="#">Prepare the Navigation</a> )

Parameter	Description
	<p><a href="#">Menu</a>).</p> <p>Alternatively, you can use the name of a menu that is created at runtime using the <a href="#">MenuGetPageNode</a> Cicode function.</p> <p>You can also enter "__EquipmentModel" to use the equipment hierarchy defined for the current project.</p>
Use Display Names	<p>Determines if display names are used to label equipment in the tree view.</p> <p>A display name is a meaningful name that can be defined for piece of equipment using the Display Name property (see <a href="#">Define Equipment in Plant SCADA Studio</a>).</p> <p>If the tree view is displaying the equipment model:</p> <p>TRUE = Equipment display names are used</p> <p>FALSE = Equipment names are used.</p> <p>If this is set to TRUE and no Display Name is configured, the equipment name will be used instead.</p> <p>If the tree view is displaying a generic menu structure:</p> <p>TRUE = Menu item comments are used</p> <p>FALSE = Menu item names are used.</p>

#### Search Parameters

Parameter	Description
Search Box Name	<p>This field specifies the name of the ciText.TextBox object that the tree view will use as its input for the search text at runtime. The name to enter is defined in the Text Object Properties dialog on the <b>Access</b> tab.</p> <p>By default, "Search" is used.</p>

#### Display Parameters

Parameter	Description
Treeview Height	<p>The height of the area the tree view will occupy (in pixels).</p> <p>If the length of the expanded tree extends beyond the height specified here, a vertical scroll bar will appear.</p>
Treeview Width	<p>The width of the area the tree view will occupy (in pixels).</p> <p>If the width of the expanded tree extends beyond the left edge of the alarm count list, a horizontal scroll bar</p>

Parameter	Description
	will appear.
Row Height	The amount of space that will be available to each row within the tree view.
Display Checkboxes	<p>Determines if a check box is displayed for each item in the tree.</p> <p>TRUE = Check boxes are displayed.</p> <p>FALSE = Check boxes are not displayed.</p>
Auto-check Children	<p>Determines if the check boxes within the lower levels of a branch are automatically checked when a parent item is checked.</p> <p>TRUE = Children are automatically checked.</p> <p>FALSE = Children are not automatically checked.</p>
Display Alarms	<p>Determines if alarm counts for the top three alarm categories are displayed for each item in the tree.</p> <p>TRUE = Alarm counts for the top three alarm categories are displayed.</p> <p>FALSE = Alarm counts for the top three alarm categories are not displayed.</p> <p>If the branch is collapsed, the values will represent the number of active alarms for each category within the current branch and its lower levels.</p> <p>If a node is expanded, the count will only shows alarms for that node, as the counts for its children will now be visible on the child nodes.</p> <p><b>Note:</b> You can use the parameter <a href="#">[Workspace]NumberOfTopPriorities</a> to display alarm counts for the top four alarm priorities (instead of the top three).</p>
Display Shelved	<p>Determines if an alarm count for shelved alarms is displayed for each item in the tree.</p> <p>If the branch is collapsed, the values will represent the number of shelved alarms for each category within the current branch and its lower levels.</p> <p>If a node is expanded, the count will only shows shelved alarms for that node, as the counts for its children will now be visible on the child nodes.</p> <p>TRUE = Alarm count for shelved alarms is displayed.</p> <p>FALSE = Alarm count for shelved alarms is not displayed.</p>

**Event Parameters**

<b>Parameter</b>	<b>Description</b>
On Init Complete Function	Represents the name of the function that is invoked when the tree view completes initialization.  <code>OnInitCompleteFunction(INT nTreeviewAN)</code> The argument <i>nTreeviewAN</i> is expected, it is the animation number of the tree view that fires the event.
On Check Function	Represents the name of the function that is invoked upon the checked event.  <code>OnCheckFunction(INT hMenuItem)</code> The argument <i>hMenuItem</i> is expected. It is the handle to the menu item that is represented by the tree node on which the check box has just been clicked.
On Left Click Function	Represents the name of the function that is invoked upon the left click event.  <code>OnLeftClickFunction(INT hMenuItem)</code> The argument <i>hMenuItem</i> is expected. It is the handle to the menu item that is represented by the tree node that has just been left-clicked.
On Right Click Function	Represents the name of the function that is invoked upon the right click event.  <code>OnRightClickFunction(INT hMenuItem)</code> The argument <i>hMenuItem</i> is expected. It is the handle to the menu item that is represented by the tree node that has just been right-clicked.

**Customization Parameters**

<b>Parameter</b>	<b>Description</b>
View Genie	Refers to the Genie that is called to display each item in the tree. By default, the <a href="#">Tree View Item Genie</a> in the "sa_controls" library is used ( <code>sa_controls.treeviewitem</code> ).  You can use this parameter to substitute the default Genie with your own. This will allow you to customize the appearance of the tree and the features it supports.
View Init Function	The Cicode function that is called to initialize the <b>View Genie</b> (see above) for each row.  By default, a system function called

Parameter	Description
	" <code>_Treeview_InitViewGenie_&lt;resolution&gt;</code> " is used (for example, " <code>_Treeview_InitViewGenie_HD1080</code> ").
Fill Function	<p>This function is called on each page scan to manage data updates in the tree view when the tree view state changes (for example, when scrolling, expanding, or collapsing occurs).</p> <p>By default, a system function called "<code>_Treeview_Update</code>" is used.</p>
View Model Data Size	<p>Specifies the number of custom fields per tree view item to be allocated for user data. The data in these fields will be updated by the function specified in the parameter <b>Datasource Item Retrieve Function</b> (see below).</p> <p>This parameter is set to zero (0) by default.</p>
View Model Init Function	<p>Custom function to configure the additional columns specified by the parameter 'View Model Data Size'. This function is called once when the tree view is initialized.</p>
Datasource Item Retrieve Function	<p>A custom function that loads additional user data for the tree view. This function is called whenever the tree view data is refreshed (for example, when scrolling, expanding, or collapsing occurs). The number of custom fields to be updated by this function is defined by the parameter <b>View Model Data Size</b> (see above).</p>
Is Source Ready Function	<p>Enter the name a function that is used to determine if the data source is ready.</p> <p>The tree view initialization code will run the function (in addition to the checks for navigation initialized and workspace system initialized) before allowing the tree view to be displayed.</p> <p>If this function does not return TRUE within five seconds of displaying the page, the tree view initialization will time out. This function needs to return an INT (TRUE or FALSE) and takes a single argument, INT nTreeviewAN.</p>
Datasource Watcher Function	<p>Enter a Cicode function that will be used to monitor the tree view's associated data source for any updates. The function will be called when the Tree View Genie is initialized, and needs to be responsible</p>

Parameter	Description
	<p>for triggering a reload if required.</p> <p>The function will run continuously, so it should not return any values. It also needs to accept the arguments <i>sDatasource</i> and <i>hTreeviewAN</i>.</p> <p>If you are using the Tree View Genie to display the entire equipment model (for example, if the <b>Datasource</b> parameter is set to "<code>__EquipmentModel</code>"), you can enter the following function name to trigger a reload of the tree whenever the equipment model changes:</p> <p><code>_Treeview_MonitorEquipmentModel</code></p>
Can Check Items Function	<p>A custom function that is used to determine if tree view item checking is allowed.</p> <p>By default, toggling the checked state of a tree view item is allowed at runtime. This function provides a way to disable the check boxes on all items if required.</p> <p>For example, you may want to disable checking if the data that the tree view will interact with is not ready, or if the current user does not have sufficient privileges.</p> <p>The function needs to return a value of TRUE or FALSE.</p>

#### AN Parameter

Parameter	Description
AN	A unique ID for the Genie object.

#### See Also

- [Configure a Genie](#)
- [Tree View Item Genie](#)
- [SA\\_Include Genie Libraries](#)

#### Tree View Item Genie

**Genie Name:** treeviewitem / treeviewitem\_uhd4k

**Genie Library:** sa\_controls

**System Project:** SA\_Include

The Tree View Item Genie is used by the [Tree View Genie](#) to build the required number of items displayed in a tree.

Two versions of the Genie exist:

- "treeviewitem" — designed for use with a 1080HD workspace
- "treeviewitem\_uhd4k" — designed for use with a UHD4K workspace.

For more information, see [Add a Tree View to a Page](#).

**Note:** The Tree View Item Genie is displayed dynamically at runtime by the Tree View Genie. There is no need to use this Genie directly on a page.

---

## See Also

[Configure a Genie](#)

[SA\\_Include Genie Libraries](#)

## Common Controls Genie Library

The SA\_Include system project includes the "sa\_controls\_common" library. It includes the following Genies:

- [Tab Base Genie](#)
- [Tab Genie](#).

**Note:** The **SA\_Controls** system project also contains a library named "sa\_controls\_common". For a list of the Genies it includes, see [Common Controls Genie Library](#).

---

## See Also

[Situational Awareness System Projects](#)

### Tab Base Genie

**Genie Name:** tab\_base

**Genie Library:** sa\_controls\_common

**System Project:** SA\_Include

The Tab Base Genie (named "tab\_base") is a component of the [Tab Bar Genie](#).

See [Create a Set of Tabs for a Pane](#).

## Genie Properties

Complete the following properties for the genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
Left Click Function	Enter a function (no arguments required) that is called when the operator clicks on the tab.
Disabled When	An expression (applied in conjunction with Area and Privilege) that can provide further logic to specify when the tab is disabled.
Tooltip	Text that displays when the operator hovers the mouse over the button at runtime.
Area	Enter the area to which the button belongs. Only users with access to this area can use the button.
Privilege	Enter the privilege level that a user needs to possess to use this button.

## See Also

[Configure a Genie](#)

[Tab Bar Genie](#)

[SA\\_Include Genie Libraries](#)

## Tab Genie

**Genie Name:** tab\_h

**Genie Library:** sa\_controls\_common

**System Project:** SA\_Include

The Tab Genie (named "tab\_h") is a component of the [Tab Bar Genie](#). It is designed to work with the Workspace.

See [Create a Set of Tabs for a Pane](#).

## Genie Properties

Complete the following properties for the genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
TabID	A unique number identifying this tab. Each tab in your tab bar should have a unique number assigned to it.

## See Also

[Configure a Genie](#)

[Tab Bar Genie](#)

[SA\\_Include Genie Libraries](#)

## Navigation Genie Library

The SA\_Include system project includes the "sa\_navigation" library. It includes the following Genies:

- [Link Down Genie](#)
- [Link Left Genie](#)
- [Link Right Genie](#)
- [Link Up Genie](#)
- [Static Link Left Genie](#)
- [Static Link Right Genie](#).

You can use these Genies to direct an operator through a production process that spans multiple graphics pages. For more information, see [Use Navigational Genies on Content Pages](#).

## See Also

[Situational Awareness System Projects](#)

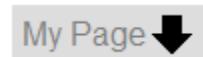
### Link Down Genie

**Genie Name:** link\_d

**Genie Library:** sa\_navigation

**System Project:** SA\_Include

The Link Down Genie (named "link\_d") can be used to direct an operator down a level in a set of hierarchical pages.



For more information, see [Use Navigational Genies on Content Pages](#).

## Genie Properties

Complete the following properties for the genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
Menu Item	The path to the linked item in the project's Menu Configuration. This is specified using the following notation: <Level1>.<Level2>.<Level3>.<Level4>.<Level5>.<Level6> For example, "Plant1.Area1.MyPage" The menu item needs to be included under the "Navigation" menu (defined in the <b>Page</b> field). As "Navigation" is assumed as the top level of the path, it is not necessary to enter it at the start of the menu item path.
Tooltip	An optional description of the Genie that appears when the mouse points at it.
Use for 4K Page?	Determines if the Genie will appear on a page that displays within a UHD_4K workspace. TRUE = The Genie will appear in a UHD4K workspace. FALSE = The Genie will not appear in a UHD4K workspace.

## See Also

[Configure a Genie](#)  
[Navigation Genie Library](#)

## Link Left Genie

**Genie Name:** link\_l  
**Genie Library:** sa\_navigation  
**System Project:** SA\_Include

The Link Left Genie (named "link\_l") can be used to direct an operator to the previous stage of a linear production process that spans multiple pages.

A blue rectangular button with a white arrow pointing left and the text "My Page" in white.

For more information, see [Use Navigational Genies on Content Pages](#).

## Genie Properties

Complete the following properties for the genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
Menu Item	The path to the linked item in the project's Menu Configuration. This is specified using the following notation: <code>&lt;Level1&gt;.&lt;Level2&gt;.&lt;Level3&gt;.&lt;Level4&gt;.&lt;Level5&gt;.&lt;Level6&gt;</code> For example, "Plant1.Area1.MyPage" The menu item needs to be included under the "Navigation" menu (defined in the <b>Page</b> field). As "Navigation" is assumed as the top level of the path, it is not necessary to enter it at the start of the menu item path.
Tooltip	An optional description of the Genie that appears when the mouse points at it.
Use for 4K Page?	Determines if the Genie will appear on a page that displays within a UHD_4K workspace. TRUE = The Genie will appear in a UHD4K workspace. FALSE = The Genie will not appear in a UHD4K workspace.
Links to another console?	Allows you to indicate that a Genie points to a destination on a different console, which is a destination that is not usually monitored by the current client. TRUE = The Genie points to a different console. FALSE = The Genie does not point to a different console. If set to TRUE, the Genie will display a gray background.

Parameter	Description
Use as Control Link?	Determines if the Genie will function as a control link. TRUE = The Genie will use a control link. FALSE = The Genie will not use a control link. If set to TRUE, the Genie will display a purple drop-shadow.
Control Signal Equipment	The piece of equipment that is associated with the left/right Genie. The Genie will only appear when their associated object is selected.
Controlled Equipment	The piece of equipment that the Control Link points to. When you click on the Genie, the page associated with the piece of equipment will display.

## See Also

[Configure a Genie](#)

[Navigation Genie Library](#)

## Link Right Genie

**Genie Name:** link\_r

**Genie Library:** sa\_navigation

**System Project:** SA\_Include

The Link Right Genie (named "link\_r") can be used to direct an operator to the next stage of a linear production process that spans multiple pages.

 My Page

For more information, see [Use Navigational Genies on Content Pages](#).

## Genie Properties

Complete the following properties for the genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
Menu Item	<p>The path to the linked item in the project's Menu Configuration. This is specified using the following notation:</p> <p>&lt;Level1&gt;.&lt;Level2&gt;.&lt;Level3&gt;.&lt;Level4&gt;.&lt;Level5&gt;.&lt;Level6&gt;</p> <p>For example, "Plant1.Area1.MyPage"</p> <p>The menu item needs to be included under the "Navigation" menu (defined in the <b>Page</b> field). As "Navigation" is assumed as the top level of the path, it is not necessary to enter it at the start of the menu item path.</p>
Tooltip	<p>An optional description of the Genie that appears when the mouse points at it.</p>
Use for 4K Page?	<p>Determines if the Genie will appear on a page that displays within a UHD_4K workspace.</p> <p>TRUE = The Genie will appear in a UHD4K workspace.</p> <p>FALSE = The Genie will not appear in a UHD4K workspace.</p>
Links to another console?	<p>Allows you to indicate that a Genie points to a destination on a different console, which is a destination that is not usually monitored by the current client.</p> <p>TRUE = The Genie points to a different console.</p> <p>FALSE = The Genie does not point to a different console.</p> <p>If set to TRUE, the Genie will display a gray background.</p>
Use as Control Link?	<p>Determines if the Genie will function as a control link.</p> <p>TRUE = The Genie will use a control link.</p> <p>FALSE = The Genie will not use a control link.</p> <p>If set to TRUE, the Genie will display a purple drop-shadow.</p>
Control Signal Equipment	<p>The piece of equipment that is associated with the left/right Genie. The Genie will only appear when their associated object is selected.</p>
Controlled Equipment	<p>The piece of equipment that the Control Link points to. When you click on the Genie, the page associated with the piece of equipment will display.</p>

## See Also

[Configure a Genie](#)  
[Navigation Genie Library](#)

## Link Up Genie

**Genie Name:** link\_u  
**Genie Library:** sa\_navigation  
**System Project:** SA\_Include

The Link Up Genie (named "link\_u") can be used to direct an operator up a level in a set of hierarchical pages.



For more information, see [Use Navigational Genies on Content Pages](#).

## Genie Properties

Complete the following properties for the genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
Menu Item	The path to the linked item in the project's Menu Configuration. This is specified using the following notation: <Level1>.<Level2>.<Level3>.<Level4>.<Level5>.<Level6> For example, "Plant1.Area1.MyPage" The menu item needs to be included under the "Navigation" menu (defined in the <b>Page</b> field). As "Navigation" is assumed as the top level of the path, it is not necessary to enter it at the start of the menu item path.
Tooltip	An optional description of the Genie that appears when the mouse points at it.
Use for 4K Page?	Determines if the Genie will appear on a page that displays within a UHD_4K workspace.

Parameter	Description
	TRUE = The Genie will appear in a UHD4K workspace. FALSE = The Genie will not appear in a UHD4K workspace.

## See Also

[Configure a Genie](#)  
[Navigation Genie Library](#)

### Static Link Left Genie

**Genie Name:** slink\_l  
**Genie Library:** sa\_navigation  
**System Project:** SA\_Include

The Static Link Left Genie (named "slink\_l") can be used to point at the next stage of a linear production process that spans multiple pages.



The Static Link Genies function primarily as a label; they are not designed to provide a direct link to a destination. For more information, see [Use Navigational Genies on Content Pages](#).

## Genie Properties

Complete the following properties for the genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
Label	The label that will appear on the Genie.
Tooltip	An optional description of the Genie that appears when the mouse points at it.
Use for 4K Page?	Determines if the Genie will appear on a page that displays within a UHD_4K workspace. TRUE = The Genie will appear in a UHD4K workspace.

Parameter	Description
	FALSE = The Genie will not appear in a UHD4K workspace.
Links to another console?	Allows you to indicate that a Genie points to a destination on a different console, which is a destination that is not usually monitored by the current client.  TRUE = The Genie points to a different console.  FALSE = The Genie does not point to a different console.  If set to TRUE, the Genie will display a gray background.

## See Also

[Configure a Genie](#)

[Navigation Genie Library](#)

### Static Link Right Genie

**Genie Name:** slink\_r

**Genie Library:** sa\_navigation

**System Project:** SA\_Include

The Static Link Left Genie (named "slink\_r") can be used to point at the next stage of a linear production process that spans multiple pages.



The Static Link Genies function primarily as a label; they are not designed to provide a direct link to a destination.

For more information, see [Use Navigational Genies on Content Pages](#).

## Genie Properties

Complete the following properties for the genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
Label	The label that will appear on the Genie.
Tooltip	An optional description of the Genie that appears when the mouse points at it.
Use for 4K Page?	Determines if the Genie will appear on a page that displays within a UHD_4K workspace. TRUE = The Genie will appear in a UHD4K workspace. FALSE = The Genie will not appear in a UHD4K workspace.
Links to another console?	Allows you to indicate that a Genie points to a destination on a different console, which is a destination that is not usually monitored by the current client. TRUE = The Genie points to a different console. FALSE = The Genie does not point to a different console. If set to TRUE, the Genie will display a gray background.

## See Also

[Configure a Genie](#)  
[Navigation Genie Library](#)

## Workspace Genie Library

The SA\_Include system project includes the "sa\_workspace" library. It includes the following Genie:

- [Pane Genie](#).

## See Also

[Situational Awareness System Projects](#)

## Pane Genie

**Genie Name:** pane  
**Genie Library:** sa\_workspace  
**System Project:** SA\_Include

A Pane Genie can be used to add a pane to a master page. See [Create a Master Page for a Customized Workspace](#).

It can also be used for popup windows that support autofill functionality. See [Create a Pop-up Window with Panes](#).

## Genie Properties

Complete the following properties for the genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Pane Properties

Property	Description
Name	The name used to identify the pane. The name needs to be unique; two panes on the same page cannot have the same name. The syslog.dat file for the client will log a message if it detects duplicate panes.
Default Page	The name of the page that the pane will display at startup. If the <b>FillMode</b> for the pane is set to an Autofill option, this page will also display when a context change results in the pane having no content to display.
Display Mode	Determines how the content of a pane will change as the pane is resized. Use one of the following integers to specify a display mode: <ul style="list-style-type: none"><li>• <b>1 = Stretch</b> — the displayed page will continue to fill the pane as it changes size.</li><li>• <b>2 = Maintain aspect ratio</b> — the displayed page will enlarge and reduce in size in proportion with the pane, however it will maintain its aspect ratio.</li><li>• <b>3 = Maintain size</b> — the displayed page will not change in size as the pane is resized. If the page is larger than the pane it will be clipped, or scroll bars will appear if enabled.</li></ul>
Is Default Pane?	Set this property to TRUE if you want the pane to be the default pane within the workspace. This will make the pane the primary point of reference for the workspace. The default pane is used to display pages that have no content type defined.  Each workspace can only contain one default pane. The syslog.dat file for a client will indicate if a default pane is not specified.

Property	Description
	The workspace Header Bar will refer to the page displayed in the default pane to source the active page title and breadcrumbs. This will also determine the current selection in the Navigation Zone.
Display Scrollbars	<p>Determines if the pane will include scrollbars at runtime.</p> <ul style="list-style-type: none"> <li>• FALSE = No scrollbars.</li> <li>• TRUE = Use scrollbars.</li> </ul> <p>Scrollbars will only appear if the pane's <b>DisplayMode</b> property is set to 3 (maintain size), and the page displayed is larger than the pane.</p>
Fill Mode	<p>Determines how the content of a pane is updated as context changes. Use one of the following integers to specify a fill mode:</p> <ul style="list-style-type: none"> <li>• <b>Static</b> — the content and equipment assignments within the pane remain unchanged when equipment context changes.</li> <li>• <b>StaticContextMustMatch</b> — the content does not change, but any associations are updated to match the new equipment context (see below).</li> <li>• <b>Autofill</b> — the pane will check the content specified for the equipment that comes into context. If it matches the pane's <b>ContentTypes</b> setting, the pane will display the content. If there is no valid content for the pane linked to the new equipment context, content may be sourced by traversing up/down the equipment hierarchy.</li> <li>• <b>AutofillContextMustMatch</b> — The same as Autofill, however content directly associated with the new equipment context will be used to fill the pane. The content will only be updated if the equipment is a direct contextual match (see below).</li> </ul> <p>See <a href="#">Autofill</a> for configuration examples.</p>
ContentTypes	Defines the type of content the pane will display. You can specify zero or more types as a comma separated list. Specifying no content type is acceptable for a static pane, but it is recommended that any other type of pane has at least one content type specified.

Property	Description
	<p>The content types you can use are configured in <b>Visualization</b> activity.</p> <p>At runtime, only pages with a content type that matches with the pane will be displayed. See <a href="#">Content Types</a>.</p>
Excluded Autofill Panes	<p>Allows you to specify panes that will not display updated content when an equipment context change is triggered from this pane. When the context changes in the current pane, the content in the excluded panes will remain static. However, the context in excluded panes will still be updated. Any pane in the workspace or a linked workspace can be excluded.</p> <p>Specify the name(s) of one or more panes or nested panes that you want to exclude from autofill. Use a comma to specify more than one pane name.</p> <p>For more information, see <i>Excluding Panes</i> (below).</p>
Tab Header Pane Name	<p>If you want the content of the pane to be controlled via a set of tabs, you can use this property to indicate the pane where the associated tab bar is located.</p> <p>Enter the name of the pane that hosts a page with the tab bar Genie on it.</p>
Tab Control Name	<p>Enter the name of the tab bar Genie that you would like to use to control the content of the pane. The tab control needs to be located on the page of the pane specified in the <b>Tab Header Pane Name</b> property (see above).</p>
Equipment Reference Associations: Categories	<p>This field allows you to associate a pane with an equipment reference category, or a comma-separated list of equipment reference categories (see "Category" in the topic <a href="#">Define Equipment References</a>). When a piece of equipment comes into context, any equipment references within the specified category will be used to create Super Genie associations for a displayed page. For more information, see <a href="#">AssEquipReferences</a>.</p>

## Excluding Panes

As described above, you can use the Pane Properties dialog box to add a list of panes that you wish to exclude from autofill. Excluded panes will show the same content regardless of the context. If the current pane changes context, content in the panes specified in the **Exclude Autofill Panes** box will not be updated.

---

**Note:** The context within excluded panes will be updated when the context of the current page changes.

It is also possible to exclude panes at runtime by using a page environment variable. To do this:

1. Open the page that contains the context that will change.
2. Open the Page Properties dialog box.
3. Select the Environment tab.
4. Click **Add**. The Edit Properties dialog box is displayed.
5. In the **Property** box, type the property name ExcludedAutoFillPanes.
6. In the **Value** box, specify the name(s) of one or more panes that you want to exclude from autofill.
7. Click **OK** to close the Edit Properties dialog box.
8. Click **OK** to close the Page Properties dialog box.

## See Also

[Configure a Genie](#)

[SA\\_Include Genie Libraries](#)

## Situational Awareness Controls Project

The Situational Awareness Controls Project (named "SA\_Controls") is an include project contains the following Genie libraries:

- [Controls Genie Library](#)
- [Common Controls Genie Library](#)
- [Priorities Genie Library](#).

---

**Note:** The SA\_Controls project is a system include project, and only appears in the Project list tree if the **System Projects** filter is applied in the Projects activity. This project is included when you create a project using the Situational Awareness Starter project.

---

## See Also

[Situational Awareness System Projects](#)

## Controls Genie Library

The SA\_Controls system project includes the "sa\_controls" library. It includes the following Genies:

- [Alarm List Genie](#)
- [Alarm List Vertical Genie](#)
- [Generic List Genie](#)
- [Generic List Vertical Genie](#)
- [List Row Genie](#).

---

**Note:** The SA\_Inlcude system project also contains a library named "sa\_controls". For a list of the Genies it

---

---

includes, see Controls Genie Library.

---

## See Also

[Situational Awareness System Projects](#)

## Alarm List Genie

**Genie Name:** alarmlist

**Genie Library:** sa\_controls

**System Project:** SA\_Controls

The Alarm List Genie (named "alarmlist") allows you to place a list of alarms on a graphics page in a pre-defined format with a header row and scroll bars. It is a variation of the [Alarm List Vertical Genie](#), which produces a list with no header row and just a vertical scroll bar.

This Genie can be used with the [List Row Genie](#), which facilitates row selection, a row context menu and row background color.

By default, an alarm list Genie displays the fields configured through the [AlarmListCreate](#) Cicode function. You can also use this function to specify the display fonts and alarm indicators.

For further instructions on how to use the Alarm List Genie, see [Add an Alarms List to a Page](#).

---

**Note:** You can also use the Citect.ini parameter [\[Format\]FormatName](#) to set a default format for the fields displayed in an alarms list.

---

## Alarm List Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Alarm List Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
AN	This is automatically populated and matches the AN in the Genie Properties dialog box. This cannot be modified.
OnRightClickFunction	Allows the user to right-click on the alarms list and perform operations displayed on the context menu.
OnColumnSelectedFunction	Allows you to select a column.
OnColumnReorderedFunction	Allows you reorder the columns displayed. For

Parameter	Description
	example, ArrayView_SetFormat().
OnColumnResizedFunction	Allows you to re-size the columns displayed.

## Alarm List Genie Page Properties

For the Genie to work at runtime, they require some arrays to be pre-configured before animation begins. In the Page Properties dialog box, Events tab, select the following event options and specify the required Cicode command.

Parameter	Example	Comments
On Page Entry	<pre>_Array View_Initialize(); AlarmListCreate(DspGetAnFromName("alarmlist"), 0, 700-14, 318, 32, 1, "", "TopActiveAlarms_UHD4K", -1, DspFontHnd("SA_AlarmHeader4K" ))</pre>	This code will create the Alarm list called "alarmlist" based on the parameters specified in the AlarmListCreate function.
On Page Exit	<pre>AlarmListDestroy(DspGetAnFromName("alarmlist"))</pre>	This code will destroy the alarm list array when the user exits the page.

## See Also

[Configure a Genie](#)

[Controls Genie Library](#)

## Alarm List Vertical Genie

**Genie Name:** alarmlist\_v

**Genie Library:** sa\_controls

**System Project:** SA\_Controls

The Alarm List Vertical Genie (named "alarmlist\_v") allows you to place a list of alarms on a graphics page in a pre-defined format. It is a variation of the [Alarm List Genie](#). It differs as it generates a list with no header row and just a vertical scroll bar.

This Genie can be used with the [List Row Genie](#), which facilitates row selection, a row context menu and row background color.

By default, an alarm list Genie displays the fields configured through the [AlarmListCreate](#) Cicode function. You can also use this function to specify the display fonts and alarm indicators.

For further instructions on how to use the List Row Vertical Genie, see [Add an Alarms List to a Page](#).

---

**Note:** You can also use the Citect.ini parameter [\[Format\]FormatName](#) to set a default format for the fields displayed in an alarms list.

## Alarm List Vertical Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- AN:** This is an automatically-generated unique ID for the Genie object.
- Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Alarm List Vertical Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
AN	This is automatically populated and matches the AN in the Genie Properties dialog box. This cannot be modified.

## Alarm List Vertical Genie Page Properties

For the Genie to work at runtime, they require some arrays to be pre-configured before animation begins. In the Page Properties dialog box, Events tab, select the following event options and specify the required Cicode command.

Parameter	Example	Comments
On Page Entry	<pre>_Array View_Initialize(); AlarmListCreate(DspGetAnFromName("alarmlist"), 0, 700-14, 318, 32, 1, "", "TopActiveAlarms_UHD4K", -1, DspFontHnd("SA_AlarmHeader4K")) )</pre>	This code will create the Alarm list called "alarmlist" based on the parameters specified in the AlarmListCreate function.
On Page Exit	<pre>AlarmListDestroy(DspGetAnFromName("alarmlist"))</pre>	This code will destroy the alarm list array when the user exits the page.

## See Also

[Configure a Genie](#)

[Controls Genie Library](#)

## Generic List Genie

**Genie Name:** genericlist

**Genie Library:** sa\_controls

**System Project:** SA\_Controls

The Generic List Genie (named "genericlist") allows you to display a custom generated list of information on a page in a pre-defined format with a header row and scroll bars. It is a variation of the [Generic List Vertical Genie](#), which produces a list with no header row and just a vertical scroll bar.

## Generic List Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Generic List Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
AN	This is automatically populated and matches the AN in the Genie Properties dialog box. This cannot be modified.
GenericListFillFunction	Specify the user Cicode function to be called at runtime to populate the list by passing the list's AN to the function. The function needs to retrieve the data to be displayed. For example, <code>AlarmPage_GetReferences</code> .
OnRightClickFunction	Allows the user right-click on a row within the alarms list and perform operations displayed on the right-click menu.
OnColumnSelectedFunction	Allows you to select a column.
OnColumnReorderedFunction	Allows you reorder the columns displayed.
OnColumnResizedFunction	Allows you to resize the columns displayed.

## Generic List Genie Page Properties

For these Genies to work at runtime, they require some arrays to be pre-configured before animation begins. In the Page Properties dialog box, Events tab, select the following event options and specify the required Cicode command.

Parameter	Description	Example
On Page Entry	This code initializes the list array to be populated when the page is displayed.	FUNCTION ListInfo_Init()
On Page Exit	When the user exits the page, clear the information stored in the array.	FUNCTION ListInfo_Exit()

### Example - Clear a list array on exit

```
FUNCTION ListInfo_Exit()
//Clear Equipment List
INT hEquipRefArray = PageGetInt("__EquipRefArray");
ArrayDestroy(hEquipRefArray);
ArrayDestroyByAn(DspGetAnFromName("MyList"));
Debug_Trace("ReferenceInfo", "ListInfo_Exit", "Closing Infoe_Default page.", 
TRACE_Information);
END
```

### Example - Initialize a list array

```
FUNCTION ListInfo_Init()
INT hEquipRefArray;
INT nListAN = DspGetAnFromName("MyList");
INT nListHeight = 160;
INT nListRowHeight = 24;
INT nListWidth = 370;
INT nListCreatedFlag = 0;
STRING sFontPostfix = "";
STRING sResolution = "HD1080";

Debug_Trace("ListInfo", "ListInfo_Init", "Initializing Info_Default page.", 
TRACE_Information);

hEquipRefArray = ArrayCreate("EquipRef_" + nReferencesListAN:#, 5, 1000); // Max of 1000
equip references
PageSetInt("__EquipRefArray", hEquipRefArray); //Cache the array handle

_ArrayView_Initialize(); //Initialize the array

nListCreatedFlag = GenericListCreate(nListAN, nListWidth-14, nListHeight ,
nListRowHeight, 1, "SideEquipRef", DspFontHnd("SA_AlmRow" + sFontPostFix),
DspFontHnd("SA_AlarmHeaderB" + sFontPostFix));
ArraySetIntByAn(nListAN, 5, 1, ARRAY_INFORMATION_ROW, 2);

Debug_Trace("ListInfo", "ListInfo_Init", "List_Default page is initialized with " +
nListCreatedFlag:#, TRACE_Information);
END
```

## Example - Populate list

```
FUNCTION ReferenceInfo_GetData(INT nListAN, STRING sEquipmentContext)
    INT hSession;
    INT nRecords;
    INT nStatus;
    INT nRow;
    INT nLength;
    INT nOffset;
    INT hModel = PageGetInt("__EquipRefArray");
    INT iArrayRow = 1;
    STRING sFilter;
    STRING sFields = "REFCLUST,REFEQUIP,REFITEM,REFCAT"
    STRING sRefCategory;
    STRING sRefCluster;
    STRING sRefEquipment;
    STRING sRefItem;
    STRING sEquipment;
    STRING sRefEquipItem;
    STRING sRefItemValue;
    IF (sEquipmentContext <> "") THEN
        nLength = StrLength(sEquipmentContext);
        nOffset = StrSearch(0, sEquipmentContext, ".");
        sEquipment = StrRight(sEquipmentContext, nLength - nOffset - 1);
        sFilter = "EQUIP=" + sEquipment;
        Debug_Trace("ReferenceInfo", "ReferenceInfo_GetData", "Populating the equipment
reference for " + sEquipment, TRACE_Information);
        hSession = EquipRefBrowseOpen(sFilter, sFields);
        IF (hSession <> BAD_HANDLE) THEN
            nRecords = EquipRefBrowseNumRecords(hSession);
            IF nRecords > 0 THEN
                nStatus = EquipRefBrowseFirst(hSession);
                FOR nRow = 0 TO (nRecords - 1) DO
                    sRefCategory = EquipRefBrowseGetField(hSession, "REFCAT");
                    sRefCluster = EquipRefBrowseGetField(hSession, "REFCLUST");
                    sRefEquipment = EquipRefBrowseGetField(hSession, "REFEQUIP");
                    sRefItem = EquipRefBrowseGetField(hSession, "REFITEM");
                    sRefEquipItem = sRefEquipment + "." + sRefItem;
                    sRefItemValue = TagRead(sRefEquipItem, 0, sRefCluster);
                    SleepMS(10);
                    ArraySetInt(hModel, 0, 0, iArrayRow);
                    ArraySetString(hModel, sRefCluster, 1, iArrayRow);
                    ArraySetString(hModel, sRefEquipItem, 2, iArrayRow);
                    ArraySetString(hModel, sRefItemValue, 3, iArrayRow);
                    ArraySetString(hModel, sRefCategory, 4, iArrayRow);
                    iArrayRow = iArrayRow + 1;
                    EquipRefBrowseNext(hSession);
                END
            END
            EquipRefBrowseClose(hSession);
        END
    END
    IF (nRecords <= 0) THEN
        FOR nRow = 0 TO ArrayGetInfo(hModel, 1) DO
            ArraySetInt(hModel, 0, 0, iArrayRow);
            ArraySetString(hModel, "", 1, iArrayRow);
```

```
        ArraySetString(hModel, "", 2, iArrayRow);
        ArraySetString(hModel, "", 3, iArrayRow);
        ArraySetString(hModel, "", 4, iArrayRow);
        //iArrayRow = iArrayRow + 1;
    END
END
ArraySetInt(hModel, iArrayRow, 0, 0);
END
```

## See Also

[Configure a Genie](#)

[Controls Genie Library](#)

## Generic List Vertical Genie

**Genie Name:** genericlist\_v

**Genie Library:** sa\_controls

**System Project:** SA\_Controls

The Generic List Vertical Genie (named "genericlist\_v") allows you to display a custom generated list of information on a page in a pre-defined format with a header row and scroll bars. It is a variation of the [Generic List Genie](#). It differs as it generates a list with no header row and just a vertical scroll bar.

## Generic List Vertical Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- AN:** This is an automatically-generated unique ID for the Genie object.
- Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Generic List Vertical Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
AN	This is automatically populated and matches the AN in the Genie Properties dialog box. This cannot be modified.
GenericListFillFunction	Specify the user Cicode function to be called at runtime to populate the list by passing the list's AN to the function. The function needs to retrieve the data to be displayed. For example, AlarmPage_FillResponse.

## Generic List Vertical Genie Page Properties

For these Genies to work at runtime, they require some arrays to be pre-configured before animation begins. In the Page Properties dialog box, Events tab, select the following event options and specify the required Cicode command.

Parameter	Description	Example
On Page Entry	This code initializes the list array to be populated when the page is displayed.	FUNCTION ListInfo_Init()
On Page Exit	When the user exits the page, clear the information stored in the array.	FUNCTION ListInfo_Exit()

### Example - Clear a list array on exit

```
FUNCTION ListInfo_Exit()
//Clear Equipment List
INT hEquipRefArray = PageGetInt("__EquipRefArray");
ArrayDestroy(hEquipRefArray);
ArrayDestroyByAn(DspGetAnFromName("MyList"));
Debug_Trace("ReferenceInfo", "ListInfo_Exit", "Closing Infoe_Default page.", 
TRACE_Information);
END
```

### Example - Initialize a list array

```
FUNCTION ListInfo_Init()
INT hEquipRefArray;
INT nListAN = DspGetAnFromName("MyList");
INT nListHeight = 160;
INT nListRowHeight = 24;
INT nListWidth = 370;
INT nListCreatedFlag = 0;
STRING sFontPostfix = "";
STRING sResolution = "HD1080";

Debug_Trace("ListInfo", "ListInfo_Init", "Initializing Info_Default page.", 
TRACE_Information);

hEquipRefArray = ArrayCreate("EquipRef_" + nReferencesListAN:#, 5, 1000); // Max of 1000
equip references
PageSetInt("__EquipRefArray", hEquipRefArray); //Cache the array handle

_ArrayView_Initialize(); //Initialize the array

nListCreatedFlag = GenericListCreate(nListAN, nListWidth-14, nListHeight ,
nListRowHeight, 1, "SideEquipRef", DspFontHnd("SA_AlmRow" + sFontPostFix),
DspFontHnd("SA_AlarmHeaderB" + sFontPostFix));
ArraySetIntByAn(nListAN, 5, 1, ARRAY_INFORMATION_ROW, 2);
```

```
    Debug_Trace("ListInfo", "ListInfo_Init", "List_Default page is initialized with " +
nListCreatedFlag:#, TRACE_Information);
END
```

## Example - Populate list

```
FUNCTION ReferenceInfo_GetData(INT nListAN, STRING sEquipmentContext)
    INT hSession;
    INT nRecords;
    INT nStatus;
    INT nRow;
    INT nLength;
    INT nOffset;
    INT hModel = PageGetInt("__EquipRefArray");
    INT iArrayRow = 1;
    STRING sFilter;
    STRING sFields = "REFCLUST,REFEQUIP,REFITEM,REFCAT"
    STRING sRefCategory;
    STRING sRefCluster;
    STRING sRefEquipment;
    STRING sRefItem;
    STRING sEquipment;
    STRING sRefEquipItem;
    STRING sRefItemValue;
    IF (sEquipmentContext <> "") THEN
        nLength = StrLength(sEquipmentContext);
        nOffset = StrSearch(0, sEquipmentContext, ".")
        sEquipment = StrRight(sEquipmentContext, nLength - nOffset - 1);
        sFilter = "EQUIP=" + sEquipment;
        Debug_Trace("ReferenceInfo", "ReferenceInfo_GetData", "Populating the equipment
reference for " + sEquipment, TRACE_Information);
        hSession = EquipRefBrowseOpen(sFilter, sFields);
        IF (hSession <> BAD_HANDLE) THEN
            nRecords = EquipRefBrowseNumRecords(hSession);
            IF nRecords > 0 THEN
                nStatus = EquipRefBrowseFirst(hSession);
                FOR nRow = 0 TO (nRecords - 1) DO
                    sRefCategory = EquipRefBrowseGetField(hSession, "REFCAT");
                    sRefCluster = EquipRefBrowseGetField(hSession, "REFCLUST");
                    sRefEquipment = EquipRefBrowseGetField(hSession, "REFEQUIP");
                    sRefItem = EquipRefBrowseGetField(hSession, "REFITEM");
                    sRefEquipItem = sRefEquipment + "." + sRefItem;
                    sRefItemValue = TagRead(sRefEquipItem, 0, sRefCluster);
                    SleepMS(10);
                    ArraySetInt(hModel, 0, 0, iArrayRow);
                    ArraySetString(hModel, sRefCluster, 1, iArrayRow);
                    ArraySetString(hModel, sRefEquipItem, 2, iArrayRow);
                    ArraySetString(hModel, sRefItemValue, 3, iArrayRow);
                    ArraySetString(hModel, sRefCategory, 4, iArrayRow);
                    iArrayRow = iArrayRow + 1;
                    EquipRefBrowseNext(hSession);
                END
            END
            EquipRefBrowseClose(hSession);
        END
    END
```

```
END
IF (nRecords <= 0) THEN
    FOR nRow = 0 TO ArrayGetInfo(hModel, 1) DO
        ArraySetInt(hModel, 0, 0, iArrayRow);
        ArraySetString(hModel, "", 1, iArrayRow);
        ArraySetString(hModel, "", 2, iArrayRow);
        ArraySetString(hModel, "", 3, iArrayRow);
        ArraySetString(hModel, "", 4, iArrayRow);
        //iArrayRow = iArrayRow + 1;
    END
END
ArraySetInt(hModel, iArrayRow, 0, 0);
END
```

## See Also

[Configure a Genie](#)

[Controls Genie Library](#)

## List Row Genie

**Genie Name:** list\_row\_x<n>

**Genie Library:** sa\_controls

**System Project:** SA\_Controls

The List Row Genie can be used in conjunction with the [Alarm List Genie](#) and [Generic List Genie](#) Genies to facilitate row selection, row context menu and row background color. Several variations of the List Row Genie are available:

- List Row x5 – use this to create a list with up to 5 rows.
- List Row x10 – use this to create a list with up to 10 rows.
- List Row x40 – use this to create a list with up to 40 rows.
- List Row x80 – use this to create a list with up to 80 rows.

---

**Note:** There is no need to adjust the size of the List Row Genie on a graphics page to fill an area. The required space will be determined at runtime by the size of the list with which it is associated.

---

For further instructions on how to use the List Row Genie, see [Add an Alarms List to a Page](#) or [Add a Generic List to a Page](#).

## List Row Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## List Row Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
ArrayAN	Specify the AN of the alarm list to which row formatting will be applied. You can use the Cicode function DspGetAnFromName to retrieve the AN. For example, DspGetAnFromName("AlarmList").
RowOffset	Row number at which to display a list if there are multiple Alarm List Genies on the page. Default is 0. If multiple List Row Genies are inserted on to a page, you need to specify the correct row offset for each List Row Genie. For example, if you insert two List Row_x10 Genies, the row offset for the first Genie will be 0 while the row offset for the second Genie will be 10.
LeftClickCommand	Allows you to call Cicode that performs operations when you click on a row within the list. For example, you can select the row using AlarmPage_SelectAlarmRecord(DspGetAnFromName("AlarmList")).
RightClickCommand	Allows you to call Cicode that performs operations when you right-click on a row within the list. For example, Alarm_TopAlarmsRow_RightClickCommand(DspGetAnFromName("AlarmList")).

## See Also

[Configure a Genie](#)

[Controls Genie Library](#)

## Common Controls Genie Library

The SA\_Controls system project includes the "sa\_controls\_common" library. It includes the following Genies:

- [Button Base Genie](#)
- [Button Label Genie](#)
- [Check Box Genie](#)
- [Scroll Bar Horizontal Genie](#)
- [Scroll Bar Vertical Genie](#)

- Toolbar Button Base Genie
- Toggle Button Genie.

---

**Note:** The SA\_Include system project also contains a library named "sa\_controls\_common". For a list of the Genies it includes, see [Common Controls Genie Library](#).

---

## See Also

[Situational Awareness System Projects](#)

### Button Base Genie

**Genie Name:** button\_base

**Genie Library:** sa\_controls\_common

**System Project:** SA\_Controls

The Button Base Genie (named "button\_base") can be used to position a button and enable its basic functionality.

Buttons are theme aware; their color will adjust according to the current theme set for the workspace.

### Genie Properties

Complete the following properties for the genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

### Genie Parameters

Select the genie, then double-click on it. The genie's parameters are displayed. Complete the following parameters:

Parameter	Description
Click Command	The function that is called when the operator clicks on the button. No specific arguments are required.
Disabled When	An expression (applied in conjunction with Area and Privilege) that can provide further logic to specify when the button is disabled.
Tooltip	Text that displays when the operator hovers the mouse over the button at runtime.
Area	Enter the area to which the button belongs. Only users with access to this area can use the button.
Privilege	Enter the privilege level that a user needs to possess to use this button.

Parameter	Description
Equipment Context	<p>Can be used to store an equipment context. This is an optional setting that is only used by navigation buttons.</p> <p><b>Note:</b> If Equipment Context is specified, then the cluster needs to be specified as well.</p>

## See Also

[Configure a Genie](#)

[Common Controls Genie Library](#)

## Button Label Genie

**Genie Name:** button\_label\_<n>

**Genie Library:** sa\_controls\_common

**System Project:** SA\_Controls

A Button Label Genie can be used to apply text to a [Button Base Genie](#).

There are two variations of the genie:

- button\_label\_12 - uses a 12 point font
- button\_label\_16 - uses a 16 point font

## Genie Properties

Complete the following properties for the genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the genie, then double-click on it. The genie's parameters are displayed. Complete the following parameters:

Parameter	Description
Label	The text that appears on the label.
Background AN	The AN of the background Button Base Genie.

## See Also

[Configure a Genie](#)

## Common Controls Genie Library

### Check Box Genie

**Genie Name:** checkbox

**Genie Library:** sa\_controls\_common

**System Project:** SA\_Controls

The Check Box Genie (named "checkbox") can be used to apply a check box control to a page.

### Genie Properties

Complete the following properties for the genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

### Genie Parameters

Select the genie, then double-click on it. The genie's parameters are displayed. Complete the following parameters:

Parameter	Description
OnToggleFunction	The function that is called when the operator clicks on the check box. No specific arguments are required.

### See Also

[Configure a Genie](#)

[Common Controls Genie Library](#)

### Scroll Bar Horizontal Genie

**Genie Name:** scrollbar\_h

**Genie Library:** sa\_controls\_common

**System Project:** SA\_Controls

The Scroll Bar Horizontal Genie (named "scrollbar\_h") allows you to add a horizontal scroll bar to a Genie. It is designed to work with the array-enabled Genies such as:

- [Alarm List Genie](#)
- [Alarm List Vertical Genie](#)
- [Generic List Genie](#)
- [Generic List Vertical Genie](#)
- [Tree View Genie](#).

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
pxOffsetX	Reflects the default vertical position of the scroll bar in relation to the Genie it is placed in. For example, if your Genie contains a scroll bar positioned 100 pixels to the right, then pxOffset will be 100.
ArrayAN	Specify the AN of the animation object where an array has been created to store positional and scroll information.

## See Also

[Configure a Genie](#)

[Common Controls Genie Library](#)

## Scroll Bar Vertical Genie

**Genie Name:** scrollbar\_v

**Genie Library:** sa\_controls\_common

**System Project:** SA\_Controls

The Scroll Bar Vertical Genie (named "scrollbar\_v") allows you to add a vertical scroll bar to a Genie. It is designed to work with the array-enabled Genies such as:

- [Alarm List Genie](#)
- [Alarm List Vertical Genie](#)
- [Generic List Genie](#)
- [Generic List Vertical Genie](#)
- [Tree View Genie](#).

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie, then double-click on it. The Genie's parameters are displayed. Complete the following parameters:

Parameter	Description
pxOffsetX	Reflects the default horizontal position of the scroll bar in relation to the Genie it is placed in. For example, if your Genie contains a scroll bar positioned 100 pixels from the bottom, then pxOffset will be 100.
ArrayAN	Specify the AN of the animation object where an array has been created to store positional and scroll information.

## See Also

[Configure a Genie](#)

[Common Controls Genie Library](#)

## Toolbar Button Base Genie

**Genie Name:** toolbar\_button\_base

**Genie Library:** sa\_controls\_common

**System Project:** SA\_Controls

The Toolbar Button Base Genie (named "toolbar\_button\_base") can be used to position a button on the Situational Awareness [Header Bar](#).

Buttons are theme aware; their color will adjust according to the current theme set for the workspace.

## Genie Properties

Complete the following properties for the genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the genie, then double-click on it. The genie's parameters are displayed. Complete the following parameters:

Parameter	Description
Click Command	The function that is called when the operator clicks on the button. No specific arguments are required.
Disabled When	An expression (applied in conjunction with Area and Privilege) that can provide further logic to specify when the button is disabled.
Tooltip	Text that displays when the operator hovers the mouse over the button at runtime.
Area	Enter the area to which the button belongs. Only users with access to this area can use the button.
Privilege	Enter the privilege level that a user needs to possess to use this button.
Equipment Context	Can be used to store an equipment context. This is an optional setting that is only used by navigation buttons.

## See Also

[Configure a Genie](#)

[Common Controls Genie Library](#)

## Toggle Button Genie

**Genie Name:** toggle\_button

**Genie Library:** sa\_controls\_common

**System Project:** SA\_Controls

The Toggle Button Genie (named "toggle\_button") is used to create a toggle button.

## Genie Properties

Complete the following properties for the genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the genie, then double-click on it. The genie's parameters are displayed. Complete the following parameters:

Parameter	Description
Click Command	The function that is called when the operator clicks on the button. No specific arguments are required.
Active When	An expression (applied in conjunction with Area and Privilege) that can provide further logic to specify when the button is active.
Disabled When	An expression (applied in conjunction with Area and Privilege) that can provide further logic to specify when the button is disabled.
Tooltip	Text that displays when the operator hovers the mouse over the button at runtime.
Area	Enter the area to which the button belongs. Only users with access to this area can use the button.
Privilege	Enter the privilege level that a user needs to possess to use this button.
Equipment Context	<p>Can be used to store an equipment context. This is an optional setting that is only used by navigation buttons.</p> <p><b>Note:</b> If Equipment Context is specified, then the cluster needs to be specified as well.</p>

## See Also

[Configure a Genie](#)

[Common Controls Genie Library](#)

## Priorities Genie Library

The SA\_Controls system project includes the "sa\_priorities" library. It includes the following Genies:

- [Disabled Normal Genie](#)
- [Disabled Small Genie](#)
- [Priority 1 Normal Genie](#)
- [Priority 1 Small Genie](#)
- [Priority 2 Normal Genie](#)
- [Priority 2 Small Genie](#)
- [Priority 3 Normal Genie](#)
- [Priority 3 Small Genie](#)
- [Priority 4 Normal Genie](#)

- Priority 4 Small Genie.

You can use these Genies to [Create Priority and State Symbols for an Alarms List](#).

## See Also

[Situational Awareness System Projects](#)

### Disabled Normal Genie

**Genie Name:** sa\_disabled\_normal

**Genie Library:** sa\_priorities

**System Project:** SA\_Controls

The Disabled Normal Genie (named "sa\_disabled\_normal") is used to represent disabled alarms for a status flag on an alarm indicator (see [Use Alarm Indicators](#)).

For information on how to configure a status flag for an alarm indicator, see [Create a Custom Flag for an Alarm Indicator](#).

## See Also

[Configure a Genie](#)

[Priorities Genie Library](#)

### Disabled Small Genie

**Genie Name:** sa\_disabled\_small

**Genie Library:** sa\_priorities

**System Project:** SA\_Controls

The Disabled Small Genie (named "sa\_disabled\_small") is used as an icon alongside a disabled alarm count.

## See Also

[Configure a Genie](#)

[Priorities Genie Library](#)

### Priority 1 Normal Genie

**Genie Name:** sa\_p1\_normal

**Genie Library:** sa\_priorities

**System Project:** SA\_Controls

The Priority 1 Normal Genie (named "sa\_p1\_normal") is used to represent priority 1 alarms for a status flag on an alarm indicator (see [Use Alarm Indicators](#)).

It is typically associated with the **Genie Name** property for an Alarm Priority (see [Configure Display Properties for an Alarm Priority](#)).

For information on how to configure a status flag for an alarm indicator, see [Create a Custom Flag for an Alarm Indicator](#).

## See Also

[Configure a Genie](#)

[Priorities Genie Library](#)

## Priority 1 Small Genie

**Genie Name:** sa\_p1\_small

**Genie Library:** sa\_priorities

**System Project:** SA\_Controls

The Priority 1 Small Genie (named "sa\_p1\_small") can be used as an icon that highlights the occurrence of priority 1 alarms. It is typically associated with the **Small Genie Name** property for an Alarm Priority (see [Configure Display Properties for an Alarm Priority](#)).

It can be used for:

- Priority And State symbols in an alarms list (see [Create Priority and State Symbols for an Alarms List](#)).
- The alarm counts on a tree-view (see [Add a Tree View to a Page](#)).
- The alarm counts in the Navigation Zone (see [Navigation Zone](#)).

## See Also

[Configure a Genie](#)

[Priorities Genie Library](#)

## Priority 2 Normal Genie

**Genie Name:** sa\_p2\_normal

**Genie Library:** sa\_priorities

**System Project:** SA\_Controls

The Priority 2 Normal Genie (named "sa\_p2\_normal") is used to represent priority 2 alarms for a status flag on an alarm indicator (see [Use Alarm Indicators](#)).

It is typically associated with the **Genie Name** property for an Alarm Priority (see [Configure Display Properties for an Alarm Priority](#)).

For information on how to configure a status flag for an alarm indicator, see [Create a Custom Flag for an Alarm Indicator](#).

## See Also

[Configure a Genie](#)

[Priorities Genie Library](#)

## Priority 2 Small Genie

**Genie Name:** sa\_p2\_small

**Genie Library:** sa\_priorities

**System Project:** SA\_Controls

The Priority 2 Small Genie (named "sa\_p2\_small") can be used as an icon that highlights the occurrence of priority 2 alarms. It is typically associated with the **Small Genie Name** property for an Alarm Priority (see [Configure Display Properties for an Alarm Priority](#)).

It can be used for:

- Priority And State symbols in an alarms list (see [Create Priority and State Symbols for an Alarms List](#)).
- The alarm counts on a tree-view (see [Add a Tree View to a Page](#)).
- The alarm counts in the Navigation Zone (see [Navigation Zone](#)).

## See Also

[Configure a Genie](#)

[Priorities Genie Library](#)

## Priority 3 Normal Genie

**Genie Name:** sa\_p3\_normal

**Genie Library:** sa\_priorities

**System Project:** SA\_Controls

The Priority 3 Normal Genie (named "sa\_p3\_normal") is used to represent priority 3 alarms for a status flag on an alarm indicator (see [Use Alarm Indicators](#)).

It is typically associated with the **Genie Name** property for an Alarm Priority (see [Configure Display Properties for an Alarm Priority](#)).

For information on how to configure a status flag for an alarm indicator, see [Create a Custom Flag for an Alarm Indicator](#).

## See Also

[Configure a Genie](#)

[Priorities Genie Library](#)

## Priority 3 Small Genie

**Genie Name:** sa\_p3\_small

**Genie Library:** sa\_priorities

**System Project:** SA\_Controls

The Priority 3 Small Genie (named "sa\_p3\_small") can be used as an icon that highlights the occurrence of priority 3 alarms. It is typically associated with the **Small Genie Name** property for an Alarm Priority (see [Configure Display Properties for an Alarm Priority](#)).

It can be used for:

- Priority And State symbols in an alarms list (see [Create Priority and State Symbols for an Alarms List](#)).
- The alarm counts on a tree-view (see [Add a Tree View to a Page](#)).
- The alarm counts in the Navigation Zone (see [Navigation Zone](#)).

## See Also

[Configure a Genie](#)

[Priorities Genie Library](#)

### Priority 4 Normal Genie

**Genie Name:** sa\_p4\_normal

**Genie Library:** sa\_priorities

**System Project:** SA\_Controls

The Priority 4 Normal Genie (named "sa\_p4\_normal") is used to represent priority 4 alarms for a status flag on an alarm indicator (see [Use Alarm Indicators](#)).

It is typically associated with the **Genie Name** property for an Alarm Priority (see [Configure Display Properties for a Fourth Alarm Priority](#)).

For information on how to configure a status flag for an alarm indicator, see [Create a Custom Flag for an Alarm Indicator](#).

## See Also

[Configure a Genie](#)

[Priorities Genie Library](#)

### Priority 4 Small Genie

**Genie Name:** sa\_p4\_small

**Genie Library:** sa\_priorities

**System Project:** SA\_Controls

The Priority 4 Small Genie (named "sa\_p4\_small") can be used as an icon that highlights the occurrence of priority 4 alarms. It is typically associated with the **Small Genie Name** property for an Alarm Priority (see [Configure Display Properties for a Fourth Alarm Priority](#)).

It can be used for:

- Priority And State symbols in an alarms list (see [Create Priority and State Symbols for an Alarms List](#)).
- The alarm counts on a tree-view (see [Add a Tree View to a Page](#)).
- The alarm counts in the Navigation Zone (see [Add an Additional Alarm Count to the Navigation Zone](#)).

## See Also

[Configure a Genie](#)

[Priorities Genie Library](#)

## Situational Awareness Library Project

The Situational Awareness Library Project (named "sa\_library") is an include project that contains a set of Genies for common user interface controls as listed in the table below.

In addition, the sa\_library projects contains Genies for meters, drives and valves that can be combined to build **Composite Genies**. Refer to the topic [Insert a Composite Genie](#) to learn how to add a meter, drive or valve object onto your graphics page.

---

**Note:** The sa\_library Include project is a system include project, and only appears in the Project list tree if the **System Projects** filter is applied in the Projects activity. This project is included when you create a project using the Situational Awareness Starter project.

---

## Object Libraries

Library objects are accessible in Graphics Builder from the following object libraries.

Library Name	Description
sa_belt	Contains <a href="#">Belt Objects</a> such as aprons, belts and conveyors.
sa_chart	Contains <a href="#">Additional Objects</a> such as polar star and XY Bar Graph.
sa_common	Contains the selection adorner objects to display borders around objects, output bar, tracker, labels and OOS indicators.
sa_dataobjects	Contains objects such as up/down buttons, numeric and text input that provide additional methods of displaying data and related properties.
sa_deviationmeter	Contains the <a href="#">Deviation Meter</a> object and its components including alarm limits, PV tracker, etc.
sa_drive	Contains <a href="#">Drive Objects</a> such as compressors, brakes, pumps, etc in different sizes and orientations.
sa_dualmeter	Contains the <a href="#">Dual Level Meter</a> object and its components including alarm limits, PV tracker, etc.
sa_faceplates	Contains the components to build a <a href="#">Faceplates</a> .
sa_meter	Contains <a href="#">Meters</a> such as analyzer, pressure,

Library Name	Description
	temperature meters, etc in different sizes and orientations.
sa_mining	Contains <a href="#">Mining Objects</a> such as crushers, cutters, magnets, etc
sa_misc	Contains <a href="#">Additional Objects</a> such as alarm light, handswitch selector, etc.
sa_trend	Contains <a href="#">Trend Objects</a> such as pens, limits, trend tail, etc.
sa_valve	Contains <a href="#">Valves</a> such as simple valves, control valves, damper valves, etc in different sizes and orientations.

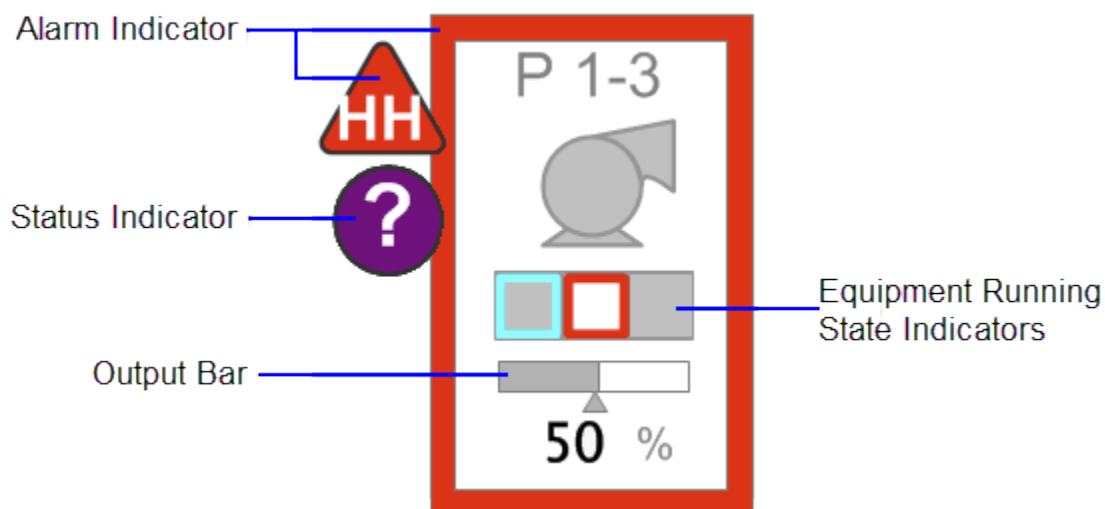
## See Also

[Common Object Elements](#)

[Show/Hide Settings](#)

## Common Object Elements

The objects within the Situational Awareness library share a set of properties and behaviors that are common across objects.



These include:

- [Use Alarm Indicators](#)
- [Status Indicators](#)
- [Output \(OP\) Bar](#)

- Equipment Running State Indicators.

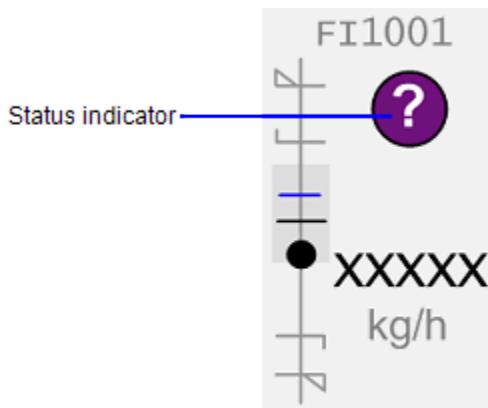
Any exceptions to the properties are noted in the individual object descriptions.

## See Also

[Show/Hide Settings](#)

### Status Indicators

Status indicators are used to represent various non-alarm conditions associated with an object, such as abnormal data quality or control system states. A status indicator will only appear while the triggering condition is true.



The status indicator is a separate genie and is used to represent various states of an object.

The status indicator symbols can be found in the SA\_Library project in a library called sa\_status\_indicator.

### Status Indicator States

Connects to: EquipmentName.EqStatus

A status indicator can represent the following states.

Symbol	State	Name
	Bad data.	d2
	Uncertain data or invalid automation state.	d1
	Simulate mode - forcing a value to a point.	d4
	Offscale.	d12

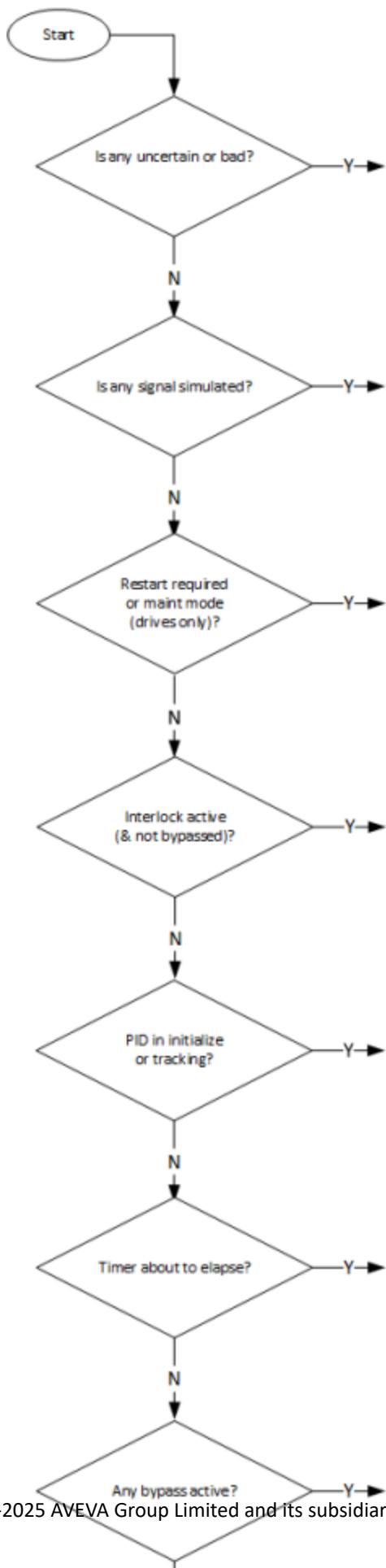
Symbol	State	Name
	Interlocked.	d6
	Interlock bypassed.	d5
	Paused.	d19
	Bad Communication between IO Server and Client	d13
	Tracking	d20
	Clamped	d21
	Off-spec lab data.	d7
	Maintenance.	d8
	Message.	d9
	Restart of the machine is required.	d11
	Initialization manual or index car.	d3
	Timer is active.	d10
	Calibrate mode. Used to calibrate instruments.	d16
	Bad Communication between Gateway and PLC.	d14

Symbol	State	Name
	Deviation	d15
	Reminder to call outside personnel	d17

In scenarios where several states may be active at the same time there is an order of precedence on what indicator will be displayed. For example, a PID controller that is in initialize mode, but also has the PV as simulated would display the "S" indicator, even if there was also an interlock condition active.

## Status Indicator Order of Precedence

The following diagram outlines the order of precedence:



Use this order (first type of failure found) →



Bad comms  
between IO  
Server and  
Client



Bad comms  
between PLC  
and IO  
Client



Bad comms  
between  
Gateway and  
PLC



Bad comms  
between  
Instrument  
and Gateway



Simulate



Calibrate



Restart  
Required



Out to  
Maintenance



Interlocked

If item has Equipment Running State Indicators,  
the lock icon will also be in the Equipment Running  
State Indicator.



Initialize



Tracking



Timer Notice



Interlock  
Bypass Active

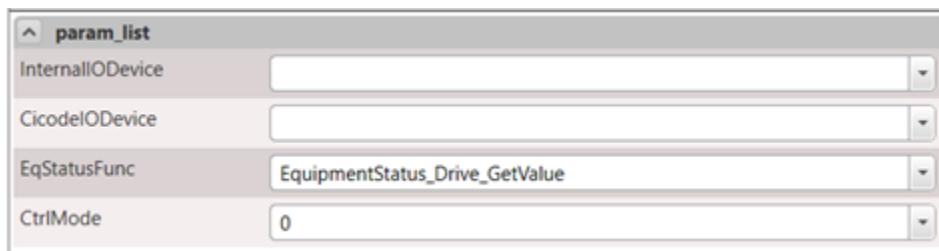
Status Indicators can be configured for most library objects excluding NumericInput and TextInput. Examples have been provided for Drives, Meters and Valves only.

## Customizing Equipment to use Status Indicators

Status Indicators can be used to highlight a non-alarm state of an object or equipment. You can configure equipment to use status indicators one of two ways:

1. Using the Equipment Editor

- Click on the Equipment Tab and from the list of Equipment select the relevant equipment.
- Open the param\_list panel
- Edit the EqStatusFunc field



2. If not using templates, you can configure the status indicator directly using the EqStatus item name of the equipment. Go to System Model -> Variables.

The SA\_Style\_1\_MultiRes starter project includes an example (Indicators.ci) of how to configure equipment using sample functions to call one of the indicators. Variations of the equipment status indicator functions have been provided for Meters, Valves and Drives only.

- [EquipmentStatus\\_Meter\\_GetValue](#)
- [EquipmentStatus\\_Valve\\_GetValue](#)
- [EquipmentStatus\\_Drive\\_GetValue](#)

These functions map the relevant status tags from a PLC's code into the values required by the various indicators. Different pieces of equipment have different relevant states for each of these indicators, so when required, make your own copy of the example Cicode file. You can then create variations or unique versions of the sample functions to use in your project.

Refer to the comments within the Cicode file Indicators.ci for more information on how to customize and use these functions.

## See Also

[Alarm Indicator](#)

## Output (OP) Bar

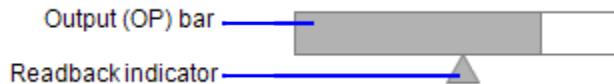
An output (OP) bar provides a visual representation of the current output for a piece of equipment. It can be used with multiple types of objects (control meters, variable speed drives, and control valves).

The information displayed by an OP bar may be presented as two values:

- Output — the value set for the output
- Actual output — the current output value indicated by the readback indicator (also referred to as feedback indicator).

Using a furnace as an example, the output would be the temperature at which the furnace is set to operate. The actual output would be the current temperature of the furnace as it moves towards the specified setting.

The output is represented by the main OP bar, while actual output is represented by a readback indicator.

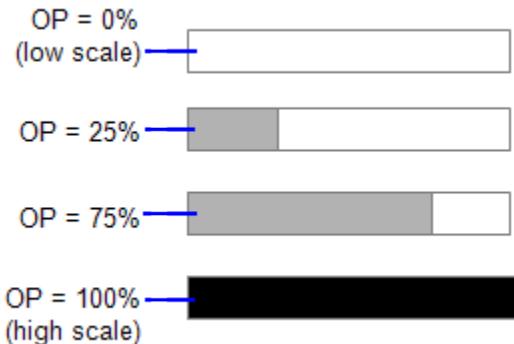


- **Output (OP) bar**

The outer rectangle represents the entire range of the output, from the low scale value to the high scale value (for example, 0—100%).

The following colors are used to represent the OP value:

- White represents 0% (or the low scale value)
- Black represents 100% (or the high scale value)
- Grey represents all output values in between.



---

**Note:** In the case of valves, the valve head will remain grey, but the OP bar will change to black when the OP reaches 100%.

- **Readback Indicator**

The readback indicator (also known as a feedback indicator) is a small filled triangle that is used to show:

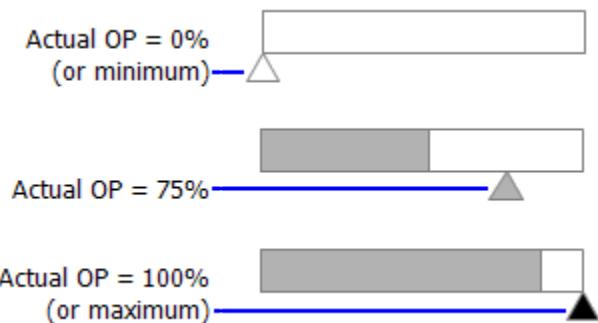
- Actual output (for controllers)
- Speed (for variable speed drives)
- Position (for valves).

If this information is not available from the control system, the readback indicator is not displayed.

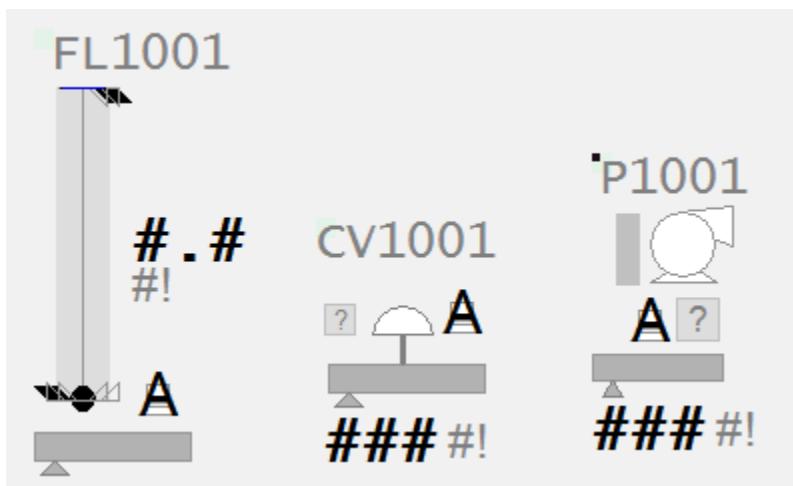
The readback indicator displays the following colors:

- White at 0% (or minimum)
- Black at 100% (or maximum)

- Grey for all values in between.



An OP bar can be specified in the Presentation Options dialog box for meters, drives and valves, but the behavior is the same in all instances. The following examples demonstrate positioning for an OP bar on a control meter, a control valve and a VSD pump.

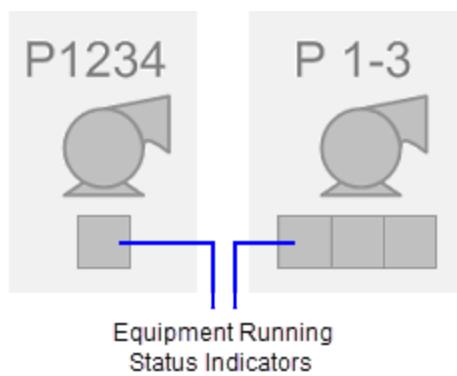


## See Also

[Common Object Elements](#)

## Equipment Running State Indicators

An Equipment Running State Indicator is a compact indicator that can be used to represent a variety of states for drive objects and some types of valves. These states can include run status, direction, interlocks, alarms, lockouts, and more.

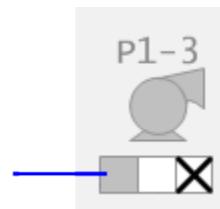


You can use Equipment Running State Indicator in the following ways:

- Represent a group of up to five drives using one symbol.**

A primary reason for using Equipment Running State Indicators is to allow equipment to be displayed on a page as group. For example, it is possible to represent a group of up to five pumps using one pump symbol and four Equipment Running State Indicators.

Equipment Running Status Indicator displaying the current state of three grouped pumps

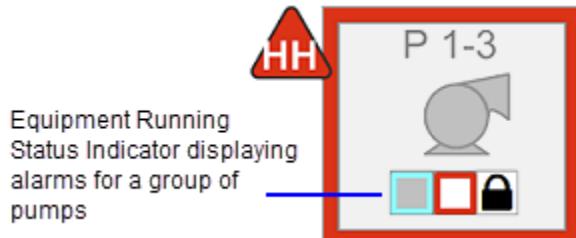


This allows an entire group of drives to be monitored using one compact object.

- Display alarms states for a group of drives.**

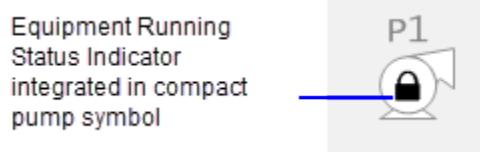
When used with a symbol representing a group of drives, an Equipment Running State Indicator will reflect the occurrence of any alarms on an individual drive within the group. An alarm condition is indicated by a border inside the Equipment Running State Indicator, allowing the status of the drive to remain visible.

The alarm status for the main object will reflect the highest priority alarm on any of the individual drives.



- Integrate an Equipment Running State Indicator into a compact drive symbol.**

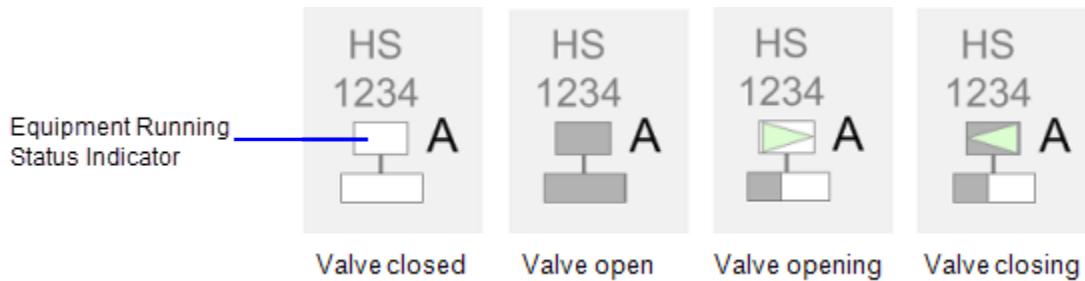
If available space on the screen is limited, compact drives can be used. Instead of showing a square Equipment Running State Indicator box under the symbol, a circular Equipment Running State Indicator is placed inside the symbol.



The Equipment Running State Indicator does not have a border around it. If the commanded and actual states are the same, the Equipment Running State Indicator circle blends into the fill of the symbol.

- **Represent the current state of valves.**

For valves, the Equipment Running State Indicator symbols are shown in the valve head instead of a separate box.



## See Also

[Equipment Running States](#)

### Equipment Running States

Equipment Running State Indicators use the following symbols to indicate the operational state of a piece of equipment. The RunStatus tag is used for indicating out-of-service states with Equipment Running State value of 2, 3 or 4.

OOS and OOSDisabled are digital tags that drive the value of equipment running states.

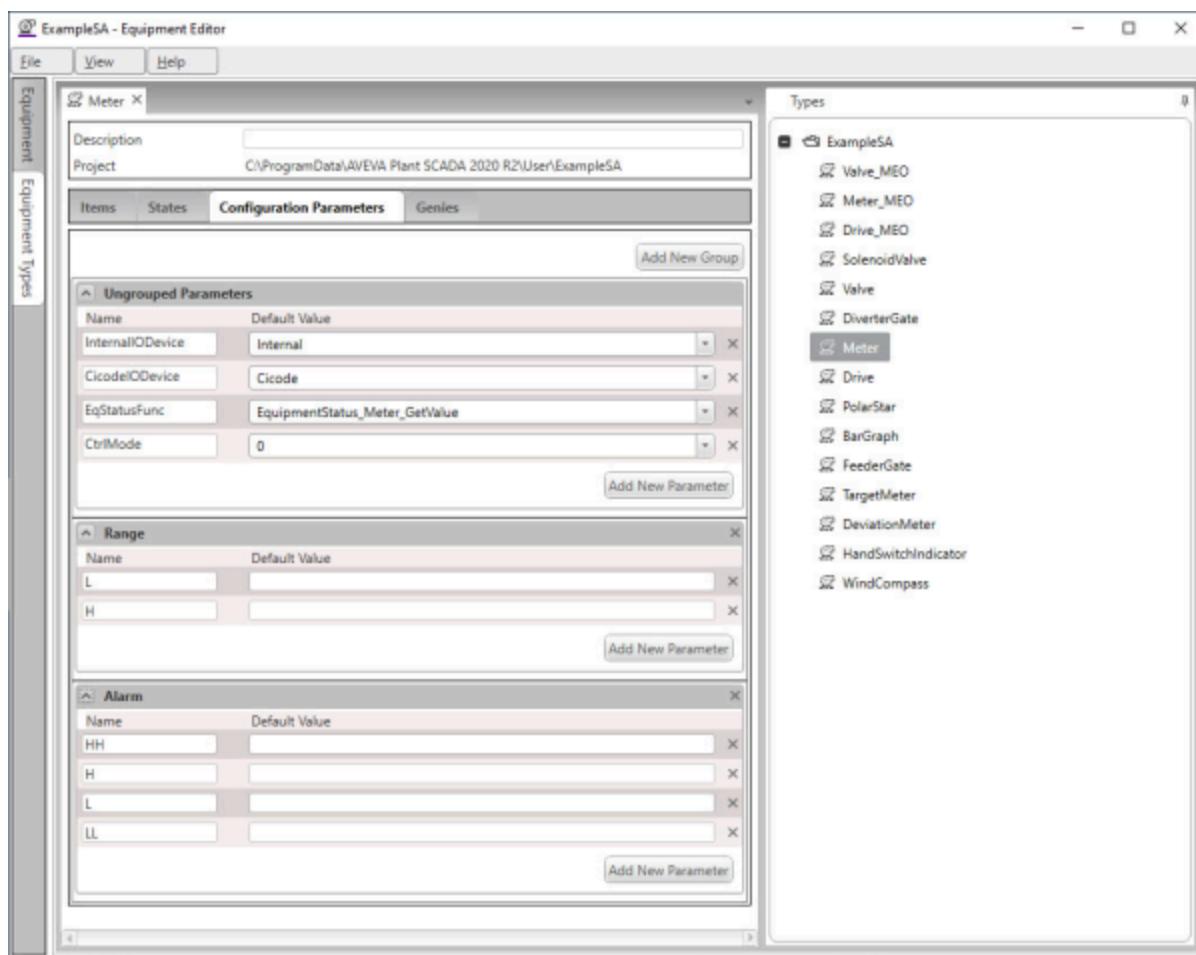
Equipment Running State	Symbol	Description
0	?	Default
1		Mnemonic
2	X	Mnemonic Out of Service - set manually
3	XX	Out To Maintenance / Out Of Service / LOTO (Unavailable)
4	XX	Off / Closed and Selected Out of Service by upstream equipment
5		Off / Stopped and available (Drives) Fully closed (Valves)

Equipment Running State	Symbol	Description
6		Drive running as standby drive in a duty standby arrangement
7		Running Forward (Single Speed Drive) Running Forward Slow Speed (Dual Speed Drive) Valve Open (MOV) Valve stopped midway (complex MOV)
8		Running Reverse (Single Speed Drive) Running Reverse Slow Speed (Dual Speed Drive)
9		Running Forward at Full Speed (Dual Speed Drives) Valve Fully Open
10		Running reverse at Full Speed (Dual Speed Drives)
11		Interlock bypassed and off/ stopped/closed
12		Interlocked and off/stopped/closed
13		Interlock bypassed and on/ running/open
14		Interlocked and on/running/open
15		Running Forward at Full Speed (Dual Speed Drives) with an Interlock Bypass Active.
16		Fully Open and Interlocked (Fail Open Valves).
17		State changing from Stopped to Running (Single speed Drive) State changing from Stopped to Running Slow (Dual speed Drive)

Equipment Running State	Symbol	Description
		Valve requested to open but still at closed limit
18		State changing from Running Slow to Running Fast (Dual Speed Drive) Valve requested to open and has left the closed limit
19		State changing from Running to Stopped (Single Speed Drive) State changing from Running Slow to Stopped (Dual Speed Drive) Valve requested to close and has left the open limit
20		State changing from Running Fast to Running Slow (Dual Speed Drive) Valve requested to close but still at open limit
21		Startup Warning Siren
22		Interlocked by a sequence and not running
23		On / Open and Out to Maintenance / Out of Service
24		Off / Closed and Selected Out of Service by upstream equipment

## Equipment Configuration Parameters

The following equipment configuration parameters defined in the equipment template are applicable to most meters, drives and valves. They are configured on each equipment instance created in the Equipment Editor. Any exception is noted in the parameter description.



- **InternalIODevice**

Name of the persisted memory PLC that stores the value of internal tags to control the value of a library object. For example, the PRHigh, PRLow, ORHigh and ORLow tags may exist in the PLC

- **CicodeIODevice**

Name of the I/O device that is used to store values for calculated variables. For example, EqStatus.

- **EqStatusFunc**

Name of the Cicode function needed to evaluate the value of the status indicator. For example, EquipmentStatus\_Drive\_GetValue.

- **CtrlMode**

Default value for the control mode, which is a value of 0, 1, 2, 3 or 4 (Automatic, Manual, etc). The CtrlDef defines the current control mode of the equipment set at the PLC level. It takes the same values as the CtrlMode.

- **Range**

Used to specify the engineering high and low values for tags that have integer values. For example, PV, PVTrack, SP, PRHigh, PRLow, ORHigh, ORLow.

Note that most tags need to have the same range values.

- **Alarm Limits**

Used to define alarm limits if using analog alarms.

---

**Note:** This parameter is not available for valve objects.

- **Height**

Used only for a Feeder Gate, this is used to specify the engineering range for the height of the gate.

- **WindSpeedRange**

Used only for a Wind Compass, this is used to specify the high and low values for wind speed tags.

- **SwitchPosition**

Used only for a Handswitch Selector, this tag indicates the position of the handswitch - whether it is left (1), right (2) or center (3).

## Show/Hide Settings

Independent controls are provided to show/hide the following elements globally on a workstation. These elements are available from the Header Bar:

Setting	Applies to	Default Setting
Numeric PV/OP and Units 	<ul style="list-style-type: none"><li>• All meters</li><li>• All VSDs and valves with numeric OP and units</li><li>• Trend objects</li></ul>	Visible
Labels 	All instances	Visible
Control Links 	All control links	Visible

## See Also

[Common Object Elements](#)

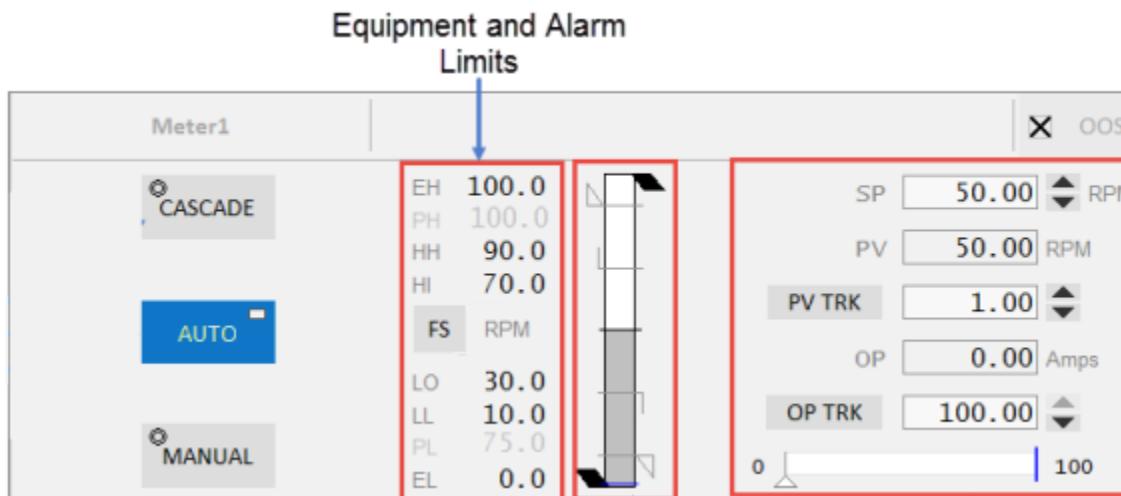
## Alarm Limits

Alarm limits indicate when a process variable will trigger an alarm in the system. The Situational Awareness support in Plant SCADA allows these limits to be visualized using the Meter Composite Genie library object and its associated faceplates.

The Meter Composite Genie and faceplates that display meters support:

- Analog alarm tag limits (a single analog alarm defining all the limits) and
- PLC tag-based alarm limits (separate I/O tags to define each limit).

Tag-based alarm limits are particularly useful if you want dynamic control of the limits and/or if you want to specify a different category for each limit.



**Note:** Up to 14 limits may be defined, but the Meter object supports display of only 4 limits. However, you can build your own meter object with 6 limits for example.

By default, all analog equipment and your project are configured to use analog alarm limits. In most situations, a meter will be displaying the ".PV" item for your equipment. The meter objects will automatically source your HighHigh, High, Low and LowLow values from a matching PV analog alarm tag.

When using Situational Awareness-based projects, this behavior can be changed in the following ways:

1. Use analog alarm limits for all equipment
2. Use analog alarm limits for most equipment, and override with PLC-based limits on a per equipment basis
3. Use PLC-based alarm limits for all equipment
4. Use PLC-based alarm limits for most equipment, and override with analog alarm limits on a per equipment basis

The default you choose will depend upon:

- The number of analog equipment in your system that will use one or the other method
- The importance of the limits in your system. Analog Alarm Limits in Plant SCADA all share the same category. If you want a different category for HH as opposed to LL, PLC-based alarm limits should be used.

## See Also

[Associate an Alarm with Equipment](#)

[Configure PLC Limits with Equipment](#)

## Configure PLC Limits with Equipment

The steps for setting up and using PLC limits are:

## 1. Choose the default mode for limits (PLC tags or analog alarm limits).

To use PLC limits for all meters, set the INI parameter [\[SA\\_Library.Meter\]UseDefaultPLCLimits](#).

To override the behavior of this parameter per equipment, add an equipment runtime parameter. For more details, see Step 5 below.

## 2. Define the Equipment Items Names that the meter limits will use as the source in [\[SA\\_Library.Meter\]PLCLimitNames](#).

When PLC limits are used, you need to configure a set of custom Equipment Item names representing each of the limits. In the **Setup** activity, **Parameters** view, add the tags (item names) as a comma separated list as shown below.

Section Name	Name	Value
SA_Library.Meter	PLCLimitNames	LLLimit,LLimit,HLimit,HHLimit

The meter library objects will then look for items with these names on your equipment.

As these items will need to exist alongside your equipment's other items (for example, PV), it is recommended to use an identifier which makes it clear that they are limit items (for example, Limit).

**Note:** Up to 14 limits may be defined, but the Meter object supports display of only 4. However, you can build your own meter object with 6 limits for example.

## 3. Define the Analog Limit Labels to display in faceplates

Configure the limit labels for each of your limits (for example, HH). The limit labels are defined at a global level as a comma separated list via the parameter [\[SA\\_Library.Meter\]PLCLimitLabels](#).

Row	Section Name	Name	Value	Comment	Project
55	SA_Library.Meter	PLCLimitLabels	LL,L,H,HH		ExampleSA

**Note:** The limits should be listed in the same order as the item names, and the number of items defined should be the same.

## 4. Define the default Analog Limits

Define the default PLC alarm limits to use for analog equipment. These are defined at a global level as a comma-separated list via the parameter [\[SA\\_Library.Meter\]DefaultPLCLimits](#).

Typically, you would assign 0 = LL, 1 = L, 2 = H and 3 = HH. This matches the order in which the limit labels are specified. If you wanted to define only H and HH as the default, you would set the **Value** to -1, -1, 2, 3.

**Note:** The -1 in the **Value** column means "ignore" the first default limits.

## 5. Configure the "PLCLimits" equipment runtime parameter

When the alarm limits need to differ for a specific piece of equipment, the **PLCLimits** equipment runtime parameter can be configured to override the default.

For example, if only H and HH are required, then this can be configured using Equipment Runtime Parameters as shown below .

▶	5	Cluster1	MyPlant.TS001	PLCLimits	-1,-1,2,3	FALSE	ExampleSA
---	---	----------	---------------	-----------	-----------	-------	-----------

To configure this:

1. In Plant SCADA Studio, navigate to **System Model** activity and select **Equipment**. Select **Runtime Parameters** from the drop-down menu.
2. For each equipment that needs to override the default limits, in the **Name** property, type "PLCLimits".
3. In the **Value** property, specify your comma-separated list of default PLC limits, that is the order in which the defined PLC alarm limits need to be used. Use the value -1 to skip limits that are not required.

**Note:** Setting the **Value** property to \* reverses the setting of the UseDefaultPLCLimits parameter. Therefore, if the UseDefaultPLCLimits parameter is set to TRUE, it will be reversed and applied as if it were set to FALSE. In this case, analog alarm limits will apply. If UseDefaultPLCLimits is FALSE, setting the value of the runtime parameter to \* will cause the associated equipment to use the default PLC limits defined at the project level.

4. Set the **Is Tag** property to FALSE so that the PLC limit labels are displayed on the meter faceplates.
5. Click **Save**.

## 6. Generate variable tag Items for each limit item on your analog equipment.

All analog equipment that will be displayed as meters will then need to be configured to include variable items for these limits. Your PLC code can then set the defaults for those limits, or it could be done via Cicode.

Variables						
Row	Equipment	Item Name	Tag Name	Cluster Name	I/O Device	Data Type
1	Meter1	CHHLimit	Meter1_CHHLimit	Cluster1	PLC	REAL
2	Meter1	CHLimit	Meter1_CHLimit	Cluster1	PLC	REAL
3	Meter1	CLLimit	Meter1_CLLimit	Cluster1	PLC	REAL
4	Meter1	CLLLimit	Meter1 CLLLimit	Cluster1	PLC	REAL

## 7. Generate variable tags representing the condition of the PV against each limit

Once limits are set up, create a digital variable to reflect when the PV signal exceeds the limit.

For the variable, this can either be driven from the PLC itself, or, you can use a [Calculated Variables](#) that will run on the I/O Server as shown below:

Variables						
Row	Equipment	Item Name	Tag Name	Cluster Name	I/O Device	Address
1	Meter1	CHH	Meter1_CHH	Cluster1	Cicode	Meter1_PV >= Meter1_CHHLimit
2	Meter1	CH	Meter1_CH	Cluster1	Cicode	Meter1_PV >= Meter1_CHLimit
3	Meter1	CL	Meter1_CL	Cluster1	Cicode	Meter1_PV <= Meter1_CLLimit
4	Meter1	CLL	Meter1 CLL	Cluster1	Cicode	Meter1_PV <= Meter1 CLLLimit

## 8. Generate digital alarm tags.

Digital Alarms can then be created for each limit item giving you granular control of alarm priority (category).

Each alarm should be bound to their respective digital condition tag defined in Step 7.

## See Also

[Alarm Limits](#)

[Add Equipment Using Equipment Editor](#)

[Define Equipment Runtime Parameters](#)

## Meters

Meters provide a graphical and a numeric representation of instrument readings. The graphical representation allows you to visually compare the PV with reference values such as alarm limits and/or user-configured reference values (e.g., the "tracker" or optimal range).

Many of the meters have variations that allow them to also be used to represent a controller. These variations include additional elements such as output (OP) and feedback indicator (FB).

You can insert multiple instances of a meter in duplicate or triplicate by using the [Multiple Meter](#).

Meters are associated with equipment, and require the following items to be defined:

Item Name	Description
FB	Feedback Indicator See <a href="#">Control Meters - Common Elements</a> .
OP	Output See <a href="#">Control Meters - Common Elements</a> .
ORDsp	Optimal Range Display See <a href="#">Meter Common Elements</a> .
ORLow	Optimal Range (OR) Low See <a href="#">Meter Common Elements</a> .
ORHigh	Optimal Range (OR) High See <a href="#">Meter Common Elements</a> .
PR	Practical Range specific to Deviation meters See <a href="#">Meter Common Elements</a> .
PRLow	Practical Range (PR) Low See <a href="#">Meter Common Elements</a> .
PRHigh	Practical Range (PR) High See <a href="#">Meter Common Elements</a> .
PV	Process Variable

Item Name	Description
	See <a href="#">Meter Common Elements</a> .
PVTarget	Process Variable specific to Target meters See <a href="#">Meter Common Elements</a> .
PVTrack	Process ValueTracker See <a href="#">Meter Common Elements</a> .
SP	Setpoint See <a href="#">Control Meters - Common Elements</a> .
TrackDsp	Controls whether or not a tracker is displayed See <a href="#">Meter Common Elements</a> .

The Meter objects available in the Situational Awareness library include:

- [Analyzer Meter](#)
- [Deviation Meter](#)
- [Dual Level Meter](#)
- [Flow Meter](#)
- [Level Meter](#)
- [Miscellaneous Meter](#)
- [Pressure Meter](#)
- [Target Meter](#)
- [Temperature Meter](#)

## See Also

[Control Meters - Common Elements](#)

[Meter Common Elements](#)

[Meter Special Elements](#)

[Meter Faceplates](#)

[Multiple Meter](#)

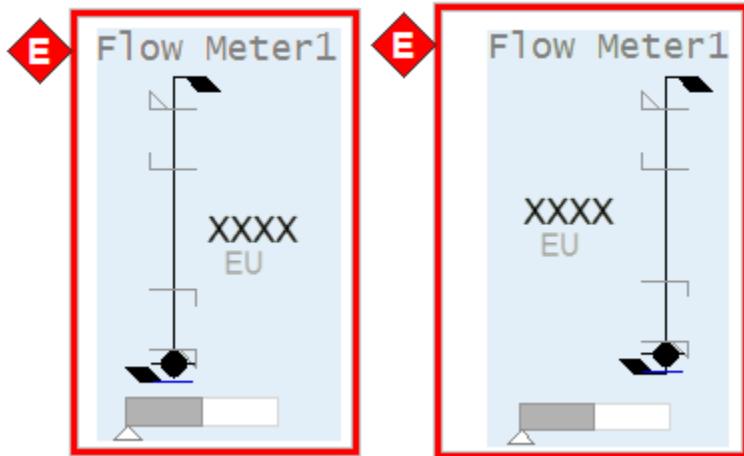
## Meter Orientations

For most meters, several orientations are provided in order to give page designers flexibility in how the graphics are laid out.

## Mirrored

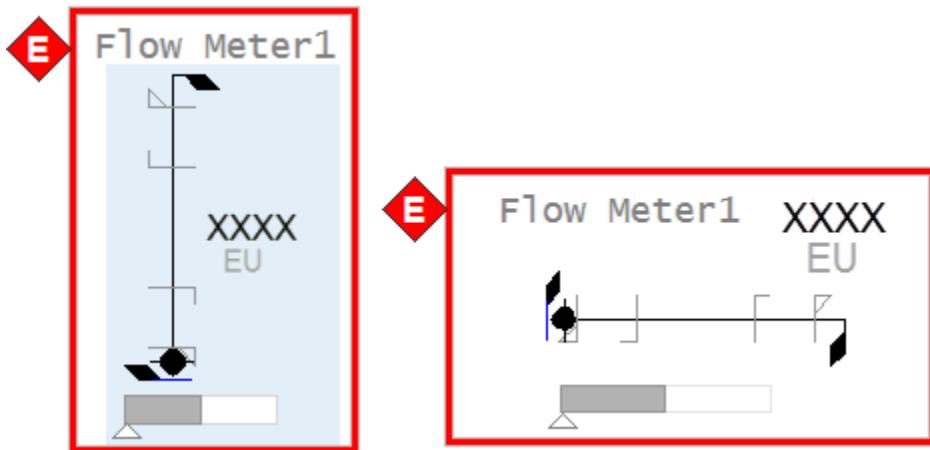
A mirrored meter reverses the left/right position of some meter elements as shown in the example below. The

positions of the running state indicator, rate of change arrow, PV, engineering units, and mode indicator (for control meters) are affected.



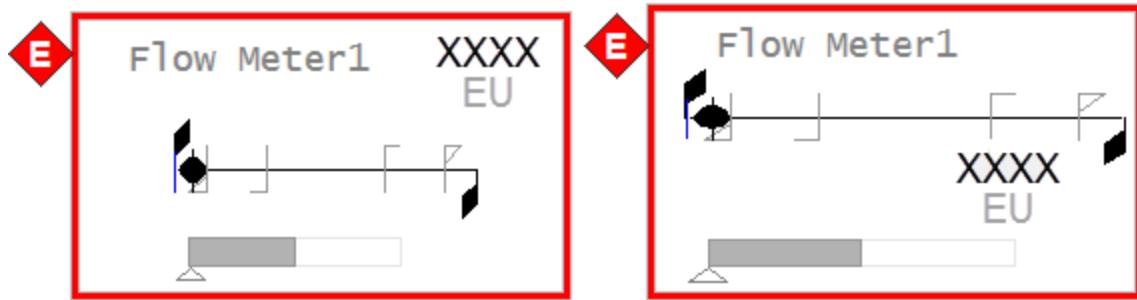
## Horizontal

The orientation of the meter is changed to horizontal as shown in the example below. Note that the proportions of the alarm border, meter symbol and controller output indicator are slightly different for the horizontal version compared to the vertical version.



## Horizontal + Mirrored

Horizontal meters can also be mirrored. Typically, this orientation is used primarily with regular meters rather than control meters.

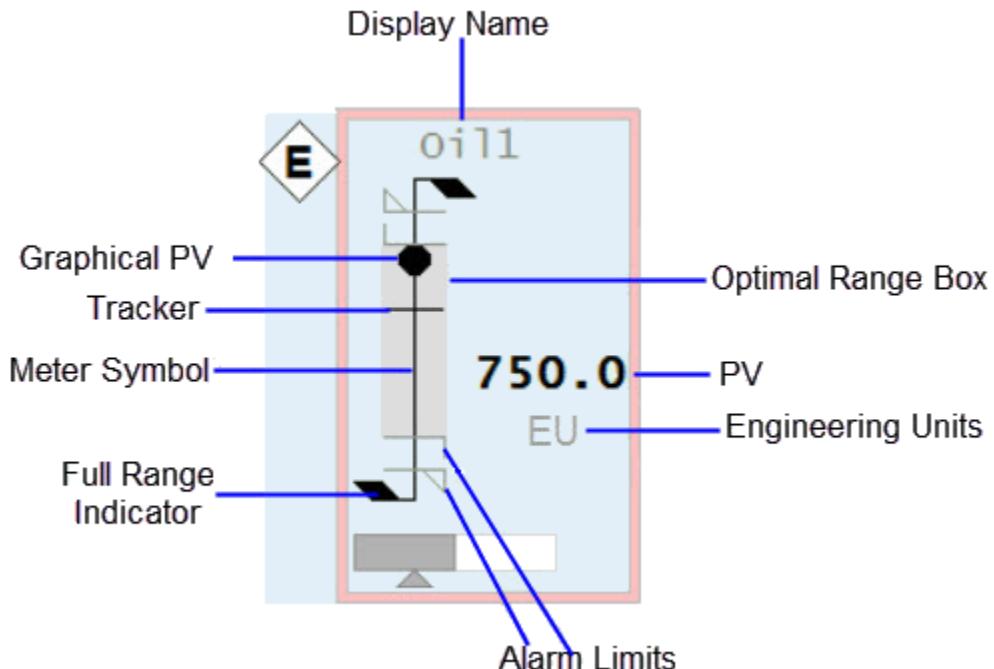


## See Also

[Meters](#)

### Meter Common Elements

The figure below shows an example of the elements that make up a meter.



### Meter Symbol and Graphical PV

Graphical PV connects to: EquipmentName.PV, EquipmentName.PVTarget

The appearance of the meter symbol and graphical PV tells operators what type of reading is being displayed (flow vs. temperature vs. pressure, etc.). For example, flow meters are represented as a basic 'stick' shape, with a black dot as the graphical PV. Temperature meters resemble a thermometer. Level meters resemble a (thin) vessel with a fill that rises and falls to represent the level, and so on. The symbols and PVs for each type of meter are shown in the following image.

Analyzer Meter	Deviation Meter	Flow Meter	Level Meter	Misc Meter	Pressure Meter	Target Meter	Temperature Meter

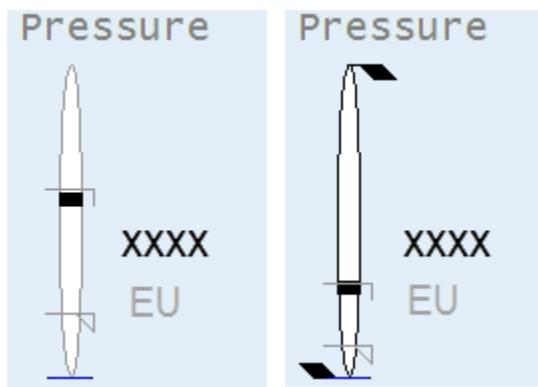
## Full Range Indicators

Connects to: EquipmentName.PRHigh, EquipmentName.PRLow, EquipmentName.PR

Each meter allows you to set a "normal" operating range ("practical" range). During most operating situations, the meter's full range is much bigger than the actual region of interest. By setting the normal operating range you can see meaningful variation in the PV when looking at the meter.

The EquipmentName.PR item is applicable to Deviation meters only. PRHigh and PRLow are not applicable to these meters.

If the PV moves outside the normal operating range, the meter scale will automatically change to show the full range of the instrument. To differentiate a meter that is showing its full range, additional flags appear at the top of the meter and the meter symbol line color changes from grey to black, as in the example below:



## Alarm Limits

Connects to: EquipmentName.PV (analog alarm - current) or (EquipmentName.HHAlm, EquipmentName.HAlm, EquipmentName.LAlm, EquipmentName.LLAlm) or PLC Alarm Limits if configured.

The alarm limit indicators appear on the meter symbol to show the location of any configured alarm limits. This allows operators to see when a PV is approaching an alarm limit. Four types of alarm limits are shown using different symbols.

**Note:** For more information about configuring PLC alarm limits, see [Configure PLC Limits with Equipment](#).

	High high alarm limit
	High alarm limit
	Low alarm limit
	Low Low alarm limit

For more information, see [Alarm Indicator](#).

An alarm limit indicator will not appear if the corresponding alarm limit is not configured for the instrument or if the limit is outside of the range shown by the meter. Note that the precise width of the alarm limit indicators varies somewhat for different meter types in order to accommodate the width of the meter symbol. For example, alarm limit indicators are wider on a level meter than on a flow meter.

## Engineering Units

Connects to: EquipmentName.PV (eng units field).

The engineering units represent the unit of measurement for the instrument.

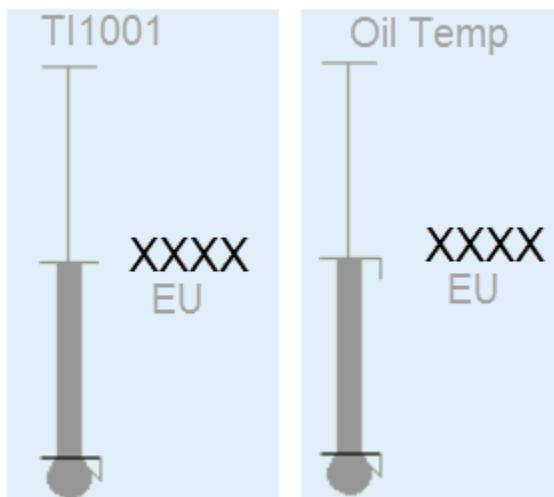
## PV

Connects to: EquipmentName.PV, EquipmentName.PVTarget

The numeric PV provides a text-based representation of the instrument reading. This provides operators with precise values when needed.

## Display Name/Nickname

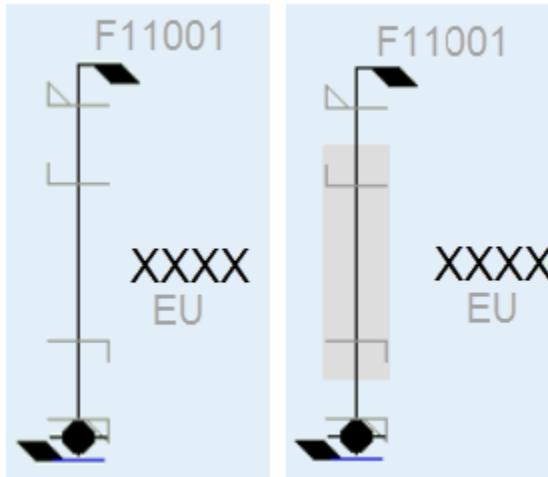
The display name/nickname shows the instrument name (by default), or, if entered by the user, a "display name". The nickname is usually a common name used to help identify the reading (for example, "Oil temp", instead of "TI1001"). The field accommodates up to 79 characters.



## Optimal Range Box

Connects to: EquipmentName.ORHigh, EquipmentName.ORLow, Visibility connected to EquipmentName.ORDsp.

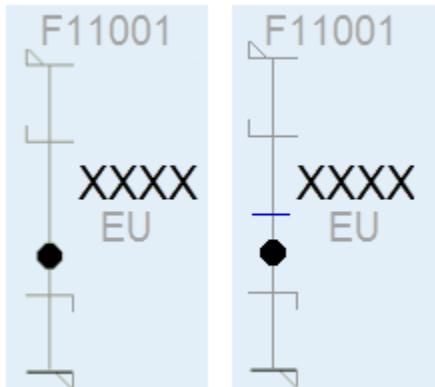
The optimal range box highlights a specific range of the meter. Typically, operators will use this feature to remind themselves of the desired or "optimal" range for the PV, depending on the operating conditions. It should be possible to enable and set the endpoints for the optimal range from the point detail. If the optimal range is not set or enabled, the box is not shown.



## Tracker

Connects to: EquipmentName.PVTrack, EquipmentName.OPTrack, EquipmentName.TrackDsp.

Similar to the optimal range box, the tracker is a user-configured indicator that marks a specific value on the meter. An operator can mark the current value of a PV so that when they look at the reading again later, they can see if it has changed. The tracker can be set and enabled/disabled from the meter faceplate.

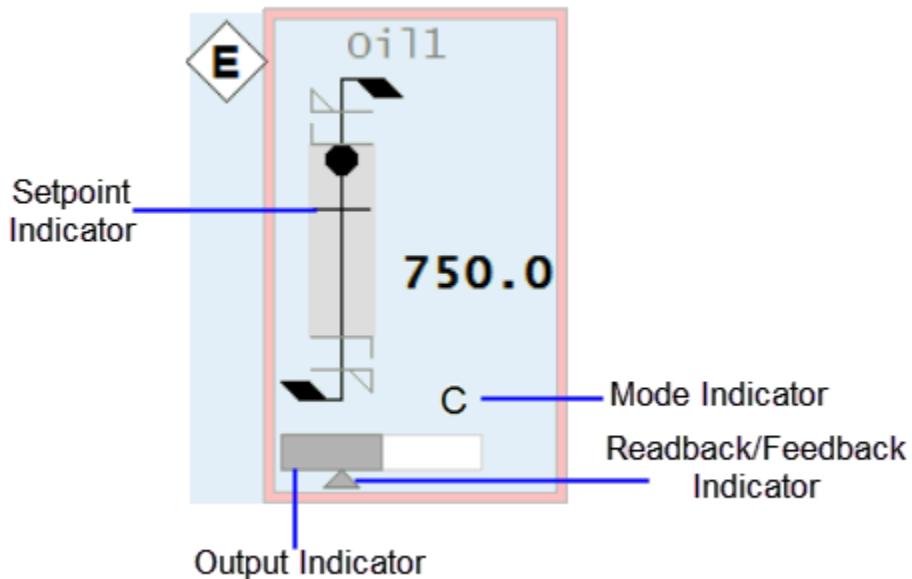


## See Also

- [Meters](#)
- [Alarm Indicator](#)
- [Alarm States](#)
- [Status Indicators](#)

## Control Meters - Common Elements

When a meter is used to show readings from a controller, additional information is included. Control meters include all of the [common elements](#) previously described for regular meters, plus the additional elements shown below.



### Setpoint Indicator

Connects to: EquipmentName.SP

The setpoint indicator is an additional mark on the meter symbol (similar to the tracker and alarm limit indicators) which shows the position of the controller setpoint value.

### Output Indicator

Connects to: EquipmentName.OP

The outer rectangle shape with white fill represents the entire range of the [Output \(OP\) Bar](#) (typically 0-100%), while the darker fill represents the output value.

### Feedback Indicator

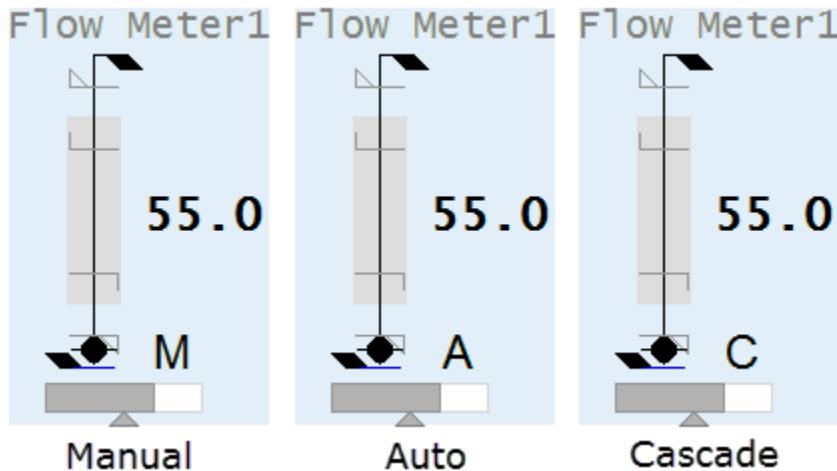
Connects to: EquipmentName.FB

This small triangular indicator represents the actual output for the controller (if available from the control system). It shares the same scale as the controller output indicator (e.g. 0-100%). This allows the operator to quickly see if the output and actual output values match. This can be particularly useful where the controller has a ramping function.

## Mode Indicator

Connects to: EquipmentName.CtrlMode, EquipmentName.CtrlModeDef

This is a single-character code that indicates the mode of the controller. Typically, the set of modes includes Manual (M), Auto (A), and Cascade (C).



**Control Mode Indicator states:**

- 0 – Auto (A)
- 1 – Manual (M)
- 2 – Cascade (C)
- 3 – Local (L)
- 4 – Special control (computer symbol)

---

**Note:** The Mode Indicator is not displayed if the tag is set to CtrlModeDef. To view the control mode at all times, remove the CtrlModeDef tag from the equipment template.

---

## See Also

[Meter Common Elements](#)

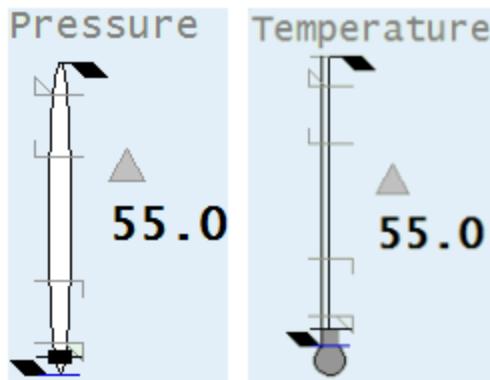
[Meter Special Elements](#)

## Meter Special Elements

Certain meters can have elements in addition to those described in [Meter Common Elements](#). There are three special elements.

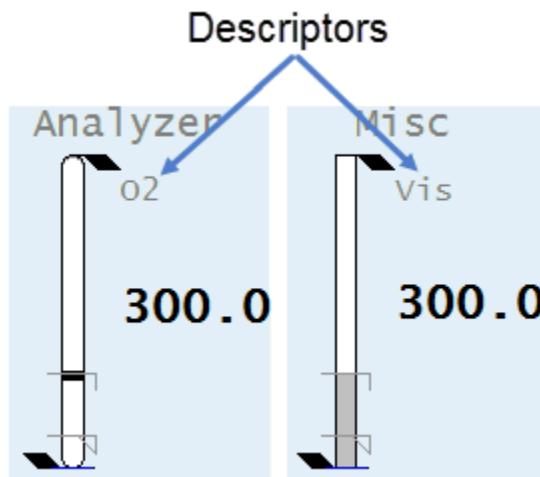
## Differential Indicator

The delta symbol ( $\Delta$ ) is used for temperature and pressure meters to indicate when a differential reading is being displayed. It is a static element which is enabled at configuration and is not connected to the control system. It consists of a simple triangle shape which is placed next to the meter.



## Descriptor

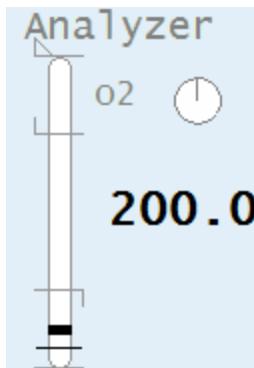
The Miscellaneous, Deviation, and Analyzer meters can all be used to show many different types of readings. Therefore, it is recommended that short text (up to 3 characters) is provided to help identify the reading (e.g., "O2" for Oxygen content, "SG" for density, "Vis" for viscosity, "Trq" for torque, etc.) .



## Clock Timer

Connects to: EquipmentName.Timer, EquipmentName.TimerEXP

The clock timer is an independent object that is integrated with Analyzer meters. It can be used in cases where the analyzer shows an intermittent sample reading. For these types of readings, the Clock Timer provides an indication of how long ago the value was updated. This helps operators to understand how current the value is. If the analyzer reads continuously the clock timer is not shown.



## See Also

[Meter Common Elements](#)

## Analyzer Meter

Property	Description
Name	Analyzer
Graphical Representation	
Example Equipment Template	Meter
Associated Composite Genie	Meter.xml
Equipment.Items that the Genie expects	<b>OP</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>ORHigh, ORLow, ORDsp</b> (see <a href="#">Meter Common Elements</a> ) <b>PRHigh, PRLow</b> (see <a href="#">Meter Common Elements</a> ) <b>PVTrack</b> (see <a href="#">Meter Common Elements</a> ) <b>PV</b> (see <a href="#">Meter Common Elements</a> ) <b>SP</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>FB</b> (see <a href="#">Control Meters - Common Elements</a> )

Property	Description
	<b>CtrlMode</b> , <b>CtrlModeDef</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>RunStatus</b> (see <a href="#">Meter Common Elements</a> ) <b>Status Indicators</b> (see <a href="#">Meter Common Elements</a> ) <b>TrackDsp</b> (see <a href="#">Meter Common Elements</a> ) <b>Timer</b> , <b>TimerExp</b> (see <a href="#">Meter Special Elements</a> )
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	Analog Controller Analog Indicator
Equipment.Items that the Faceplate Expects	<b>OP</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>ORHigh</b> , <b>ORLow</b> , <b>ORDsp</b> (see <a href="#">Meter Common Elements</a> ) <b>PRHigh</b> , <b>PRLow</b> (see <a href="#">Meter Common Elements</a> ) <b>OPTrack</b> , <b>PVTrack</b> , <b>TrackDsp</b> (see <a href="#">Meter Common Elements</a> ) <b>PV</b> (see <a href="#">Meter Common Elements</a> ) <b>SP</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>CtrlMode</b> , <b>CtrlModeDef</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>AutoCmd</b> , <b>ManCmd</b> , <b>CasCmd</b> <b>RunStatus</b> (see <a href="#">Meter Common Elements</a> ) <b>OOS</b> , <b>OOSDisabled</b> (see <a href="#">Equipment Running States</a> )

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated

with the equipment you have defined in your project.

- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Meter Type	From the drop-down select one of the following meters: Level, Flow, Temperature, Pressure, Analyzer, Miscellaneous, Target, Deviation and Dual Level
Equipment Name	Enter a name for the equipment associated with the meter. You can enter a maximum of 160 characters for this option.
Equipment Item Prefix	<p>Prefix applied to the equipment item name. This allows for displaying multiple values for a single piece of equipment. For example, you want to create three meters to monitor power, voltage and current for a single pump. You would create three meters on the pump and assign the equipment item prefix "Power", "Voltage" and "Current". So, the PV values for the meters would be PowerPV, VoltagePV and CurrentPV respectively.</p> <p><b>Note:</b> To get a meter with a prefix to work correctly, you need to create tags with the required item names. From the example above, you will need to create the tags PowerPV, PowerOP, VoltagePV, VoltageOP, CurrentPV, CurrentOP and so on.</p> <p>The following equipment item names (where applicable) can be prefixed:</p> <ul style="list-style-type: none"><li>• FB</li><li>• FullScale</li><li>• OP</li><li>• ORDsp</li><li>• ORLow</li><li>• ORHigh</li><li>• PRLow</li><li>• PRHigh</li></ul>

Option	Description
	<ul style="list-style-type: none"> <li>• PV</li> <li>• PVTrack</li> <li>• PVTarget</li> <li>• SP</li> <li>• Timer</li> <li>• TrackDsp</li> </ul>
Size	Size of the meter object image - small or large
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Use this setting to display a label at the selected position. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm border around the Genie.
Display Alarm Flag	<p>Use this setting to display an alarm flag at the selected position.</p> <p><b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.</p>
Display Status Indicator	<p>Use this setting to display a status indicator at the selected location. Select <b>None</b> if you do not want to display the status indicator. For more information, see <a href="#">Status Indicators</a>.</p>
Display Process Variable	Select to display a process variable (numeric PV) with the meter. This option is selected by default.
Display Control Mode	<p>Select this option to display the Controller <a href="#">Control Meters - Common Elements</a>. On selecting this option the <b>Display Control Readback</b> option box is also displayed.</p> <p>The Display Control <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.</p>

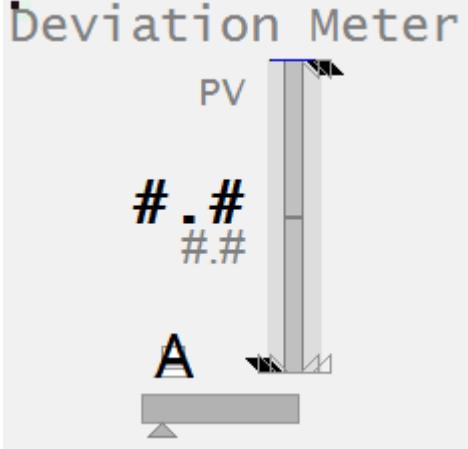
Option	Description
Display Controller Output	Select to display an output indicator with the meter. For more information, see <a href="#">Output (OP) Bar</a> .
Display Setpoint	Select to display a setpoint indicator with the meter.
Display OOS	Select to display Out of Service indicator.
Display Differential Indication	Select to display an indicator when a differential reading is detected. <b>This is visible only on Pressure and Temperature meters.</b> For more information, see <a href="#">Meter Special Elements</a> .
Descriptor	Enter a short 3-character text descriptor to indicate what is being measured. If left blank no descriptor will be displayed. <b>This is visible only on Miscellaneous, Deviation and Analyzer meters.</b>
Display Clock Timer	Select to display a clock timer on the meter. <b>This is visible only on Analyzer meters.</b> For more information, see <a href="#">Meter Special Elements</a> .
Display Control Readback	Select to display a readback indicator (also known as feedback indicator) with the meter.
Display Trend	Select this option to display a <a href="#">Trend Objects</a> object with the meter. On selecting this option, the <b>Trend Type</b> dropdown is displayed. Select either <b>Tail</b> or <b>Full</b> . <b>Note:</b> When the <b>Display Trend</b> option is selected, the Mirrored option is not available.
Trend Type	Select the type of trend to be displayed with the meter: <a href="#">Trend Tail</a> or <a href="#">Full Trend</a> .
Mirrored	Inserts a mirror image of the meter based on the selected presentation options.

## See Also

[Meter Faceplates](#)

## Deviation Meter

The deviation meter is used to represent the difference between two values. A positive difference is shown as a bar extending upwards from the middle of the meter symbol. A negative difference extends downwards. The configured input range for the deviation meter needs to be in equal proportions around a zero (for example -200 to +200 or -50 to +50). The PV will be shown as a deviation from setpoint, where zero=setpoint. If the PV exceeds this range, the color of the bar will change to black.

Property	Description
Name	Deviation
Graphical Representation	
Example Equipment Template	DeviationMeter
Associated Composite Genie	Meter.xml
Equipment.Items that the Genie expects	<p><b>OP</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>ORHigh, ORLow, ORDsp</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>PRHigh, PRLow</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>PVTrack</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>PV</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>SP</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>FB</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>RunStatus</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>EqStatus</b> (see <a href="#">Status Indicators</a>)</p> <p><b>TrackDsp</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>Timer, TimerExp</b> (see <a href="#">Meter Special Elements</a>)</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .

Property	Description
Associated Faceplate(s)	<a href="#">Analog Controller</a> <a href="#">Analog Indicator</a>
Equipment.Items that the Faceplate Expects	<b>OP</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>ORHigh</b> , <b>ORLow</b> , <b>ORDsp</b> (see <a href="#">Meter Common Elements</a> ) <b>PRHigh</b> , <b>PRLow</b> (see <a href="#">Meter Common Elements</a> ) <b>OPTrack</b> , <b>PVTrack</b> , <b>TrackDsp</b> (see <a href="#">Meter Common Elements</a> ) <b>PV</b> (see <a href="#">Meter Common Elements</a> ) <b>SP</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>CtrlMode</b> , <b>CtrlModeDef</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>AutoCmd</b> , <b>ManCmd</b> , <b>CasCmd</b> <b>RunStatus</b> (see <a href="#">Meter Common Elements</a> ) <b>OOS</b> , <b>OOSDisabled</b> (see <a href="#">Equipment Running States</a> )

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Meter Type	From the drop-down select one of the following meters: Level, Flow, Temperature, Pressure, Analyzer, Miscellaneous, Target, Deviation and Dual Level
Equipment Name	Enter a name for the equipment associated with the

Option	Description
	meter. You can enter a maximum of 160 characters for this option.
Equipment Item Prefix	<p>Prefix applied to the equipment item name. This allows for displaying multiple values for a single piece of equipment. For example, you want to create three meters to monitor power, voltage and current for a single pump. You would create three meters on the pump and assign the equipment item prefix "Power", "Voltage" and "Current". So, the PV values for the meters would be PowerPV, VoltagePV and CurrentPV respectively.</p> <p><b>Note:</b> To get a meter with a prefix to work correctly, you need to create tags with the required item names. From the example above, you will need to create the tags PowerPV, PowerOP, VoltagePV, VoltageOP, CurrentPV, CurrentOP and so on.</p> <p>The following equipment item names (where applicable) can be prefixed:</p> <ul style="list-style-type: none"> <li>• FB</li> <li>• FullScale</li> <li>• OP</li> <li>• ORDsp</li> <li>• ORLow</li> <li>• ORHigh</li> <li>• PRLow</li> <li>• PRHigh</li> <li>• PV</li> <li>• PVTrack</li> <li>• PVTarget</li> <li>• SP</li> <li>• Timer</li> <li>• TrackDsp</li> </ul>
Size	Size of the meter object image - small or large
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Use this setting to display a label at the selected position. Select <b>None</b> if you do not want to display a

Option	Description
	label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm border around the Genie.
Display Alarm Flag	<p>Use this setting to display an alarm flag at the selected position.</p> <p><b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.</p>
Display Status Indicator	Use this setting to display a status indicator at the selected location. Select <b>None</b> if you do not want to display the status indicator. For more information, see <a href="#">Status Indicators</a> .
Display Process Variable	Select to display a process variable (numeric PV) with the meter. This option is selected by default.
Display Control Mode	<p>Select this option to display the Controller <a href="#">Control Meters - Common Elements</a>. On selecting this option the <b>Display Control Readback</b> option box is also displayed.</p> <p>The Display Control <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.</p>
Display Controller Output	Select to display an output indicator with the meter. For more information, see <a href="#">Output (OP) Bar</a> .
Display Setpoint	Select to display a setpoint indicator with the meter.
Display OOS	Select to display Out of Service indicator.
Display Differential Indication	Select to display an indicator when a differential reading is detected. <b>This is visible only on Pressure and Temperature meters.</b> For more information, see <a href="#">Meter Special Elements</a> .
Descriptor	Enter a short 3-character text descriptor to indicate what is being measured. If left blank no descriptor will be displayed. <b>This is visible only on Miscellaneous, Deviation and Analyzer meters.</b>

Option	Description
Display Clock Timer	Select to display a clock timer on the meter. <b>This is visible only on Analyzer meters.</b> For more information, see <a href="#">Meter Special Elements</a> .
Display Control Readback	Select to display a readback indicator (also known as feedback indicator) with the meter.
Display Trend	Select this option to display a <a href="#">Trend Objects</a> object with the meter. On selecting this option, the <b>Trend Type</b> dropdown is displayed. Select either <b>Tail</b> or <b>Full</b> . <b>Note:</b> When the <b>Display Trend</b> option is selected, the Mirrored option is not available.
Trend Type	Select the type of trend to be displayed with the meter: <a href="#">Trend Tail</a> or <a href="#">Full Trend</a> .
Mirrored	Inserts a mirror image of the meter based on the selected presentation options.

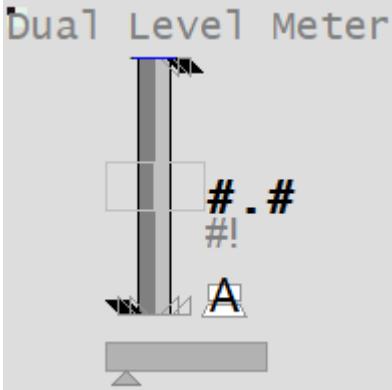
## See Also

[Meter Faceplates](#)

## Dual Level Meter

This object provides a compact way to monitor two related levels simultaneously. The level indication is presented on a linear scale. For example, you could use a Dual Level meter to indicate the levels of two tanks. One of the tanks is shown in a slightly darker shade of gray than the other. The Dual Level meter is designed to display the values of two levels that share a common set of scales (engineering range, practical range and alarm limits).

**Note:** If you need to display the values of two levels with different scales, use the Multiple Meter.

Property	Description
Name	Dual Level
Graphical Representation	
Example Equipment Template	Meter
Associated Composite Genie	Meter.xml
Equipment.Items that the Genie expects	<p><b>OP</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>ORHigh, ORLow, ORDsp</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>PRHigh, PRLow</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>PV</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>SP</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>FB</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>RunStatus</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>EqStatus</b> (see <a href="#">Status Indicators</a>)</p> <p><b>TrackDsp</b> (see <a href="#">Meter Common Elements</a>)</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	Analog Controller Analog Indicator
Equipment.Items that the Faceplate Expects	<b>OP</b> (see <a href="#">Control Meters - Common Elements</a> )

Property	Description
	<b>ORHigh, ORLow, ORDsp</b> (see <a href="#">Meter Common Elements</a> ) <b>PRHigh, PRLow</b> (see <a href="#">Meter Common Elements</a> ) <b>OPTrack, PVTrack, TrackDsp</b> (see <a href="#">Meter Common Elements</a> ) <b>PV</b> (see <a href="#">Meter Common Elements</a> ) <b>SP</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>AutoCmd, ManCmd, CasCmd</b> <b>RunStatus</b> (see <a href="#">Meter Common Elements</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> )

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

**Note:** It is recommended that for a Dual Level meter you set these values on both meters to the same value.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Meter Type	From the dropdown list, select Dual Level.
Reference Equipment Name	Enter the full name and path for the equipment associated with the dual meter. The path and name should be prefixed with the equipment's cluster name for multi-cluster systems. You can enter a maximum of 160 characters for this option.

Option	Description
Equipment Name and Equipment #2 Name	Enter the full name and path of the two equipment objects. You can enter a maximum of 160 characters for this option.
Equipment Item Prefix and Equipment #2 Item Prefix	<p>Prefix applied to the equipment item name. This allows for displaying multiple values for a single piece of equipment. For example, you want to create two meters to monitor power or voltage for a single pump. You would create two meters on the pump and assign the equipment item prefix "Power1" and "Power2" or "Voltage1" and "Voltage2". So, the PV values for the meters would be Power1PV, Power2PV, etc, respectively.</p> <p><b>Note:</b> To get a meter with a prefix to work correctly, you need to create tags with the required item names. From the example above, you will need to create the tags Power1PV, Power2PV and so on.</p> <p>The following equipment item names can be prefixed:</p> <ul style="list-style-type: none"> <li>• PV</li> <li>• OP</li> <li>• ORDsp</li> <li>• TrackDsp</li> <li>• FB</li> <li>• PRLow</li> <li>• PRHigh</li> <li>• ORLow</li> <li>• ORHigh</li> <li>• SP</li> <li>• PVTrack</li> <li>• FullScale</li> <li>• OPTTrack</li> <li>• Timer</li> <li>• PVTarget</li> </ul>
Size	Size of the meter object image - small or large
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Use this setting to display a label at the selected position.

Option	Description
	Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Status Indicator	Select the location to display a status indicator. Select None if you do not wish to see the status indicator. For more information, see <a href="#">Status Indicators</a> .
Display Process Variable	Select to display a process variable (numeric PV) with the meter. This option is selected by default.
Display Control Mode	<p>Select this option to display the Controller <a href="#">Control Meters - Common Elements</a>. On selecting this option the "Display Control Readback" option box is also displayed.</p> <p>The Display Control <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.</p>
Display Controller Output	Select to display an output indicator with the meter. For more information, see <a href="#">Output (OP) Bar</a> .
Display Setpoint	Select to display a setpoint indicator with the meter.
Display OOS	Select to display Out of Service indicator.
Display Control Readback	Select to display a readback indicator (also known as feedback indicator) with the meter.
Mirrored	Inserts a mirror image of the meter based on the selected presentation options.

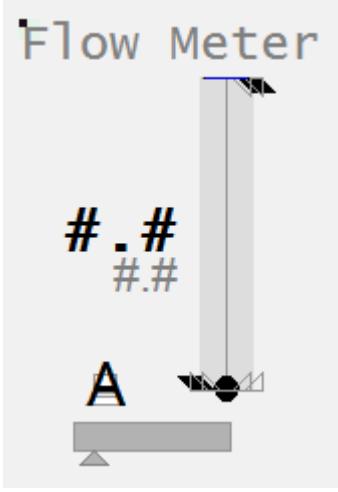
**Note:** The Control Mode, PV and OP bar take values only from **Equipment Name**. **Equipment Name #2** is only used to display the height of the second bar (the light grey bar).

## See Also

[Meter Faceplates](#)

## Flow Meter

The flow indication is presented on a linear scale, containing a "floating ball" PV indicator like commonly found physical flow meters. The meter limit markers will appear only if there are no alarm limits shown, in order to better define the meter range visually.

Property	Description
Name	Flow
Graphical Representation	
Example Equipment Template	Meter
Associated Composite Genie	Meter.xml
Equipment.Items that the Genie expects	<p><b>OP</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>ORHigh, ORLow, ORDsp</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>PRHigh, PRLow</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>OPTrack, PVTrack</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>PV</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>SP</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>FB</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>RunStatus</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>EqStatus</b> (see <a href="#">Status Indicators</a>)</p> <p><b>TrackDsp</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>Timer, TimerExp</b> (see <a href="#">Meter Special Elements</a>)</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .

Property	Description
Associated Faceplate(s)	<a href="#">Analog Controller</a> <a href="#">Analog Indicator</a>
Equipment.Items that the Faceplate Expects	<b>OP</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>ORHigh</b> , <b>ORLow</b> , <b>ORDsp</b> (see <a href="#">Meter Common Elements</a> ) <b>PRHigh</b> , <b>PRLow</b> (see <a href="#">Meter Common Elements</a> ) <b>OPTrack</b> , <b>PVTrack</b> , <b>TrackDsp</b> (see <a href="#">Meter Common Elements</a> ) <b>PV</b> (see <a href="#">Meter Common Elements</a> ) <b>SP</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>CtrlMode</b> , <b>CtrlModeDef</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>AutoCmd</b> , <b>ManCmd</b> , <b>CasCmd</b> <b>RunStatus</b> (see <a href="#">Meter Common Elements</a> ) <b>OOS</b> , <b>OOSDisabled</b> (see <a href="#">Equipment Running States</a> )

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Meter Type	From the drop-down select one of the following meters: Level, Flow, Temperature, Pressure, Analyzer, Miscellaneous, Target, Deviation and Dual Level
Equipment Name	Enter a name for the equipment associated with the

Option	Description
	meter. You can enter a maximum of 160 characters for this option.
Equipment Item Prefix	<p>Prefix applied to the equipment item name. This allows for displaying multiple values for a single piece of equipment. For example, you want to create three meters to monitor power, voltage and current for a single pump. You would create three meters on the pump and assign the equipment item prefix "Power", "Voltage" and "Current". So, the PV values for the meters would be PowerPV, VoltagePV and CurrentPV respectively.</p> <p><b>Note:</b> To get a meter with a prefix to work correctly, you need to create tags with the required item names. From the example above, you will need to create the tags PowerPV, PowerOP, VoltagePV, VoltageOP, CurrentPV, CurrentOP and so on.</p> <p>The following equipment item names (where applicable) can be prefixed:</p> <ul style="list-style-type: none"> <li>• FB</li> <li>• FullScale</li> <li>• OP</li> <li>• ORDsp</li> <li>• ORLow</li> <li>• ORHigh</li> <li>• PRLow</li> <li>• PRHigh</li> <li>• PV</li> <li>• PVTrack</li> <li>• PVTarget</li> <li>• SP</li> <li>• Timer</li> <li>• TrackDsp</li> </ul>
Size	Size of the meter object image - small or large
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Use this setting to display a label at the selected position. Select <b>None</b> if you do not want to display a

Option	Description
	label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm border around the Genie.
Display Alarm Flag	<p>Use this setting to display an alarm flag at the selected position.</p> <p><b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.</p>
Display Status Indicator	Use this setting to display a status indicator at the selected location. Select <b>None</b> if you do not want to display the status indicator. For more information, see <a href="#">Status Indicators</a> .
Display Process Variable	Select to display a process variable (numeric PV) with the meter. This option is selected by default.
Display Control Mode	<p>Select this option to display the Controller <a href="#">Control Meters - Common Elements</a>. On selecting this option the <b>Display Control Readback</b> option box is also displayed.</p> <p>The Display Control <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.</p>
Display Controller Output	Select to display an output indicator with the meter. For more information, see <a href="#">Output (OP) Bar</a> .
Display Setpoint	Select to display a setpoint indicator with the meter.
Display OOS	Select to display Out of Service indicator.
Display Differential Indication	Select to display an indicator when a differential reading is detected. <b>This is visible only on Pressure and Temperature meters.</b> For more information, see <a href="#">Meter Special Elements</a> .
Descriptor	Enter a short 3-character text descriptor to indicate what is being measured. If left blank no descriptor will be displayed. <b>This is visible only on Miscellaneous, Deviation and Analyzer meters.</b>

Option	Description
Display Clock Timer	Select to display a clock timer on the meter. <b>This is visible only on Analyzer meters.</b> For more information, see <a href="#">Meter Special Elements</a> .
Display Control Readback	Select to display a readback indicator (also known as feedback indicator) with the meter.
Display Trend	Select this option to display a <a href="#">Trend Objects</a> object with the meter. On selecting this option, the <b>Trend Type</b> dropdown is displayed. Select either <b>Tail</b> or <b>Full</b> . <b>Note:</b> When the <b>Display Trend</b> option is selected, the Mirrored option is not available.
Trend Type	Select the type of trend to be displayed with the meter: <a href="#">Trend Tail</a> or <a href="#">Full Trend</a> .
Mirrored	Inserts a mirror image of the meter based on the selected presentation options.

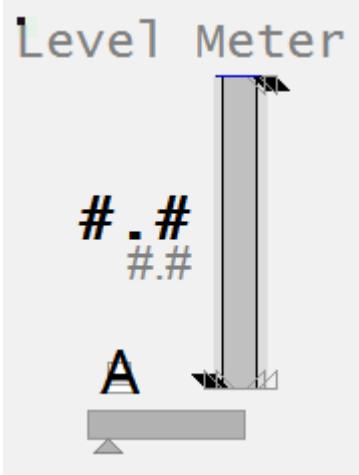
## See Also

[Meter Faceplates](#)

## Level Meter

The meter symbol for a Level meter is a narrow box shape that resembles a "skinny" vessel. The graphical PV is shown as a darker fill within the vessel. The fill rises and falls to indicate the actual level.

Level meters are provided in both vertical and horizontal orientations, although it is expected they will be used primarily in the vertical orientation.

Property	Description
Name	Level
Graphical Representation	
Example Equipment Template	Meter
Associated Composite Genie	Meter.xml
Equipment.Items that the Genie expects	<p><b>OP</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>ORHigh, ORLow, ORDsp</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>PRHigh, PRLow</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>OPTrack, PVTrack</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>PV</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>SP</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>FB</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>RunStatus</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>EqStatus</b> (see <a href="#">Status Indicators</a>)</p> <p><b>TrackDsp</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>Timer, TimerExp</b> (see <a href="#">Meter Special Elements</a>)</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .

Property	Description
Associated Faceplate(s)	<a href="#">Analog Controller</a> <a href="#">Analog Indicator</a>
Equipment.Items that the Faceplate Expects	<b>OP</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>ORHigh</b> , <b>ORLow</b> , <b>ORDsp</b> (see <a href="#">Meter Common Elements</a> ) <b>PRHigh</b> , <b>PRLow</b> (see <a href="#">Meter Common Elements</a> ) <b>OPTrack</b> , <b>PVTrack</b> , <b>TrackDsp</b> (see <a href="#">Meter Common Elements</a> ) <b>PV</b> (see <a href="#">Meter Common Elements</a> ) <b>SP</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>CtrlMode</b> , <b>CtrlModeDef</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>AutoCmd</b> , <b>ManCmd</b> , <b>CasCmd</b> <b>RunStatus</b> (see <a href="#">Meter Common Elements</a> ) <b>OOS</b> , <b>OOSDisabled</b> (see <a href="#">Equipment Running States</a> )

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Meter Type	From the drop-down select one of the following meters: Level, Flow, Temperature, Pressure, Analyzer, Miscellaneous, Target, Deviation and Dual Level
Equipment Name	Enter a name for the equipment associated with the

Option	Description
	meter. You can enter a maximum of 160 characters for this option.
Equipment Item Prefix	<p>Prefix applied to the equipment item name. This allows for displaying multiple values for a single piece of equipment. For example, you want to create three meters to monitor power, voltage and current for a single pump. You would create three meters on the pump and assign the equipment item prefix "Power", "Voltage" and "Current". So, the PV values for the meters would be PowerPV, VoltagePV and CurrentPV respectively.</p> <p><b>Note:</b> To get a meter with a prefix to work correctly, you need to create tags with the required item names. From the example above, you will need to create the tags PowerPV, PowerOP, VoltagePV, VoltageOP, CurrentPV, CurrentOP and so on.</p> <p>The following equipment item names (where applicable) can be prefixed:</p> <ul style="list-style-type: none"> <li>• FB</li> <li>• FullScale</li> <li>• OP</li> <li>• ORDsp</li> <li>• ORLow</li> <li>• ORHigh</li> <li>• PRLow</li> <li>• PRHigh</li> <li>• PV</li> <li>• PVTrack</li> <li>• PVTarget</li> <li>• SP</li> <li>• Timer</li> <li>• TrackDsp</li> </ul>
Size	Size of the meter object image - small or large
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Use this setting to display a label at the selected position. Select <b>None</b> if you do not want to display a

Option	Description
	label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm border around the Genie.
Display Alarm Flag	<p>Use this setting to display an alarm flag at the selected position.</p> <p><b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.</p>
Display Status Indicator	Use this setting to display a status indicator at the selected location. Select <b>None</b> if you do not want to display the status indicator. For more information, see <a href="#">Status Indicators</a> .
Display Process Variable	Select to display a process variable (numeric PV) with the meter. This option is selected by default.
Display Control Mode	<p>Select this option to display the Controller <a href="#">Control Meters - Common Elements</a>. On selecting this option the <b>Display Control Readback</b> option box is also displayed.</p> <p>The Display Control <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.</p>
Display Controller Output	Select to display an output indicator with the meter. For more information, see <a href="#">Output (OP) Bar</a> .
Display Setpoint	Select to display a setpoint indicator with the meter.
Display OOS	Select to display Out of Service indicator.
Display Differential Indication	Select to display an indicator when a differential reading is detected. <b>This is visible only on Pressure and Temperature meters.</b> For more information, see <a href="#">Meter Special Elements</a> .
Descriptor	Enter a short 3-character text descriptor to indicate what is being measured. If left blank no descriptor will be displayed. <b>This is visible only on Miscellaneous, Deviation and Analyzer meters.</b>

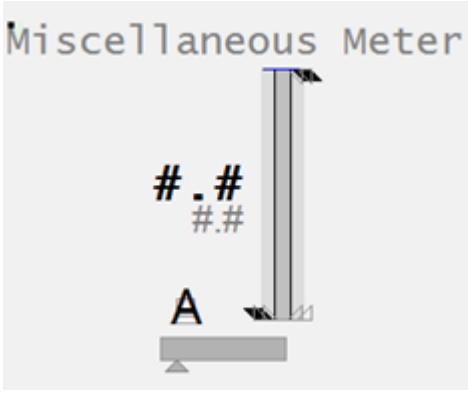
Option	Description
Display Clock Timer	Select to display a clock timer on the meter. <b>This is visible only on Analyzer meters.</b> For more information, see <a href="#">Meter Special Elements</a> .
Display Control Readback	Select to display a readback indicator (also known as feedback indicator) with the meter.
Display Trend	Select this option to display a <a href="#">Trend Objects</a> object with the meter. On selecting this option, the <b>Trend Type</b> dropdown is displayed. Select either <b>Tail</b> or <b>Full</b> . <b>Note:</b> When the <b>Display Trend</b> option is selected, the Mirrored option is not available.
Trend Type	Select the type of trend to be displayed with the meter: <a href="#">Trend Tail</a> or <a href="#">Full Trend</a> .
Mirrored	Inserts a mirror image of the meter based on the selected presentation options.

## See Also

[Meter Faceplates](#)

## Miscellaneous Meter

The miscellaneous meter indication is presented on a linear scale, containing a filled rectangle as a PV indicator. The miscellaneous meter is usually used to represent readings that do not have a dedicated meter type (e.g., weight, speed, torque, etc.).

Property	Description
Name	Miscellaneous
Graphical Representation	
Example Equipment Template	Meter

Property	Description
Associated Composite Genie	Meter.xml
Equipment.Items that the Genie expects	<p><b>OP</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>ORHigh, ORLow, ORDsp</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>PRHigh, PRLow</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>OPTrack, PVTrack</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>PV</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>SP</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>FB</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>RunStatus</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>EqStatus</b> (see <a href="#">Status Indicators</a>)</p> <p><b>TrackDsp</b> (see <a href="#">Meter Common Elements</a>)</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	Analog Controller Analog Indicator
Equipment.Items that the Faceplate Expects	<p><b>OP</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>ORHigh, ORLow, ORDsp</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>PRHigh, PRLow</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>OPTrack, PVTrack, TrackDsp</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>PV</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>SP</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>AutoCmd, ManCmd, CasCmd</b></p> <p><b>RunStatus</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a>)</p>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Meter Type	From the drop-down select one of the following meters: Level, Flow, Temperature, Pressure, Analyzer, Miscellaneous, Target, Deviation and Dual Level
Equipment Name	Enter a name for the equipment associated with the meter. You can enter a maximum of 160 characters for this option.
Equipment Item Prefix	<p>Prefix applied to the equipment item name. This allows for displaying multiple values for a single piece of equipment. For example, you want to create three meters to monitor power, voltage and current for a single pump. You would create three meters on the pump and assign the equipment item prefix "Power", "Voltage" and "Current". So, the PV values for the meters would be PowerPV, VoltagePV and CurrentPV respectively.</p> <p><b>Note:</b> To get a meter with a prefix to work correctly, you need to create tags with the required item names. From the example above, you will need to create the tags PowerPV, PowerOP, VoltagePV, VoltageOP, CurrentPV, CurrentOP and so on.</p> <p>The following equipment item names (where applicable) can be prefixed:</p> <ul style="list-style-type: none"><li>• FB</li></ul>

Option	Description
	<ul style="list-style-type: none"> <li>• FullScale</li> <li>• OP</li> <li>• ORDsp</li> <li>• ORLow</li> <li>• ORHigh</li> <li>• PRLow</li> <li>• PRHigh</li> <li>• PV</li> <li>• PVTrack</li> <li>• PVTarget</li> <li>• SP</li> <li>• Timer</li> <li>• TrackDsp</li> </ul>
Size	Size of the meter object image - small or large
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Use this setting to display a label at the selected position. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm border around the Genie.
Display Alarm Flag	<p>Use this setting to display an alarm flag at the selected position.</p> <p><b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.</p>
Display Status Indicator	Use this setting to display a status indicator at the selected location. Select <b>None</b> if you do not want to display the status indicator. For more information, see <a href="#">Status Indicators</a> .
Display Process Variable	Select to display a process variable (numeric PV) with the meter. This option is selected by default.

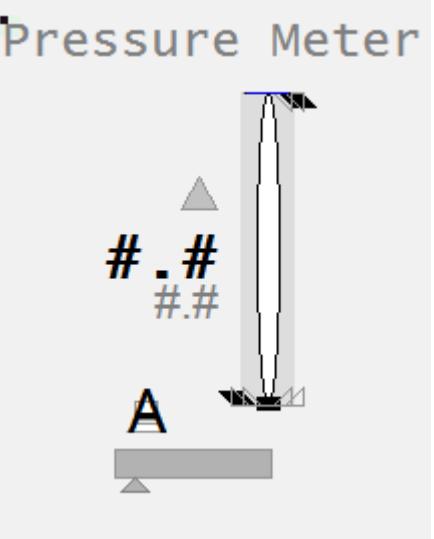
Option	Description
Display Control Mode	<p>Select this option to display the Controller <a href="#">Control Meters - Common Elements</a>. On selecting this option the <b>Display Control Readback</b> option box is also displayed.</p> <p>The Display Control <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.</p>
Display Controller Output	<p>Select to display an output indicator with the meter. For more information, see <a href="#">Output (OP) Bar</a>.</p>
Display Setpoint	<p>Select to display a setpoint indicator with the meter.</p>
Display OOS	<p>Select to display Out of Service indicator.</p>
Display Differential Indication	<p>Select to display an indicator when a differential reading is detected. <b>This is visible only on Pressure and Temperature meters</b>. For more information, see <a href="#">Meter Special Elements</a>.</p>
Descriptor	<p>Enter a short 3-character text descriptor to indicate what is being measured. If left blank no descriptor will be displayed. <b>This is visible only on Miscellaneous, Deviation and Analyzer meters</b>.</p>
Display Clock Timer	<p>Select to display a clock timer on the meter. <b>This is visible only on Analyzer meters</b>. For more information, see <a href="#">Meter Special Elements</a>.</p>
Display Control Readback	<p>Select to display a readback indicator (also known as feedback indicator) with the meter.</p>
Display Trend	<p>Select this option to display a <a href="#">Trend Objects</a> object with the meter. On selecting this option, the <b>Trend Type</b> dropdown is displayed. Select either <b>Tail</b> or <b>Full</b>.</p> <p><b>Note:</b> When the <b>Display Trend</b> option is selected, the Mirrored option is not available.</p>
Trend Type	<p>Select the type of trend to be displayed with the meter: <a href="#">Trend Tail</a> or <a href="#">Full Trend</a>.</p>
Mirrored	<p>Inserts a mirror image of the meter based on the selected presentation options.</p>

## See Also

[Meter Faceplates](#)

## Pressure Meter

Pressure meters can be used to show regular pressure readings, or differential pressure readings.

Property	Description
Name	Pressure
Graphical Representation	
Example Equipment Template	Meter
Associated Composite Genie	Meter.xml
Equipment.Items that the Genie expects	<b>OP</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>ORHigh, ORLow, ORDsp</b> (see <a href="#">Meter Common Elements</a> ) <b>PRHigh, PRLow</b> (see <a href="#">Meter Common Elements</a> ) <b>OPTrack, PVTrack</b> (see <a href="#">Meter Common Elements</a> ) <b>PV</b> (see <a href="#">Meter Common Elements</a> ) <b>SP</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>FB</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>RunStatus</b> (see <a href="#">Meter Common Elements</a> ) <b>EqStatus</b> (see <a href="#">Status Indicators</a> ) <b>TrackDsp</b> (see <a href="#">Meter Common Elements</a> )
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode

Property	Description
	Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	<a href="#">Analog Controller</a> <a href="#">Analog Indicator</a>
Equipment.Items that the Faceplate Expects	<b>OP</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>ORHigh</b> , <b>ORLow</b> , <b>ORDsp</b> (see <a href="#">Meter Common Elements</a> ) <b>PRHigh</b> , <b>PRLow</b> (see <a href="#">Meter Common Elements</a> ) <b>OPTrack</b> , <b>PVTrack</b> , <b>TrackDsp</b> (see <a href="#">Meter Common Elements</a> ) <b>PV</b> (see <a href="#">Meter Common Elements</a> ) <b>SP</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>CtrlMode</b> , <b>CtrlModeDef</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>AutoCmd</b> , <b>ManCmd</b> , <b>CasCmd</b> <b>RunStatus</b> (see <a href="#">Meter Common Elements</a> ) <b>OOS</b> , <b>OOSDisabled</b> (see <a href="#">Equipment Running States</a> )

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Meter Type	From the drop-down select one of the following

Option	Description
	meters: Level, Flow, Temperature, Pressure, Analyzer, Miscellaneous, Target, Deviation and Dual Level
Equipment Name	Enter a name for the equipment associated with the meter. You can enter a maximum of 160 characters for this option.
Equipment Item Prefix	<p>Prefix applied to the equipment item name. This allows for displaying multiple values for a single piece of equipment. For example, you want to create three meters to monitor power, voltage and current for a single pump. You would create three meters on the pump and assign the equipment item prefix "Power", "Voltage" and "Current". So, the PV values for the meters would be PowerPV, VoltagePV and CurrentPV respectively.</p> <p><b>Note:</b> To get a meter with a prefix to work correctly, you need to create tags with the required item names. From the example above, you will need to create the tags PowerPV, PowerOP, VoltagePV, VoltageOP, CurrentPV, CurrentOP and so on.</p> <p>The following equipment item names (where applicable) can be prefixed:</p> <ul style="list-style-type: none"> <li>• FB</li> <li>• FullScale</li> <li>• OP</li> <li>• ORDsp</li> <li>• ORLow</li> <li>• ORHigh</li> <li>• PRLow</li> <li>• PRHigh</li> <li>• PV</li> <li>• PVTrack</li> <li>• PVTarget</li> <li>• SP</li> <li>• Timer</li> <li>• TrackDsp</li> </ul>
Size	Size of the meter object image - small or large
Orientation	Select the orientation that is appropriate for the

Option	Description
	presentation of the object on the graphics page.
Display Label	Use this setting to display a label at the selected position. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm border around the Genie.
Display Alarm Flag	Use this setting to display an alarm flag at the selected position.  <b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.
Display Status Indicator	Use this setting to display a status indicator at the selected location. Select <b>None</b> if you do not want to display the status indicator. For more information, see <a href="#">Status Indicators</a> .
Display Process Variable	Select to display a process variable (numeric PV) with the meter. This option is selected by default.
Display Control Mode	Select this option to display the Controller <a href="#">Control Meters - Common Elements</a> . On selecting this option the <b>Display Control Readback</b> option box is also displayed.  The <a href="#">Display Control Control Meters - Common Elements</a> represents the actual output for the controller.
Display Controller Output	Select to display an output indicator with the meter. For more information, see <a href="#">Output (OP) Bar</a> .
Display Setpoint	Select to display a setpoint indicator with the meter.
Display OOS	Select to display Out of Service indicator.
Display Differential Indication	Select to display an indicator when a differential reading is detected. <b>This is visible only on Pressure and Temperature meters</b> . For more information, see <a href="#">Meter Special Elements</a> .
Descriptor	Enter a short 3-character text descriptor to indicate

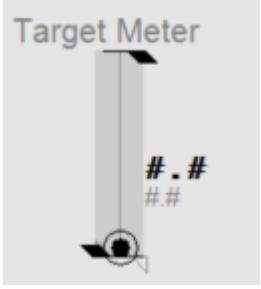
Option	Description
	what is being measured. If left blank no descriptor will be displayed. <b>This is visible only on Miscellaneous, Deviation and Analyzer meters.</b>
Display Clock Timer	Select to display a clock timer on the meter. <b>This is visible only on Analyzer meters.</b> For more information, see <a href="#">Meter Special Elements</a> .
Display Control Readback	Select to display a readback indicator (also known as feedback indicator) with the meter.
Display Trend	Select this option to display a <a href="#">Trend Objects</a> object with the meter. On selecting this option, the <b>Trend Type</b> dropdown is displayed. Select either <b>Tail</b> or <b>Full</b> . <b>Note:</b> When the <b>Display Trend</b> option is selected, the Mirrored option is not available.
Trend Type	Select the type of trend to be displayed with the meter: <a href="#">Trend Tail</a> or <a href="#">Full Trend</a> .
Mirrored	Inserts a mirror image of the meter based on the selected presentation options.

## See Also

[Meter Faceplates](#)

## Target Meter

The Target meter is used to show deviations from setpoint in level.

Property	Description
Name	Target
Graphical Representation	
Example Equipment Template	TargetMeter

Property	Description
Associated Composite Genie	Meter.xml
Equipment.Items that the Genie expects	<p><b>OP</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>ORHigh, ORLow, ORDsp</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>PRHigh, PRLow</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>OPTrack, PVTrack</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>PV</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>SP</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>FB</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>RunStatus</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>EqStatus</b> (see <a href="#">Status Indicators</a>)</p> <p><b>TrackDsp</b> (see <a href="#">Meter Common Elements</a>)</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	Analog Controller Analog Indicator
Equipment.Items that the Faceplate Expects	<p><b>OP</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>ORHigh, ORLow, ORDsp</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>PRHigh, PRLow</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>OPTrack, PVTrack, TrackDsp</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>PV</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>SP</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>AutoCmd, ManCmd, CasCmd</b></p> <p><b>RunStatus</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a>)</p>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Meter Type	From the drop-down select one of the following meters: Level, Flow, Temperature, Pressure, Analyzer, Miscellaneous, Target, Deviation and Dual Level
Equipment Name	Enter a name for the equipment associated with the meter. You can enter a maximum of 160 characters for this option.
Equipment Item Prefix	<p>Prefix applied to the equipment item name. This allows for displaying multiple values for a single piece of equipment. For example, you want to create three meters to monitor power, voltage and current for a single pump. You would create three meters on the pump and assign the equipment item prefix "Power", "Voltage" and "Current". So, the PV values for the meters would be PowerPV, VoltagePV and CurrentPV respectively.</p> <p><b>Note:</b> To get a meter with a prefix to work correctly, you need to create tags with the required item names. From the example above, you will need to create the tags PowerPV, PowerOP, VoltagePV, VoltageOP, CurrentPV, CurrentOP and so on.</p> <p>The following equipment item names (where applicable) can be prefixed:</p> <ul style="list-style-type: none"><li>• FB</li></ul>

Option	Description
	<ul style="list-style-type: none"> <li>• FullScale</li> <li>• OP</li> <li>• ORDsp</li> <li>• ORLow</li> <li>• ORHigh</li> <li>• PRLow</li> <li>• PRHigh</li> <li>• PV</li> <li>• PVTrack</li> <li>• PVTarget</li> <li>• SP</li> <li>• Timer</li> <li>• TrackDsp</li> </ul>
Size	Size of the meter object image - small or large
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Use this setting to display a label at the selected position. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm border around the Genie.
Display Alarm Flag	<p>Use this setting to display an alarm flag at the selected position.</p> <p><b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.</p>
Display Status Indicator	Use this setting to display a status indicator at the selected location. Select <b>None</b> if you do not want to display the status indicator. For more information, see <a href="#">Status Indicators</a> .
Display Process Variable	Select to display a process variable (numeric PV) with the meter. This option is selected by default.

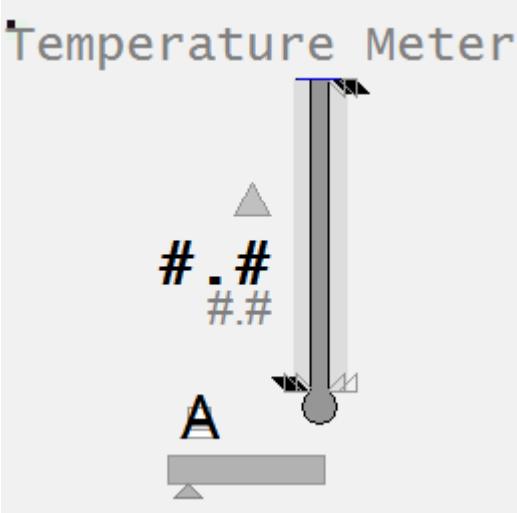
Option	Description
Display Control Mode	<p>Select this option to display the Controller <a href="#">Control Meters - Common Elements</a>. On selecting this option the <b>Display Control Readback</b> option box is also displayed.</p> <p>The Display Control <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.</p>
Display Controller Output	<p>Select to display an output indicator with the meter. For more information, see <a href="#">Output (OP) Bar</a>.</p>
Display Setpoint	<p>Select to display a setpoint indicator with the meter.</p>
Display OOS	<p>Select to display Out of Service indicator.</p>
Display Differential Indication	<p>Select to display an indicator when a differential reading is detected. <b>This is visible only on Pressure and Temperature meters</b>. For more information, see <a href="#">Meter Special Elements</a>.</p>
Descriptor	<p>Enter a short 3-character text descriptor to indicate what is being measured. If left blank no descriptor will be displayed. <b>This is visible only on Miscellaneous, Deviation and Analyzer meters</b>.</p>
Display Clock Timer	<p>Select to display a clock timer on the meter. <b>This is visible only on Analyzer meters</b>. For more information, see <a href="#">Meter Special Elements</a>.</p>
Display Control Readback	<p>Select to display a readback indicator (also known as feedback indicator) with the meter.</p>
Display Trend	<p>Select this option to display a <a href="#">Trend Objects</a> object with the meter. On selecting this option, the <b>Trend Type</b> dropdown is displayed. Select either <b>Tail</b> or <b>Full</b>.</p> <p><b>Note:</b> When the <b>Display Trend</b> option is selected, the Mirrored option is not available.</p>
Trend Type	<p>Select the type of trend to be displayed with the meter: <a href="#">Trend Tail</a> or <a href="#">Full Trend</a>.</p>
Mirrored	<p>Inserts a mirror image of the meter based on the selected presentation options.</p>

## See Also

[Meter Faceplates](#)

## Temperature Meter

Temperatures are indicated on a linear scale. Temperature meters can be used to show regular temperature readings, or differential temperature readings.

Property	Description
Name	Deviation
Graphical Representation	
Example Equipment Template	Meter
Associated Composite Genie	Meter.xml
Equipment.Items that the Genie expects	<b>OP</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>ORHigh, ORLow, ORDsp</b> (see <a href="#">Meter Common Elements</a> ) <b>PRHigh, PRLow</b> (see <a href="#">Meter Common Elements</a> ) <b>OPTrack, PVTrack</b> (see <a href="#">Meter Common Elements</a> ) <b>PV</b> (see <a href="#">Meter Common Elements</a> ) <b>SP</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>FB</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>RunStatus</b> (see <a href="#">Meter Common Elements</a> ) <b>EqStatus</b> (see <a href="#">Status Indicators</a> ) <b>TrackDsp</b> (see <a href="#">Meter Common Elements</a> )
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodelIODevice EqStatusFunc

	CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	<a href="#">Analog Controller</a> <a href="#">Analog Indicator</a>
Equipment.Items that the Faceplate Expects	<b>OP</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>ORHigh, ORLow, ORDsp</b> (see <a href="#">Meter Common Elements</a> ) <b>PRHigh, PRLow</b> (see <a href="#">Meter Common Elements</a> ) <b>OPTrack, PVTrack, TrackDsp</b> (see <a href="#">Meter Common Elements</a> ) <b>PV</b> (see <a href="#">Meter Common Elements</a> ) <b>SP</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Control Meters - Common Elements</a> ) <b>AutoCmd, ManCmd, CasCmd</b> <b>RunStatus</b> (see <a href="#">Meter Common Elements</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> )

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Meter Type	From the drop-down select one of the following

Option	Description
	meters: Level, Flow, Temperature, Pressure, Analyzer, Miscellaneous, Target, Deviation and Dual Level
Equipment Name	Enter a name for the equipment associated with the meter. You can enter a maximum of 160 characters for this option.
Equipment Item Prefix	<p>Prefix applied to the equipment item name. This allows for displaying multiple values for a single piece of equipment. For example, you want to create three meters to monitor power, voltage and current for a single pump. You would create three meters on the pump and assign the equipment item prefix "Power", "Voltage" and "Current". So, the PV values for the meters would be PowerPV, VoltagePV and CurrentPV respectively.</p> <p><b>Note:</b> To get a meter with a prefix to work correctly, you need to create tags with the required item names. From the example above, you will need to create the tags PowerPV, PowerOP, VoltagePV, VoltageOP, CurrentPV, CurrentOP and so on.</p> <p>The following equipment item names (where applicable) can be prefixed:</p> <ul style="list-style-type: none"> <li>• FB</li> <li>• FullScale</li> <li>• OP</li> <li>• ORDsp</li> <li>• ORLow</li> <li>• ORHigh</li> <li>• PRLow</li> <li>• PRHigh</li> <li>• PV</li> <li>• PVTrack</li> <li>• PVTarget</li> <li>• SP</li> <li>• Timer</li> <li>• TrackDsp</li> </ul>
Size	Size of the meter object image - small or large
Orientation	Select the orientation that is appropriate for the

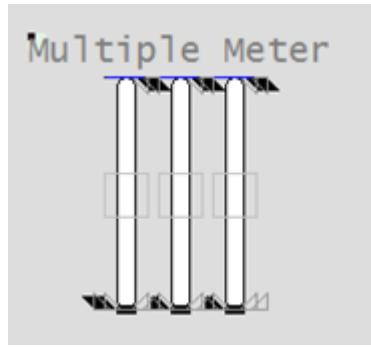
Option	Description
	presentation of the object on the graphics page.
Display Label	Use this setting to display a label at the selected position. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm border around the Genie.
Display Alarm Flag	Use this setting to display an alarm flag at the selected position.  <b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.
Display Status Indicator	Use this setting to display a status indicator at the selected location. Select <b>None</b> if you do not want to display the status indicator. For more information, see <a href="#">Status Indicators</a> .
Display Process Variable	Select to display a process variable (numeric PV) with the meter. This option is selected by default.
Display Control Mode	Select this option to display the Controller <a href="#">Control Meters - Common Elements</a> . On selecting this option the <b>Display Control Readback</b> option box is also displayed.  The <a href="#">Display Control Control Meters - Common Elements</a> represents the actual output for the controller.
Display Controller Output	Select to display an output indicator with the meter. For more information, see <a href="#">Output (OP) Bar</a> .
Display Setpoint	Select to display a setpoint indicator with the meter.
Display OOS	Select to display Out of Service indicator.
Display Differential Indication	Select to display an indicator when a differential reading is detected. <b>This is visible only on Pressure and Temperature meters</b> . For more information, see <a href="#">Meter Special Elements</a> .
Descriptor	Enter a short 3-character text descriptor to indicate

Option	Description
	what is being measured. If left blank no descriptor will be displayed. <b>This is visible only on Miscellaneous, Deviation and Analyzer meters.</b>
Display Clock Timer	Select to display a clock timer on the meter. <b>This is visible only on Analyzer meters.</b> For more information, see <a href="#">Meter Special Elements</a> .
Display Control Readback	Select to display a readback indicator (also known as feedback indicator) with the meter.
Display Trend	Select this option to display a <a href="#">Trend Objects</a> object with the meter. On selecting this option, the <b>Trend Type</b> dropdown is displayed. Select either <b>Tail</b> or <b>Full</b> . <b>Note:</b> When the <b>Display Trend</b> option is selected, the Mirrored option is not available.
Trend Type	Select the type of trend to be displayed with the meter: <a href="#">Trend Tail</a> or <a href="#">Full Trend</a> .
Mirrored	Inserts a mirror image of the meter based on the selected presentation options.

## See Also

[Meter Faceplates](#)

## Multiple Meter

Property	Description
Name	Multiple
Graphical Representation	
Example Equipment Template	Two or three meter objects

Associated Composite Genie	Meter_Multiple.xml
Equipment.Items that the Genie expects	<p><b>OP</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>ORHigh, ORLow, ORDsp</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>PRHigh, PRLow</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>OPTTrack, PVTrack</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>PV</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>SP</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>FB</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>RunStatus</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>EqStatus</b> (see <a href="#">Status Indicators</a>)</p> <p><b>TrackDsp</b> (see <a href="#">Meter Common Elements</a>)</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	Analog Controller Analog Indicator
Equipment.Items that the Faceplate Expects	<p><b>OP</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>ORHigh, ORLow, ORDsp</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>PRHigh, PRLow</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>OPTTrack, PVTrack, TrackDsp</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>PV</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>SP</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p><b>AutoCmd, ManCmd, CasCmd</b></p> <p><b>RunStatus</b> (see <a href="#">Meter Common Elements</a>)</p> <p><b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a>)</p>

## Configuration Tasks

1. [Add Equipment Using Equipment Editor](#): Create the equipment for the required objects. Organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
2. [Equipment References](#): Create the required relationships (equipment references) between a piece of equipment to items of another piece of equipment. This allows you to refer to equipment.items outside of the equipment hierarchy.
3. [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
4. [Insert a Composite Genie](#): Create graphics pages with instances of Composite Genies that are associated with the equipment you have defined in your project.
5. [Create a New Faceplate](#): Add a faceplate for the equipment to allow operators to engage with the equipment at runtime.
6. Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the equipment associated with the meter. You can enter a maximum of 160 characters for this option. <b>Note:</b> The Multiple Meter equipment needs to be manually configured to reference the name of each meter to be included.
Meter Count	Number of meters - select from Duplicated for two meters or Triplicated for three meters.
Equipment #1/#2/#3 Name	Enter the name of the equipment object for the first/second/third meter (displayed only if meter count is triplicated). You can enter a maximum of 160 characters for this option.
Equipment #1/#2/#3 Prefix	Prefix applied to the equipment item name for the meters. This allows for displaying multiple values for a single piece of equipment. For example, you want to create three meters to monitor power, voltage and current for a single pump. You would create three meters on the pump and assign the equipment item prefix "Power", "Voltage" and "Current". So, the PV values for the meters would be PowerPV, VoltagePV and CurrentPV respectively.

Option	Description
	<p><b>Note:</b> To get a meter with a prefix to work correctly, you need to create tags with the required item names. From the example above, you will need to create the tags PowerPV, PowerOP, VoltagePV, VoltageOP, CurrentPV, CurrentOP and so on.</p> <p>The following equipment item names (where applicable) can be prefixed:</p> <ul style="list-style-type: none"> <li>• PV</li> <li>• OP</li> <li>• ORDsp</li> <li>• TrackDsp</li> <li>• FB</li> <li>• PRLow</li> <li>• PRHigh</li> <li>• ORLow</li> <li>• ORHigh</li> <li>• SP</li> <li>• PVTrack</li> <li>• FullScale</li> <li>• OPTrack</li> <li>• Timer</li> <li>• PVTtarget</li> </ul>
Meter Type	From a drop-down select one of the following meters: Level, Flow, Temperature, Pressure, Analyzer, Miscellaneous, and Deviation.
Size	Size of the meter object image - small or large
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Use this setting to display a label at the selected position. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.

Option	Description
Display Alarm Indicator	Shows or hides the Alarm Indicator which indicates the highest priority alarm and its state for this group of meters.
Display Setpoint	Select to display a setpoint indicator with the meter.
Display OOS	Select to display Out of Service indicator.
Mirrored	Inserts a mirror image of the meter based on the selected presentation options.

## See Also

[Meter Faceplates](#)

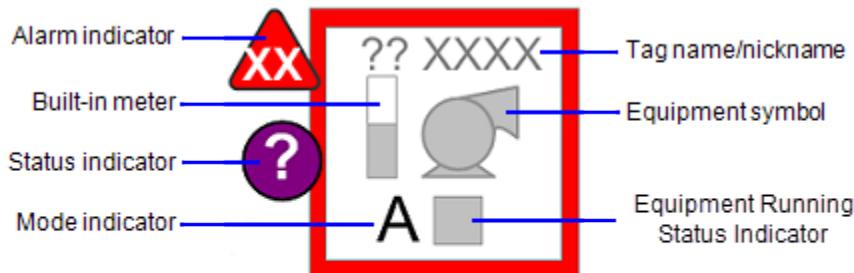
## Drive, Belt and Mining Objects

The drive objects that are available in the Situational Awareness library are classified into the following groups:

- [Drive Objects](#)
- [Belt Objects](#)
- [Rotary Valve](#).

They are all similar in their elements and functionality. The equipment symbol (see below) is the only element that is unique to each drive type.

The image below shows the elements that make up a basic drive object.




---

**Note:** The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the following item: PV.

If you want your drive to have multiple analog values, you can display those as separate meter objects using meter Composite Genies. When using meters the addresses and tag names need to be unique. Use a prefix on the item name. For more information refer to [Meters](#).

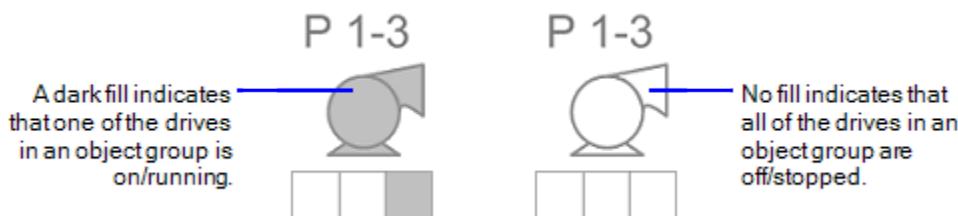
## Equipment symbol

Connects to: Running, Stopped

The equipment symbol indicates the following:

- The type of equipment being represented.
- The run status of the drive.

A dark fill indicates the run status of a drive is 'on' or 'running'. A white fill indicates the drive is 'off' or 'stopped'.



Some equipment symbols are available in variations that point in a particular direction.

## Alarm indicator

An alarm indicator is used to show the occurrence and status of alarms associated with an object. See [Use Alarm Indicators](#) for more information.

## Status indicator

Connects to: EqStatus.

A running state indicator is used to represent various non-alarm conditions associated with an object, such as abnormal data quality or control system states. See Status Indicators in the section on Common Object Properties.

## Equipment Running State Indicator

Connects to: RunStatus.

An Equipment Running State Indicator is a compact indicator that can be used to represent a variety of states for drive objects. They allow an object to represent a group of drives. See [Equipment Running State Indicators](#) in the section on Common Object Properties.

## Built-in meter

Connects to: PV.

This item is available to accommodate drive objects that have an integrated meter for motor amps or kilowatts.

It is built using a small version of the meter symbol and graphical PV from the Miscellaneous meter. To give the graphics designer flexibility to put the relevant information next to the object, the item name that drives this graph has been given a name that describes its use on the genie.

The built-in meter element is only shown if the reading is available. It is only available for single-drive objects, it is not used with groups of multiple drive objects.

For drive objects with a left-facing option, the built-in meter is moved to the right-hand side of the equipment symbol.



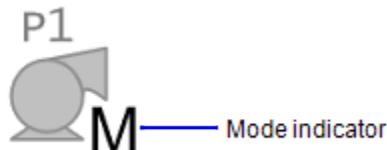
## Tag name or nickname

The tag name/nickname field shows the equipment tag name (by default), or, if entered by the user, a "nickname". The nickname is usually a common name used to help identify the reading (for example, "Oil pump", instead of "P4509").

## Mode indicator

Connects to: CtrlMode, CtrlModeDef

This is a single-character code that indicates the current mode of the drive.



### Control Mode Indicator states:

- 0 – Auto (A)
- 1 – Manual (M)
- 2 – Cascade (C)
- 3 – Local (L)
- 4 – Special control (computer symbol)

---

**Note:** The Mode Indicator is not displayed if the tag is set to CtrlModeDef. To view the control mode at all times, remove the CtrlModeDef tag from the equipment template.

---

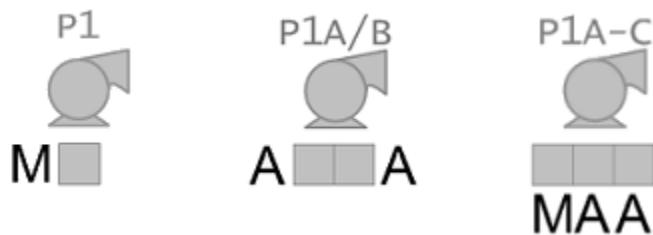
The AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd tags work in conjunction with the CtrlMode tag. For more information, see [Create a New Faceplate](#).

If a drive object includes Equipment Running State Indicators, the following positioning is used:

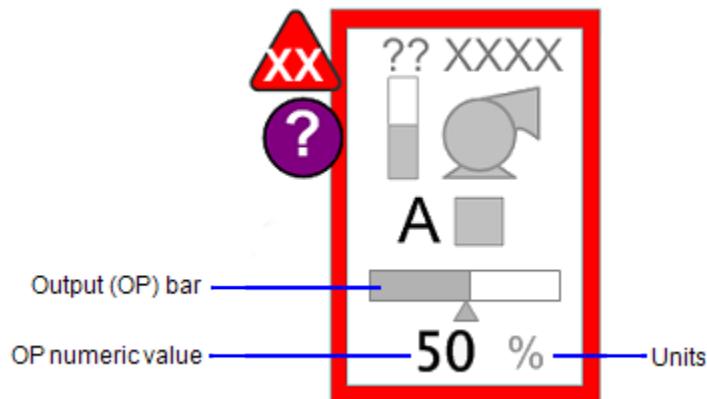
- For a single Equipment Running State Indicator, the mode indicator is positioned to the left of the Equipment Running State Indicator.
- For two Equipment Running State Indicators, the mode indicators are placed on either side of the Equipment

Running State Indicator.

- For groups of three Equipment Running State Indicators, (or more), the mode indicator is shown underneath the Equipment Running State Indicator.



If the object represents a variable speed drive (VSD), it may also include the following additional elements.



## Output (OP) bar

Connects to: OP.

The output (OP) bar provides a graphical representation of the commanded output for a variable speed drive (VSD). See [Output \(OP\) Bar](#) in the section on Common Object Properties.

When dealing with a group of 2 or more drives, the VSD output bar should only be shown if there is only one VSD in the group, or if there is a common controller for all of the drives in the group. If the group includes two or more separate VSDs, either the VSD output indicator should not be shown, or two separate drive objects should be used.

The readback indicator is only shown for drives where readback is available. Generally this will show the actual speed of the drive, measured as a percentage. When a group of drives with a VSD is shown, and readback is available, readback indicators are shown for each drive.

## Numeric OP

Connects to: OP.

For variable speed drives, a numeric representation of the OP can provide operators with precise values when needed. This is measured as a percentage value.

The visibility of the numeric OP is tied to the visibility state for PVs (as specified in the Show/Hide Settings). If the user selects to turn off the visibility for PVs, the numeric OP values is also hidden.

## Units

Represents the unit of measurement for the output. Typically, this will be percentage (%) for VSD output.

## See Also

[Drive Faceplates](#)

## Drive Objects

The drive objects that are available in the Situational Awareness library include:

- [Agitator/Rake](#)
- [Blower](#)
- [Bore Pump](#)
- [Brake](#)
- [Compressor](#)
- [Electric Heater](#)
- [Fan](#)
- [Motor](#)
- [Pump](#)
- [Rotary Valve](#)
- [Shell and Tube Heat Exchanger](#)
- [Sump Pump](#)
- [Turbine](#)

## See Also

[Belt Objects](#)

[Mining Objects](#)

[Drive Faceplates](#)

## Agitator/Rake

Property	Description
Name	Agitator/Rake
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Drive.xml
Equipment.Items the Genie expects	<p><b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	Single and Multiple DOL Drive Single and Multiple VSD Drive - Single Direction Single DOL Drive with Forward/Reverse Capability Single VSD Drive with Forward/Reverse Capability
Equipment.Items that the Faceplate expects	<b>OPTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> )

Property	Description
	<p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>Fwd, Rev, Maint</b></p> <p><b>Stopped</b></p>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.

Option	Description
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	<p>Use this setting to display an alarm flag at the selected position.</p> <p><b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.</p>
Display Status Indicator	<p>Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a>.</p>
Number of Equipment	<p>If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of <a href="#">Equipment Running State Indicators</a>.</p> <p>If you select 1, the <b>Display Compact Running State</b> option will be displayed.</p> <p>If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>
Display Compact Running State	Select to display a compact <a href="#">Equipment Running State Indicators</a> .
Display Related Equipment Alarm Indicator	Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.
Related Equipment #n Name	Enter a name for each piece of equipment in the group.
Display Control Mode	Select this option to display a mode indicator for the object.
Display OOS	If the number of sub equipment associated with the drive object is more than 1, you can select this option

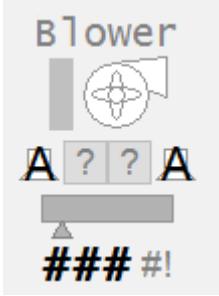
Option	Description
	to display an "Out of Service" indicator for each related equipment.
Display Meter	Select this option to display a built-in meter for the object.
Display Output Bar	Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed. The Display Control <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)

[Drive Faceplates](#)

## Blower

Property	Description
Name	Blower
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Drive.xml
Equipment.Items the Genie expects	<b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a> )

Property	Description
	<p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	<a href="#">Single and Multiple DOL Drive</a> <a href="#">Single and Multiple VSD Drive - Single Direction</a> <a href="#">Single DOL Drive with Forward/Reverse Capability</a> <a href="#">Single VSD Drive with Forward/Reverse Capability</a>
Equipment.Items that the Faceplate expects	<b>OPTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, <b>OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize

the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.

- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	Use this setting to display an alarm flag at the selected position. <b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.
Display Status Indicator	Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a> .

Option	Description
Number of Equipment	<p>If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of <a href="#">Equipment Running State Indicators</a>.</p> <p>If you select 1, the <b>Display Compact Running State</b> option will be displayed.</p> <p>If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>
Display Compact Running State	<p>Select to display a compact <a href="#">Equipment Running State Indicators</a>.</p>
Display Related Equipment Alarm Indicator	<p>Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.</p>
Related Equipment #n Name	<p>Enter a name for each piece of equipment in the group.</p>
Display Control Mode	<p>Select this option to display a mode indicator for the object.</p>
Display OOS	<p>If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.</p>
Display Meter	<p>Select this option to display a built-in meter for the object.</p>
Display Output Bar	<p>Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed.</p> <p>The Display Control <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.</p>
Display Readback	<p>Select this option to display a visual marker for the readback value on the drive's output.</p>
Display Value	<p>Select this option to display the output value of the</p>

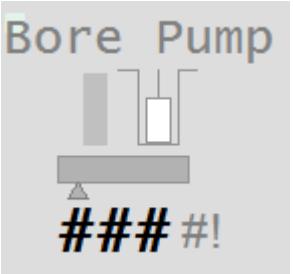
Option	Description
	drive.

## See Also

[Drives](#)

[Drive Faceplates](#)

## Bore Pump

Property	Description
Name	Bore Pump
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Drive.xml
Equipment.Items the Genie expects	<p><b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice

Property	Description
	<p>EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a>.</p>
Associated Faceplate(s)	<p><a href="#">Single and Multiple DOL Drive</a> <a href="#">Single and Multiple VSD Drive - Single Direction</a> <a href="#">Single DOL Drive with Forward/Reverse Capability</a> <a href="#">Single VSD Drive with Forward/Reverse Capability</a></p>
Equipment.Items that the Faceplate expects	<p><b>OPTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a>)  <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)  <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a>)  <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a>)  <b>Fwd, Rev, Maint</b>  <b>Stopped</b></p>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can

Option	Description
	enter a maximum of 160 characters for this option.
Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	Use this setting to display an alarm flag at the selected position. <b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.
Display Status Indicator	Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a> .
Number of Equipment	If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of <a href="#">Equipment Running State Indicators</a> . If you select 1, the <b>Display Compact Running State</b> option will be displayed. If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed. <b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.
Display Compact Running State	Select to display a compact <a href="#">Equipment Running State</a>

Option	Description
	<a href="#">Indicators</a> .
Display Related Equipment Alarm Indicator	Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.
Related Equipment #n Name	Enter a name for each piece of equipment in the group.
Display Control Mode	Select this option to display a mode indicator for the object.
Display OOS	If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.
Display Meter	Select this option to display a built-in meter for the object.
Display Output Bar	Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed. The Display Control <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)

[Drive Faceplates](#)

## Brake

Property	Description
Name	Brake
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Drive.xml
Equipment.Items the Genie expects	<p><b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	<a href="#">Single and Multiple DOL Drive</a> <a href="#">Single and Multiple VSD Drive - Single Direction</a> <a href="#">Single DOL Drive with Forward/Reverse Capability</a> <a href="#">Single VSD Drive with Forward/Reverse Capability</a>

Property	Description
Equipment.Items that the Faceplate expects	<b>OPTTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.

Option	Description
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	<p>Use this setting to display an alarm flag at the selected position.</p> <p><b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.</p>
Display Status Indicator	<p>Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a>.</p>
Number of Equipment	<p>If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of <a href="#">Equipment Running State Indicators</a>.</p> <p>If you select 1, the <b>Display Compact Running State</b> option will be displayed.</p> <p>If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>
Display Compact Running State	Select to display a compact <a href="#">Equipment Running State Indicators</a> .
Display Related Equipment Alarm Indicator	Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.
Related Equipment #n Name	Enter a name for each piece of equipment in the group.
Display Control Mode	Select this option to display a mode indicator for the object.
Display OOS	If the number of sub equipment associated with the drive object is more than 1, you can select this option

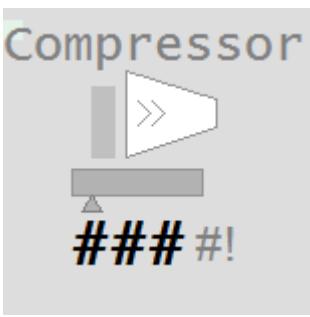
Option	Description
	to display an "Out of Service" indicator for each related equipment.
Display Meter	Select this option to display a built-in meter for the object.
Display Output Bar	Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed. The Display Control <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)

[Drive Faceplates](#)

## Compressor

Property	Description
Name	Compressor
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Drive.xml
Equipment.Items the Genie expects	<b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a> )

Property	Description
	<p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	<a href="#">Single and Multiple DOL Drive</a> <a href="#">Single and Multiple VSD Drive - Single Direction</a> <a href="#">Single DOL Drive with Forward/Reverse Capability</a> <a href="#">Single VSD Drive with Forward/Reverse Capability</a>
Equipment.Items that the Faceplate expects	<b>OPTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, <b>OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize

the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.

- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	Use this setting to display an alarm flag at the selected position. <b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.
Display Status Indicator	Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a> .

Option	Description
Number of Equipment	<p>If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of <a href="#">Equipment Running State Indicators</a>.</p> <p>If you select 1, the <b>Display Compact Running State</b> option will be displayed.</p> <p>If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>
Display Compact Running State	<p>Select to display a compact <a href="#">Equipment Running State Indicators</a>.</p>
Display Related Equipment Alarm Indicator	<p>Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.</p>
Related Equipment #n Name	<p>Enter a name for each piece of equipment in the group.</p>
Display Control Mode	<p>Select this option to display a mode indicator for the object.</p>
Display OOS	<p>If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.</p>
Display Meter	<p>Select this option to display a built-in meter for the object.</p>
Display Output Bar	<p>Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed.</p> <p>The Display Control <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.</p>
Display Readback	<p>Select this option to display a visual marker for the readback value on the drive's output.</p>
Display Value	<p>Select this option to display the output value of the</p>

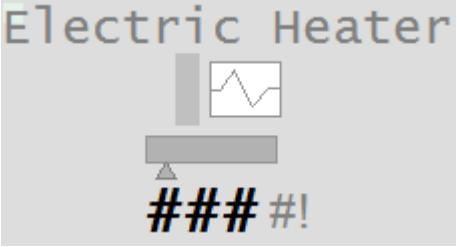
Option	Description
	drive.

## See Also

[Drives](#)

[Drive Faceplates](#)

## Electric Heater

Property	Description
Name	Electric Heater
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Drive.xml
Equipment.Items the Genie expects	<p><b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice

Property	Description
	<p>EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a>.</p>
Associated Faceplate(s)	<p><a href="#">Single and Multiple DOL Drive</a> <a href="#">Single and Multiple VSD Drive - Single Direction</a> <a href="#">Single DOL Drive with Forward/Reverse Capability</a> <a href="#">Single VSD Drive with Forward/Reverse Capability</a></p>
Equipment.Items that the Faceplate expects	<p><b>OPTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a>)  <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)  <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a>)  <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a>)  <b>Fwd, Rev, Maint</b>  <b>Stopped</b></p>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can

Option	Description
	enter a maximum of 160 characters for this option.
Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	Use this setting to display an alarm flag at the selected position.  <b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.
Display Status Indicator	Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a> .
Number of Equipment	If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of <a href="#">Equipment Running State Indicators</a> .  If you select 1, the <b>Display Compact Running State</b> option will be displayed.  If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.  <b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.
Display Compact Running State	Select to display a compact <a href="#">Equipment Running State</a>

Option	Description
	<a href="#">Indicators</a> .
Display Related Equipment Alarm Indicator	Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.
Related Equipment #n Name	Enter a name for each piece of equipment in the group.
Display Control Mode	Select this option to display a mode indicator for the object.
Display OOS	If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.
Display Meter	Select this option to display a built-in meter for the object.
Display Output Bar	Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed. The Display Control <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)

[Drive Faceplates](#)

## Fan

Property	Description
Name	Fan
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Drive.xml
Equipment.Items the Genie expects	<p><b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	<a href="#">Single and Multiple DOL Drive</a> <a href="#">Single and Multiple VSD Drive - Single Direction</a> <a href="#">Single DOL Drive with Forward/Reverse Capability</a>

Property	Description
	Single VSD Drive with Forward/Reverse Capability
Equipment.Items that the Faceplate expects	<b>OPTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.

Option	Description
Display Label	<p>Select the location to display a label.</p> <p>Select <b>None</b> if you do not want to display a label.</p>
Label	<p>Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.</p>
Display Alarm Indicator	<p>Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.</p>
Display Alarm Flag	<p>Use this setting to display an alarm flag at the selected position.</p> <p><b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.</p>
Display Status Indicator	<p>Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a>.</p>
Number of Equipment	<p>If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of <a href="#">Equipment Running State Indicators</a>.</p> <p>If you select 1, the <b>Display Compact Running State</b> option will be displayed.</p> <p>If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>
Display Compact Running State	<p>Select to display a compact <a href="#">Equipment Running State Indicators</a>.</p>
Display Related Equipment Alarm Indicator	<p>Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.</p>
Related Equipment #n Name	<p>Enter a name for each piece of equipment in the group.</p>
Display Control Mode	<p>Select this option to display a mode indicator for the object.</p>

Option	Description
Display OOS	If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.
Display Meter	Select this option to display a built-in meter for the object.
Display Output Bar	<p>Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed.</p> <p>The Display Control <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.</p>
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)

[Drive Faceplates](#)

## Motor

Property	Description
Name	Motor
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Drive.xml

Property	Description
Equipment.Items the Genie expects	<p><b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	Single and Multiple DOL Drive Single and Multiple VSD Drive - Single Direction Single DOL Drive with Forward/Reverse Capability Single VSD Drive with Forward/Reverse Capability
Equipment.Items that the Faceplate expects	<b>OPTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	Use this setting to display an alarm flag at the selected position. <b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.

Option	Description
Display Status Indicator	<p>Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a>.</p>
Number of Equipment	<p>If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of <a href="#">Equipment Running State Indicators</a>.</p> <p>If you select 1, the <b>Display Compact Running State</b> option will be displayed.</p> <p>If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>
Display Compact Running State	<p>Select to display a compact <a href="#">Equipment Running State Indicators</a>.</p>
Display Related Equipment Alarm Indicator	<p>Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.</p>
Related Equipment #n Name	<p>Enter a name for each piece of equipment in the group.</p>
Display Control Mode	<p>Select this option to display a mode indicator for the object.</p>
Display OOS	<p>If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.</p>
Display Meter	<p>Select this option to display a built-in meter for the object.</p>
Display Output Bar	<p>Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed.</p> <p>The Display Control <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.</p>

Option	Description
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)

[Drive Faceplates](#)

## Pump

Property	Description
Name	Pump
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Drive.xml
Equipment.Items the Genie expects	<p><b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for</p>

Property	Description
	more information about these items.
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	Single and Multiple DOL Drive Single and Multiple VSD Drive - Single Direction Single DOL Drive with Forward/Reverse Capability Single VSD Drive with Forward/Reverse Capability
Equipment.Items that the Faceplate expects	<b>OPTTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	Use this setting to display an alarm flag at the selected position. <b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.
Display Status Indicator	Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a> .
Number of Equipment	If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of <a href="#">Equipment Running State Indicators</a> . If you select 1, the <b>Display Compact Running State</b> option will be displayed. If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be

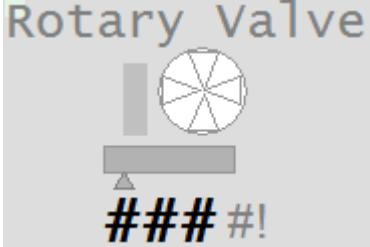
Option	Description
	<p>displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>
Display Compact Running State	<p>Select to display a compact <a href="#">Equipment Running State Indicators</a>.</p>
Display Related Equipment Alarm Indicator	<p>Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.</p>
Related Equipment #n Name	<p>Enter a name for each piece of equipment in the group.</p>
Display Control Mode	<p>Select this option to display a mode indicator for the object.</p>
Display OOS	<p>If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.</p>
Display Meter	<p>Select this option to display a built-in meter for the object.</p>
Display Output Bar	<p>Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed.</p> <p>The Display Control <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.</p>
Display Readback	<p>Select this option to display a visual marker for the readback value on the drive's output.</p>
Display Value	<p>Select this option to display the output value of the drive.</p>

## See Also

[Drives](#)

[Drive Faceplates](#)

## Rotary Valve

Property	Description
Name	Rotary Valve
Graphical Representation	 <b>#####!</b>
Example Equipment Template	Drive
Associated Composite Genie	Drive.xml
Equipment.Items the Genie expects	<p><b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	<a href="#">Single and Multiple DOL Drive</a> <a href="#">Single and Multiple VSD Drive - Single Direction</a> <a href="#">Single DOL Drive with Forward/Reverse Capability</a> <a href="#">Single VSD Drive with Forward/Reverse Capability</a>

Property	Description
Equipment.Items that the Faceplate expects	<b>OPTTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.

Option	Description
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	<p>Use this setting to display an alarm flag at the selected position.</p> <p><b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.</p>
Display Status Indicator	<p>Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a>.</p>
Number of Equipment	<p>If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of <a href="#">Equipment Running State Indicators</a>.</p> <p>If you select 1, the <b>Display Compact Running State</b> option will be displayed.</p> <p>If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>
Display Compact Running State	Select to display a compact <a href="#">Equipment Running State Indicators</a> .
Display Related Equipment Alarm Indicator	Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.
Related Equipment #n Name	Enter a name for each piece of equipment in the group.
Display Control Mode	Select this option to display a mode indicator for the object.
Display OOS	If the number of sub equipment associated with the drive object is more than 1, you can select this option

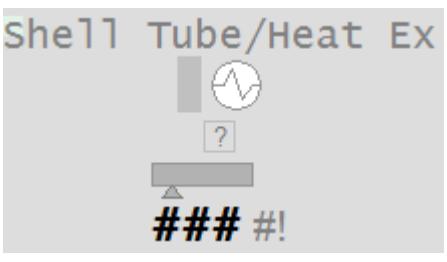
Option	Description
	to display an "Out of Service" indicator for each related equipment.
Display Meter	Select this option to display a built-in meter for the object.
Display Output Bar	Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed. The Display Control <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)

[Drive Faceplates](#)

## Shell and Tube Heat Exchanger

Property	Description
Name	Shell and Tube Heat Exchanger
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Drive.xml
Equipment.Items the Genie expects	<b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining</a> )

Property	Description
	<p><a href="#">Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	Single and Multiple DOL Drive Single and Multiple VSD Drive - Single Direction Single DOL Drive with Forward/Reverse Capability Single VSD Drive with Forward/Reverse Capability
Equipment.Items that the Faceplate expects	<b>OPTTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a

graphics page. You can add one or more instances of an equipment to a page.

- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	Use this setting to display an alarm flag at the selected position. <b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.
Display Status Indicator	Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a> .

Option	Description
Number of Equipment	<p>If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of <a href="#">Equipment Running State Indicators</a>.</p> <p>If you select 1, the <b>Display Compact Running State</b> option will be displayed.</p> <p>If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>
Display Compact Running State	<p>Select to display a compact <a href="#">Equipment Running State Indicators</a>.</p>
Display Related Equipment Alarm Indicator	<p>Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.</p>
Related Equipment #n Name	<p>Enter a name for each piece of equipment in the group.</p>
Display Control Mode	<p>Select this option to display a mode indicator for the object.</p>
Display OOS	<p>If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.</p>
Display Meter	<p>Select this option to display a built-in meter for the object.</p>
Display Output Bar	<p>Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed.</p> <p>The Display Control <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.</p>
Display Readback	<p>Select this option to display a visual marker for the readback value on the drive's output.</p>
Display Value	<p>Select this option to display the output value of the</p>

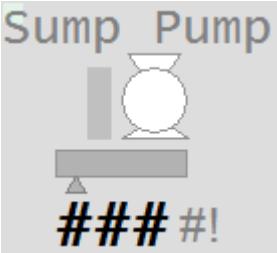
Option	Description
	drive.

## See Also

[Drives](#)

[Drive Faceplates](#)

## Sump Pump

Property	Description
Name	Sump Pump
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Drive.xml
Equipment.Items the Genie expects	<p><b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice

Property	Description
	<p>EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a>.</p>
Associated Faceplate(s)	<p><a href="#">Single and Multiple DOL Drive</a> <a href="#">Single and Multiple VSD Drive - Single Direction</a> <a href="#">Single DOL Drive with Forward/Reverse Capability</a> <a href="#">Single VSD Drive with Forward/Reverse Capability</a></p>
Equipment.Items that the Faceplate expects	<p><b>OPTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a>)  <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)  <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a>)  <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a>)  <b>Fwd, Rev, Maint</b>  <b>Stopped</b></p>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can

Option	Description
	enter a maximum of 160 characters for this option.
Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	Use this setting to display an alarm flag at the selected position.  <b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.
Display Status Indicator	Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a> .
Number of Equipment	If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of <a href="#">Equipment Running State Indicators</a> .  If you select 1, the <b>Display Compact Running State</b> option will be displayed.  If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.  <b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.
Display Compact Running State	Select to display a compact <a href="#">Equipment Running State</a>

Option	Description
	<a href="#">Indicators</a> .
Display Related Equipment Alarm Indicator	Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.
Related Equipment #n Name	Enter a name for each piece of equipment in the group.
Display Control Mode	Select this option to display a mode indicator for the object.
Display OOS	If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.
Display Meter	Select this option to display a built-in meter for the object.
Display Output Bar	<p>Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed.</p> <p>The Display Control <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.</p>
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)

[Drive Faceplates](#)

## Turbine

Property	Description
Name	Turbine
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Drive.xml
Equipment.Items the Genie expects	<p><b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	<a href="#">Single and Multiple DOL Drive</a> <a href="#">Single and Multiple VSD Drive - Single Direction</a> <a href="#">Single DOL Drive with Forward/Reverse Capability</a> <a href="#">Single VSD Drive with Forward/Reverse Capability</a>

Property	Description
Equipment.Items that the Faceplate expects	<b>OPTTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.

Option	Description
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	<p>Use this setting to display an alarm flag at the selected position.</p> <p><b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.</p>
Display Status Indicator	<p>Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a>.</p>
Number of Equipment	<p>If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of <a href="#">Equipment Running State Indicators</a>.</p> <p>If you select 1, the <b>Display Compact Running State</b> option will be displayed.</p> <p>If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>
Display Compact Running State	Select to display a compact <a href="#">Equipment Running State Indicators</a> .
Display Related Equipment Alarm Indicator	Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.
Related Equipment #n Name	Enter a name for each piece of equipment in the group.
Display Control Mode	Select this option to display a mode indicator for the object.
Display OOS	If the number of sub equipment associated with the drive object is more than 1, you can select this option

Option	Description
	to display an "Out of Service" indicator for each related equipment.
Display Meter	Select this option to display a built-in meter for the object.
Display Output Bar	Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed. The Display Control <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)

[Drive Faceplates](#)

## Belt Objects

The belt objects that are available in the Situational Awareness library include:

- [Apron](#)
- [Belt Feeder](#)
- [Conveyor](#)

There are also a set of drive objects for use within the mining industry, see [Mining Objects](#).

## See Also

[Drives](#)

[Drive Faceplates](#)

## Apron

Property	Description
Name	Apron
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Belt.xml
Equipment.Items the Genie expects	<p><b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	<a href="#">Single and Multiple DOL Drive</a> <a href="#">Single and Multiple VSD Drive - Single Direction</a> <a href="#">Single DOL Drive with Forward/Reverse Capability</a> <a href="#">Single VSD Drive with Forward/Reverse Capability</a>

Property	Description
Equipment.Items that the Faceplate expects	<b>OPTTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Belt Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.

Option	Description
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	<p>Use this setting to display an alarm flag at the selected position.</p> <p><b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.</p>
Display Status Indicator	Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a> .
Number of Equipment	<p>If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of <a href="#">Equipment Running State Indicators</a>.</p> <p>If you select 1, the <b>Display Compact Running State</b> option will be displayed.</p> <p>If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>
Display Compact Running State	Select to display a compact <a href="#">Equipment Running State Indicators</a> .
Display Related Equipment Alarm Indicator	Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.
Related Equipment #n Name	Enter a name for each piece of equipment in the group.
Display Control Mode	Select this option to display a <a href="#">mode indicator</a> for the object.
Display OOS	If the number of sub equipment associated with the drive object is more than 1, you can select this option

Option	Description
	to display an "Out of Service" indicator for each related equipment.
Display Meter	Select this option to display a <a href="#">built-in meter</a> for the object.
Display Output Bar	Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed. The <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)

[Drive Faceplates](#)

## Belt Feeder

Property	Description
Name	Belt
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Belt.xml
Equipment.Items the Genie expects	<b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> )

Property	Description
	<p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)  <b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)  <b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)  <b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	Single and Multiple DOL Drive Single and Multiple VSD Drive - Single Direction Single DOL Drive with Forward/Reverse Capability Single VSD Drive with Forward/Reverse Capability
Equipment.Items that the Faceplate expects	<b>OPTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- **Add Equipment Using Equipment Editor:** Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.

- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Belt Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	Use this setting to display an alarm flag at the selected position. <b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.
Display Status Indicator	Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a> .
Number of Equipment	If the object represents multiple pieces of equipment,

Option	Description
	<p>select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of <a href="#">Equipment Running State Indicators</a>.</p> <p>If you select 1, the <b>Display Compact Running State</b> option will be displayed.</p> <p>If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and Display OOS options will be displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>
Display Compact Running State	Select to display a compact <a href="#">Equipment Running State Indicators</a> .
Display Related Equipment Alarm Indicator	Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.
Related Equipment #n Name	Enter a name for each piece of equipment in the group.
Display Control Mode	Select this option to display a <a href="#">mode indicator</a> for the object.
Display OOS	If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.
Display Meter	Select this option to display a <a href="#">built-in meter</a> for the object.
Display Output Bar	Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed. The <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)  
[Drive Faceplates](#)

## Conveyor

Property	Description
Name	Conveyor
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Belt.xml
Equipment.Items the Genie expects	<b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>RunStatus</b> (see <a href="#">Equipment Running States</a> ) <b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits

Property	Description
	See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	Single and Multiple DOL Drive Single and Multiple VSD Drive - Single Direction Single DOL Drive with Forward/Reverse Capability Single VSD Drive with Forward/Reverse Capability
Equipment.Items that the Faceplate expects	<b>OPTTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Belt Drive Type	From the drop down select the type of drive object to be configured.

Option	Description
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	Use this setting to display an alarm flag at the selected position.  <b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.
Display Status Indicator	Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a> .
Number of Equipment	If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of <a href="#">Equipment Running State Indicators</a> .  If you select 1, the <b>Display Compact Running State</b> option will be displayed.  If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.  <b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.
Display Compact Running State	Select to display a compact <a href="#">Equipment Running State Indicators</a> .
Display Related Equipment Alarm Indicator	Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.

Option	Description
Related Equipment #n Name	Enter a name for each piece of equipment in the group.
Display Control Mode	Select this option to display a <a href="#">mode indicator</a> for the object.
Display OOS	If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.
Display Meter	Select this option to display a <a href="#">built-in meter</a> for the object.
Display Output Bar	Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed. The <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)  
[Drive Faceplates](#)

## Mining Objects

The following objects are intended for use with mining industry projects:

- [Crusher](#)
- [Diverter Gate](#)
- [Dust Collector](#)
- [Feeder Gate](#)
- [Magnet](#)
- [Metal Detector](#)
- [Primary Cutter](#)
- [Screen](#)

- Secondary Cutter
- Slider Gate
- Sprayer
- Vezin Cutter
- Vibrating Feeder

There is also a set of [Drive Objects](#) and [Belt Objects](#).

## See Also

[Drives](#)

[Drive Faceplates](#)

## Crusher

Property	Description
Name	Crusher
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Mining.xml
Equipment.Items the Genie expects	<b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>RunStatus</b> (see <a href="#">Equipment Running States</a> ) <b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for

Property	Description
	more information about these items.
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	Single and Multiple DOL Drive Single and Multiple VSD Drive - Single Direction Single DOL Drive with Forward/Reverse Capability Single VSD Drive with Forward/Reverse Capability
Equipment.Items that the Faceplate expects	<b>OPTTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Mining Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	Use this setting to display an alarm flag at the selected position. <b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.
Display Status Indicator	Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a> .
Number of Equipment	If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of Running State Indicators. If you select 1, the <b>Display Compact Running State</b> option will be displayed. If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be

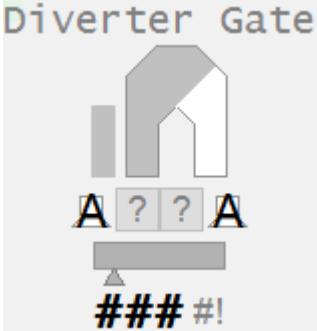
Option	Description
	<p>displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>
Display Compact Running State	<p>Select to display a compact <a href="#">Equipment Running State Indicators</a>.</p>
Display Related Equipment Alarm Indicator	<p>Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.</p>
Related Equipment #n Name	<p>Enter a name for each piece of equipment in the group.</p>
Display Control Mode	<p>Select this option to display a <a href="#">mode indicator</a> for the object.</p>
Display OOS	<p>If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.</p>
Display Meter	<p>Select this option to display a <a href="#">built-in meter</a> for the object.</p>
Display Output Bar	<p>Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed.</p> <p>The <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.</p>
Display Readback	<p>Select this option to display a visual marker for the readback value on the drive's output.</p>
Display Value	<p>Select this option to display the output value of the drive.</p>

## See Also

[Drives](#)

[Drive Faceplates](#)

## Diverter Gate

Property	Description
Name	Diverter Gate
Graphical Representation	
Example Equipment Template	DiverterGate
Associated Composite Genie	Mining.xml
Equipment.Items the Genie expects	<p><b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	<a href="#">Single and Multiple DOL Drive</a> <a href="#">Single and Multiple VSD Drive - Single Direction</a> <a href="#">Single DOL Drive with Forward/Reverse Capability</a>

Property	Description
	Single VSD Drive with Forward/Reverse Capability
Equipment.Items that the Faceplate expects	<b>OPTTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Mining Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.

Option	Description
Display Label	<p>Select the location to display a label.</p> <p>Select <b>None</b> if you do not want to display a label.</p>
Label	<p>Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.</p>
Display Alarm Indicator	<p>Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.</p>
Display Alarm Flag	<p>Use this setting to display an alarm flag at the selected position.</p> <p><b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.</p>
Display Status Indicator	<p>Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator.</p> <p>For more information, see <a href="#">Status Indicators</a>.</p>
Number of Equipment	<p>If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of Running State Indicators.</p> <p>If you select 1, the <b>Display Compact Running State</b> option will be displayed.</p> <p>If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>
Display Compact Running State	<p>Select to display a compact <a href="#">Equipment Running State Indicators</a>.</p>
Display Related Equipment Alarm Indicator	<p>Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.</p>
Related Equipment #n Name	<p>Enter a name for each piece of equipment in the group.</p>
Display Control Mode	<p>Select this option to display a <a href="#">mode indicator</a> for the object.</p>

Option	Description
Display OOS	If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.
Display Meter	Select this option to display a <a href="#">built-in meter</a> for the object.
Display Output Bar	Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed. The <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)

[Drive Faceplates](#)

## Dust Collector

Property	Description
Name	Dust Collector
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Mining.xml
Equipment.Items the Genie expects	<b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a> )

Property	Description
	<p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	<a href="#">Single and Multiple DOL Drive</a> <a href="#">Single and Multiple VSD Drive - Single Direction</a> <a href="#">Single DOL Drive with Forward/Reverse Capability</a> <a href="#">Single VSD Drive with Forward/Reverse Capability</a>
Equipment.Items that the Faceplate expects	<b>OPTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, <b>OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize

the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.

- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Mining Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	Use this setting to display an alarm flag at the selected position. <b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.
Display Status Indicator	Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a> .

Option	Description
Number of Equipment	<p>If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of Running State Indicators.</p> <p>If you select 1, the <b>Display Compact Running State</b> option will be displayed.</p> <p>If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>
Display Compact Running State	Select to display a compact <a href="#">Equipment Running State Indicators</a> .
Display Related Equipment Alarm Indicator	Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.
Related Equipment #n Name	Enter a name for each piece of equipment in the group.
Display Control Mode	Select this option to display a <a href="#">mode indicator</a> for the object.
Display OOS	If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.
Display Meter	Select this option to display a <a href="#">built-in meter</a> for the object.
Display Output Bar	<p>Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed.</p> <p>The <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.</p>
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)  
[Drive Faceplates](#)

## Feeder Gate

Property	Description
Name	Feeder Gate
Graphical Representation	
Example Equipment Template	FeederGate
Associated Composite Genie	Mining.xml
Equipment.Items the Genie expects	<b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>RunStatus</b> (see <a href="#">Equipment Running States</a> ) <b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits

Property	Description
	See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	Single and Multiple DOL Drive Single and Multiple VSD Drive - Single Direction Single DOL Drive with Forward/Reverse Capability Single VSD Drive with Forward/Reverse Capability
Equipment.Items that the Faceplate expects	<b>OPTTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Mining Drive Type	From the drop down select the type of drive object to be configured.

Option	Description
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	Use this setting to display an alarm flag at the selected position.  <b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.
Display Status Indicator	Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a> .
Number of Equipment	If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of Running State Indicators.  If you select 1, the <b>Display Compact Running State</b> option will be displayed.  If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.  <b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.
Display Compact Running State	Select to display a compact <a href="#">Equipment Running State Indicators</a> .
Display Related Equipment Alarm Indicator	Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.

Option	Description
Related Equipment #n Name	Enter a name for each piece of equipment in the group.
Display Control Mode	Select this option to display a <a href="#">mode indicator</a> for the object.
Display OOS	If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.
Display Meter	Select this option to display a <a href="#">built-in meter</a> for the object.
Display Output Bar	Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed. The <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)

[Drive Faceplates](#)

## Magnet

Property	Description
Name	Magnet
Graphical Representation	

Property	Description
Example Equipment Template	Drive
Associated Composite Genie	Mining.xml
Equipment.Items the Genie expects	<p><b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	Single and Multiple DOL Drive Single and Multiple VSD Drive - Single Direction Single DOL Drive with Forward/Reverse Capability Single VSD Drive with Forward/Reverse Capability
Equipment.Items that the Faceplate expects	<b>OPTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Mining Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	Use this setting to display an alarm flag at the selected position. <b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.

Option	Description
Display Status Indicator	<p>Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a>.</p>
Number of Equipment	<p>If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of Running State Indicators.</p> <p>If you select 1, the <b>Display Compact Running State</b> option will be displayed.</p> <p>If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>
Display Compact Running State	<p>Select to display a compact <a href="#">Equipment Running State Indicators</a>.</p>
Display Related Equipment Alarm Indicator	<p>Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.</p>
Related Equipment #n Name	<p>Enter a name for each piece of equipment in the group.</p>
Display Control Mode	<p>Select this option to display a <a href="#">mode indicator</a> for the object.</p>
Display OOS	<p>If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.</p>
Display Meter	<p>Select this option to display a <a href="#">built-in meter</a> for the object.</p>
Display Output Bar	<p>Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed.</p> <p>The <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.</p>
Display Readback	<p>Select this option to display a visual marker for the</p>

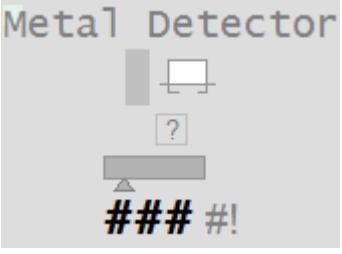
Option	Description
	readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)

[Drive Faceplates](#)

## Metal Detector

Property	Description
Name	Metal Detector
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Mining.xml
Equipment.Items the Genie expects	<p><b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>

Property	Description
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	Single and Multiple DOL Drive Single and Multiple VSD Drive - Single Direction Single DOL Drive with Forward/Reverse Capability Single VSD Drive with Forward/Reverse Capability
Equipment.Items that the Faceplate expects	<b>OPTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Mining Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	<p>Select the location to display a label. Select <b>None</b> if you do not want to display a label.</p>
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	<p>Use this setting to display an alarm flag at the selected position.</p> <p><b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.</p>
Display Status Indicator	<p>Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator.</p> <p>For more information, see <a href="#">Status Indicators</a>.</p>
Number of Equipment	<p>If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of Running State Indicators.</p> <p>If you select 1, the <b>Display Compact Running State</b> option will be displayed.</p> <p>If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>

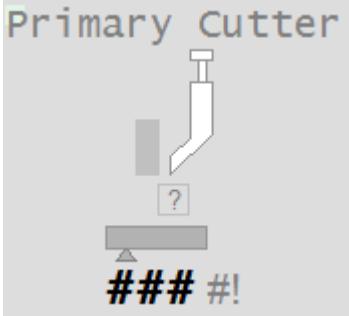
Option	Description
Display Compact Running State	Select to display a compact <a href="#">Equipment Running State Indicators</a> .
Display Related Equipment Alarm Indicator	Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.
Related Equipment #n Name	Enter a name for each piece of equipment in the group.
Display Control Mode	Select this option to display a <a href="#">mode indicator</a> for the object.
Display OOS	If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.
Display Meter	Select this option to display a <a href="#">built-in meter</a> for the object.
Display Output Bar	Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed. <a href="#">The Control Meters - Common Elements</a> represents the actual output for the controller.
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)

[Drive Faceplates](#)

## Primary Cutter

Property	Description
Name	Primary Cutter
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Mining.xml
Equipment.Items the Genie expects	<p><b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	<a href="#">Single and Multiple DOL Drive</a> <a href="#">Single and Multiple VSD Drive - Single Direction</a> <a href="#">Single DOL Drive with Forward/Reverse Capability</a>

Property	Description
	Single VSD Drive with Forward/Reverse Capability
Equipment.Items that the Faceplate expects	<p><b>OPTTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>Fwd, Rev, Maint</b></p> <p><b>Stopped</b></p>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Mining Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.

Option	Description
Display Label	<p>Select the location to display a label.</p> <p>Select <b>None</b> if you do not want to display a label.</p>
Label	<p>Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.</p>
Display Alarm Indicator	<p>Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.</p>
Display Alarm Flag	<p>Use this setting to display an alarm flag at the selected position.</p> <p><b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.</p>
Display Status Indicator	<p>Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator.</p> <p>For more information, see <a href="#">Status Indicators</a>.</p>
Number of Equipment	<p>If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of Running State Indicators.</p> <p>If you select 1, the <b>Display Compact Running State</b> option will be displayed.</p> <p>If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>
Display Compact Running State	<p>Select to display a compact <a href="#">Equipment Running State Indicators</a>.</p>
Display Related Equipment Alarm Indicator	<p>Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.</p>
Related Equipment #n Name	<p>Enter a name for each piece of equipment in the group.</p>
Display Control Mode	<p>Select this option to display a <a href="#">mode indicator</a> for the object.</p>

Option	Description
Display OOS	If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.
Display Meter	Select this option to display a <a href="#">built-in meter</a> for the object.
Display Output Bar	Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed. The <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)

[Drive Faceplates](#)

## Screen

Property	Description
Name	Screen
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Mining.xml
Equipment.Items the Genie expects	<b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a> )

Property	Description
	<p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	<a href="#">Single and Multiple DOL Drive</a> <a href="#">Single and Multiple VSD Drive - Single Direction</a> <a href="#">Single DOL Drive with Forward/Reverse Capability</a> <a href="#">Single VSD Drive with Forward/Reverse Capability</a>
Equipment.Items that the Faceplate expects	<b>OPTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, <b>OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize

the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.

- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

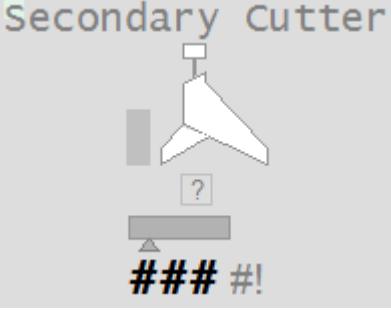
Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Mining Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	Use this setting to display an alarm flag at the selected position. <b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.
Display Status Indicator	Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a> .

Option	Description
Number of Equipment	<p>If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of Running State Indicators.</p> <p>If you select 1, the <b>Display Compact Running State</b> option will be displayed.</p> <p>If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>
Display Compact Running State	Select to display a compact <a href="#">Equipment Running State Indicators</a> .
Display Related Equipment Alarm Indicator	Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.
Related Equipment #n Name	Enter a name for each piece of equipment in the group.
Display Control Mode	Select this option to display a <a href="#">mode indicator</a> for the object.
Display OOS	If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.
Display Meter	Select this option to display a <a href="#">built-in meter</a> for the object.
Display Output Bar	<p>Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed.</p> <p>The <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.</p>
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)  
[Drive Faceplates](#)

### Secondary Cutter

Property	Description
Name	Secondary Cutter
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Mining.xml
Equipment.Items the Genie expects	<p><b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range

Property	Description
	Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	<a href="#">Single and Multiple DOL Drive</a> <a href="#">Single and Multiple VSD Drive - Single Direction</a> <a href="#">Single DOL Drive with Forward/Reverse Capability</a> <a href="#">Single VSD Drive with Forward/Reverse Capability</a>
Equipment.Items that the Faceplate expects	<b>OPTTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Mining Drive Type	From the drop down select the type of drive object to be configured.

Option	Description
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	Use this setting to display an alarm flag at the selected position.  <b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.
Display Status Indicator	Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a> .
Number of Equipment	If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of Running State Indicators.  If you select 1, the <b>Display Compact Running State</b> option will be displayed.  If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.  <b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.
Display Compact Running State	Select to display a compact <a href="#">Equipment Running State Indicators</a> .
Display Related Equipment Alarm Indicator	Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.

Option	Description
Related Equipment #n Name	Enter a name for each piece of equipment in the group.
Display Control Mode	Select this option to display a <a href="#">mode indicator</a> for the object.
Display OOS	If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.
Display Meter	Select this option to display a <a href="#">built-in meter</a> for the object.
Display Output Bar	Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed. The <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)  
[Drive Faceplates](#)

## Slider Gate

Property	Description
Name	Slider Gate
Graphical Representation	

Property	Description
Example Equipment Template	Drive
Associated Composite Genie	Mining.xml
Equipment.Items the Genie expects	<p><b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	Single and Multiple DOL Drive Single and Multiple VSD Drive - Single Direction Single DOL Drive with Forward/Reverse Capability Single VSD Drive with Forward/Reverse Capability
Equipment.Items that the Faceplate expects	<b>OPTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Mining Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	Use this setting to display an alarm flag at the selected position. <b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.

Option	Description
Display Status Indicator	<p>Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a>.</p>
Number of Equipment	<p>If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of Running State Indicators.</p> <p>If you select 1, the <b>Display Compact Running State</b> option will be displayed.</p> <p>If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>
Display Compact Running State	<p>Select to display a compact <a href="#">Equipment Running State Indicators</a>.</p>
Display Related Equipment Alarm Indicator	<p>Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.</p>
Related Equipment #n Name	<p>Enter a name for each piece of equipment in the group.</p>
Display Control Mode	<p>Select this option to display a <a href="#">mode indicator</a> for the object.</p>
Display OOS	<p>If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.</p>
Display Meter	<p>Select this option to display a <a href="#">built-in meter</a> for the object.</p>
Display Output Bar	<p>Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed.</p> <p>The <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.</p>
Display Readback	<p>Select this option to display a visual marker for the</p>

Option	Description
	readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)

[Drive Faceplates](#)

## Sprayer

Property	Description
Name	Sprayer
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Mining.xml
Equipment.Items the Genie expects	<ul style="list-style-type: none"> <li>• <a href="#">Running</a></li> <li>• <a href="#">CtrlMode, CtrlModeDef</a></li> <li>• <a href="#">RunStatus</a></li> <li>• <a href="#">OP</a></li> <li>• <a href="#">PV</a></li> <li>• <a href="#">EqStatus</a></li> </ul> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these</p>

Property	Description
	items.
Equipment Parameters the Equipment Items Expect	<a href="#">Equipment Configuration Parameters</a> <a href="#">Equipment Configuration Parameters</a>
Associated Faceplate(s)	<a href="#">Single and Multiple DOL Drive</a> <a href="#">Single and Multiple VSD Drive - Single Direction</a> <a href="#">Single DOL Drive with Forward/Reverse Capability</a> <a href="#">Single VSD Drive with Forward/Reverse Capability</a>
Equipment.Items that the Faceplate expects	<a href="#">OPTTrack</a> <a href="#">CtrlMode, CtrlModeDef</a> <a href="#">AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</a> <a href="#">OOS, OOSDisabled</a> <a href="#">Fwd, Rev, Maint</a> <a href="#">Stopped</a>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Mining Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	<p>Select the location to display a label. Select <b>None</b> if you do not want to display a label.</p>
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	<p>Use this setting to display an alarm flag at the selected position.</p> <p><b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.</p>
Display Status Indicator	<p>Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator.</p> <p>For more information, see <a href="#">Status Indicators</a>.</p>
Number of Equipment	<p>If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of Running State Indicators.</p> <p>If you select 1, the <b>Display Compact Running State</b> option will be displayed.</p> <p>If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>

Option	Description
Display Compact Running State	Select to display a compact <a href="#">Equipment Running State Indicators</a> .
Display Related Equipment Alarm Indicator	Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.
Related Equipment #n Name	Enter a name for each piece of equipment in the group.
Display Control Mode	Select this option to display a <a href="#">mode indicator</a> for the object.
Display OOS	If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.
Display Meter	Select this option to display a <a href="#">built-in meter</a> for the object.
Display Output Bar	Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed. <a href="#">The Control Meters - Common Elements</a> represents the actual output for the controller.
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)

[Drive Faceplates](#)

[Vezin Cutter](#)

Property	Description
Name	Vezin Cutter
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Mining.xml
Equipment.Items the Genie expects	<p><b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	<a href="#">Single and Multiple DOL Drive</a> <a href="#">Single and Multiple VSD Drive - Single Direction</a> <a href="#">Single DOL Drive with Forward/Reverse Capability</a> <a href="#">Single VSD Drive with Forward/Reverse Capability</a>

Property	Description
Equipment.Items that the Faceplate expects	<b>OPTTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Mining Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.

Option	Description
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	<p>Use this setting to display an alarm flag at the selected position.</p> <p><b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.</p>
Display Status Indicator	<p>Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a>.</p>
Number of Equipment	<p>If the object represents multiple pieces of equipment, select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of Running State Indicators.</p> <p>If you select 1, the <b>Display Compact Running State</b> option will be displayed.</p> <p>If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and <b>Display OOS</b> options will be displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>
Display Compact Running State	Select to display a compact <a href="#">Equipment Running State Indicators</a> .
Display Related Equipment Alarm Indicator	Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.
Related Equipment #n Name	Enter a name for each piece of equipment in the group.
Display Control Mode	Select this option to display a <a href="#">mode indicator</a> for the object.
Display OOS	If the number of sub equipment associated with the drive object is more than 1, you can select this option

Option	Description
	to display an "Out of Service" indicator for each related equipment.
Display Meter	Select this option to display a <a href="#">built-in meter</a> for the object.
Display Output Bar	Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed. The <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

[Drives](#)

[Drive Faceplates](#)

## Vibrating Feeder

Property	Description
Name	Vibrating Feeder
Graphical Representation	
Example Equipment Template	Drive
Associated Composite Genie	Mining.xml
Equipment.Items the Genie expects	<b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> )

Property	Description
	<p><b>RunStatus</b> (see <a href="#">Equipment Running States</a>)</p> <p><b>OP</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>EqStatus</b> (see <a href="#">Drive,Belt and Mining Objects</a>)</p> <p><b>Note:</b> The drive object includes a built-in meter so that you can display a key analog value next to the drive symbol. If you do not want the side meter, you do not need the PV item. If you want your drive to have multiple analog values, you can display those as separate meter objects that have a set of items associated with them. Refer to the <a href="#">Meters</a> topics for more information about these items.</p>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range Alarm Limits See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	Single and Multiple DOL Drive Single and Multiple VSD Drive - Single Direction Single DOL Drive with Forward/Reverse Capability Single VSD Drive with Forward/Reverse Capability
Equipment.Items that the Faceplate expects	<b>OPTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd, ManCmd, CasCmd, FwdCmd, LocalCmd, MaintCmd, RemCmd, RevCmd, StartCmd, StopCmd, OOSCmd</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>OOS, OOSDisabled</b> (see <a href="#">Equipment Running States</a> ) <b>Fwd, Rev, Maint</b> <b>Stopped</b>

## Configuration Tasks

- **Add Equipment Using Equipment Editor:** Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.

- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Mining Drive Type	From the drop down select the type of drive object to be configured.
Size	Select the size that is appropriate for the presentation of the object on the graphics page.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page.
Display Label	Select the location to display a label. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this drive.
Display Alarm Flag	Use this setting to display an alarm flag at the selected position. <b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.
Display Status Indicator	Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see <a href="#">Status Indicators</a> .
Number of Equipment	If the object represents multiple pieces of equipment,

Option	Description
	<p>select how many pieces of equipment the object group will include. When the object displays, it will include a corresponding number of Running State Indicators.</p> <p>If you select 1, the <b>Display Compact Running State</b> option will be displayed.</p> <p>If more than 1 is selected the corresponding Sub Equipment #n Name fields will be displayed and the <b>Display MEO Alarm</b> and Display OOS options will be displayed.</p> <p><b>Note:</b> Multiple pieces of equipment (MEO) are also known as Equipment Running State Indicators.</p>
Display Compact Running State	Select to display a compact <a href="#">Equipment Running State Indicators</a> .
Display Related Equipment Alarm Indicator	Select to display the alarm indicator which indicates the highest priority alarm and its state for every related piece of equipment in this object.
Related Equipment #n Name	Enter a name for each piece of equipment in the group.
Display Control Mode	Select this option to display a <a href="#">mode indicator</a> for the object.
Display OOS	If the number of sub equipment associated with the drive object is more than 1, you can select this option to display an "Out of Service" indicator for each related equipment.
Display Meter	Select this option to display a <a href="#">built-in meter</a> for the object.
Display Output Bar	Select to display an output indicator with the object. On selecting this option, the "Display Control Readback" and "Display Control Value" options are displayed. The <a href="#">Control Meters - Common Elements</a> represents the actual output for the controller.
Display Readback	Select this option to display a visual marker for the readback value on the drive's output.
Display Value	Select this option to display the output value of the drive.

## See Also

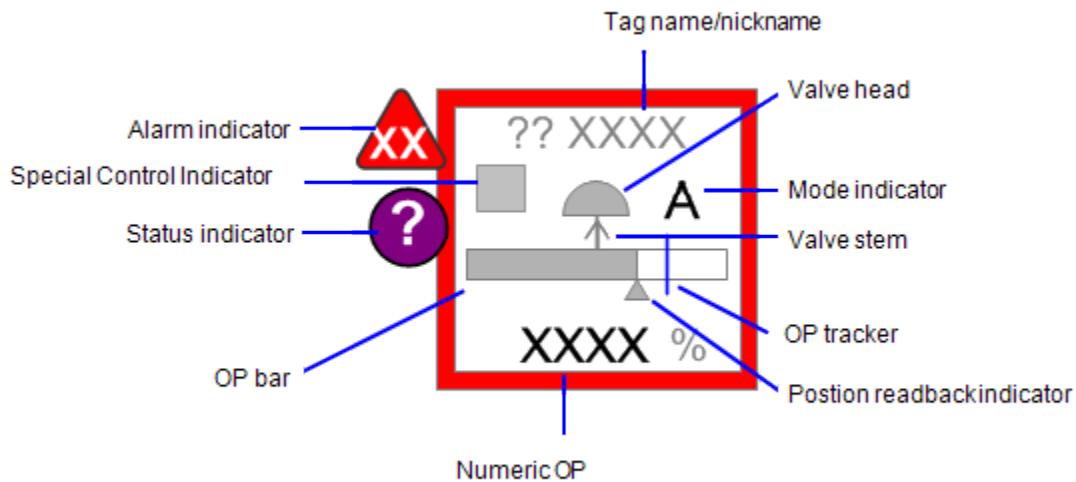
[Drives](#)  
[Drive Faceplates](#)

## Valves

The Valve objects that are available in the Situational Awareness library include:

- [Block Valve](#)
- [Control Valve](#)
- [Damper](#)
- [Hand Valve](#)

The figure below shows an example of the elements that may make up a control valve object.



## Alarm indicator

An alarm indicator is used to show the occurrence and status of alarms associated with an object. See [Alarm Indicators](#) in the section on [Common Object Elements](#).

## Status indicator

Connects to: EquipmentName.EqStatus

A status indicator is used to represent various non-alarm conditions associated with an object, such as abnormal data quality or control system states. See [Status Indicators](#) in the section on [Common Object Elements](#).

**Note:** Status Indicator is not part of the basic genie composite.

## Valve Objects - Tagname/Nickname

The tagname/nickname field shows the valve tagname (by default), or, if entered by the user, a "nickname". The nickname is usually a common name used to help identify the reading (e.g., "Fuel Gas", instead of "CV1001").

Visibility of the tagname/nickname field shall be tied to the settings for visibility of Tagnames in the "Show/Hide Settings" controls.

## OP Bar (for valves with continuous position control)

Connects to: EquipmentName.OP

For control valves and other valves that can be positioned between 0-100% open, this bar shows the commanded OP for the valve (see section 0 for a description of functionality).

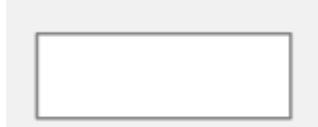
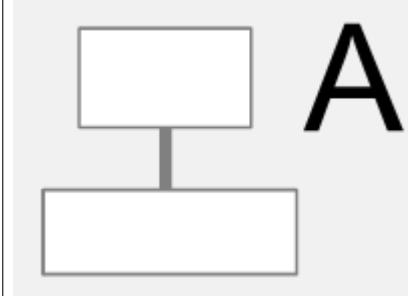
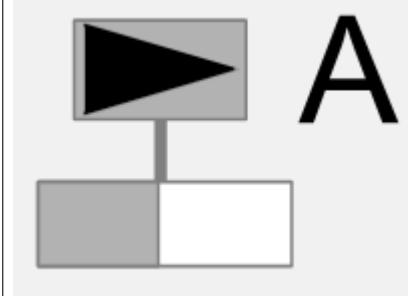
Note that the OP bar does not turn black when the valve is fully open.

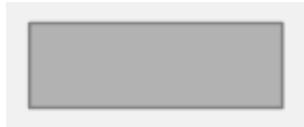
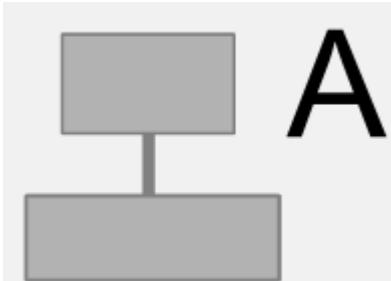
## OP Bar (for valves with discrete position control)

Connects to: EquipmentName.OP

For two-position valves (open/closed), such as block valves or simple MOVs, the OP bar is used to distinguish between closed, opening/closing, and closed (as available). When opening or closing, the bar is shown with the fill at the half-way point of the bar.

Note that the OP bar does not turn black in the open state.

State	OP Bar	Example (Block Valve)
Fully closed		
Opening or closing		

State	OP Bar	Example (Block Valve)
Fully open		

## Position Readback Indicator

Connects to: EquipmentName.FB

This triangle is used to show the actual (readback) valve position (0-100%), if available (see section 0 for a description of functionality).

---

**Note:** Should only be used on complex valves that have OP and FB items; otherwise it will not compile.

---

## Valve Objects - Numeric OP

Connects to: EquipmentName.OP

This is a numeric field used to show the commanded OP (0-100%), if applicable. It is always shown as a % value. This provides operators with precise values when needed.

The visibility of the numeric OP should be tied to the visibility state for PVs as selected in the "Show/Hide Settings" controls. If the user selects to turn off the visibility for PVs, the numeric OP values should also be hidden. The visibility should be forced to "on" if the valve goes into an alarm state.

## OP Tracker

Connects to: EquipmentName.OPTacker

The tracker is a user-configured indicator that marks a specific output value on the OP bar. A typical use case is where an operator wants to mark the current position so that when they look at the position again (some time later) they can see if it has changed. The tracker position is set and enabled/disabled from the valve faceplate (see APG dashboard specification). The tracker is only available on valves with continuous position control (control valves, dampers, and complex MOV/EOVs).

## Mode Indicator

Connects to: EquipmentName.CtrlMode, EquipmentName.CtrlModeDef

This is a single-character code that indicates the mode of the valve. It is shown next to the valve head. Typically, the set of modes includes Manual (M), Auto (A), Cascade (C) and Local (L) and special control (a computer symbol) but others may be added as required for the Control System platform.

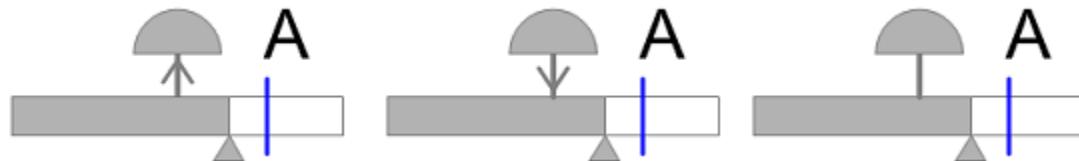
---

**Note:** The Mode Indicator is not displayed if the tag is set to CtrlModeDef. To view the control mode at all times, remove the CtrlModeDef tag from the equipment template.

---

## Valve Stem

Where applicable, the valve stem is used to show the fail action for the valve (fail open, fail closed, or fail last). These states are shown as follows:



## Valve Head (control valves and dampers)

Connects to: EquipmentName.Open, EquipmentName.POS, EquipmentName.Closed, EquipmentName.Opening, EquipmentName.Closing

For [Control Valve](#) and [Damper](#), the fill color of the valve head is used to indicate whether the valve is fully closed, partially open, or fully open. This is similar to the way fill color is used for the OP bar and readback indicators. When fully closed the fill is white. When partially open the fill is grey, and when fully open the fill turns black.

If readback is available the valve head fill should indicate the readback position. Otherwise it should indicate the commanded position.

State	Control Valve	Damper
Fully closed		
Partially Open		
Fully open		

## Valve Head (MOV/EOVs and block valves)

Connects to: EquipmentName.Open, EquipmentName.POS, EquipmentName.Closed, EquipmentName.Opening, EquipmentName.Closing

For MOV/EOVs and block valves, the valve head fill color follows the same scheme as for control valves and dampers.

In addition to the position however, the valve heads for MOV/EOVs and block valves are also used to indicate transition states. That is, the valve head uses symbols from the Equipment Running State Indicator symbol set to show when the valve is opening or closing. Opening or closing is indicated with a triangle in the valve head. See examples below:



Fully closed, commanded to open



Partially open, opening further



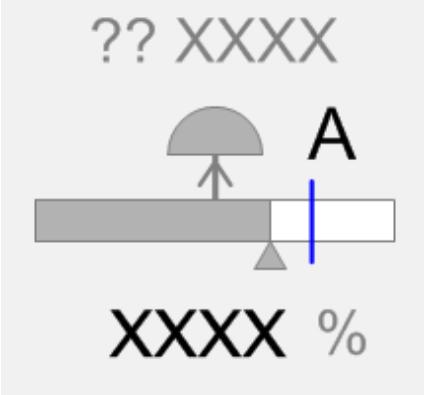
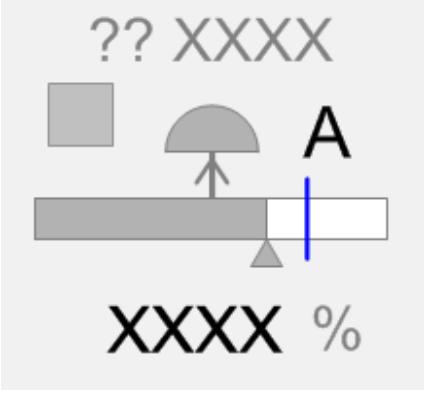
Partially open, and closing



Fully open, commanded to close

## Special Control Indicator

The special control indicator can be used to indicate either of two special states: field control and computer control. No symbol appears for valves that are under normal control via the SCADA system.

State	Appearance	
Normal		The valve is being controlled normally via the control system (no symbol shown)
Handswitch control		The handswitch control indicator appears when the valve is being controlled from the field via a local handswitch.

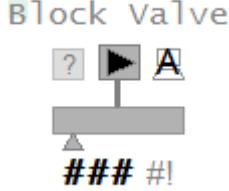
**Note:** Check that the address configured for a valve in the PLC is the same as the tag address specified in Plant SCADA. If this is not configured correctly, the state and output value of the valve will not be accurate.

## See Also

[Valve Versions](#)

[Valve Orientations](#)

## Block Valve

Property	Description
Name	Block Valve
Graphical Representation	
Example Equipment Template	Valve
Associated Composite Genie	Valve.xml
Equipment.Items the Genie expects	<b>Open, POS, Closed, Opening, Closing</b> (see <a href="#">Valves</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Valves</a> ) <b>RunStatus</b> (see <a href="#">Equipment Running States</a> ) <b>OP</b> (see <a href="#">Valves</a> ) <b>OPTTrack</b> (see <a href="#">Valves</a> ) <b>FB</b> (see <a href="#">Valves</a> ) <b>OOS, OOSDisable</b> (see <a href="#">Equipment Running States</a> ) <b>OpenCmd</b> <b>CloseCmd</b> <b>StopCmd</b> <b>AutoCmd</b> <b>ManCmd</b> <b>Stopped</b>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	<a href="#">Simple Valve Faceplate</a> <a href="#">Complex Valve</a>
Equipment.Items that the Faceplate expects	<b>OPTTrack</b> (see <a href="#">Valves</a> ) <b>CtrlModeDef</b> (see <a href="#">Valves</a> ) <b>OpenCmd</b>

Property	Description
	<b>CloseCmd</b> <b>StopCmd</b> <b>AutoCmd</b> <b>ManCmd</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

**Note:** Check that the address configured for a valve in the PLC is the same as the tag address specified in Plant SCADA. If this is not configured correctly, the state and output value of the valve will not be accurate.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Equipment Item Prefix	Specify the prefix and the equipment.item the prefix will be applied to.
Valve Type	Select the type of Valve - Block, Control or Damper, Hand.
Size	Size of the Valve - small or large.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page. Horizontal or Vertical. Horizontal is selected by default.

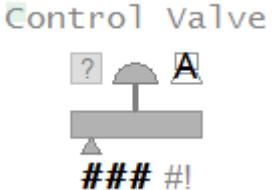
Option	Description
Fail Type	Select the fail action for the valve (open, closed, or last).
Display Label	Use this setting to display a label at the selected position. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this Valve's equipment.
Display Alarm Flag	Use this setting to display an alarm flag at the selected position. <b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.
Display Status Indicator	Select the location to display a status indicator. Select None if you do not wish to see the status indicator. For more information, see Status Indicators.
Display Output Bar	Select this option to display a field control or computer control for this object.
Display OOS	Select this option to display an out of service indicator.
Display Control Mode	Select this option to display a mode indicator for the object.
Display Readback	The Display Readback represents the actual output for the controller.
Display Value	Select this option to display the output value.

## See Also

[Valve Faceplates](#)

## Control Valve

The control valve object shows the position and status of control valves. The elements and behavior are as described in Common Elements.

Property	Description
Name	Control Valve
Graphical Representation	
Example Equipment Template	Valve
Associated Composite Genie	Valve.xml
Equipment.Items the Genie expects	<b>Open, POS, Closed, Opening, Closing</b> (see <a href="#">Valves</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Valves</a> ) <b>RunStatus</b> (see <a href="#">Equipment Running States</a> ) <b>OP</b> (see <a href="#">Valves</a> ) <b>OPTTrack</b> (see <a href="#">Valves</a> ) <b>FB</b> (see <a href="#">Valves</a> ) <b>OOS, OOSDisable</b> (see <a href="#">Equipment Running States</a> ) <b>OpenCmd</b> <b>CloseCmd</b> <b>StopCmd</b> <b>AutoCmd</b> <b>ManCmd</b> <b>Stopped</b>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	<a href="#">Simple Valve Faceplate</a> <a href="#">Complex Valve</a>
Equipment.Items that the Faceplate expects	<b>OPTTrack</b> (see <a href="#">Valves</a> ) <b>CtrlModeDef</b> (see <a href="#">Valves</a> ) <b>OpenCmd</b> <b>CloseCmd</b>

Property	Description
	<b>StopCmd</b> <b>AutoCmd</b> <b>ManCmd</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

**Note:** Check that the address configured for a valve in the PLC is the same as the tag address specified in Plant SCADA. If this is not configured correctly, the state and output value of the valve will not be accurate.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Equipment Item Prefix	Specify the prefix and the equipment.item the prefix will be applied to.
Valve Type	Select the type of Valve - Block, Control or Damper, Hand.
Size	Size of the Valve - small or large.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page. Horizontal or Vertical. Horizontal is selected by default.
Fail Type	Select the fail action for the valve (open, closed, or

Option	Description
	last).
Display Label	<p>Use this setting to display a label at the selected position.</p> <p>Select <b>None</b> if you do not want to display a label.</p>
Label	<p>Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.</p>
Display Alarm Indicator	<p>Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this Valve's equipment.</p>
Display Alarm Flag	<p>Use this setting to display an alarm flag at the selected position.</p> <p><b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.</p>
Display Status Indicator	<p>Select the location to display a status indicator. Select None if you do not wish to see the status indicator. For more information, see Status Indicators.</p>
Display Output Bar	<p>Select this option to display a field control or computer control for this object.</p>
Display OOS	<p>Select this option to display an out of service indicator.</p>
Display Control Mode	<p>Select this option to display a mode indicator for the object.</p>
Display Readback	<p>The Display Readback represents the actual output for the controller.</p>
Display Value	<p>Select this option to display the output value.</p>

## See Also

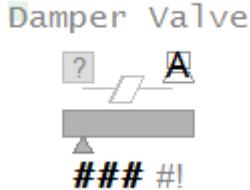
[Valve Faceplates](#)

## Damper

The damper object is based closely on the control valve object. It shows the position and status of dampers. The elements and behavior are as described in the Common Elements.

Levels:

- Level 3 (vertical, horizontal, compact vertical, compact horizontal)
- Level 2 (vertical, horizontal, compact vertical, compact horizontal)
- Level 1 (vertical, horizontal, compact vertical, compact horizontal)
- Basic (vertical, horizontal, compact vertical, compact horizontal)
- Basic with tag (vertical, horizontal, compact vertical, compact horizontal).

Property	Description
Name	Damper
Graphical Representation	
Example Equipment Template	Valve
Associated Composite Genie	Valve.xml
Equipment.Items the Genie expects	<b>Open, POS, Closed, Opening, Closing</b> (see <a href="#">Valves</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Valves</a> ) <b>RunStatus</b> (see <a href="#">Equipment Running States</a> ) <b>OP</b> (see <a href="#">Valves</a> ) <b>OPTrack</b> (see <a href="#">Valves</a> ) <b>FB</b> (see <a href="#">Valves</a> ) <b>OOS, OOSDisable</b> (see <a href="#">Equipment Running States</a> ) <b>OpenCmd</b> <b>CloseCmd</b> <b>StopCmd</b> <b>AutoCmd</b> <b>ManCmd</b> <b>Stopped</b>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range See <a href="#">Equipment Configuration Parameters</a> .

Property	Description
Associated Faceplate(s)	<a href="#">Simple Valve Faceplate</a> <a href="#">Complex Valve</a>
Equipment.Items that the Faceplate expects	<b>OPTrack</b> (see <a href="#">Valves</a> ) <b>CtrlModeDef</b> (see <a href="#">Valves</a> ) <b>OpenCmd</b> <b>CloseCmd</b> <b>StopCmd</b> <b>AutoCmd</b> <b>ManCmd</b> <b>Stopped</b>

**Note:** Check that the address configured for a valve in the PLC is the same as the tag address specified in Plant SCADA. If this is not configured correctly, the state and output value of the valve will not be accurate.

## Presentation Options

The following presentation options are available for this object.

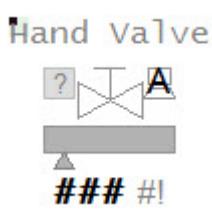
Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Equipment Item Prefix	Specify the prefix and the equipment.item the prefix will be applied to.
Valve Type	Select the type of Valve - Block, Control or Damper, Hand.
Size	Size of the Valve - small or large.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page. Horizontal or Vertical. Horizontal is selected by default.
Fail Type	Select the fail action for the valve (open, closed, or last).
Display Label	Use this setting to display a label at the selected position. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30

Option	Description
	characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this Valve's equipment.
Display Alarm Flag	<p>Use this setting to display an alarm flag at the selected position.</p> <p><b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.</p>
Display Status Indicator	Select the location to display a status indicator. Select None if you do not wish to see the status indicator. For more information, see Status Indicators.
Display Output Bar	Select this option to display a field control or computer control for this object.
Display OOS	Select this option to display an out of service indicator.
Display Control Mode	Select this option to display a mode indicator for the object.
Display Readback	The Display Readback represents the actual output for the controller.
Display Value	Select this option to display the output value.

## See Also

[Valve Faceplates](#)

## Hand Valve

Property	Description
Name	Hand Valve
Graphical Representation	

Property	Description
Example Equipment Template	Valve
Associated Composite Genie	Valve.xml
Equipment.Items the Genie expects	<b>Open, POS, Closed, Opening, Closing</b> (see <a href="#">Valves</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Valves</a> ) <b>RunStatus</b> (see <a href="#">Equipment Running States</a> ) <b>OP</b> (see <a href="#">Valves</a> ) <b>OPTTrack</b> (see <a href="#">Valves</a> ) <b>FB</b> (see <a href="#">Valves</a> ) <b>OOS, OOSDisable</b> (see <a href="#">Equipment Running States</a> ) <b>OpenCmd</b> <b>CloseCmd</b> <b>StopCmd</b> <b>AutoCmd</b> <b>ManCmd</b> <b>Stopped</b>
Equipment Parameters the Equipment Items Expect	InternalIODevice CicodeIODevice EqStatusFunc CtrlMode Range See <a href="#">Equipment Configuration Parameters</a> .
Associated Faceplate(s)	<a href="#">Simple Valve Faceplate</a> <a href="#">Complex Valve</a>
Equipment.Items that the Faceplate expects	<b>OPTTrack</b> (see <a href="#">Valves</a> ) <b>CtrlModeDef</b> (see <a href="#">Valves</a> ) <b>OpenCmd</b> <b>CloseCmd</b> <b>StopCmd</b> <b>AutoCmd</b> <b>ManCmd</b> <b>Stopped</b>

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize

the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.

- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

---

**Note:** Check that the address configured for a valve in the PLC is the same as the tag address specified in Plant SCADA. If this is not configured correctly, the state and output value of the valve will not be accurate.

---

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter a name for the piece of equipment. You can enter a maximum of 160 characters for this option.
Equipment Item Prefix	Specify the prefix and the equipment.item the prefix will be applied to.
Valve Type	Select the type of Valve - Block, Control or Damper, Hand.
Size	Size of the Valve - small or large.
Orientation	Select the orientation that is appropriate for the presentation of the object on the graphics page. Horizontal or Vertical. Horizontal is selected by default.
Fail Type	Select the fail action for the valve (open, closed, or last).
Display Label	Use this setting to display a label at the selected position. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for

Option	Description
	this Valve's equipment.
Display Alarm Flag	<p>Use this setting to display an alarm flag at the selected position.</p> <p><b>Note:</b> If you select the same position for the alarm flag and <b>Display Status</b> indicator (see below), they will overlap. The alarm flag will not be visible.</p>
Display Status Indicator	Select the location to display a status indicator. Select None if you do not wish to see the status indicator. For more information, see Status Indicators.
Display Output Bar	Select this option to display a field control or computer control for this object.
Display OOS	Select this option to display an out of service indicator.
Display Control Mode	Select this option to display a mode indicator for the object.
Display Readback	The Display Readback represents the actual output for the controller.
Display Value	Select this option to display the output value.

## See Also

[Valve Faceplates](#)

## Valve Orientations

All valves can be oriented horizontally or vertically. Block valves have additional orientation options to provide extra flexibility. Refer to the individual object specifications for details.

## See Also

[Valve Versions](#)

## Valve Versions

Most valves appear in several versions, which differ in the amount of information shown surrounding the valve. The different versions correspond roughly to the types of display pages they are typically expected to be used on. Alarm borders are re-sized as needed for the different versions.

## Level 3 Valves

Level 3 valves are the baseline version for every valve. They contain all of the information elements available. All of the examples described above show Level 3 valves. These objects are expected to be used heavily on detailed operating graphics ("Level 3" operating graphics) where showing a full complement of information is most important.

## Level 2 Valves

Level 2 valves are expected to be used on "Level 2" operating graphics, which provide overviews of major areas or units. Level 2 valves are identical to their Level 3 counterparts with the exception that the visibility of some of the elements can be "switched off" (i.e., hidden from view) by operators (during run time). The intent is to allow operators some flexibility in the amount of detail shown on their Level 2 operating graphics.

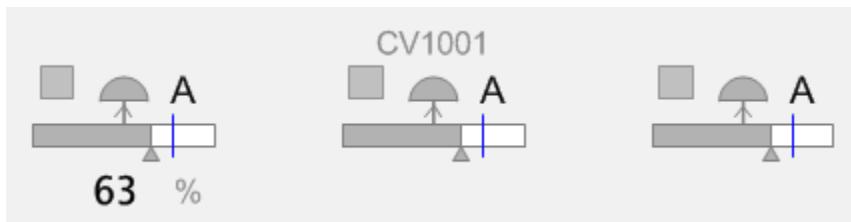
**Note:** Whether this feature is implemented using visibility properties for a single object or by creating separate objects is at the discretion of the implementation team.

The "Show/Hide Settings" controls are used by the operator to select which elements are shown for Level 2 valves. The intent is to provide this functionality globally on a workstation via a button on the toolbar and/or via a right click menu.

Elements which can be (independently) hidden are:

- Tagname / nickname
- Numeric OP and Units

All of the other common elements for valves remain unchanged.



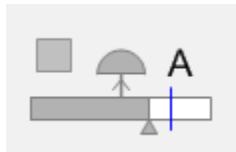
For Level 2 valves the visibility of the numeric OP will be forced on if the valve goes into an alarm state.

## Level 1 Valves

Level 1 valves can be used on the overview displays covering an operators entire span of control ("Level 1" operating graphics). These displays tend to focus more on visual presentation of process status, and less on details. Level 1 objects therefore have the visibility of tagnames/nicknames and the numeric OP elements set to "off". Unlike Level 2 valves, these elements cannot be re-enabled by operators using the Show/Hide Settings selections.

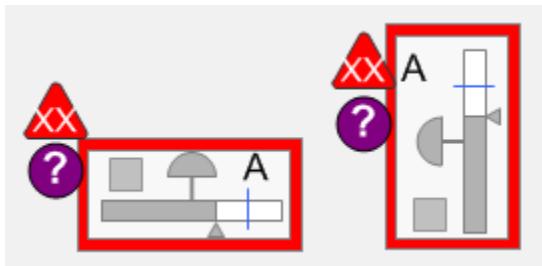
On Level 1 valves, the visibility of the numeric OP will be forced on if the valve goes into an alarm state.

All of the other common elements for valves remain unchanged.



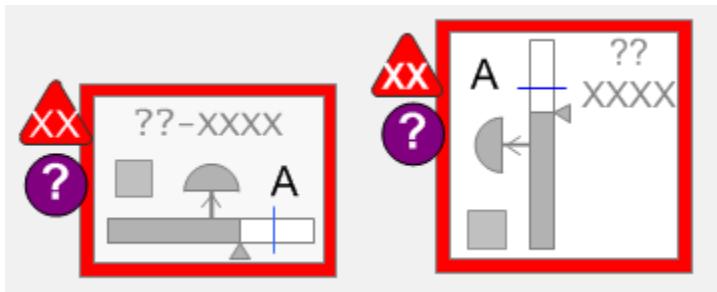
## Basic Valves

Basic valves provide page designers with a small-footprint option for special situations requiring only a basic set of information for a given valve. Basic valves do not show the tagname/nickname field, or the numeric OP and units (if applicable). Basic valves can be vertical or horizontal. Basic valves can also be mirrored in either orientation.



## Basic Valves (with Tags)

Basic valves with tags are identical to basic valves except that they retain space for a tagname/nickname field. This makes the footprint slightly larger. Basic (with tag) valves can be vertical or horizontal as shown below. They can also be mirrored in either orientation.



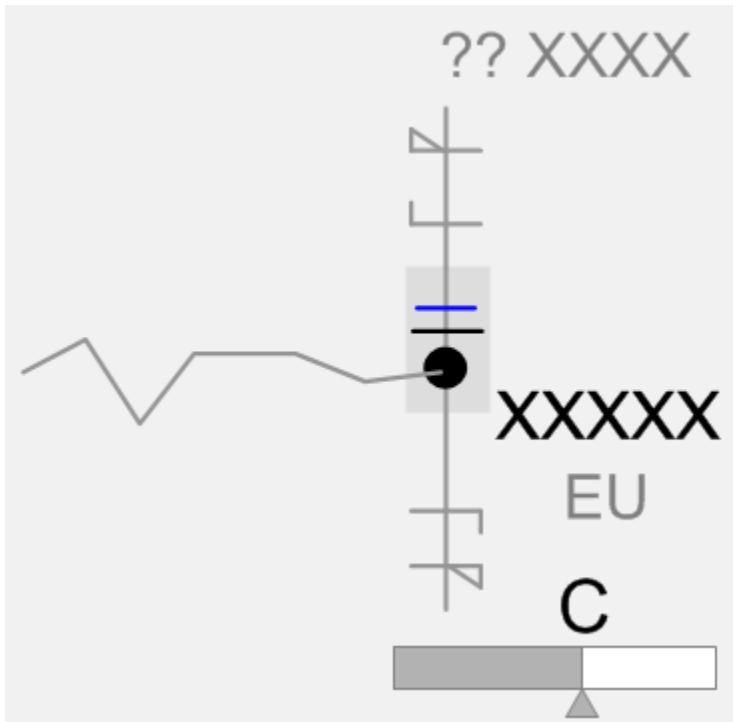
**Note:** All valve versions should include orientation options as per the individual object specifications.

## Trend Objects

The trend objects provide the ability to embed trend information in display pages. The following trend objects are available in the Situational Awareness Library:

- [Trend Tail](#)
- [Full Trend](#)
- [Four-Pen Trend](#)

## Trend Tail



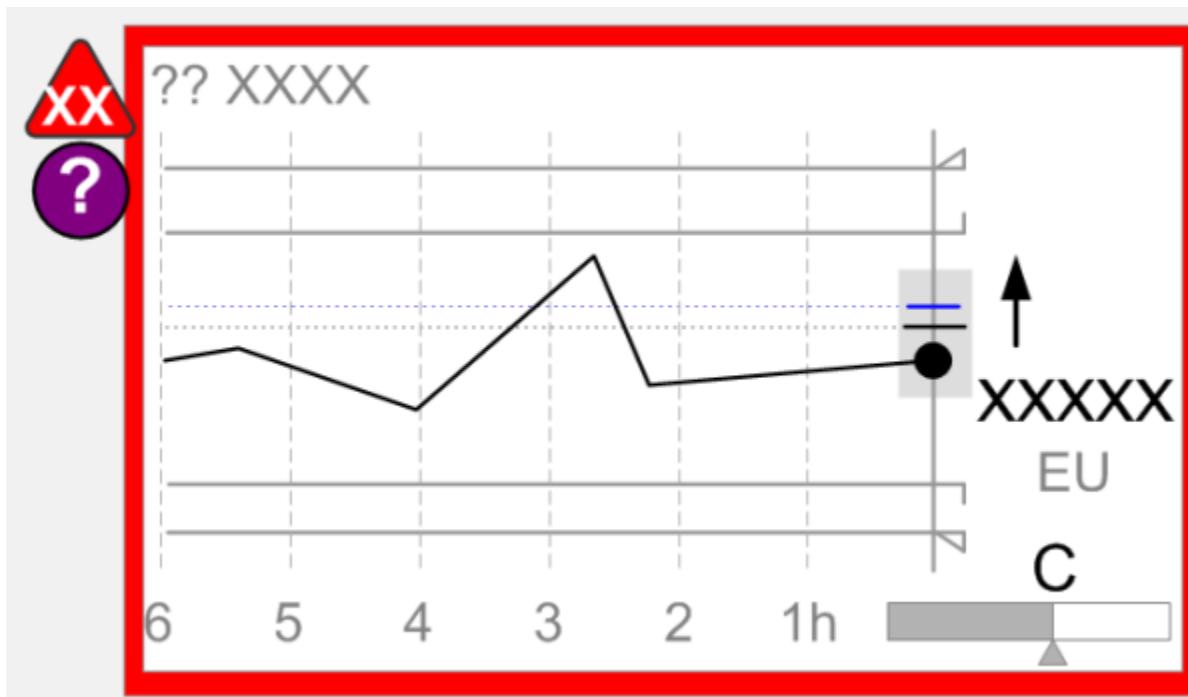
**Note:** Trend Tails are included in Meters and are not separate objects.

The Trend Tail object provides the operator with information about the historical behavior of a point. It is always used together with a meter object, and provides a relatively simple and compact method for showing trend information directly on the display.

The vertical scale of trend tail needs to be tied to the meter it is connected to. If the meter has auto-scaling turned on, or if the PV exceeds the practical range and the meter goes into "full-range" mode, then the scale of the trend must change accordingly. There is no alarming associated with the trend tail, however, the meter to which the trend tail is connected will display alarms as configured.

There are no orientation options for the trend tail. Therefore, it can only be used with meters that are in their vertical orientation.

## Full Trend



**Note:** Full Trends are included in Meters and are not separate objects.

This trend object provides basic trending capability for any instrument. This can be created by setting the **Trend Type** option to **Full** on any meter composite. The alarm limits are projected leftward onto the trend to provide reference lines for the PV. The tracker is also displayed as a time-varying value.

**Note:** The High and High High alarm limit indicators are 'flipped', so that the flags are shown on the right hand side of the meter.

## Four-Pen Trend

The Four-Pen Trend object shows between 1 and 4 Equipment Trend Items trended on the same set of axes.

The tag names corresponding to each displayed Equipment Trend Item are shown across the top of the graph. The box surrounding each tagname matches the line style and color of the trend line for that Equipment Trend Item. At runtime when you hover the mouse cursor over the boxes, the full Equipment Name is displayed. Above each box, flags are displayed to indicate the full range.

The number of vertical gridlines and the corresponding text indicating the time intervals (i.e., 1h, 2h, 3h, etc.) depends upon the **Time Span** selected in the Presentation Options for the trend.

If the trend has fewer than 4 displayed Equipment Trend Items, the objects associated with the remaining items are not displayed.

Alarming is based on the referenced equipment.

Property	Description
Name	Four-Pen Trend
Graphical Representation	
Associated Composite Genie	Trend.xml
Equipment.Item Names Associated with the Genie	PV

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter the name of the equipment for which you want to view the trend. You can enter a maximum of 160 characters for this option.
Size	Size of the Trend object image - small or large
Display Label	Use this setting to display a label at the selected position. Select <b>None</b> if you do not want to display a label.
Label	Enter the text that will display at the location specified in <b>Display Label</b> field. You can enter a maximum of 30 characters for this option.
Display Status Indicator	Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see Status Indicators.
Time Span	Select the time span to be used for the trend. The minimum is 30 seconds and the maximum is 12 hours.

## Additional Objects

This category includes a variety of objects with various functions and features. All are described with individual specifications.

- [Polar Star](#)
- [Dump Light Genie](#)
- [Polar Star](#)
- [Numeric Input](#)
- [Polar Star](#)
- [Status Light Genie](#)
- [Text Input](#)
- [Wind Compass](#)
- [XY Bar Graph](#)

### Alarm Light

The alarm light object is a text object used to draw the operator's attention to an alarm that may not have a numerical equivalent (for example, LOW FIRE CHECK).

When the alarm is inactive, the text is barely visible so as to not draw the operator's attention to the text, but still makes the operator aware that there is an alarm associated with a piece of equipment. If the alarm is triggered, the visibility will be forced to "on".

Property	Description
Name	Alarm Light
Graphical Representation	
Associated Composite Genie	AlarmLight.xml

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the

equipment at runtime.

- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Equipment Name	Enter the name of the equipment associated with the object. You can enter a maximum of 160 characters for this option.
Item Name	Item Name of the alarm for which to display the status. You can enter a maximum of 63 characters for this option.
Size	Size of the alarm light object image - small or large
Label	Text that describes the alarm condition. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator that indicates the highest priority alarm and its state for this Alarm Light equipment object.
Display Alarm Flag	Select this option to display an alarm flag. This option is available for selection only if the <b>Display Alarm Border</b> option is selected.
Display Status Indicator	Select the location to display a status indicator. Select None if you do not wish to see the status indicator. For more information, see Status Indicators.

## Dump Light Genie

**Genie Name:** dumplight\_32 / dumplight\_48

**Genie Library:** sa\_misc

**System Project:** SA\_Library

The Dump Light Genie allows you to add a stop/go indicator to a page or faceplate. The "proceed" indication is a green triangle in a white circle. The "stop" indication is a large black X.

The Genie is available in two sizes:

- dumplight\_32
- dumplight\_48

The Genie binds to the following equipment item:

- **Running** (see [Drive, Belt and Mining Objects](#))

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

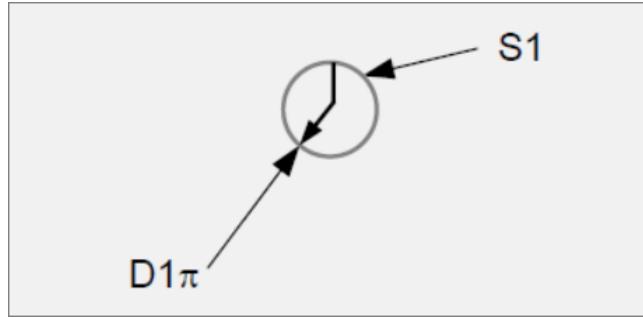
## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
Equipment	The name of the piece of equipment with which the Genie is associated.

## Handswitch Selector

The Handswitch Selector object can be used to show which of two pieces of equipment is selected "on" through the use of handswitching. The solid line connects the output indicator line from the selected controller to the active piece of equipment.

Property	Description
Name	Handswitch Selector
Graphical Representation	
Example Equipment Template	HandSwitchIndicator
Associated Composite Genie	Handswitch.xml
Equipment.Item that the Genie Expects	<b>SwitchPosition</b> (see <a href="#">Equipment Configuration Parameters</a> )

## Presentation Options

Option	Description
Equipment Name	Enter a name for the equipment associated with the handswitch selector. You can enter a maximum of 160 characters for this option.
Size	Size of the Handswitch Selector object image - small or large

## Numeric Input

The numerical input object can be used when you need to input data that might be measured remotely, or may be manually entered for use within a Genie. There is no alarming associated with this object.

Property	Description
Name	Numeric Display
Graphical Representation	
Associated Composite Genie	NumericInput.xml

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

Option	Description
Equipment Name	Enter a name for the equipment associated with numeric display. You can enter a maximum of 160 characters for this option.
Item Name	Enter the item name of the value to be displayed. You

Option	Description
	can enter a maximum of 63 characters for this option.
Display Label	Use this setting to display a label
Label	Enter the text that will display. This is displayed only if the <b>Display Label</b> option is selected. You can enter a maximum of 30 characters for this option.
Size	Size of the Numeric Display object image - small or large
Is Editable	Select this option make the displayed value editable.
Disabled When	Enter a Boolean condition for disabling the input functionality. This option is displayed only if the <b>Disabled When</b> option is selected.
Privilege	Privilege required to change the value at runtime. For more information, see <a href="#">Privileges</a> .
Area	Area access required to change the value at runtime. Area 0 includes every privilege a role may be assigned. For more information, see <a href="#">Areas</a> .
Separate Read/Write Items	Select whether changing the value writes to a different item. This option is displayed only if the <b>Disabled When</b> option is selected.
Write Item Name	Enter the item name that is written to when the value is changed. This option is displayed only if the <b>Separate Read/Write Items</b> option is selected. You can enter a maximum of 63 characters for this option.
Display Adjustment Controls	<p>Select to add controls to increase or decrease the value.</p> <p><b>Note:</b> In a multi-cluster system, any instance of Numeric Input with "Show Adjustment Controls" enabled needs to specify the cluster on the Equipment Name parameter.</p>
Use Absolute Step Value	<p>Select whether to increase and decrease by a fixed value or by a value defined by a runtime parameter on the equipment.</p> <p>If this is not selected, it will display a field in which you need to enter the name of the <a href="#">Define Equipment Runtime Parameters</a> that holds the step value.</p>

Option	Description
Step Value	Enter the value to increment/decrement value. This option is displayed only if the <b>Use Absolute Step Value</b> option is selected.
Step Equipment Parameter	Enter the name of the equipment runtime parameter that defines the increment/decrement value. This option is displayed only if the <b>Use Absolute Step Value</b> option is not selected.
Display Units	Select this option to display the engineering units for the equipment item.

## Polar Star

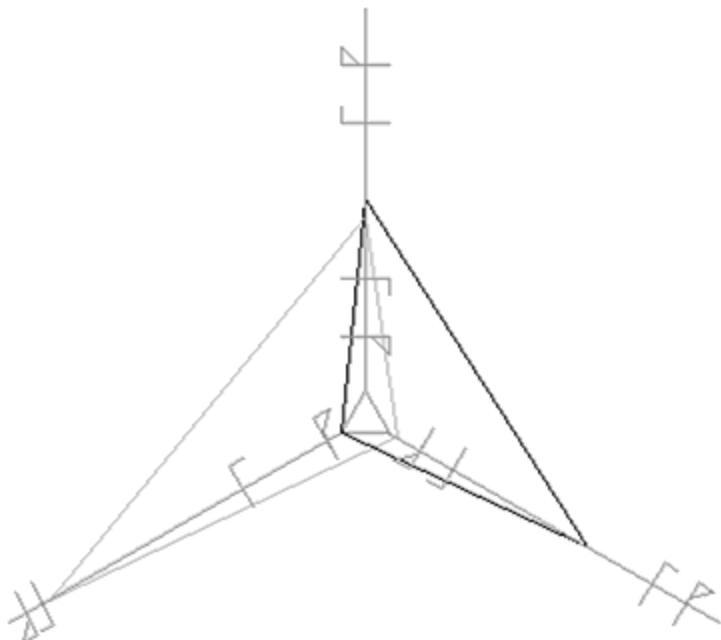
The Polar Star can be used for monitoring purposes. However, its use is limited to a few applications where information is for the most part static yet needs to be monitored. With Plant SCADA, you can create a 3-spoke Polar Star. Each spoke can be used to measure one parameter of an equipment and the 3 parameters are represented by a triangle. For example, you can compare three meters or three drives.

**Note:** The equipment that you select to compare needs to have a PV equipment.item.

Whenever one of the parameters records a deviation from its SP, the shape of the triangle will be disfigured and the operator will be able identify the abnormal parameter. When an alarm is triggered by an alarm flag, the spoke that triggered the alarm will be highlighted in the same color as the alarm border. When a spoke is selected, it displays the faceplate associated with it.

At runtime, the triangle is displayed in gray when the SP changes and in black when the PV changes (as shown below).

Meter1



Meter3

Meter2

Property	Description
Name	Polar Star
Graphical Representation	
Example Equipment Template	PolarStar
Associated Composite Genie	PolarStar.xml
Equipment.Item Names Associated with the Genie	<b>EqStatus</b> (see <a href="#">Equipment Configuration Parameters</a> ) <b>RunStatus</b> (see <a href="#">Equipment Running States</a> )

## Configuration Tasks

1. [Add Equipment Using Equipment Editor](#): Create the equipment for the required objects. Organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
2. [Equipment References](#): Create the required relationships (equipment references) between a piece of equipment to items of another piece of equipment. This allows you to refer to equipment.items outside of the equipment hierarchy.
3. [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
4. [Insert a Composite Genie](#): Create graphics pages with instances of Composite Genies that are associated with the equipment you have defined in your project.
5. [Create a New Faceplate](#): Add a faceplate for the equipment to allow operators to engage with the equipment at runtime.
6. Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Polar Star Equipment	Enter a name for the equipment associated with the Polar Star. You can enter a maximum of 160 characters for this option. <b>Note:</b> The Polar Star equipment needs to be manually configured to reference each equipment to be monitored.
Spoke #1/#2/#3 Equipment Name	Enter the name of the first/second/third equipment that you want to measure. You can enter a maximum of 160 characters for this option.
Spoke #1/#2/#3 Equipment Item Prefix	Prefix applied to the equipment item name on each spoke. This allows for displaying multiple values for a single piece of equipment.
Size	Size of the Polar Star object image - small or large
Display Label	Use this setting to display a label at the selected position. Select <b>None</b> if you do not want to display a label.
Label / Spoke#2 Label/Spoke #3 Label	Enter the text that will display for each spoke of the Polar Star. You can enter a maximum of 30 characters for this option.

Option	Description
Display Alarm Limits	Select this option to display alarm limits on the selected equipment.
Display Alarm Indicator	Select this option to display an alarm indicator which indicates the highest priority alarm and its state for this Polar Star's equipment. The Polar Star will also include the alarms for each equipment specified for each spoke if you create an equipment reference between the Polar Star and each equipment specified for the spokes.
Display Alarm Flag	Select this option to display an alarm flag. This option is available for selection only if the <b>Display Alarm Border</b> option is selected.
Display Status Indicator	Select the location to display a status indicator. Select <b>None</b> if you do not want to see the status indicator. For more information, see Status Indicators.
Display OOS	Select to display Out of Service indicator.

## Status Light Genie

**Genie Name:** statuslight\_20 / statuslight\_30

**Genie Library:** sa\_misc

**System Project:** SA\_Library

The Status Light Genie allows you to add a status indicator to a page or faceplate. It can be used to show basic status information (on/off/alarm) for equipment. It is based on the multiple equipment object, but with a limited range of states.

The Genie is available in two sizes:

- statuslight\_20
- statuslight\_30

This Genie binds to the following equipment item:

- Running

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
Equipment	The name of the piece of equipment with which the Genie is associated.

## Text Input

The text input-output object can be used for entry and/or display of text (non-numeric) data. There is no alarming associated with this object.

Property	Description
Name	Text Input
Graphical Representation	
Associated Composite Genie	TextInput.xml

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

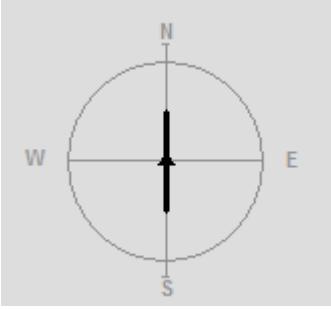
## Presentation Options

Option	Description
Equipment Name	Enter a name for the equipment associated with numeric display. You can enter a maximum of 160 characters for this option.
Item Name	Enter the item name of the value to be displayed. You can enter a maximum of 63 characters for this option.
Show Label	Use this setting to display a label
Label	Enter the text that will display. This is displayed only if the <b>Display Label</b> option is selected. You can enter a maximum of 30 characters for this option.
Size	Size of the Text Input object image - small or large
Width	Select the length of text display - Normal or Wide
Is Editable	Select this option make the displayed value editable.
Disabled When	Enter a Boolean condition for disabling the input functionality.
Privilege	Privilege required to change the value at runtime. For more information, see <a href="#">Privileges</a> .
Area	Area access required to change the value at runtime. Area 0 includes every privilege a role may be assigned. For more information, see <a href="#">Areas</a> .
Separate Read/Write Items	Select whether changing the value writes to a different item. This option is displayed only if the <b>Disabled When</b> option is selected.
Write Item Name	Enter the item name that is written to when the value is changed. This option is displayed only if the <b>Separate Read/Write Items</b> option is selected. You can enter a maximum of 63 characters for this option.

## Wind Compass

This object shows both current and average wind speed and direction. A black arrow points in the direction in which the wind is blowing, and a dark grey arrow points in the average direction in which the wind is blowing. The lengths of the arrows are proportional to the wind speeds, with a dot representing zero. The arrows are anchored in the middle and rotate around the middle of the compass. For light winds, the arrowhead is not drawn.

All the spokes have the same scale (operating range), which is configured at design time and can be changed during runtime. If the wind speed exceeds this range, the spokes change to full scale and the black full scale indicators are shown.

Property	Description
Name	Wind Compass
Graphical Representation	
Example Equipment Template	WindCompass
Associated Composite Genie	WindCompass.xml
Equipment.Item Names Associated with the Genie	<b>WindSpeed, WindSpeedAvg, WindSpeedThreshold</b> <b>PRHigh, PRLow</b> <b>WindPV, WindPVAvg</b> See <a href="#">Wind Compass Elements</a>
Equipment Parameters the Equipment Items Expect	<b>WindSpeedRange</b> (see <a href="#">Equipment Configuration Parameters</a> )

## Configuration Tasks

- [Add Equipment Using Equipment Editor](#): Create equipment instances for the required objects, and organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
- [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
- [Insert a Composite Genie](#): Create graphics pages with instances of [Composite Genies](#) that are associated with the equipment you have defined in your project.
- [Create a New Faceplate](#): Add a [Faceplates](#) for the equipment to allow operators to engage with the equipment at runtime.
- Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

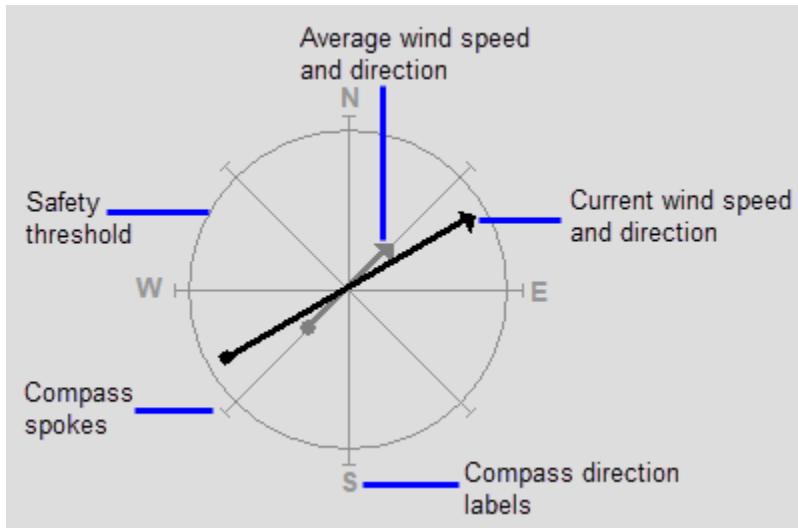
## Presentation Options

Option	Description
Equipment Name	Enter a name for the equipment associated with the wind compass.

Option	Description
	You can enter a maximum of 160 characters for this option.
Size	Size of the Wind Compass object image - small or large

## Wind Compass Elements

The image below shows the elements that make up the wind compass object.



## Wind Speed and Direction

Connects to: Equipment.WindSpeed, Equipment.WindSpeedAvg, Equipment.WindSpeedThreshold

Current and average speed and direction of the wind. A black arrow points in the direction in which the wind is blowing. A dark grey arrow points in the average direction in which the wind is blowing. The lengths of the arrows are proportional to the wind speed, with a dot representing zero. The radius of the compass measures the wind speed.

The WindSpeedThreshold represents the safety threshold circle. This item controls the size of the circle.

## Range Indicators

Connects to: Equipment.PRHigh, Equipment.PRLow,

Allows you to set a "normal" operating range ( "practical" range) with maximum (high) and minimum (low) values.

## Compass Direction

Connects to: Equipment.WindPV, Equipment.WindPVAvg

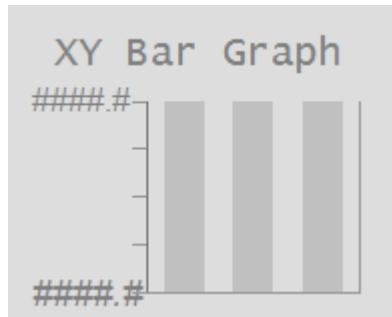
This is the angle between 0 and 360 degrees which denotes the direction in which the wind is blowing.

## Compass Spokes

All the spokes have the same scale (operating range). If the wind speed exceeds this range, the spokes change to full scale and black full scale indicators are shown.

## XY Bar Graph

The XY Bar Graph object can be used to compare sets of related PV values from similar equipment.

Property	Description
Name	XY Bar Graph
Graphical Representation	
Example Equipment Template	BarGraph
Associated Composite Genie	XY Bar Graph.XML
Equipment.Item Names Associated with the Genie	<b>PRHigh</b> , <b>PRLow</b> (see <a href="#">Meter Common Elements</a> ) <b>ORHigh</b> , <b>ORLow</b> , <b>ORDsp</b> (see <a href="#">Meter Common Elements</a> ) <b>EqStatus</b> (see <a href="#">Status Indicators</a> )
Equipment Parameters the Equipment Items Expect	<b>EqStatusFunc</b> <b>Range</b> See <a href="#">Equipment Configuration Parameters</a>

## Configuration Tasks

1. [Add Equipment Using Equipment Editor](#): Create the equipment for the required objects. Organize the equipment in a hierarchy. Objects can be referred to by their name when you add Composite Genies to a graphics page. You can add one or more instances of an equipment to a page.
2. [Equipment References](#): Create the required relationships (equipment references) between a piece of equipment to items of another piece of equipment. This allows you to refer to equipment.items outside of the equipment hierarchy.
3. [Configure PLC Limits with Equipment](#): This is an optional task that is required if you want to use PLC alarm limits instead of the standard analog alarm limits.
4. [Insert a Composite Genie](#): Create graphics pages with instances of Composite Genies that are associated

with the equipment you have defined in your project.

5. [Create a New Faceplate](#): Add a faceplate for the equipment to allow operators to engage with the equipment at runtime.
6. Write Cicode functions to initialize PR and OR values. Refer to the [Cicode Reference](#) documentation for more information.

## Presentation Options

The following presentation options are available for this object.

Option	Description
Main Equipment Name	<p>Enter a name for the equipment associated with the XY Bar Graph. You can enter a maximum of 160 characters for this option.</p> <p><b>Note:</b> The XY Bar Graph equipment needs to be <a href="#">Define Equipment References</a> each equipment to be monitored.</p>
Main Equipment Item Prefix	<p>Prefix applied to the main equipment item name. This allows for displaying multiple values for a single piece of equipment.</p>
Number of Bars	<p>Select the number of bars to be displayed in the graph. You can select a minimum of 1 and a maximum of 5 bars</p>
Bar #n Equipment Name	<p>Enter the name of each equipment to measure. The number of <b>Bar #n Equipment Name</b> options displayed depends on the <b>Number of Bars</b> selected for display. A maximum of 5 pieces of equipment can be selected for monitoring.</p> <p>You can enter a maximum of 160 characters for this option.</p>
Bar #n Equipment Item Prefix	<p>Prefix applied to the equipment item name on each bar of the XY Bar Graph. This allows for displaying multiple values for a single piece of equipment.</p>
Size	<p>Size of the XY Bar Graph object image - small or large</p>

Option	Description
Label	Enter the text that will display for the XY Bar Graph. You can enter a maximum of 30 characters for this option.
Display Alarm Indicator	Select this option to display an alarm indicator, which indicates the highest priority alarm and its state for this XY Bar Graph. The XY Bar Graph will also include alarms from each equipment included for comparison if you create an equipment reference between the XY bar Graph and each equipment specified for the bars.
Display Alarm Flag	Select this option to display an alarm flag. This option is available for selection only if the <b>Display Alarm Border</b> option is selected.
Display Status Indicator	Select the location to display a status indicator. Select <b>None</b> if you do not wish to see the status indicator. For more information, see <a href="#">Status Indicators</a> .

## Faceplates

A faceplate is a page that is associated with a piece of equipment. It typically contains a graphical representation of the equipment, measurements and controls that allow an operator to engage with the equipment at runtime. When the piece of equipment comes into context at runtime, the faceplate will be displayed. To support this, the default layout for a Situational Awareness project includes a Faceplate Zone in the Information Zone.

**Note:** When the equipment is out of service, all controls (buttons and numeric inputs) on the faceplate are disabled.

Plant SCADA provides a set of sample faceplates for meters, drives and valves. Note that the sample faceplates are fully configured and ready to use. However, you can [Create a New Faceplate](#) if it is required.

Faceplates are made up Genies and/or Composite Genies. User interaction with faceplates is achieved through items (tags) associated with the various components of a faceplate. The items associated with each faceplate are listed in the Associated Tags column of the faceplate table.

## See Also

- [Meter Faceplates](#)
- [Drive Faceplates](#)
- [Valve Faceplates](#)

## Create a New Faceplate

You can use one of the sample faceplates in the required resolution (HD1080 or 4K) to create your own faceplate. Sample faceplates are located in the SA\_Style\_1\_MultiRes project.

### To create a new faceplate, follow these steps:

1. Modify an existing sample faceplate.

Or:

Create a copy of an existing faceplate.

Or:

Create a new page in Graphics Builder using the "faceplate" template in the required resolution.

**Note:** The sample faceplates use a specific set of items to display controls at runtime. You can build your [Faceplates](#) using these items, which are listed in the topics that describe the faceplates for [Meter Faceplates](#), [Drive Faceplates](#) and [Valve Faceplates](#).

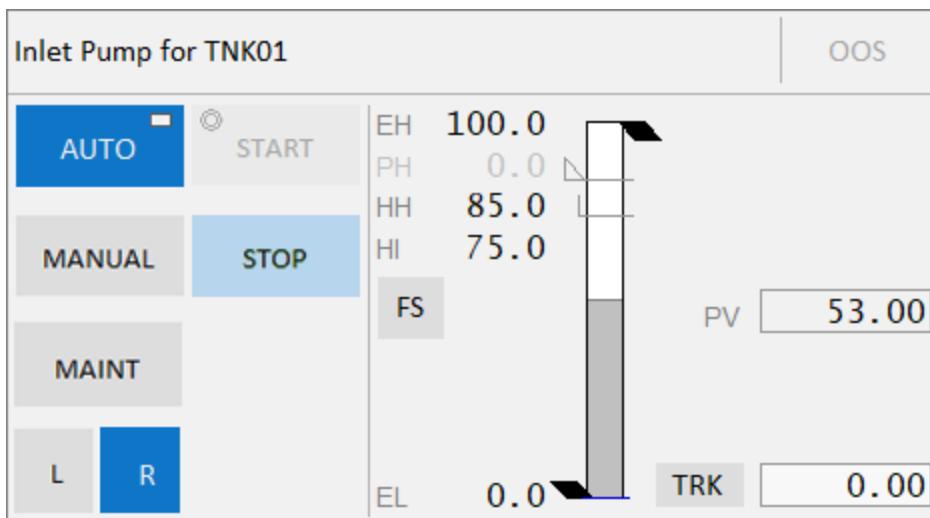
2. Save your page.
3. Go to Plant SCADA Studio.
4. In the **Visualization activity | Pages**, set the **Content Type** for the above page to "FP". Content of this type will automatically display in the [Faceplate Zone](#) of the workspace.
5. Click **Save**.
6. In Plant SCADA Studio, **System Model activity | Equipment**, locate the equipment for which you are creating the faceplate.
7. Add the name of the page you created earlier to the **Content** field for the equipment. It can include a comma separated list.
8. Click **Save**.

**Note:** User interaction with faceplates is achieved through command tags (items) associated with the various components of a faceplate. Items are listed in the individual faceplate topic. Buttons on a faceplate are designed to support this interaction. However, tags in Plant SCADA need to be mapped to the tags that you have configured on your PLCs. For example, tags for starting a drive (StartCmd) and displaying its status (CtrlMode) available in Plant SCADA. Similarly, tags for starting the drive and displaying its status would exist on your PLC. These two sets of tags need to be mapped to each other for the buttons on the drive faceplate to work. If the mapping is not created, buttons on the faceplates will not function. Refer to the example below to understand how this works.

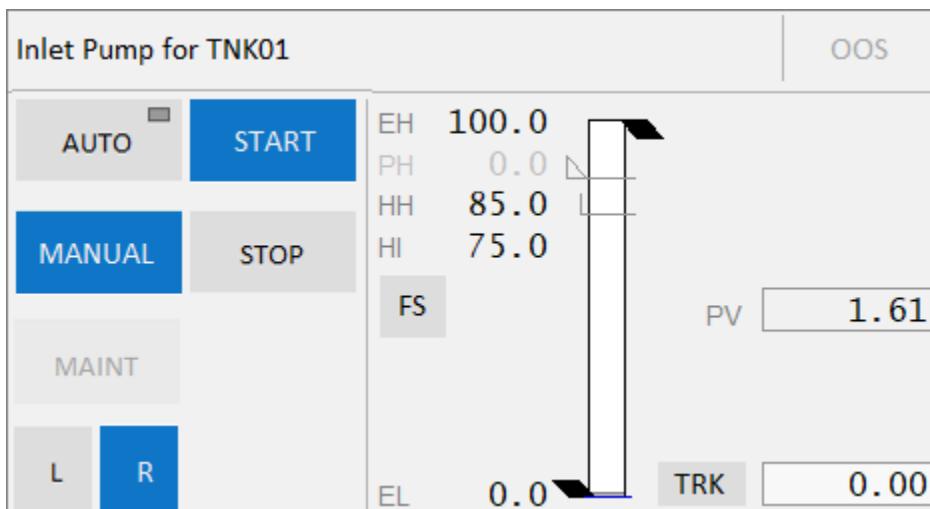
## Example

This example illustrates how tags are used to achieve interaction with faceplates.

The CtrlModeDef tag sets the default mode of operation for a piece of equipment. If you set this tag to 0 for a pump, the default mode of a pump will be Auto. That is indicated on the faceplate by a rectangle in the top-right corner of the **Auto** button on the faceplate as shown below.



The CtrlMode tag represents the current mode for a piece of equipment. If the CtrlModeDef (default) and CtrlMode (current) have different values, the current mode will be displayed in green. For example, the CtrlModeDef = 0 (Auto) and the CtrlMode = 1 (Manual), the **Manual** button will be displayed in green. Clicking the **Manual** button will set the ManCmd tag to 1.

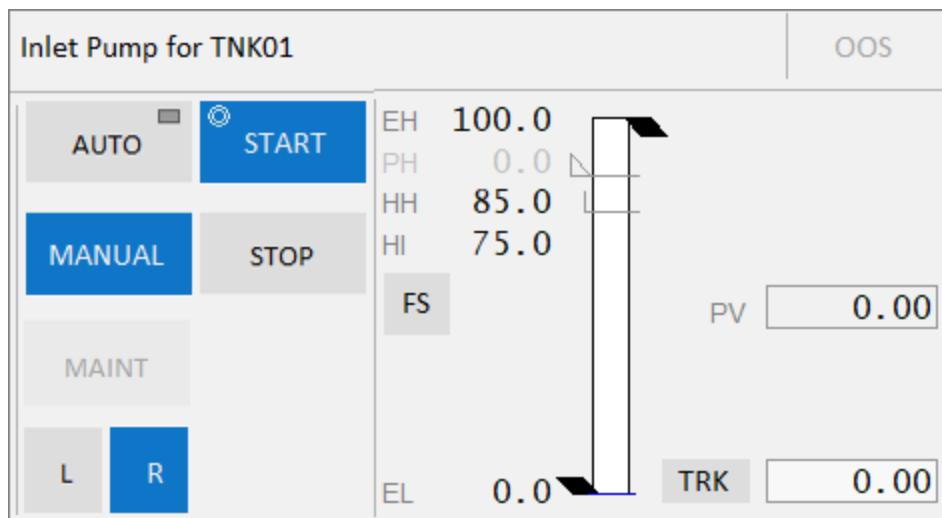


If the pump is not running, the **Start** button will also be displayed in green to allow you to manually start the pump. If the pump is running, the **Stop** button will be green allowing you to manually stop the pump. The **Auto** button will continue to display the rectangle to indicate that it is the default mode. Note that the **Start** and **Stop** buttons are available only when the mode is Manual.

---

**Note:** Clicking any button will set its <name>Cmd tag to 1. For example, clicking the **Manual** button will set ManCmd to 1.

---



When you click one of the buttons, a circle will appear in the top-left corner to indicate that a command (**Start** in the above image) has been sent to the PLC mapped to the corresponding tags. Once the Start command is completed, the circle will disappear as the PLC sets StartCmd to 0 (or StopCmd to 0 in case you clicked the **Stop** button). In addition, the PLC sets the value of the Running tag to 1 (Stopped tag if you clicked the **Stop** button). When you click Stop, the value of the Stopped tag is 1.

## See Also

- [Faceplates](#)
- [Faceplate Zone](#)
- [Meter Faceplates](#)
- [Drive Faceplates](#)
- [Valve Faceplates](#)

## Meter Faceplates

Meters can be associated with one the following faceplates:

- [Analog Controller](#)
- [Analog Indicator](#)

The following sample meter faceplates are available for use:

- FP\_AI\_HD1080
- FP\_AI\_UHD4K
- FP\_CTRL\_HD1080
- FP\_CTRL\_UHD4K

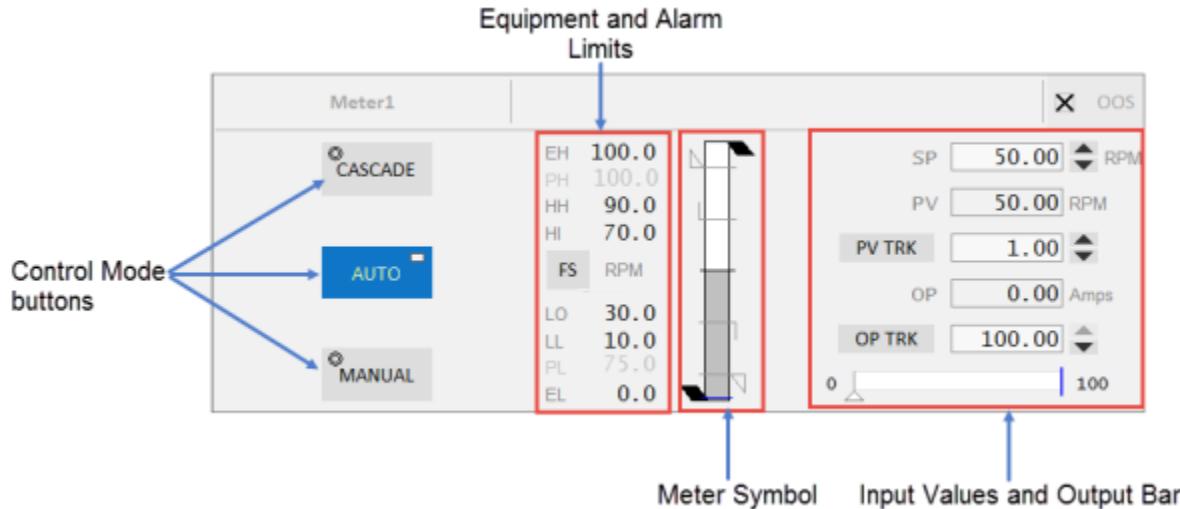
These faceplates are pages in the sa\_style\_1\_multires project, and are fully configured. You can use them directly in your projects.

## See Also

[Faceplates](#)  
[Drive Faceplates](#)  
[Valve Faceplates](#)

### Analog Controller

The Analog Controller faceplate can be associated with control meters such as a Flow meter.



The Analog Controller faceplate comprises the following components:

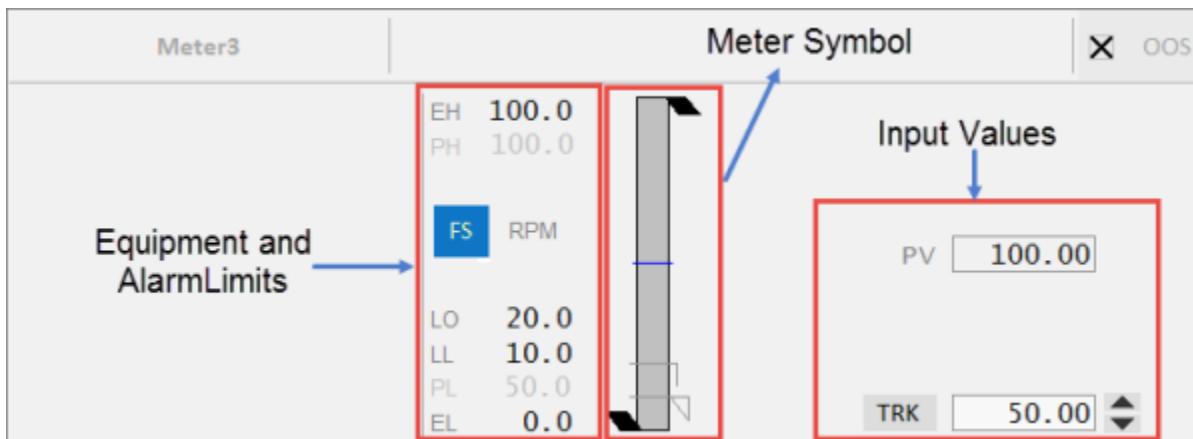
Component	Description	Associated Tags
Control Mode buttons	<p>Auto, Manual and Cascade buttons are displayed on this faceplate. The button is green when a mode is active. When the tag associated with the equipment is 0, the mode is Auto. When the tag is 1, the mode is Manual and when the tag is 2, the mode is Cascade.</p> <p>Clicking a mode button sends a command to the linked PLC.</p> <p>The Mode button displays two symbols:</p> <ul style="list-style-type: none"> <li>The circle in the top-left corner indicates that a command has been sent to the linked PLC. This indicator is not visible otherwise.</li> </ul>	<p><b>CtrlMode, CtrlDef</b> (see <a href="#">Control Meters - Common Elements</a>)</p> <p>AutoCmd, CasCmd, ManCmd</p>

Component	Description	Associated Tags
	<ul style="list-style-type: none"> <li>The rectangle in the top-right corner indicates that the current mode is the normal(default) mode of operation for the equipment.</li> </ul>	
Equipment Limits	<p>The following limits are displayed on the left-hand side of the meter symbol:</p> <ul style="list-style-type: none"> <li>Engineering High and Low values</li> <li>Practical Value High and Low values</li> </ul> <p>If the values lie outside the practical values, additional indicators are displayed. The color of the equipment limits changes as the meter values change.</p> <p>If the practical range is the same as the engineering range, the practical range will be hidden.</p>	<b>PRHigh, PRLow</b> (see <a href="#">Meter Common Elements</a> ) <b>PV</b> (see <a href="#">Meter Common Elements</a> )
Alarm Limits	<p>Displays the following alarm limits on the left-hand side of the meter symbol:</p> <ul style="list-style-type: none"> <li>HiHi</li> <li>Hi</li> <li>Lo</li> <li>LoLo</li> </ul> <p>These are the standard analog alarm limits. If these alarm limits are not configured, they will not be displayed.</p> <p><b>Note:</b> You can now configure custom digital alarm limits based on the PLC alarm limits.</p> <p>The <b>FS</b> (Full scale) button can be used to toggle the meter into full scale even when the PV is within the practical range. The units of</p>	<b>HHAlm, HAlm, LAlm, LLAlm</b> (see <a href="#">Meter Common Elements</a> ) <b>Note:</b> These tags are used if the UsePLCLimits parameter is "true" unless the PLCLimits parameter has been changed from the default. If the UsePLCLimits parameter is "false", values come from the alarm limit configuration of the PV tag.

Component	Description	Associated Tags
	measurement of the PV are displayed to the right of the <b>FS</b> button.	
Meter Symbol	Symbol for the currently selected meter.	
Faceplate Inputs	<p>Shown on the right-hand side of the meter symbol. Most of these are selected in the Composite Genie presentation options. These include:</p> <ul style="list-style-type: none"> <li>• Setpoint (SP) - this is editable only in Auto mode.</li> <li>• PV (comes from the equipment)</li> <li>• PV tracker</li> <li>• OP - this is editable only in Manual mode.</li> <li>• OP tracker</li> </ul> <p><b>Note:</b> You can configure the Setpoint increment by using an Equipment Runtime Parameter on your equipment. This is not supported on the default faceplates, but you can modify the faceplate to achieve this. Specify a name for the parameter (for example, SPStep) on the Composite Genie of the faceplate. Also, the <b>IsTag</b> property needs to be set to FALSE.</p>	<a href="#">SP</a> (see <a href="#">Control Meters - Common Elements</a> ) <a href="#">PV, PVTrack</a> (see <a href="#">Meter Common Elements</a> ) <a href="#">OP, OPTrack</a> (see <a href="#">Control Meters - Common Elements</a> )
Output Bar	Displayed below the input values. This shows the expected output along the bar and the actual output indicated by a triangle below the bar. The output indicator is visible only when the controller is in Auto or Cascade mode. It is not visible in Manual mode.	<a href="#">OP, OPTrack</a> (see <a href="#">Control Meters - Common Elements</a> ) <a href="#">FB</a> (see <a href="#">Control Meters - Common Elements</a> )

## Analog Indicator

The Analog Indicator faceplate can be associated with regular meters such as an Analyzer meter.



The Analog Indicator faceplate comprises the following components:

Component	Description	Associated Tags
Equipment Limits	<p>The following limits are displayed for this faceplate:</p> <ul style="list-style-type: none"> <li>Engineering High and Low values</li> <li>Practical Value High and Low values</li> </ul> <p>If the values lie outside the practical values, additional indicators are displayed. The color of the equipment limits changes as the meter values change.</p> <p>If the practical range is the same as the engineering range, the practical range will be hidden.</p>	<a href="#">PRHigh</a> , <a href="#">PRLow</a> <a href="#">PV</a>
Alarm Limits	<p>Displays the following alarm limits:</p> <ul style="list-style-type: none"> <li>HiHi</li> <li>Hi</li> <li>Lo</li> <li>LoLo</li> </ul> <p>These are the standard analog alarm limits. If these alarm limits are not configured, they will not be displayed.</p>	<a href="#">HHAlm</a> , <a href="#">HAlm</a> , <a href="#">LAlm</a> , <a href="#">LLAlm</a> <b>Note:</b> These tags are used if the UsePLCLimits parameter is "true" unless the PLCLimits parameter has been changed from the default. If UsePLCLimits Parameter is "false", values come from the alarm limit configuration of the PV tag.

Component	Description	Associated Tags
	<p><b>Note:</b> You can now configure custom digital alarm limits based on the PLC alarm limits.</p> <p>The <b>FS</b> (Full scale) button can be used to toggle the meter into full scale even when the PV is within the practical range. The units of measurement of the PV are displayed to the right of the <b>FS</b> button.</p>	
Faceplate Inputs	<p>Shown on the right hand side. Most of these are selected in the Composite Genie presentation options. These include:</p> <ul style="list-style-type: none"> <li>• PV (comes from the equipment)</li> <li>• PV tracker</li> </ul>	PV, PVTrack

## Drive Faceplates

Drive objects can call different faceplates depending on their specific properties as outlined below:

- [Single and Multiple DOL Drive](#)
- [Single and Multiple VSD Drive - Single Direction](#)
- [Single DOL Drive with Forward/Reverse Capability](#)
- [Single VSD Drive with Forward/Reverse Capability.](#)

The following sample drive faceplates are available for use:

- FP\_DOL\_HD1080
- FP\_DOL\_UHD4K
- FP\_VSD\_HD1080
- FP\_VSD\_UHD4K
- FP\_DOL\_FR\_HD1080
- FP\_DOL\_FR\_UHD4K
- FP\_VSD\_FR\_HD1080
- FP\_VSD\_FR\_UHD4K

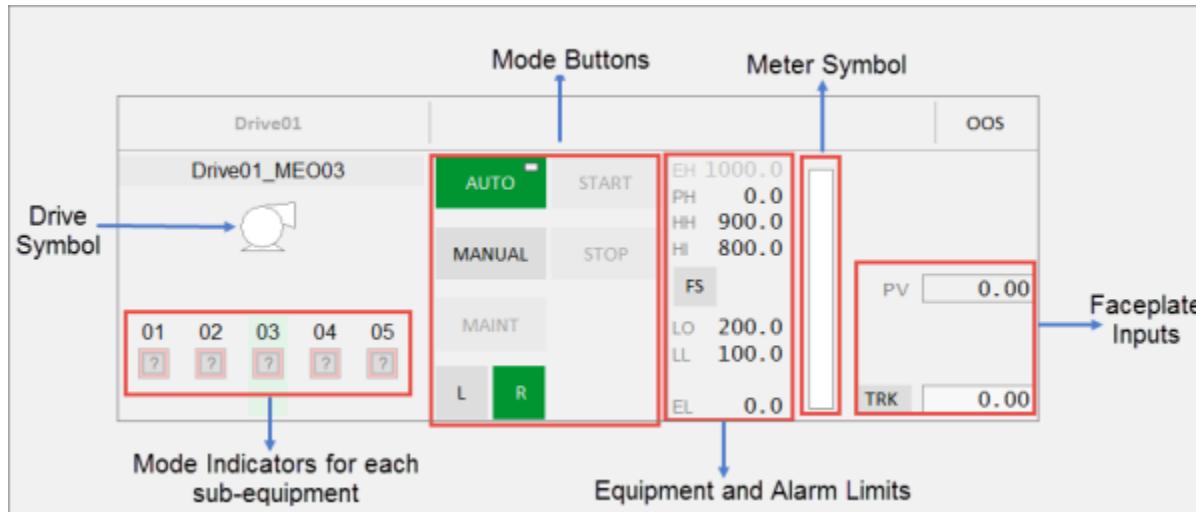
These faceplates are pages in the sa\_style\_1\_multires project, and are fully configured. You can use them directly in your projects.

## See Also

[Faceplates](#)  
[Meter Faceplates](#)  
[Valve Faceplates](#)

### Single and Multiple DOL Drive

This faceplate is used for all single and multiple Direct Online (DOL) drives.



The Single or Multiple DOL Drive faceplate comprises the following components:

Component	Description	Associated Tags
Drive Symbol	Symbol for the drive associated with the selected equipment. Clicking on the multiple equipment object selects that particular drive. The buttons, meter and readings are then populated for that drive.	
Mode Indicators for each related equipment	Status Indicator and control modes for each related equipment displayed below the drive symbol. The last two characters of the tagname for each drive are shown over each multiple equipment indicator to distinguish between the drives.  <b>Note:</b> If the drive does not have any MEOs, a Running Status indicator and control mode will be displayed for the drive itself in the	<a href="#">RunStatus</a> (see <a href="#">Equipment Running States</a> )  <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> )

Component	Description	Associated Tags
	<p>center position. The last two characters' label and the label above the drive are not shown. The mode indicator is not selectable.</p> <p>Indicators for up to 5 DOL drives can be displayed here.</p>	
<b>Mode Buttons</b>	<p>Mode buttons are green when a mode is active, and grey when inactive. When the CtrlMode tag associated with the equipment is 0, the mode is Auto. When the tag is 1, the mode is Manual. This tag applies to Auto and Manual buttons.</p> <p>The Mode button displays two symbols:</p> <ul style="list-style-type: none"> <li>• The circle in the top-left corner indicates that a command has been sent to the linked PLC. This indicator is not visible otherwise.</li> <li>• The rectangle in the top-right corner indicates that the current mode is the normal mode of operation for the equipment.</li> </ul> <p>The following mode buttons are displayed:</p>	
	<ul style="list-style-type: none"> <li>• Auto: Automatic mode</li> </ul>	<b>CtrlMode</b> , <b>CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd</b>
	<ul style="list-style-type: none"> <li>• Manual: Manual mode</li> </ul>	<b>CtrlMode</b> , <b>CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>ManCmd</b>
	<ul style="list-style-type: none"> <li>• Maint: Maintenance mode - This is a toggle button that toggles the "MaintCmd" item from 0 to 1 or vice versa. It is green (active) when the</li> </ul>	<b>Maint</b> , <b>MaintCmd</b>

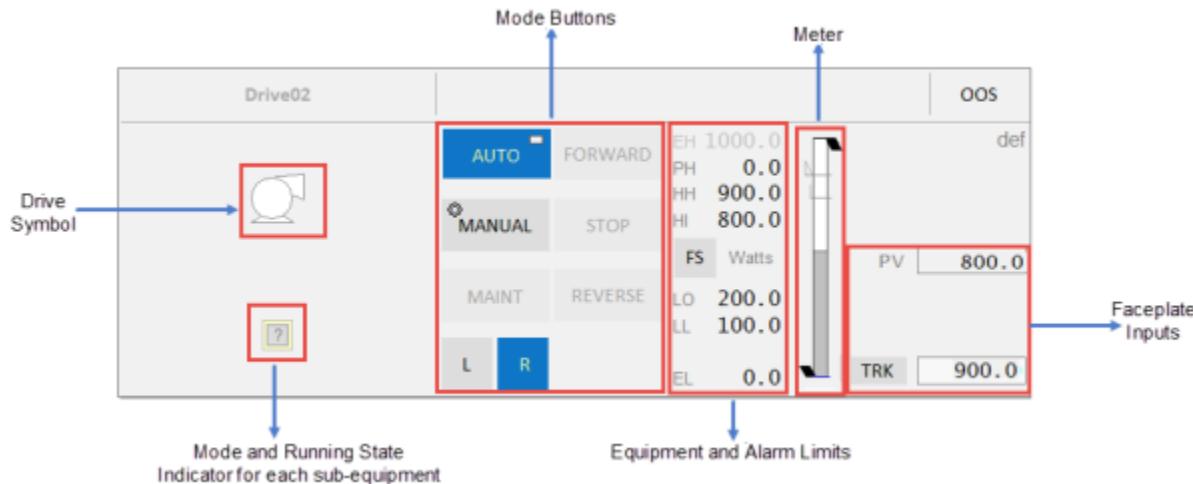
Component	Description	Associated Tags
	<p>"Maint" item is 1 and disabled when the drive is running. During this time, all non-critical interlocks are bypassed. The circle or rectangle symbols are not displayed on this button.</p>	
	<ul style="list-style-type: none"> <li>Start: This is a toggle button to manually start the drive. This button is available only when the drive is in Manual (MAN) mode. This button does not have a "Normal" mode, and therefore does not display the rectangle symbol.</li> </ul>	<a href="#">Running</a> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>StartCmd</b>
	<ul style="list-style-type: none"> <li>Stop: This is a toggle button to manually stop the drive. This button is available only when the drive is in Manual (MAN) mode. It is disabled when the drive is in Maintenance mode. It is disabled when the drive is in Maintenance mode. This button does not have a "Normal" mode, and therefore does not display the rectangle symbol.</li> </ul>	<a href="#">Running</a> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>StopCmd</b>
	<ul style="list-style-type: none"> <li>Local: When active, this button indicates that the drive is in local control.</li> </ul>	
	<ul style="list-style-type: none"> <li>Remote: When active, this button indicates that the drive is in remote control.</li> </ul>	
Equipment Limits	The following limits are displayed on the right-hand side:	<b>PRHigh, PRLow</b> (see <a href="#">Meter Common Elements</a> ) <b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a> )

Component	Description	Associated Tags
	<ul style="list-style-type: none"> <li>Engineering High and Low values</li> <li>Practical Value High and Low values</li> </ul> <p>If the values lie outside the practical values, additional indicators are displayed. The color of the equipment limits changes as the meter values change.</p> <p>If the practical range is the same as the engineering range, the practical range will be hidden.</p>	<a href="#">Objects</a> )
Alarm Limits	<p>Displays the following alarm limits on the right-hand side:</p> <ul style="list-style-type: none"> <li>HiHi</li> <li>Hi</li> <li>Lo</li> <li>LoLo</li> </ul> <p>These are the standard analog alarm limits. If these alarm limits are not configured, they will not be displayed.</p> <p><b>Note:</b> You can now configure custom digital alarm limits based on the PLC alarm limits.</p> <p>The <b>FS</b> (Full scale) button can be used to toggle the meter into full scale even when the PV is within the practical range. The units of measurement of the PV are displayed to the right of the <b>FS</b> button.</p>	<b>HHAlm, HAlm, LAlm, LLAlm</b> (see <a href="#">Meter Common Elements</a> ) <b>Note:</b> These tags are used if the UsePLCLimits parameter is "true" unless the PLCLimits parameter has been changed from the default. If UsePLCLimits parameter is "false", values come from the alarm limit configuration of the PV tag.
Meter	Meter displayed for the PV of the equipment.	
Faceplate Inputs	<p>Shown on the right-hand side of the meter. These include:</p> <ul style="list-style-type: none"> <li>PV (comes from the</li> </ul>	<ul style="list-style-type: none"> <li><b>PV, PVTrack</b> (see <a href="#">Meter Common Elements</a>)</li> </ul>

Component	Description	Associated Tags
	equipment) <ul style="list-style-type: none"> <li>PV tracker</li> </ul>	

### Single DOL Drive with Forward/Reverse Capability

This faceplate is used for a customized Direct Online (DOL) drive that has different commands for forward and reverse.



Component	Description	Associated Tags
Drive Symbol	Symbol for the drive associated with the selected equipment. Clicking on the multiple equipment object selects that particular drive. The buttons, meter and readings are then populated for that drive.	
Mode Indicators for each related equipment	Running State Indicator and control modes for each related equipment displayed below the drive symbol. The last two characters of the tagname for each drive are shown over each multiple equipment indicator to distinguish between the drives. <b>Note:</b> If the drive does not have any MEOs, a Running Status indicator and control mode will be displayed for the drive itself in the	<a href="#">RunStatus</a> (see <a href="#">Equipment Running States</a> ) <a href="#">CtrlMode</a> , <a href="#">CtrlModeDef</a> (see <a href="#">Drive, Belt and Mining Objects</a> )

Component	Description	Associated Tags
	<p>center position. The last two characters' label and the label above the drive are not shown. The mode indicator is not selectable.</p> <p>Indicators for up to 5 DOL drives can be displayed here.</p>	
<b>Mode Buttons</b>	<p>Mode buttons are green when a mode is active, and grey when inactive. When the CtrlMode tag associated with the equipment is 0, the mode is Auto. When the tag is 1, the mode is Manual. This tag applies to Auto and Manual buttons.</p> <p>The Mode button displays two symbols:</p> <ul style="list-style-type: none"> <li>• The circle in the top-left corner indicates that a command has been sent to the linked PLC. This indicator is not visible otherwise.</li> <li>• The rectangle in the top-right corner indicates that the current mode is the normal mode of operation for the equipment.</li> </ul> <p>The following mode buttons are displayed:</p>	
	<ul style="list-style-type: none"> <li>• Auto: Automatic mode</li> </ul>	<b>CtrlMode</b> , <b>CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd</b>
	<ul style="list-style-type: none"> <li>• Manual: Manual mode</li> </ul>	<b>CtrlMode</b> , <b>CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>ManCmd</b>
	<ul style="list-style-type: none"> <li>• Maint: Maintenance mode - This is a toggle button that toggles the "MaintCmd" item from 0 to 1 or vice versa. It is green (active) when the</li> </ul>	<b>Maint</b> , <b>MaintCmd</b>

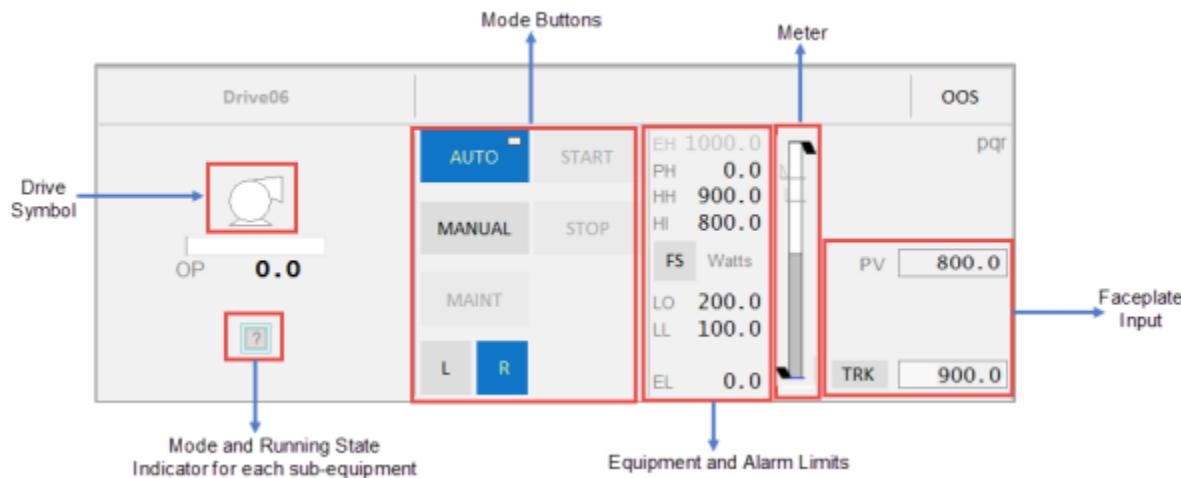
Component	Description	Associated Tags
	<p>"Maint" item is 1 and disabled when the drive is running. During this time, all non-critical interlocks are bypassed. The circle or rectangle symbols are not displayed on this button.</p>	
	<ul style="list-style-type: none"> <li>Stop: This is a toggle button to manually stop the drive. This button is available only when the drive is in Manual (MAN) mode. It is disabled when the drive is in Maintenance mode. It is disabled when the drive is in Maintenance mode. This button does not have a "Normal" mode, and therefore does not display the rectangle symbol.</li> </ul>	<b>Stopped</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>StopCmd</b>
	<ul style="list-style-type: none"> <li>Forward: This is a toggle button to manually start the drive in the forward direction. This button is available only when the drive is in Manual (MAN) mode. This button does not have a "Normal" mode, and therefore does not display the rectangle symbol.</li> </ul>	<b>Fwd, FwdCmd</b>
	<ul style="list-style-type: none"> <li>Reverse: This is a toggle button to manually start the drive in the reverse direction. This button is available only when the drive is in Manual (MAN) mode. This button does not have a "Normal" mode, and therefore does not display the rectangle symbol.</li> </ul>	<b>Rev, RevCmd</b>
	<ul style="list-style-type: none"> <li>Local: When active, this button</li> </ul>	

Component	Description	Associated Tags
	indicates that the drive is in local control.	
	<ul style="list-style-type: none"> <li>• Remote: When active, this button indicates that the drive is in remote control.</li> </ul>	
Equipment Limits	<p>The following limits are displayed on the right-hand side:</p> <ul style="list-style-type: none"> <li>• Engineering High and Low values</li> <li>• Practical Value High and Low values</li> </ul> <p>If the values lie outside the practical values, additional indicators are displayed. The color of the equipment limits changes as the meter values change.</p> <p>If the practical range is the same as the engineering range, the practical range will be hidden.</p>	<b>PRHigh</b> , <b>PRLow</b> (see <a href="#">Meter Common Elements</a> ) <b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a> )
Alarm Limits	<p>Displays the following alarm limits on the right-hand side:</p> <ul style="list-style-type: none"> <li>• HiHi</li> <li>• Hi</li> <li>• Lo</li> <li>• LoLo</li> </ul> <p>These are the standard analog alarm limits. If these alarm limits are not configured, they will not be displayed.</p> <p><b>Note:</b> You can now configure custom digital alarm limits based on the PLC alarm limits.</p> <p>The <b>FS</b> (Full scale) button can be used to toggle the meter into full scale even when the PV is within the practical range. The units of</p>	<b>HHAlm</b> , <b>HAlm</b> , <b>LAlm</b> , <b>LLAlm</b> (see <a href="#">Meter Common Elements</a> ) <b>Note:</b> These tags are used if the UsePLCLimits parameter is "true" unless the PLCLimits parameter has been changed from the default. If UsePLCLimits parameter is "false", values come from the alarm limit configuration of the PV tag.

Component	Description	Associated Tags
	measurement of the PV are displayed to the right of the <b>FS</b> button.	
Meter	Meter displayed for the PV of the equipment.	
Faceplate Inputs	Shown on the right-hand side of the meter. These include: <ul style="list-style-type: none"> <li>PV (comes from the equipment)</li> <li>PV tracker</li> </ul>	<b>PV, PVTrack</b> (see <a href="#">Drive,Belt and Mining Objects</a> )

### Single and Multiple VSD Drive - Single Direction

This faceplate is used for single and multiple Variable Speed Drive (VSD) control. It is used for variable drives and groups of variable speed drives. This faceplate functions similar to the multiple DOL drive faceplate. The Equipment Running State Indicators are positioned like the multiple DOL drive faceplate based upon the number of drives.



Component	Description	Associated Tags
Drive Symbol	Symbol for the drive associated with the selected equipment. Clicking on the multiple equipment object selects that particular drive. The buttons, meter and readings are then populated for that drive.	
Output Bar	This shows the expected output	

Component	Description	Associated Tags
	along the bar and the actual output indicated by a triangle below the bar.	
Mode Indicators for each related equipment	<p>Running State Indicator and control modes for each related equipment displayed below the drive symbol. The last two characters of the tagname for each drive are shown over each multiple equipment indicator to distinguish between the drives.</p> <p><b>Note:</b> If the drive does not have any MEOs, a Running Status indicator and control mode will be displayed for the drive itself in the center position. The last two characters' label and the label above the drive are not shown. The mode indicator is not selectable.</p> <p>Indicators for up to 5 VS drives can be displayed here.</p>	<b>RunStatus</b> (see <a href="#">Equipment Running States</a> ) <b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> )
Mode Buttons	<p>Mode buttons are green when a mode is active, and grey when inactive. When the CtrlMode tag associated with the equipment is 0, the mode is Auto. When the tag is 1, the mode is Manual. This tag applies to Auto and Manual buttons.</p> <p>The Mode button displays two symbols:</p> <ul style="list-style-type: none"> <li>The circle in the top-left corner indicates that a command has been sent to the linked PLC. This indicator is not visible otherwise.</li> <li>The rectangle in the top-right corner indicates that the current mode is the normal mode of operation for the equipment.</li> </ul> <p>The following mode buttons are</p>	

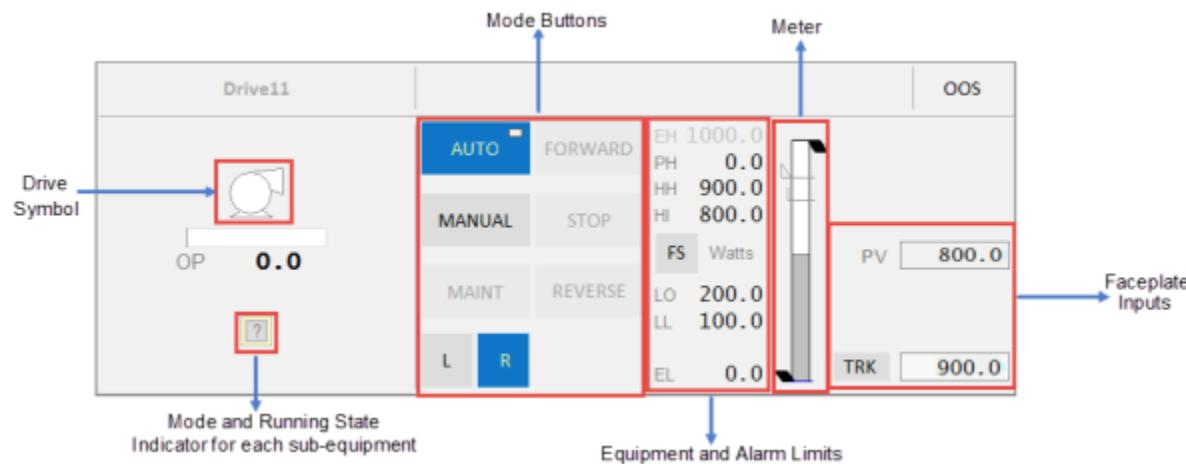
Component	Description	Associated Tags
	displayed:	
	<ul style="list-style-type: none"> <li>• Auto: Automatic mode</li> </ul>	<b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>AutoCmd</b>
	<ul style="list-style-type: none"> <li>• Manual: Manual mode</li> </ul>	<b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>ManCmd</b>
	<ul style="list-style-type: none"> <li>• Maint: Maintenance mode - This is a toggle button that toggles the "MaintCmd" item from 0 to 1 or vice versa. It is green (active) when the "Maint" item is 1 and disabled when the drive is running. During this time, all non-critical interlocks are bypassed. The circle or rectangle symbols are not displayed on this button.</li> </ul>	<b>Maint, MaintCmd</b>
	<ul style="list-style-type: none"> <li>• Start: This is a toggle button to manually start the drive. This button is available only when the drive is in Manual (MAN) mode. This button does not have a "Normal" mode, and therefore does not display the rectangle symbol.</li> </ul>	<b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>StartCmd</b>
	<ul style="list-style-type: none"> <li>• Stop: This is a toggle button to manually stop the drive. This button is available only when the drive is in Manual (MAN) mode. It is disabled when the drive is in Maintenance mode. It is disabled when the drive is in Maintenance mode. This button does not have a "Normal" mode, and therefore does not display the rectangle symbol.</li> </ul>	<b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>StopCmd</b>

Component	Description	Associated Tags
	<ul style="list-style-type: none"> <li>• Local: When active, this button indicates that the drive is in local control.</li> </ul>	
	<ul style="list-style-type: none"> <li>• Remote: When active, this button indicates that the drive is in remote control.</li> </ul>	
Equipment Limits	<p>The following limits are displayed on the right-hand side:</p> <ul style="list-style-type: none"> <li>• Engineering High and Low values</li> <li>• Practical Value High and Low values</li> </ul> <p>If the values lie outside the practical values, additional indicators are displayed. The color of the equipment limits changes as the meter values change.</p> <p>If the practical range is the same as the engineering range, the practical range will be hidden.</p>	<b>PRHigh</b> , <b>PRLow</b> (see <a href="#">Meter Common Elements</a> ) <b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a> )
Alarm Limits	<p>Displays the following alarm limits on the right-hand side:</p> <ul style="list-style-type: none"> <li>• HiHi</li> <li>• Hi</li> <li>• Lo</li> <li>• LoLo</li> </ul> <p>These are the standard analog alarm limits. If these alarm limits are not configured, they will not be displayed.</p> <p><b>Note:</b> You can now configure custom digital alarm limits based on the PLC alarm limits.</p> <p>The <b>FS</b> (Full scale) button can be used to toggle the meter into full</p>	<b>HHAlm</b> , <b>HAlm</b> , <b>LAlm</b> , <b>LLAlm</b> (see <a href="#">Meter Common Elements</a> ) <b>Note:</b> These tags are used if the UsePLCLimits parameter is "true" unless the PLCLimits parameter has been changed from the default. If UsePLCLimits parameter is "false", values come from the alarm limit configuration of the PV tag.

Component	Description	Associated Tags
	scale even when the PV is within the practical range. The units of measurement of the PV are displayed to the right of the FS button.	
Meter	Meter displayed for the PV of the equipment.	

### Single VSD Drive with Forward/Reverse Capability

This faceplate is used for a customized VSD drive that has different commands for forward and reverse. This faceplate is similar to the VSD drive faceplate.



Component	Description	Associated Tags
Drive Symbol	Symbol for the drive associated with the selected equipment. Clicking on the multiple equipment object selects that particular drive. The buttons, meter and readings are then populated for that drive.	
Output Bar	This shows the expected output along the bar and the actual output indicated by a triangle below the bar.	
Mode Indicators for each related equipment	Running State Indicator and control modes for each related equipment displayed below the drive symbol. The last two characters of the	<a href="#">RunStatus</a> (see <a href="#">Equipment Running States</a> ) <a href="#">CtrlMode</a> , <a href="#">CtrlModeDef</a> (see <a href="#">Drive,Belt and Mining Objects</a> )

Component	Description	Associated Tags
	<p>tagname for each drive are shown over each multiple equipment indicator to distinguish between the drives.</p> <p><b>Note:</b> If the drive does not have any MEOs, a Running Status indicator and control mode will be displayed for the drive itself in the center position. The last two characters' label and the label above the drive are not shown. The mode indicator is not selectable.</p> <p>Indicators for up to 5 VS drives can be displayed here.</p>	
Mode Buttons	<p>Mode buttons are green when a mode is active, and grey when inactive. When the CtrlMode tag associated with the equipment is 0, the mode is Auto. When the tag is 1, the mode is Manual. This tag applies to Auto and Manual buttons.</p> <p>The Mode button displays two symbols:</p> <ul style="list-style-type: none"> <li>• The circle in the top-left corner indicates that a command has been sent to the linked PLC. This indicator is not visible otherwise.</li> <li>• The rectangle in the top-right corner indicates that the current mode is the normal mode of operation for the equipment.</li> </ul> <p>The following mode buttons are displayed:</p>	
	<ul style="list-style-type: none"> <li>• Auto: Automatic mode</li> </ul>	<b>CtrlMode, CtrlModeDef</b> (see <a href="#">Drive, Belt and Mining Objects</a> ) <b>AutoCmd</b>

Component	Description	Associated Tags
	<ul style="list-style-type: none"> <li>Manual: Manual mode</li> </ul>	<b>CtrlMode</b> , <b>CtrlModeDef</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>ManCmd</b>
	<ul style="list-style-type: none"> <li>Maint: Maintenance mode - This is a toggle button that toggles the "MaintCmd" item from 0 to 1 or vice versa. It is green (active) when the "Maint" item is 1 and disabled when the drive is running. During this time, all non-critical interlocks are bypassed. The circle or rectangle symbols are not displayed on this button.</li> </ul>	<b>Maint</b> , <b>MaintCmd</b>
	<ul style="list-style-type: none"> <li>Start: This is a toggle button to manually start the drive. This button is available only when the drive is in Manual (MAN) mode. This button does not have a "Normal" mode, and therefore does not display the rectangle symbol.</li> </ul>	<b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>StartCmd</b>
	<ul style="list-style-type: none"> <li>Stop: This is a toggle button to manually stop the drive. This button is available only when the drive is in Manual (MAN) mode. It is disabled when the drive is in Maintenance mode. It is disabled when the drive is in Maintenance mode. This button does not have a "Normal" mode, and therefore does not display the rectangle symbol.</li> </ul>	<b>Running</b> (see <a href="#">Drive,Belt and Mining Objects</a> ) <b>StopCmd</b>
	<ul style="list-style-type: none"> <li>Forward: This is a toggle button to manually start the drive in the forward direction. This button is available only when</li> </ul>	<b>Fwd</b> , <b>FwdCmd</b>

Component	Description	Associated Tags
	<p>the drive is in Manual (MAN) mode. This button does not have a "Normal" mode, and therefore does not display the rectangle symbol.</p>	
	<ul style="list-style-type: none"> <li>Reverse: This is a toggle button to manually start the drive in the reverse direction. This button is available only when the drive is in Manual (MAN) mode. This button does not have a "Normal" mode, and therefore does not display the rectangle symbol.</li> </ul>	<b>Rev, RevCmd</b>
	<ul style="list-style-type: none"> <li>Local: When active, this button indicates that the drive is in local control.</li> </ul>	
	<ul style="list-style-type: none"> <li>Remote: When active, this button indicates that the drive is in remote control.</li> </ul>	
Equipment Limits	<p>The following limits are displayed on the right-hand side:</p> <ul style="list-style-type: none"> <li>Engineering High and Low values</li> <li>Practical Value High and Low values</li> </ul> <p>If the values lie outside the practical values, additional indicators are displayed. The color of the equipment limits changes as the meter values change.</p> <p>If the practical range is the same as the engineering range, the practical range will be hidden.</p>	<b>PRHigh, PRLow</b> (see <a href="#">Meter Common Elements</a> ) <b>PV</b> (see <a href="#">Drive,Belt and Mining Objects</a> )
Alarm Limits	Displays the following alarm limits	<b>HHAlm, HAlm, LAlm, LLAlm</b> (see <a href="#">Alarms</a> )

Component	Description	Associated Tags
	<p>on the right-hand side:</p> <ul style="list-style-type: none"> <li>• HiHi</li> <li>• Hi</li> <li>• Lo</li> <li>• LoLo</li> </ul> <p>These are the standard analog alarm limits. If these alarm limits are not configured, they will not be displayed.</p> <p><b>Note:</b> You can now configure custom digital alarm limits based on the PLC alarm limits.</p> <p>The <b>FS</b> (Full scale) button can be used to toggle the meter into full scale even when the PV is within the practical range. The units of measurement of the PV are displayed to the right of the <b>FS</b> button.</p>	<a href="#">Meter Common Elements</a> ) <b>Note:</b> These tags are used if the UsePLCLimits parameter is "true" unless the PLCLimits parameter has been changed from the default. If UsePLCLimits parameter is "false", values come from the alarm limit configuration of the PV tag.
Meter	Meter displayed for the PV of the equipment.	
Faceplate Inputs	<p>Shown on the right-hand side of the meter. These include:</p> <ul style="list-style-type: none"> <li>• PV (comes from the equipment)</li> <li>• PV tracker</li> </ul>	<a href="#">PV</a> , <a href="#">PVTrack</a> (see <a href="#">Meter Common Elements</a> )

## Valve Faceplates

Valves can be associated with one of the following faceplates:

- [Simple Valve Faceplate](#)
- [Complex Valve](#)

The following sample valve faceplates are available for use:

- FP\_COMPLX\_VLV\_HD1080
- FP\_COMPLX\_VLV\_UHD4K
- FP\_VLV\_HD1080

- FP\_VLV\_UHD4K

These faceplates are pages in the sa\_style\_1\_multires project, and are fully configured. You can use them directly in your projects.

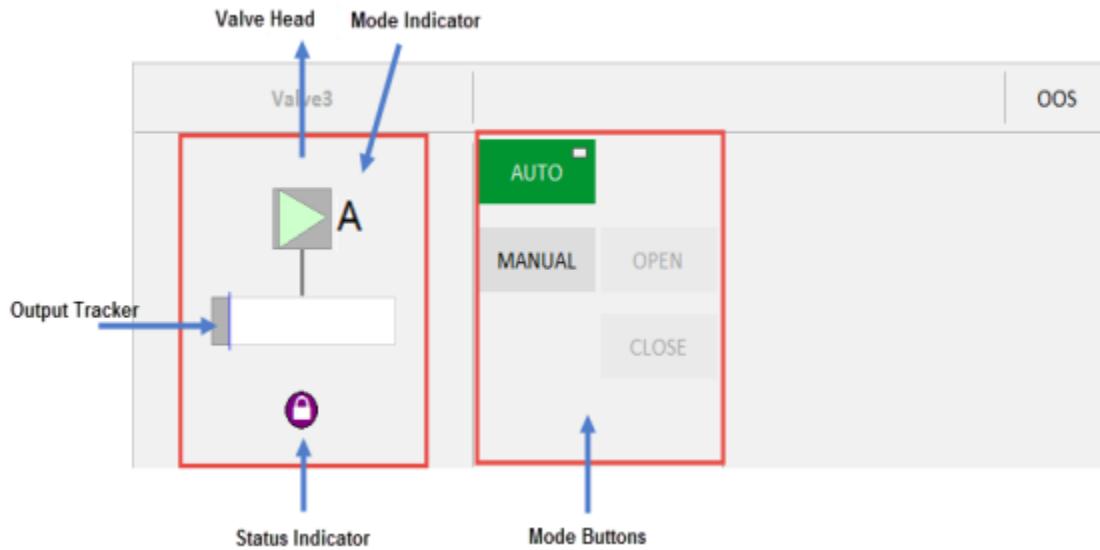
## See Also

[Faceplates](#)  
[Drive Faceplates](#)  
[Meter Faceplates](#)

### Simple Valve Faceplate

The Simple Valve faceplate example can be associated with a Block Valve as it has the square valve head.

**Note:** To use for other valve types, create a copy of the faceplate and update the valve type in the composite genie.



The Simple Valve faceplate comprises the following components:

Component	Description	Associated Tags
Valve Head	Valve Symbol associated with the selected equipment.	
Mode buttons	Mode buttons are green when a mode is active, and grey when inactive. When the CtrlMode tag associated with the equipment is 0, the mode is Auto. When the tag is 1, the mode is Manual. This tag	<b>CtrlMode, CtrlModeDef (see Valves)</b> <b>AutoCmd</b> <b>ManCmd</b>

Component	Description	Associated Tags
	<p>applies to Auto and Manual buttons.</p> <p>The Mode button displays two symbols:</p> <ul style="list-style-type: none"> <li>• The circle in the top-left corner indicates that a command has been sent to the linked PLC. This indicator is not visible otherwise.</li> <li>• The rectangle in the top-right corner indicates that the current mode is the normal mode of operation for the equipment. Displayed on the Auto/Manual buttons, the rectangle depends on 'CtrlModeDef'</li> </ul> <p>The following Mode Buttons are available:</p>	
	<ul style="list-style-type: none"> <li>• Auto: Automatic mode</li> </ul>	<b>CtrlMode, CtrlModeDef</b> (see <a href="#">Valves</a> ) <b>AutoCmd</b>
	<ul style="list-style-type: none"> <li>• Manual: Manual mode</li> </ul>	<b>CtrlMode, CtrlModeDef</b> (see <a href="#">Valves</a> ) <b>ManCmd</b>
	<ul style="list-style-type: none"> <li>• Open: Open mode - This is a toggle button that toggles the "OpenCmd". The Open button is active (green) when 'OpenCmd' tag is 1. The button is disabled when the valve is in AUTO mode or the valve has an interlock/shutdown condition preventing the valve from being operated.</li> </ul>	<b>OpenCmd</b>
	<ul style="list-style-type: none"> <li>• Close: Close mode - This is a toggle button that toggles the "CloseCmd". The Close button</li> </ul>	<b>CloseCmd</b>

Component	Description	Associated Tags
	is active (green) when 'Closedmd' tag is 1. The button is disabled when the valve is in AUTO mode or the valve has an interlock/shutdown condition preventing the valve from being closed.	
Mode Indicators	<p>Local, Manual, Auto and Cascade Mode.</p> <p>Local - Local Mode – field or MCC control only.</p> <p>Manual - Manual Mode - start/stop or valve position set by control room operator</p> <p>Auto Mode - start/stop and valve position set by control system, control operator sets Setpoint target.</p> <p>Cascade mode – control system sets setpoint target from another loop/logic.</p>	<b>CtrlMode, CtrlModeDef</b> (see <a href="#">Valves</a> )
Equipment Status Indicator	Displays if there is an abnormal status condition e.g. bad comms (stop IO server).	<b>EqStatus</b> (see <a href="#">Status Indicators</a> )
Output Tracker	Blue line on output bar. The position of the output tracker is based on the 'OPTrack' tag and visible when 'TrackDsp' tag is 1.	<b>OPTrack</b> (see <a href="#">Valves</a> ) TrackDsp

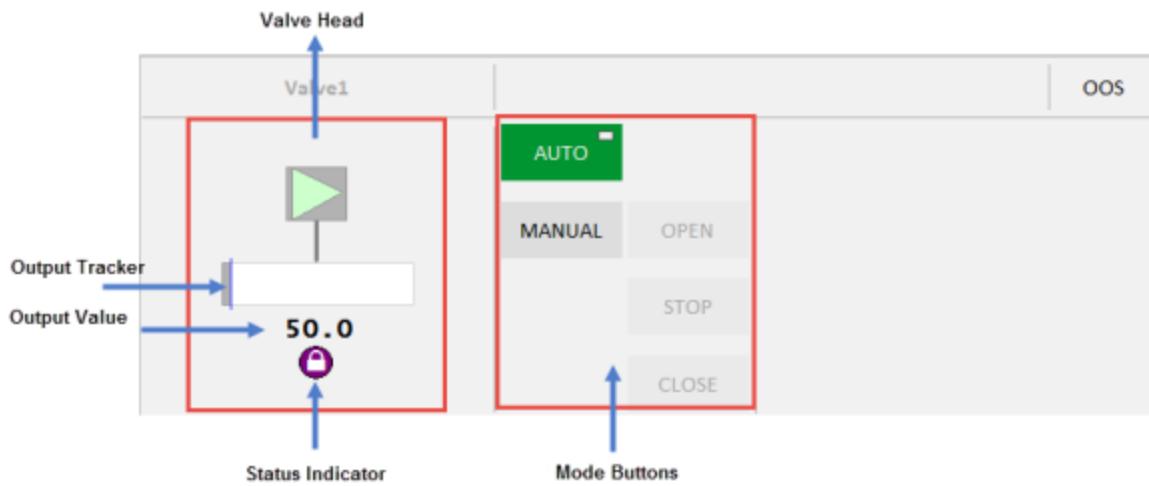
## Complex Valve

The Complex Valve faceplate can be associated with the Block Valve.

---

**Note:** To use for other valve types, create a copy of the faceplate and update the valve type in the composite genie.

---



The Complex Valve faceplate comprises the following components:

Component	Description	Associated Tags
Valve Head	Valve Symbol associated with the selected equipment.	
Mode buttons	<p>Auto, Manual, Open, Close and Stop buttons are displayed on this faceplate. The button is green when a mode is active. When the tag associated with the equipment is 0, the mode is Auto. When the tag is 1, the mode is Manual.</p> <p>The Control Mode button displays two symbols:</p> <ul style="list-style-type: none"> <li>The circle in the top-left corner indicates that a command has been sent to the linked PLC. This indicator is not visible otherwise.</li> <li>The rectangle in the top-right corner indicates that the current mode is the normal mode of operation for the equipment. Displayed on the Auto/Manual buttons, the rectangle depends on 'CtrlModeDef'</li> </ul>	<b>CtrlMode, CtrlModeDef</b> (see <a href="#">Valves</a> ) <b>AutoCmd</b> <b>ManCmd</b>

Component	Description	Associated Tags
	<ul style="list-style-type: none"> <li>• Auto: Automatic mode</li> </ul>	<b>CtrlMode, CtrlModeDef</b> (see <a href="#">Valves</a> ) <b>AutoCmd</b>
	<ul style="list-style-type: none"> <li>• Manual: Manual mode</li> </ul>	<b>CtrlMode, CtrlModeDef</b> (see <a href="#">Valves</a> ) <b>ManCmd</b>
	<ul style="list-style-type: none"> <li>• Open: Open mode - This is a toggle button that toggles the "OpenCmd". The Open button is active (green) when 'OpenCmd' tag is 1. The button is disabled when the valve is in AUTO mode or the valve has an interlock/shutdown condition preventing the valve from being operated.</li> </ul>	<b>OpenCmd</b>
	<ul style="list-style-type: none"> <li>• Close: Close mode - This is a toggle button that toggles the "CloseCmd". The Close button is active (green) when 'Closedmd' tag is 1. The button is disabled when the valve is in AUTO mode or the valve has an interlock/shutdown condition preventing the valve from being closed.</li> </ul>	<b>CloseCmd</b>
	<ul style="list-style-type: none"> <li>• Stop: Stop mode - This is a toggle button that toggles the .The Stop button is active (green )when 'Stopped' is 1. The button shall be disabled when the valve is in AUTO mode.</li> </ul>	<b>StopCmd</b>
Mode Indicators	Local, Manual, Auto and Cascade Mode. Local - Local Mode – field or MCC control only. Manual - Manual Mode - start/stop	<b>CtrlMode, CtrlModeDef</b> (see <a href="#">Valves</a> )

Component	Description	Associated Tags
	or valve position set by control room operator Auto Mode - start/stop and valve position set by control system, control operator sets Setpoint target. Cascade mode – control system sets setpoint target from another loop/logic.	
Equipment Status Indicator	Displays if there is an abnormal status condition e.g. bad comms (stop IO server).	<b>EqStatus</b> (see <a href="#">Status Indicators</a> )
Output Tracker	Blue line on output bar. The position of the output tracker is based on the 'OPTrack' tag and visible when 'TrackDsp' tag is 1.	<b>OPTrack</b> (see <a href="#">Valves</a> ) TrackDsp
Output Value	Number below the output bar. This uses the 'OP' tag which is the same for the output bar. The units of the OP will be displayed to the right of the number if configured.	

## Core Component Genie Libraries

This section of the help describes the components that are used to build the Composite Genies and faceplates provided with a Situational Awareness project.

These components are located in the following libraries:

### Common Components Genie Library

The Common Components Library (named "sa\_common") includes the following components:

- [Clock Timer Genie](#)
- [Control Mode Genie](#)
- [Label Genie](#)
- [MEO Genie](#)
- [Meter Value Genie](#)
- [Out Of Service \(OOS\) Genie](#)
- [Output Bar Genie](#)
- [Output Value Genie](#)

- PV Value Genie
- Selection Adorner Genie
- Selection Adorner Auto Genie
- Status Indicator Genie.

## Miscellaneous Components Genie Library

The Miscellaneous Library (named "sa\_misc") includes the following component:

- Direction Arrow Genie

## Faceplate Components Library

The Faceplate Library (named "sa\_faceplates") includes Genies that can be used as components for equipment faceplates.

- Faceplate Button Genie
- Faceplate Command Button Genie
- Limit Label Genie
- Local/Remote Indicator Genie
- MEO Selector Genie
- Output Tracker Genie
- Faceplate Output Value Genie
- Faceplate Selection Adorner Genie

## See Also

[Create a New Faceplate](#)

## Common Components Genie Library

This Common Components Genie Library (named "sa\_common") includes the following Genies that are used to build the Composite Genies and faceplates provided with a Situational Awareness project.

- Clock Timer Genie
- Control Mode Genie
- Label Genie
- MEO Genie
- Meter Value Genie
- Out Of Service (OOS) Genie
- Output Bar Genie
- Output Value Genie
- PV Value Genie

- Selection Adorner Genie
- Selection Adorner Auto Genie
- Status Indicator Genie.

## See Also

[Create a New Faceplate](#)  
[Core Component Genie Libraries](#)

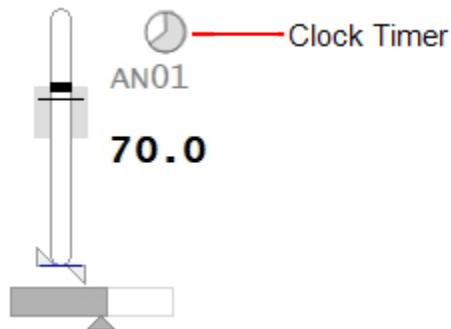
### Clock Timer Genie

**Genie Name:** clocktimer\_24

**Genie Library:** sa\_common

**System Project:** SA\_Library

The Clock Timer Genie (named "clocktimer\_24") allows you to add an indicator to a page or faceplate that uses a grey fill to show how much time has passed since the PV last changed.



If the clock reaches 100% and no new sample has been received, the clock continues to fill but the grey segments are replaced with black. If a second cycle passes without an updated sample reading, the clock will remain solid black.

This Genie binds to the following equipment items:

- **PV** (see [Drive, Belt and Mining Objects](#))
- **Timer** (see [Meter Special Elements](#))

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
Equipment	The name of the piece of equipment with which the Genie is associated.
Prefix	If the piece of equipment has a prefix specified for the PV or Timer items (enabling support for multiple values), enter the prefix you want the Genie to access.

## See Also

[Configure a Genie](#)

[Core Component Genie Libraries](#)

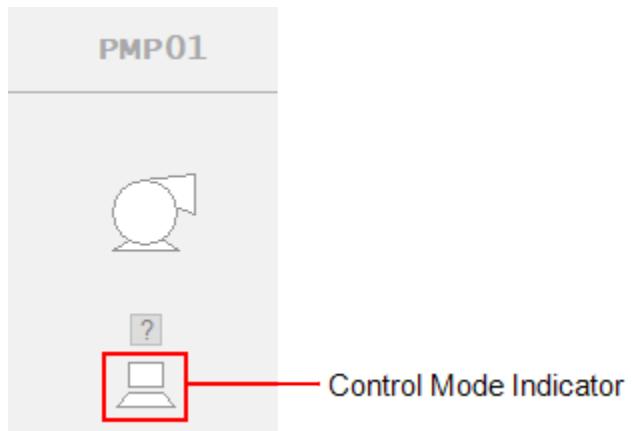
## Control Mode Genie

**Genie Name:** control\_mode\_<n>

**Genie Library:** sa\_common

**System Project:** SA\_Library

The Control Mode Genie allows you to add a control mode indicator to a faceplate.



It indicates the following modes:

- 0 – Automatic (A)
- 1 – Manual (M)
- 2 – Cascade (C)
- 3 – Local (L)
- 4 – Special control (computer symbol).

The Genie is available in three sizes:

- control\_mode\_14
- control\_mode\_18
- control\_mode\_24

This Genie binds to the following equipment items:

- **CtrlMode** (see [Drive, Belt and Mining Objects](#))
- **CtrlModeDef** (see [Drive, Belt and Mining Objects](#))

See [Create a New Faceplate](#) for an example of how this Genie is activated.

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
Equipment	The name of the piece of equipment with which the Genie is associated.

## See Also

[Configure a Genie](#)

[Core Component Genie Libraries](#)

## Label Genie

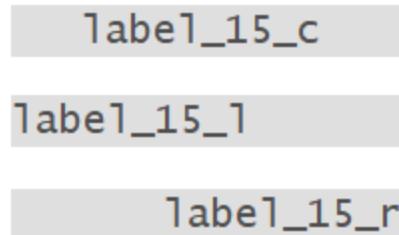
**Genie Name:** label\_<n>\_c / label\_<n>\_l / label\_<n>\_r

**Genie Library:** sa\_common

**System Project:** SA\_Library

The Label Genie allows you to add a label to a page or faceplate. Six variations of the Genie are included in the SA\_Library project:

- label\_12\_c — a 12 pixel center label
- label\_12\_l — a 12 pixel left label
- label\_12\_r — a 12 pixel right label
- label\_15\_c — a 15 pixel center label
- label\_15\_l — a 15 pixel left label
- label\_15\_r — a 15 pixel right label



## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
Label	Enter the text that you would like to appear on the label.

## See Also

[Configure a Genie](#)

[Core Component Genie Libraries](#)

## MEO Genie

**Genie Name:** meo\_<n> / meo\_<n>\_r

**Genie Library:** sa\_common

**System Project:** SA\_Library

The MEO Genie is used to create a Multiple Equipment Object (MEO). It is called by the [MEO Selector Genie](#), which is the Genie you should use to add an MEO to a faceplate.

Four variations of the Genie are included in the SA\_Library project:

- meo\_16 — a small square MEO
- meo\_17\_r — a small round MEO
- meo\_22 — a larger square MEO
- meo\_25\_r — a larger round MEO

The Genie binds to the following equipment item:

- **RunStatus** (see [Drive, Belt and Mining Objects](#))

For more information, see [Configure Equipment Selection for Group Objects](#).

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
Equipment	The name of the piece of equipment with which the Genie is associated.

## See Also

[Configure a Genie](#)

[Core Component Genie Libraries](#)

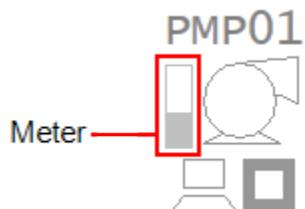
## Meter Value Genie

**Genie Name:** meter\_value\_<n>

**Genie Library:** sa\_common

**System Project:** SA\_Library

The Meter Value Genie allows you to add a meter value indicator to a page or faceplate.



Two variations of the Genie are included in the SA\_Library project:

- meter\_value\_28
- meter\_value\_35

This Genie binds to the following equipment item:

- **PV** (see [Drive, Belt and Mining Objects](#))

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
Equipment	The name of the piece of equipment with which the Genie is associated.
Prefix	If the piece of equipment has a prefix specified for the PV item (enabling support for multiple values), enter the prefix you want the Genie to access.

## See Also

[Configure a Genie](#)

[Core Component Genie Libraries](#)

## Out Of Service (OOS) Genie

**Genie Name:** oos\_<n>

**Genie Library:** sa\_misc

**System Project:** SA\_Library

The Out Of Service (OOS) Genie allows you to add an OOS button to a page or faceplate. Four variations of the Genie are included in the SA\_Library project:

- oos\_16
- oos\_22
- oos\_25
- oos\_33

This Genie binds to the following equipment item:

- **RunStatus** (see [Drive, Belt and Mining Objects](#))

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
Equipment	The name of the piece of equipment with which the Genie is associated.

## See Also

[Configure a Genie](#)

[Core Component Genie Libraries](#)

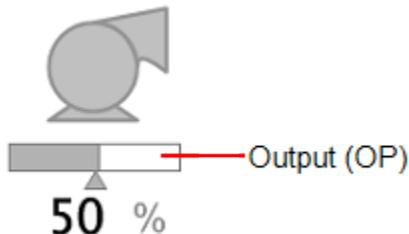
## Output Bar Genie

**Genie Name:** output\_<n> / output\_<n>\_rb / output\_<n>\_rb\_v / output\_<n>\_v

**Genie Library:** sa\_misc

**System Project:** SA\_Library

The Output Bar Genie allows you to add an Output (OP) Bar to a faceplate.

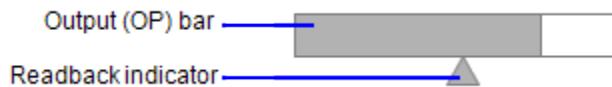


Fourteen variations of the Genie are included in the SA\_Library project:

- output\_110
- output\_110\_rb
- output\_50
- output\_50\_rb
- output\_50\_rb\_v
- output\_50\_v
- output\_65
- output\_65\_rb
- output\_65\_rb\_v
- output\_65\_v
- output\_80

- output\_80\_rb
- output\_80\_rb\_v
- output\_80\_v

The "\_rb" variations will include a readback indicator.



The "\_v" variations are a vertical version of the OP bar.

This Genie binds to the following equipment items:

- **FB** (see [Control Meters - Common Elements](#))
- **OP** (see [Control Meters - Common Elements](#))

---

**Note:** The 'output\_50\_rb\_v' and 'output\_65\_rb\_v' variations of this Genie can show the feedback position for a set of sub-equipment. They can be used in conjunction with the MEO Genie.

---

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
Equipment	The name of the piece of equipment with which the Genie is associated.
Prefix	If the piece of equipment has a prefix specified for the FB or OP items (enabling support for multiple values), enter the prefix you want the Genie to access.
Equipment1 — Equipment5	Applies to the 'output_50_rb_v' and 'output_65_rb_v' variations of the Genie. To access a set of sub-equipment, enter up to five equipment names.

## See Also

[Configure a Genie](#)

[Core Component Genie Libraries](#)

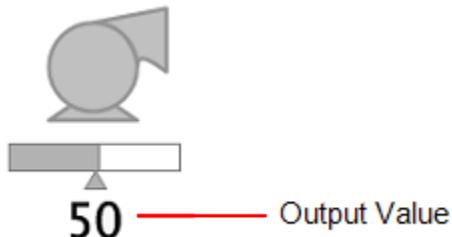
## Output Value Genie

**Genie Name:** output\_value\_<n>\_r

**Genie Library:** sa\_misc

**System Project:** SA\_Library

The Output Value Genie allows you to add a value label to an output bar on a page.



Three variations of the Genie are included in the SA\_Library project:

- output\_value\_11\_r
- output\_value\_14\_r
- output\_value\_18\_r

The Genie binds to the following equipment item:

- **OP** (see [Control Meters - Common Elements](#))

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
Equipment	The name of the piece of equipment with which the Genie is associated.
Prefix	If the piece of equipment has a prefix specified for the OP item (enabling support for multiple values), enter the prefix you want the Genie to access.

## See Also

- [Configure a Genie](#)
- [Core Component Genie Libraries](#)

### PV Value Genie

**Genie Name:** pv\_value\_<n>\_l / pv\_value\_<n>\_r

**Genie Library:** sa\_misc

**System Project:** SA\_Library

The PV Value Genie allows you to add a value label for a process variable (PV) on a faceplate. Four variations of the Genie are included in the SA\_Library project:

- pv\_value\_14\_l
- pv\_value\_14\_r
- pv\_value\_18\_l
- pv\_value\_18\_r

The Genie binds to the following equipment item:

- **PV** (see [Drive, Belt and Mining Objects](#))

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
Equipment	The name of the piece of equipment with which the Genie is associated.
Prefix	If the piece of equipment has a prefix specified for the PV item (enabling support for multiple values), enter the prefix you want the Genie to access.

## See Also

- [Configure a Genie](#)

## Selection Adorner Genie

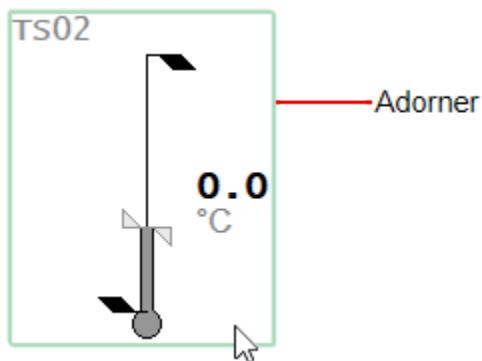
**Genie Name:** selection\_adorner

**Genie Library:** sa\_misc

**System Project:** SA\_Library

The Selection Adorner Genie (named "selection\_adorner") is a Genie that enables you to set the context of the workspace by clicking in an area that surrounds a piece of equipment. This Genie is useful with existing pages that do not include the Situational Awareness library objects. It allows legacy objects to be selected.

The border of the adorner is hidden at runtime until an operator points the mouse at the associated piece of equipment.



Clicking within the adorner will set the context for the workspace.

**Note:** A variation of this Genie is available that will automatically adjust to the size of the associated piece of equipment. See [Selection Adorner Auto Genie](#).

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- AN:** This is an automatically-generated unique ID for the Genie object.
- Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
OnClickFunction	A Cicode function that is executed when an operator clicks on the adorner around a piece of equipment with which the adorner is associated.
OnRightClickFunction	A Cicode function that is executed when an operator right-clicks on the adorner around a piece of equipment with which the adorner is associated..
Equipment	The name of the associated piece of equipment that

Parameter	Description
	you want use to set the context of the workspace.
EquipmentSecondary	The name of the piece of equipment that references any sub-equipment.

## See Also

[Configure a Genie](#)

### Selection Adorner Auto Genie

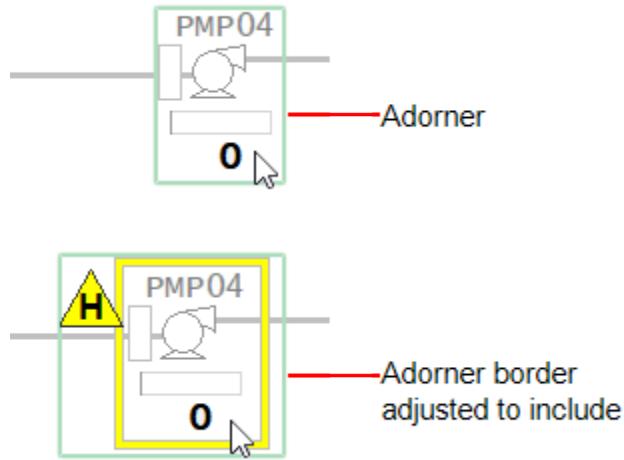
**Genie Name:** selection\_adorner\_auto

**Genie Library:** sa\_misc

**System Project:** SA\_Library

The Selection Adorner Auto Genie (named "selection\_adorner\_auto") is a Genie that enables you to set the context of the workspace by clicking in an area that surrounds a piece of equipment. The border of the adorner is hidden at runtime until an operator points the mouse at the associated piece of equipment. Clicking within the adorner will set the context for the workspace.

The size of the Adorner Auto Genie will automatically adjust to the size of the associated piece of equipment. For example, it will adjust to include an alarm indicator when it appears.



## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
RefAN	The AN number of the object you want the adorner to surround.  By default, the following function is specified: Adorner_GetAnFromName("..\\..\\Composite")  This will retrieve the AN of an assumed sibling object named "Composite".  If required, delete this function and enter the AN for the object you would like to use the adorner with.
Padding	If you want to add padding to the adorner, enter the required padding size in pixels.
IsParentRefAN	Specifies if the parent AN is used.  TRUE = Parent AN is used.  FALSE = Parent AN is not used.
OnClickFunction	A Cicode function that is executed when an operator clicks on the adorner around a piece of equipment with which the adorner is associated.
OnRightClickFunction	A Cicode function that is executed when an operator right-clicks on the adorner around a piece of equipment with which the adorner is associated..
Equipment	The name of the associated piece of equipment that you want use to set the context of the workspace.
EquipmentSecondary	The name of the piece of equipment that references any sub-equipment.

## See Also

[Configure a Genie](#)

[Core Component Genie Libraries](#)

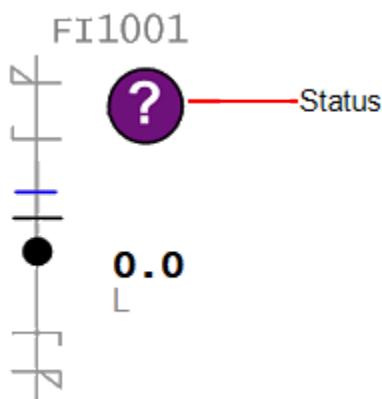
## Status Indicator Genie

**Genie Name:** status\_<n>

**Genie Library:** sa\_misc

**System Project:** SA\_Library

The Status Indicator Genie allows you to add a status indicator to a page or faceplate. See Status Indicators.



Two variations of the Genie are included in the SA\_Library project:

- status\_24
- status\_35

This Genie binds to the following equipment items:

- **EQStatus** (see [Drive, Belt and Mining Objects](#))

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
Equipment	The name of the piece of equipment with which the Genie is associated.

## See Also

[Configure a Genie](#)

[Core Component Genie Libraries](#)

## Miscellaneous Components Genie Library

The Miscellaneous Components Library (named "sa\_misc") includes the following Genie:

- Direction Arrow Genie

## See Also

[Create a New Faceplate](#)  
[Core Component Genie Libraries](#)

### Direction Arrow Genie

**Genie Name:** directionarrow\_31 / directionarrow\_45

**Genie Library:** sa\_misc

**System Project:** SA\_Library

The Direction Arrow Genie allows you to add a direction indicator to a page or faceplate. The Genie will indicate forward and reverse states.



Forward



Reverse

The Genie is available in two sizes:

- directionarrow\_31
- directionarrow\_45

The Genie binds to the following equipment item:

- Left
- Right

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
Equipment	The name of the piece of equipment with which the Genie is associated.

## See Also

- [Configure a Genie](#)
- [Core Component Genie Libraries](#)
- [Miscellaneous Components Genie Library](#)

## Faceplate Components Genie Library

The Faceplate Library (named "sa\_faceplates") includes Genies that can be used as components for equipment faceplates.

- [Faceplate Button Genie](#)
- [Faceplate Command Button Genie](#)
- [Limit Label Genie](#)
- [Local/Remote Indicator Genie](#)
- [MEO Selector Genie](#)
- [Output Tracker Genie](#)
- [Faceplate Output Value Genie](#)
- [Faceplate Selection Adorner Genie](#)

## See Also

- [Create a New Faceplate](#)
- [Core Component Genie Libraries](#)

## Faceplate Button Genie

**Genie Name:** button\_fp\_<n>\_<n>

**Genie Library:** sa\_faceplate

**System Project:** SA\_Library

The Faceplate Button Genie allows you to add a button to a faceplate.

---

**Note:** If you need a button that binds directly to an item, use the [Faceplate Command Button Genie](#).

---

The button features two indicators.



- Targeted indicator - indicates that a command has been issued, but has not completed.
- Normal mode indicator - indicates that the associated piece of equipment is in its normal operational state. For example, if the button represents Auto/Manual, then Auto maybe the normal operational state and is therefore indicated as such.

Two variations of the Genie are included in the SA\_Library project:

- button\_fp\_112\_44
- button\_fp\_84\_40

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
Equipment Context	An equipment name that is used to define equipment context, or a Super Genie association that defines equipment context.  By default, the following is used: <code>?__EquipmentRef?</code>  This specifies that the current workspace context is used.
Label	The text label that appears on the button.
Tooltip	Additional text that appears when the mouse hovers over the button.
Click Command	An expression that is called when the button is clicked on.
Active When	An expression (applied in conjunction with Area and Privilege) that can provide further logic to specify when the button is active.
Normal When	An expression that determines when the normal mode indicator displays.
Targeted When	An expression that determines when the targeted indicator displays.
Disabled When	An expression (applied in conjunction with Area and Privilege) that can provide further logic to specify when the button is disabled.
Hidden When	An expression that determines when the button is hidden.

Parameter	Description
Security: Area	Enter the area to which the button belongs. Only users with access to this area can use the button.
Security: Privilege	Enter the privilege level that a user needs to possess to use this button.

## See Also

- [Create a New Faceplate](#)
- [Configure a Genie](#)
- [Faceplate Components Genie Library](#)
- [Core Component Genie Libraries](#)

### Faceplate Command Button Genie

**Genie Name:** button\_fp\_cmd\_<n>\_<n>

**Genie Library:** sa\_faceplate

**System Project:** SA\_Library

The Faceplate Command Button Genie allows you to add a button to a faceplate that binds directly to an item. The button features two indicators.



- Targeted indicator - indicates that a command has been issued, but has not completed.
- Normal mode indicator - indicates that the associated piece of equipment is in its normal operational state.

Two variations of the Genie are included in the SA\_Library project:

- button\_fp\_cmd\_112\_44
- button\_fp\_cmd\_84\_40

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table

below:

Parameter	Description
Equipment Context	An equipment name that is used to define equipment context, or a Super Genie association that defines equipment context.  By default, the following is used: <code>?__EquipmentRef?</code> This specifies that the current workspace context is used.
Label	The text label that appears on the button.
Tooltip	Additional text that appears when the mouse hovers over the button.
Command Item	The name of the command tag that, when written to, will update the status item.
Status Item	The name of the item that sets the button to its active state.
Normal Mode Item	The name of the item that controls the appearance of the normal mode indicator.
Status Value	The value of the normal mode item that controls the normal mode indicator.
Disabled When	An expression (applied in conjunction with Area and Privilege) that can provide further logic to specify when the button is disabled.
Hidden When	An expression that determines when the button is hidden.
Security: Area	Enter the area to which the button belongs. Only users with access to this area can use the button.
Security: Privilege	Enter the privilege level that a user needs to possess to use this button.

## See Also

- [Create a New Faceplate](#)
- [Configure a Genie](#)
- [Faceplate Components Genie Library](#)
- [Core Component Genie Libraries](#)

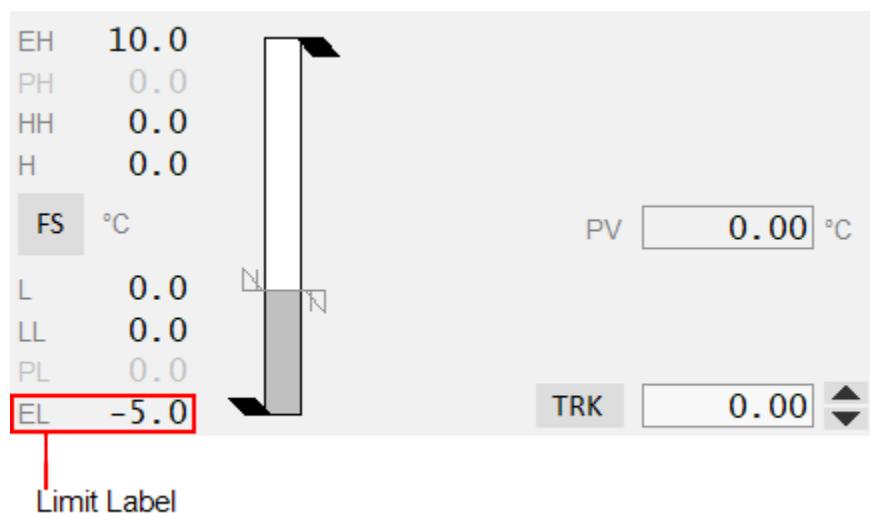
## Limit Label Genie

**Genie Name:**limitlabel\_<n>

**Genie Library:** sa\_faceplate

**System Project:** SA\_Library

The Limit Label Genie allows you to add a label to a limit indicator on a faceplate.



It can display alarming limits, or the practical engineering range.

Practical engineering range:

- EH = engineering high.
- PH = practical high.
- PL = practical low.
- EL = engineering low.

Alarming limits:

- HH = high high alarm.
- HI = high alarm.
- LO = low alarm.
- LL = low low alarm.

Two variations of the Genie are included in the SA\_Library project:

- limitlabel\_12
- limitlabel\_15

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
Equipment Context	An equipment name that is used to define equipment context, or a Super Genie association that defines equipment context. By default, the following is used: ?__EquipmentRef? This specifies that the current workspace context is used.
Item Prefix	Enter the equipment item prefix you want the Genie to access.
Limit Index	Enter the value that indicates the limit index you would like the label to use. You can enter one of the following values, or one of the pre-defined labels included in the SA_Library system project. <ul style="list-style-type: none"><li>• HH = 3 (METER_LIMIT_HH)</li><li>• HI = 2 (METER_LIMIT_H)</li><li>• LO = 1 (METER_LIMIT_L)</li><li>• LL = 0 (METER_LIMIT_LL)</li><li>• PH = -1 (METER_LIMIT_PH)</li><li>• PL = -2 (METER_LIMIT_PL)</li><li>• EH = -3 (METER_LIMIT_EH)</li><li>• EL = -4 (METER_LIMIT_EL)</li></ul> <b>Note:</b> If you want to use PLC limits, you can add additional labels that correspond to those configured by the [SA_Library.Meter]PLCLimitNames parameter. See Associate PLC Limits with Equipment.

## See Also

[Create a New Faceplate](#)

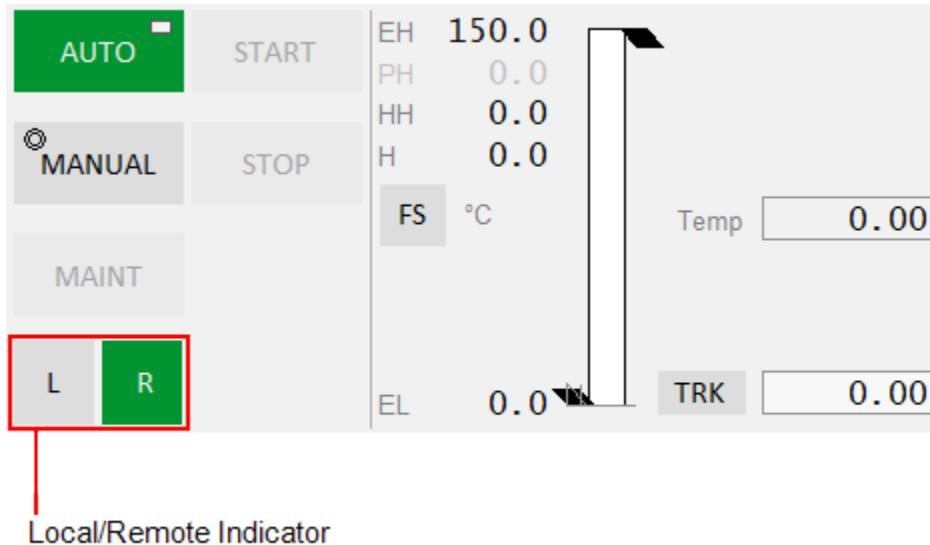
[Configure a Genie](#)

[Faceplate Components Genie Library](#)[Core Component Genie Libraries](#)

## Local/Remote Indicator Genie

**Genie Name:**localremote\_indicator\_<n>**Genie Library:** sa\_faceplate**System Project:** SA\_Library

The Local/Remote Indicator Genie allows you to add a local/remote indicator to a faceplate.



If the control mode for a piece of equipment is set to "Local" (mode 3), the indicator will highlight "L". For all other control modes, the indicator will highlight "R".

Two variations of the Genie are included in the SA\_Library project:

- localremote\_indicator\_12
- localremote\_indicator\_16

The Genie binds to the following equipment item:

- **CtrlMode** (see [Drive, Belt and Mining Objects](#))

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
Equipment	The name of the piece of equipment with which the Genie is associated.

## See Also

- [Create a New Faceplate](#)
- [Configure a Genie](#)
- [Faceplate Components Genie Library](#)
- [Core Component Genie Libraries](#)

## MEO Selector Genie

**Genie Name:** meo\_selector\_<n>  
**Genie Library:** sa\_faceplate  
**System Project:** SA\_Library

The MEO Selector Genie allows you to add an MEO selector to a faceplate, allowing you to switch between the objects within the Multiple Equipment Object. Two variations of the Genie are included in the SA\_Library project:

- meo\_selector\_18
- meo\_selector\_22

The Genie binds to the following equipment item:

- **RunStatus** (see [Drive, Belt and Mining Objects](#))

For more information, see [Configure Equipment Selection for Group Objects](#).

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
Equipment	The name of the piece of equipment with which the Genie is associated.
Equipment Secondary	The name of the piece of equipment that references any sub-equipment.

## See Also

- [Create a New Faceplate](#)
- [Configure a Genie](#)
- [Faceplate Components Genie Library](#)
- [Core Component Genie Libraries](#)

## Output Tracker Genie

**Genie Name:** optrack\_<n>  
**Genie Library:** sa\_faceplate  
**System Project:** SA\_Library

The Output Tracker Genie allows you to add an [output tracker](#) to a faceplate. Two variations of the Genie are included in the SA\_Library project:

- optrack\_135
- optrack\_169

This Genie binds to the following equipment items:

- **OPTTrack** (see [Meter Common Elements](#))
- **TrackDsp** (see [Meter Common Elements](#))

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
Equipment	The name of the piece of equipment with which the Genie is associated.

## See Also

- [Create a New Faceplate](#)
- [Configure a Genie](#)
- [Faceplate Components Genie Library](#)
- [Core Component Genie Libraries](#)

## Output Indicator Genie

**Genie Name:** output\_<n>\_fp

**Genie Library:** sa\_faceplate

**System Project:** SA\_Library

The Output Indicator Genie allows you to add an output indicator to a faceplate. Two variations of the Genie are included in the SA\_Library project:

- output\_160\_fp
- output\_170\_fp

This Genie binds to the following equipment items:

- **OP** (see [Control Meters - Common Elements](#))
- **FB** (see [Control Meters - Common Elements](#))

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
Show Readback	Determines if the readback indicator is displayed. TRUE = Readback indicator is displayed (default). FALSE = Readback indicator is not displayed.
Equipment	The name of the piece of equipment with which the Genie is associated.

## See Also

- [Create a New Faceplate](#)
- [Configure a Genie](#)
- [Faceplate Components Genie Library](#)
- [Core Component Genie Libraries](#)

### Faceplate Output Value Genie

**Genie Name:** output\_value\_<n>\_r

**Genie Library:** sa\_faceplate

**System Project:** SA\_Library

The Faceplate Output Value Genie allows you to add an output value indicator to a faceplate.

Two variations of the Genie are included in the SA\_Library project:

- output\_value\_12\_r
- output\_value\_15\_r

This Genie binds to the following equipment item:

- **OP** (see [Control Meters - Common Elements](#))

### Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

### Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
Equipment	The name of the piece of equipment with which the Genie is associated.

## See Also

- [Create a New Faceplate](#)
- [Configure a Genie](#)
- [Faceplate Components Genie Library](#)
- [Core Component Genie Libraries](#)

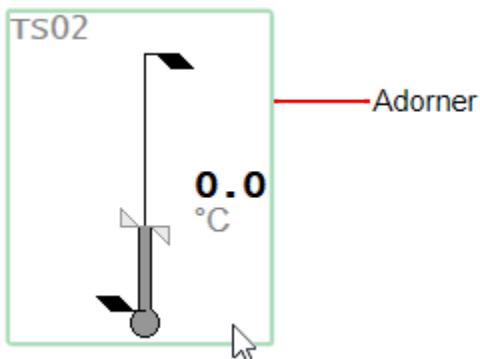
### Faceplate Selection Adorner Genie

**Genie Name:** selection\_adorner\_fp

**Genie Library:** sa\_faceplate

**System Project:** SA\_Library

The Faceplate Selection Adorner Genie (named "selection\_adorner\_fp") allows you to set the context of the workspace by clicking in the area surrounding a highlighted piece of equipment on a faceplate. The border of the adorner is hidden at runtime until an operator points the mouse at the associated piece of equipment.



Clicking within the adorner will set the context for the workspace.

## Genie Properties

Complete the following properties for the Genie in the Genie Properties dialog box.

- **AN:** This is an automatically-generated unique ID for the Genie object.
- **Name:** Optional. If you need to reference the Genie at runtime using Cicode, specify a unique name.

## Genie Parameters

Select the Genie and double-click on it to view its parameters. The parameters are used as described in the table below:

Parameter	Description
OnClickFunction	A Cicode function that is executed when an operator clicks on the adorner around a piece of equipment with which the adorner is associated. By default, the following function is used: <code>Workspace_SelectEquipment(TRUE)</code>
Equipment	The name of the associated piece of equipment that you want use to set the context of the workspace.
EquipmentSecondary	The name of the piece of equipment that references any sub-equipment.

## See Also

- [Create a New Faceplate](#)
- [Configure a Genie](#)
- [Faceplate Components Genie Library](#)
- [Core Component Genie Libraries](#)

## StruxureWare (SxW) Templates Project

The SxW Template project (named "SxW\_Style\_Include") is a system project that includes a set of templates designed for use with StruxureWare systems.

---

**Note:** These templates are not designed for direct upgrade from existing Tab Style templates. If you want to use these templates you will need to manipulate the page objects so that the objects are rendered correctly on screen.

When a new Plant SCADA project is created to use SxW\_Style\_1 as the default template style, the SxW Template (SxW\_Style\_Include) project is automatically incorporated as an included project. This means the project's templates are available for implementation when creating your graphics pages in Graphics Builder.

As well as including a standard graphics page template for creating plant mimics, the project also includes predefined alarm templates, a set of file templates for displaying text, Rich Text Format and HTML files. They have identical page navigation, alarm banners and menus for consistent "look and feel", and usability across an entire project.

---

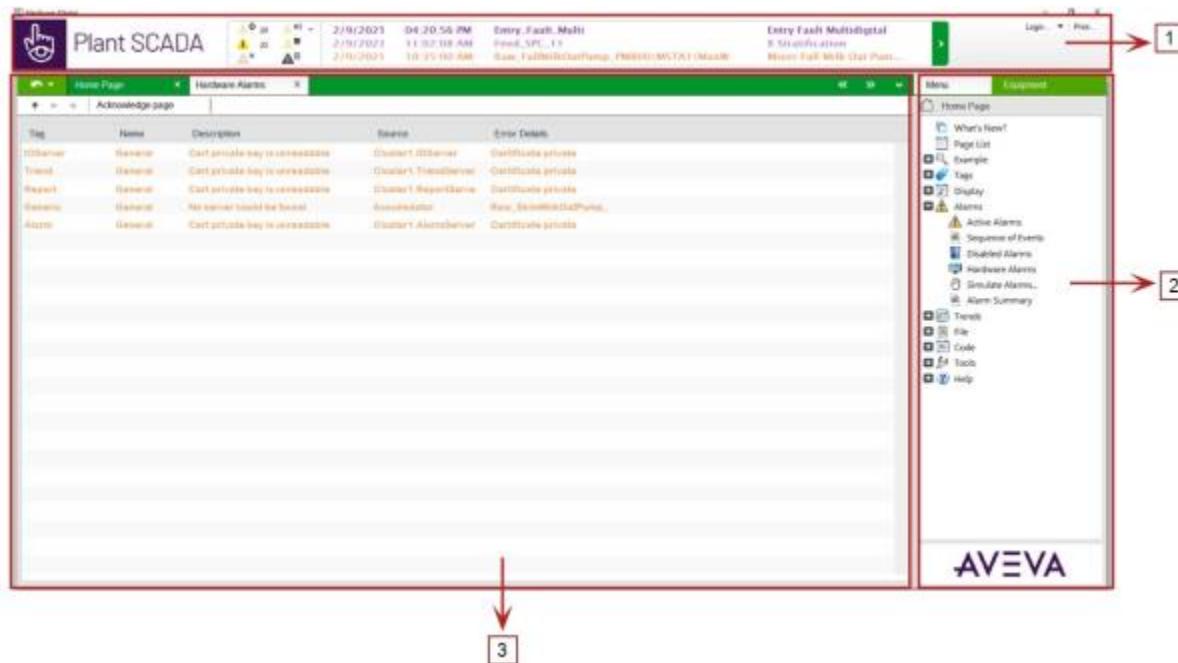
**Note:** Avoid making changes to the SxW Template project for use as a runtime project. A Plant SCADA upgrade will install a new version of the project, which will overwrite any changes you make.

## See Also

- [The SxW Interface](#)
- [Use SxW Pages and Templates](#)
- [Creating a New Project](#)

## The SxW Interface

The SxW template is divided into 3 parts:



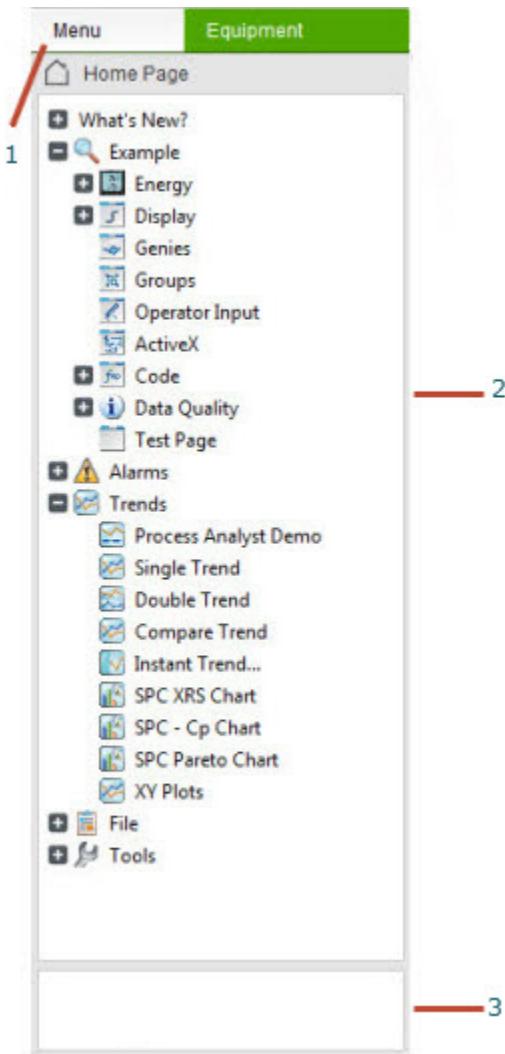
### 1. The Header



<b>1</b>	The icon and name of the product. They are always visible, and cannot be re-sized.
<b>2</b>	The Alarm panel. It displays status (and counts) for various types of alarms, including active, unacknowledged, disabled and hardware alarms, shortcuts to relevant alarm and events page, and controls to silence alarm (sound).
<b>3</b>	The Alarm banner. It lists recent alarms. Click on the white arrow in the side tab to expand/reduce the alarm banner.
<b>4</b>	Currently logged in user name. click on the drop-down arrow for more login related options, and print commands. The options for the login drop-down menu are user configurable via the Menu Configuration form. The items for this menu should be assigned with page name of "Template" and Level 1

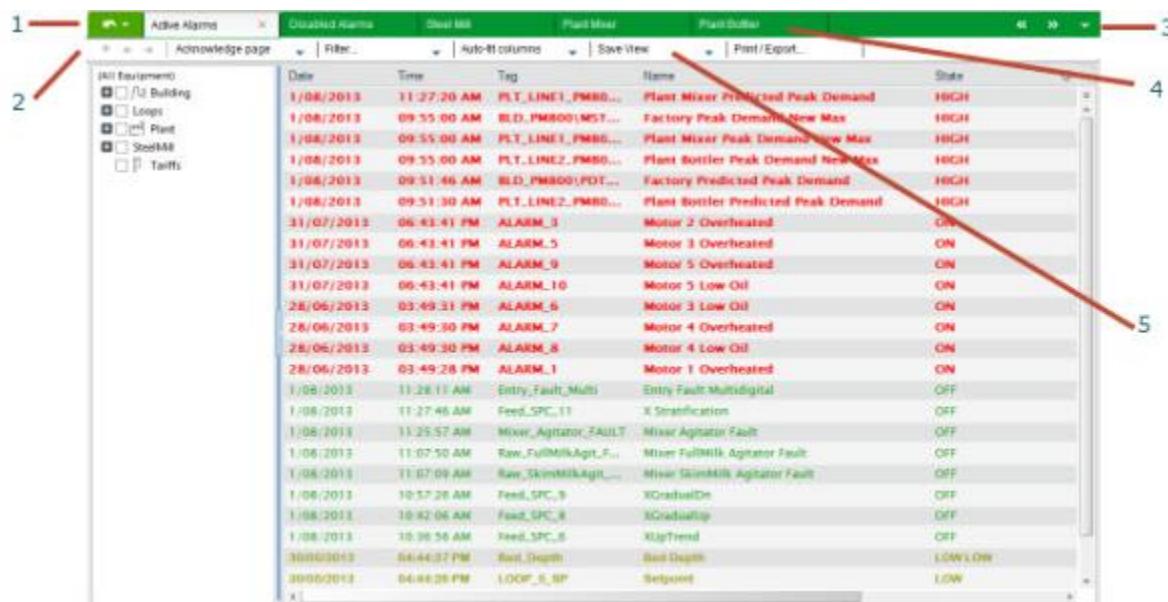
name of "Login". When the alarm banner is expanded, this section will collapse to a single drop-down arrow. The information shown here will be automatically incorporated to the drop-down menu.

## 2. The Right Panel



1	<p>Option to switch between navigation tree representing your Menu Configuration or Equipment hierarchy. Click on the Menu option to view common menu configuration for all pages in your project. On selecting a page it will open in the main window. The icon dimensions for optimal display in the menu and equipment tree are 16x16.</p> <p>To view the Equipment tree (lists all equipment defined in your project) click on the <b>Equipment</b> option.</p>
2	<p>Click on the plus and minus signs next to the nodes to expand and collapse items.</p>
3	<p>User configurable logo. To display your logo in the allocated space, define your logo as bitmap symbol in your project. The dimension of the bitmap should be 230 x 50. Assign the name of your symbol (in format of library.symbol) to project / INI parameter [Page]Logo.</p>

### 3. The Main Window



1	<p>History button. Move back through the tabs that have been opened.</p>
2	<p>Page specific navigation commands.</p> <ul style="list-style-type: none"> <li>Up navigation command changes the display to the "parent" page. You can assign a parent page by setting an environment variable in Graphics Builder. To do so,</li> </ul>

	open the page properties dialog of a page (via File   Properties menu command), click the Environment tab, and add a variable called "ParentPage". Assign the name of the parent page to the value of this variable <ul style="list-style-type: none"> <li>• Left / Right navigation commands display the previous / next page relative to the current page. To configure previous / next page, open the page properties dialog in Graphics Builder (via File   Properties menu command), click the General tab, and set page name to the Previous page / Next page fields.</li> </ul>
<b>3</b>	Forward and back navigation command buttons and the Display list button. Use to scroll and display the list of opened pages.
<b>4</b>	Tabs of opened pages. Allows you to access and navigate between pages that have been selected from the right-panel or launched from other pages.
<b>5</b>	Page specific menu commands. Menu configuration defined specified for this page appears here. (Menu configuration that is common to all pages appears on the navigation tree on the right panel).

**Note:** The Alarm templates include a left-hand panel that enables you to filter using the Equipment hierarchy.

## See Also

[Use SxW Pages and Templates](#)

## Use SxW Pages and Templates

When you create a new project based on the SxW templates starter project, the following templates are available for you to use as you create your graphics pages. The templates are divided into three parts with each part sharing common functionality, including page navigation, menus and tools. Options available may vary according to the type of page.

The SxW template starter project has the following templates:

Template Type	Description
Normal	Standard graphics page. See <a href="#">Normal Page Template</a> .
Blank	For users who want to create their pages from scratch. See <a href="#">Blank Page Template</a> .
Data Browse	For users who want to display a table of data retrieved from built-in data browse functions on a page. See <a href="#">Data Browse Page Template</a> .

Template Type	Description
Alarm	Includes Alarm, Disabled, Hardware, Sequence of events and Summary for displaying different types of alarm list. See <a href="#">Alarm Page Templates</a> .
Process Analyst	Includes SinglePA, DoublePA, PopPA. See <a href="#">Process Analyst Page Templates</a> .
SPC	Includes SpcPareto, SpcCpk, SpcXRSChart, EventSPCXRS, MeanMeanChart, RangeChart and StandardChart. See <a href="#">SPC Trend Page Templates</a> .
File	Includes File, File_RTF and File_HTML for displaying a plain text file, rich text format file and HTML file / URL respectively. See <a href="#">File Page Templates</a> .

## See Also

[Creating a New Project](#)

[Creating New Pages](#)

## Normal Page Template

A template designed for creating user-necessary content and plant mimics. This page contains the navigation and alarm panels featured on SxW templates. If you are creating a project based on the SxW style templates, use this template for your standard graphics pages.

The Normal template creates a page in display format to be shown with a title bar. Two wide-screen sizes are available: HD768 (1388x768,16:9) and HD1080 (1920x1080,16:9).

## See Also

[Create Pages](#)

[Secure the Window Titlebar](#)

## Blank Page Template

The blank template is included for users who want to create their pages from scratch. There are no differences between the SxW blank template and the template in the include project.

## See Also

[Create Pages](#)

## Data Browse Page Template

The data browse template is included for users who want to create a page that displays a table of data retrieved from built-in data browse functions. The template includes the Equipment tree view on the left side of the main window. This is used to filter the display list based on the selected equipment node .

When using this template to create a new page within your project, configure the Data Browse pop up form. When configured the new page will display the appropriate data session at runtime.

Field	Description
Equipment Symbol Library (Optional)	The symbol library for displaying equipment symbol for each item in the equipment tree. The symbol is displayed based on the type of the equipment. It should have the same name as the equipment type assigned to an equipment record. If not specified, it defaults to a symbol library named "Equipment".
Equipment Default Symbol (Optional)	The default symbol to be displayed next to the item in the equipment tree if symbol corresponding to the equipment type of the item cannot be found from the supplied Equipment Symbol Library. The symbol should be specified in format of <library name>.<symbol name>
Browse Type	The type of project configuration to be browsed and displayed, including: AccumBrowse – for listing accumulators AlmBrowse – for listing alarm tags EquipBrowse - for listing equipment ServerBrowse – for listing servers TagBrowse – for listing variable tags TrnBrowse – for listing trend tags
Clusters (Optional)	A comma delimited string specifying only items from a subset of clusters to be included in the display. If not specified, items from all clusters will be included.
Filter (Optional)	A filter expression specifying the records to return during the browse If not specified, all records will be returned. Please see the browse open Cicode function of the selected browse type for the details of filter expression supported.
Fields (Optional)	A comma delimited string specifying the fields (columns) to be returned during the browse. If it is not specified, all fields of the selected browse type will be displayed: Tag, Tag Item, Value, Comment, Cluster, Equipment, Item, Type, IODev, Addr, Arr_Size, Eng_Zero, Eng_Full, Eng_Units, Format, Deadband,

Field	Description
	Raw_Zero,Raw_Full,Scaled_Type,Ovr_Mode,Ctrl_Mode,Override,Valid,Status,Field,Num_Subs.
Sort By (Optional)	A comma delimited string specifying the order of sorting preferences. This is only applicable to TagBrowse. Please see the Sort parameter of the built-in Cicode function <a href="#">TagBrowseOpen</a> for details.
View Name (Optional)	<p>A name to represent a view setting about the placement and width of the columns displayed on the browse table. The view name works with the Save view and Restored saved view options at runtime. You can assign a unique view on different data browse pages or share the same view among multiple pages. When user selects the Save view command from the Action tab, the current column settings will be saved under the view name. Once a view is saved, when user re-enters the page, previous saved view settings will be retrieved automatically. The saved view can also be restored on demand by selecting the Restored saved view from the Action tab. User can reset (forget) the view to the default as how it was pre-configured by selecting the Restore view to default command under the Action tab.</p> <p>For system engineers, the information associated with a view name is saved under INI parameters:</p> <pre>[BrowseTableView] &lt;BrowseType&gt;.&lt;ViewName&gt;.Fields &lt;BrowseType&gt;.&lt;ViewName&gt;.ColWidths</pre> <p>in comma delimited format. You can pre-configure these parameters in your project as the defaults.</p>
Privilege (Optional)	Only applicable to Browse Type of "TagBrowse", this specifies the privilege required by the logged on user to change the values / operating mode of variable tags displayed on the table. If no privilege is defined, no modification will be allowed.

Use the equipment tree to filter the data browse table at runtime. Items belonging to selected parent and child branches will be displayed with the exception of 'ServerBrowse' which is not associated with equipment.

Refer to Interactive Data Browse List for list display options available at runtime.

## See Also

[Create Pages](#)

## Interactive Data Browse List

The Data Browse template features an interactive list that supports the following functionality:

### Adjustable Columns

The user can adjust the columns displayed on the data browse list. Options include:

- Reposition a column by drag-and-drop its column heading to the new position.
- Adjust the size of a column by dragging the column separator that appears at the right edge of a column heading.

### Selection of data rows

The user can select single or multiple rows on the display list so that the same operation can be applied to multiple selected rows in one go. Row selection works as follows:

- To select a single data browse row: Left-click a row that is not already selected to select it.
- To select a continuous range of data browse rows: Select a single row to mark the start of the selection. Then hold the SHIFT key and left-click another row to extend the selection. Alternatively, you can select rows by dragging the mouse cursor from the starting to the ending row. Once rows are selected, you can extend its range by SHIFT left-clicking another row of alarm.
- To select disjointed rows, hold the CTRL key while selecting a row.
- To deselect alarm(s): Left-click any alarm within a selection range to deselect the whole range. Only the row that you have clicked will remain selected. Click the selected row again to deselect it.

### Context Menu

If the page is set to display tag browse data, right-click on any row will bring up the context menu for operations. Options available in the context menu will vary depending on whether single or multiple items are selected:

The commands will only work if:

- Tag field is included in the table display
- For system with multiple clusters, the cluster field will also need to be included

The items available on the menu are selection sensitive depending on whether single or multiple items are selected at the time. The following options are available only when single row is selected:

- Set Value

---

**Note:** To disable the Set Value command leave the privilege field empty. See [Data Browse Page Template](#) topic for more information.

---

The following options are available:

- Set Override Mode
- Set Control Mode

## Alarm Page Templates

Alarm page templates are included for the following types of alarm displays:

- **Active Alarm Page:** Used to create a page displaying active alarms that are unacknowledged or acknowledged and still in alarm state. The template has the equipment tree-view panel displayed on the left hand side of the page for filtering of the display list. The tree-view will display a hierarchy of equipment available to the currently logged on user. The template also includes vertical and horizontal scroll bars to allow users to view columns and rows off screen. To create a default active alarm page for your project, create a new page based on the **alarm** template. By default, the name of the page is expected to be "Alarm". However, you can change the name of the page by adjusting the setting for the INI parameter [\[Page\]AlarmPage](#).
- **Disabled Alarm Page :** Used to create a page displaying alarms that are presently disabled in the system. The alarms will appear on this page when they have been disabled from the normal alarm system due to a maintenance shutdown, nuisance tripping etc. The template has the equipment tree-view panel displayed on the left hand side of the page for filtering of the display list. The tree-view will display a hierarchy of equipment available to the currently logged on user. To create a default disabled alarm page for your project, create a new page based on the **disabled** template. By default, the name of the page is expected to be "Disabled". However, you can change the name of the page by adjusting the setting for the INI parameter [\[Page\]DisabledPage](#).
- **Sequence of Events Page:** Used to create a page displaying a snapshot of every state transition of an alarm instance that have occurred. The template has the equipment tree-view panel displayed on the left hand side of the page for filtering of the display list. When configured the tree-view will display a hierarchy of equipment available to the currently logged on user. To create a default SOE alarm page for your project, create a new page based on the **soe** template. By default, the name of the page is expected to be "SOE". However, you can change the name of the page by adjusting the setting for the INI parameter, [\[Page\]SOEPage](#). Unlike the active, hardware, and disabled alarm pages, that are refreshed automatically at runtime to display real time data, the SOE and summary pages display a snapshot of events that have occurred, therefore is not updated automatically. To retrieve the latest events you can manually refresh the page.
- **Alarm Summary Page :** Used to create a page displaying a historical log of alarms that have occurred. This page can be used for troubleshooting purposes. The template has the equipment tree-view panel displayed on the left hand side of the page for filtering of the display list. When configured the tree-view will display a hierarchy of equipment available to the currently logged on user. To create a default summary alarm page for your project, create a new page based on the **summary** template. By default, the name of the page is expected to be "Summary". However, you can change the name of the page by adjusting the setting for the INI parameter [\[Page\]SummaryPage](#).



### SUMMARY DATA NOT AUTOMATICALLY REFRESHED

Summary data is not automatically refreshed at runtime. Manually refresh the Summary and SOE page before using its data for analysis of plant state.

**Failure to follow these instructions can result in injury or equipment damage.**

The alarm templates display alarm lists using a "no draw" mode. The display of alarm fields is handled by a new set of template specific Cicode functions and genies to control each of the alarm fields' position and size. The

alarm list is installed with a callback event function to redraw the alarm list upon data change. The "no draw" mode and data change callback event are supported by a modified form of the existing Cicode function [AlarmDsp\(\)](#).

## Alarm Page-specific Menus

If your project is created using SxW starter project, a set of menus are already pre-configured for each of the alarm pages. When the user displays an alarm page, its page specific menu items will be automatically displayed. Options available on the page specific menu vary according to the type of Alarm page.

Name	Description
Acknowledge Page	Acknowledge all alarms on the current page. Available on Active Alarm page.
Disable Page	Disable all alarms on the current page. Available on Active Alarm page.
Enable Page	Enables all alarms on Page. Available on Disabled Alarm Page.
Filter...	Opens the alarm filter form. Filter form varies according to alarm page open.
Reset Filter	Clears the effect of the filter criteria specified through the filter form.
Auto size columns	Auto size column widths in the alarm display list.
Add column...	Add column to the alarm display list.
Save View	Save the column settings of the alarm display. The same settings will then be used for this alarm page.
Restore Saved View	Restore the column settings according to the alarm format set in parameter <a href="#">[Format]FormatName</a> .
Reset view to default	Reset the column settings of alarm display to default. Define your own default by setting parameters <a href="#">[Format]FormatName</a> as project parameters.
Print/Export	Print or Export the current page

## Interactive Alarm List

The alarm page templates feature an interactive alarm list that supports the following functionality:

## Adjustable Columns

The user can adjust the columns displayed on the alarm list. Options include:

- Reposition a column by drag-and-drop its column heading to the new position.
- Delete a column by drag-and-drop its column heading (vertically) beyond the column heading row.
- Insert or delete a column via a pop-up menu by right-clicking a column heading
- Adjust the size of a column by dragging the column separator that appears at the right edge of a column heading.

## Dockable Equipment tree-view

- Adjust the width of the equipment tree-view by dragging the column separator that appears at the right edge of the column. When selected the column separator will change from blue to gray.
- To hide the equipment tree-view click the arrow icon on the column separator. On clicking it the equipment tree-view will collapse and be hidden from view. Click the arrow icon at the edge of the page to show the equipment tree-view on screen.

## Selection of Alarms

The user can select single or multiple alarms on the alarm list so that the same operation can be applied to multiple selected alarms in one go. If some of the selected alarms are off-screen, before the required operation is completed, a message dialog will popup, reminding the user that some of the selected records are not viewable. On clicking Continue or the Enter key the selected records on and off the screen will be acted on. On clicking Cancel or the ESC key the command is canceled. If the selected records are viewable on-screen the message dialog will not open.

Alarm selection works as follows:

- To select a single alarm: Left-click an alarm that is not already selected to select it.
- To select a continuous range of alarms: Select a single alarm to mark the start of the selection. Then hold the SHIFT key and left-click another alarm to extend the selection. Alternatively, you can select rows by dragging the mouse cursor from the starting to the ending row. Once rows are selected, you can extend the range of the selection by SHIFT left-clicking another row of alarm.
- To select disjointed rows, hold the CTRL key while selecting a row.
- To deselect alarm(s): Left-click any alarm within a selection range to deselect the whole range. Only the row that you have clicked will remain selected. Click the selected row again to deselect it.

## Sorting the Alarm List

Users can sort the alarm list display using most fields except for 'State' and 'Type'. To sort click on the header of the relevant column.

**Note:** Alarms on the Alarm Summary Page can be sorted on only on date and time.

The default order of alarms is dependent on the [\[Alarm\]SummarySortMode](#) parameter and the type of alarm page.

Refer to the topic [Alarm List Default Sort Order](#) for more information.

## Context Menu

Right-click on any alarm row to bring up the context menu for alarm operations. Options available in the context menu will vary depending on whether single or multiple items are selected. If you right-click on an alarm that is not already selected, this will select it. Subsequent commands (selected from the context menu) will then only operate on this single alarm. On the other hand, if you right-click on any alarm within a selection range, this will not change the selection. Subsequent commands selected will operate on each alarm within the selection range. Several operation options are available in the context menu:

- Acknowledge: acknowledges selected alarms. This option is restricted by the privilege set by the parameter [\[Privilege\]AckAlarms](#). This is not available on the Alarm Summary page.
- Disable: disables selected alarms. This option is restricted by the privilege set by the parameter [\[Privilege\]DisableAlarms](#). It is not available on the Disabled Alarms page.
- Enable: enables selected alarms. This option is restricted by the privilege set by the parameter [\[Privilege\]DisableAlarms](#). It is not available on the Active Alarms page.
- Comment: Add a comment to an event. This option is only available when a single row is selected. This option is restricted by the privilege set by the parameter [\[Privilege\]AckAlarms](#). This option is available to Alarm Summary and SOE pages only.
- Information: displays the detailed information of a single alarm.
- Help: displays the Help page of a single alarm.

---

**Note:** The Hardware Alarms template only allows a single alarm to be selected. Its context menu has only the Acknowledge option which is also restricted by the privilege set by parameter [\[Privilege\]AckAlarms](#).

---

## Filtering Alarms using the Equipment Panel

You can filter the alarm display list to show alarms belonging to selected parent and child equipment branches. If no equipment is selected in the tree-view then all alarms are displayed.

To select equipment, check the box to the left of the relevant branches. If you select equipment with child branches, then the child equipment will be automatically selected, with their check boxes grayed out.

---

**Note:** If the parent equipment branch is checked, child equipment cannot be deselected.

---

If the parent is not selected you can change the checked states of the child equipment.

Next to each item in the tree, an alarm count is displayed. The count includes the total number of alarms belonging to the equipment and its child hierarchy. The type of alarms that are counted vary according to the type of alarms that are displayed on the list. For example, on the active alarms page, the count represents the active alarms associated to that equipment, while on the disabled alarms page, the count represents the disabled alarms associated to that equipment. Other points to consider with the equipment count include:

1. The count on the tree node represents the total number of alarms that are associated with that equipment hierarchy branch.
2. For alarms that are not associated to equipment, the count is not available due to the nature of the equipment tree view.
3. The counts are not affected by the filter applied to the alarm list.
4. Due to point 2 and 3, the total sum of the counts from the equipment tree does not equal to the alarm count on the alarm banner, and they never mean to be comparable. The alarm count on the banner represents the

total number of (unfiltered) unacknowledged alarms in the system (subjected to the viewable areas of the currently logged on user).

## Process Analyst Page Templates

The Process Analyst template is the default template for you to create trend pages. The SxW Style project includes templates for the three following types of trend display:

- **SinglePA** - A template that displays a single Process Analyst chart on the page. To create a default trend page for your project, please create a new page based on this template and name it "ProcessAnalyst". You can change the default page name by setting parameter [\[Page\]ProcessAnalystPage](#)
- **DoublePA** - A template that displays two Process Analyst charts on the page.
- **PopPA** - A template that displays a single Process Analyst chart on the page. The size of the page is reduced to make it suitable for pop-up pages. To create a default trend pop-up page for your project, please create a new page based on this template and name it "!ProcessAnalystPopup". You can change the default page name by setting parameter [\[Page\]ProcessAnalystPopupPage](#).

The functionality of traditional trends, double trends, and popup trends are largely incorporated into the Process Analyst on the new templates.

There is a key difference between the new PA templates and the existing trend templates, which is pen assignment. The trend templates can get pens assigned to the page at both design time and runtime, but every PA template needs the pens assigned at runtime only. This can be done by calling the [Process Analyst Functions](#) and pass the PAV file or individual trend pens to those functions.

There are two new items on the PA templates:

- The Show/Hide Trend Statistics button shows and hides trend statistics columns in the Process Analyst.
- The Std Deviation column works out the sample standard deviation value of the samples within the display area.

## See Also

[Process Analyst](#)

## SPC Trend Page Templates

The following trend templates are provided for charting different types of statistical process control:

- **SPCPareto**: Displays a Pareto chart for multiple variable tags
- **SPCCpk**: Displays an SPC mean chart and a Capability chart
- **SPCXRSChart**: Displays SPC mean, range and standard deviation charts
- **EventSPCXRS**: Displays SPC mean, range and standard deviation charts for Event SPC trend tag
- **MeanMeanChart**: Displays SPC mean charts for two SPC trend tags
- **RangeChart**: Displays SPC mean and range charts for an SPC trend tag
- **StandardChart**: Displays SPC mean and standard deviation charts for an SPC trend tag

## File Page Templates

The following templates have been included for displaying different types of files:

- File: Displays a plain text file on the page. This template replicates the functionality of the original file template found in the standard Include project. The new template is compatible with any existing `PageFile()` or `DspFile()` Cicode functions. It can display up to 24 lines of text from a file at a time. Buttons are provided for the user to browse to different rows and columns of the file.
- File\_RTF: Displays a rich text format (RTF) file on the page using the CiTextBox ActiveX control. It can also display plain text file.
- File\_HTML: Displays a hypertext markup language (HTML) file or an universal resource locator (URL) link on the page. It needs Microsoft Internet Explorer to be installed on the machine.

Several common features are presented on the templates:

The new file template replicates the functionality of the original file template in the Include project. The new template is compatible with any existing `PageFile()` or `DspFile()` Cicode functions. The HD768 version of the template can display up to 23 lines of text from a file on screen, while the HD1080 version of the template can display up to 33 lines of text on screen. Buttons are provided for the user to browse to different rows and columns of the file.

Several new features are added to the templates:

- When you create a page based on any of these templates in Graphics Builder, you can predefine the file for display at runtime. Clicking anywhere on the page will show a genie style dialog to allow you to enter a file path to assign a file to the page. A Plant SCADA path substitution string can be used in the file path. When the page is recalled at runtime, for example via `PageDisplay()`, the specified file will be automatically displayed.
- The "Open File" button displays an Open File dialog to select a different file to display.
- The "Refresh" button reopens the already opened file to get the latest contents.
- The full path name of the currently opened file is displayed.

## Creating a New Project

Creating a project based on the SxW templates project is simple; by default, it is incorporated as an included project in new projects. This means whenever you start a new project, the SxW templates are ready to use as necessary.

There are two ways to create a project based on the SxW templates:

- Use the predefined starter project
- Create a project from scratch

---

**Note:** It is recommended when creating a project, you base it on one of the starter projects. Offering a basic level of functionality, the project can be modified and extended to suit your needs.

---

## Using the pre-defined Starter Project

### To base a project on an existing starter project:

#### 1. Create a Project Using a Starter Project.

Two pre-defined starter projects with page resolutions HD1080 and HD768 for wide screens only are provided as part of the product installation. Each of the starter projects will be based on project "SxW\_Style\_1" and contain:

- A cluster named "Cluster1".
- A server of each server types: "IOServer1", "AlarmServer1", "ReportServer1" and "TrendServer1".
- A memory I/O devices named "Internal".
- Equipment type "Motor" and its associated template file, graphical genie and super-genie pop-up pages.
- A role named "Administrators" which is linked to the "BUILTIN\Administrators" Windows group and have global privilege of 1 to 8. This allows a user to log in Plant SCADA at runtime using a Windows user account who belongs to the Administrators group of the local machine.
- Pages of Alarm, Summary, Disabled, Hardware, ProcessAnalyst and !ProcessAnalystPopup, Data Browse -Control Inhibit, and Manual Override, based on the relevant templates found in the SxW\_Style\_Include project.
- Page specific menu items. For example, the active alarm page will have page specific menu items such as Simulate, History, Acknowledge and Disable, and Filter options.

The newly created project will be immediately compilable, and will contain a basic level of built-in functions such as viewing alarms and trends, and a menu displayed in the right-panel to easily access and navigate the included pages.

## Create a project from scratch

### To configure a project based on the SxW templates from scratch:

1. Create a privileged user; see [Create a Privileged User](#).
2. Run the Setup Wizard; see [Run the Setup Wizard](#).
3. Set up instant trending; see [Use Instant Trending](#).
4. Implement audible alarms; see [Implement Audible Alarms](#).
5. Add Pages; see [Creating New Pages](#).
6. Create/define custom menu; see [Creating Custom Menus for SxW Templates](#).

Avoid modifications to the SxW template project for use as a runtime project; set it aside for use as a template for new projects.

If creating a project based on the SxW templates, it is not recommended to include pages based on templates that use a different style, including the earlier Include project. Doing so might affect functionality.

## See Also

[Create Pages](#)

## Create a Privileged User

Some SxW template project content is restricted via a user login. Without a valid login, some project functionality will be disabled. For example, the Close window (the cross) button on the title bar will not work if you log in as a user with restricted privileges.

By default, the following elements within the SxW templates project are restricted by global privileges.

Element	Global Privilege	Associated Parameter
Editing users	8	[Privilege]EditUser
Project shutdown	8	[Privilege]Shutdown
Acknowledge alarms	1	[Privilege]AckAlarms
Disable alarms	8	[Privilege]DisableAlarms
Silence alarms	0	[Privilege]SilenceAlarms

When configuring a SxW template project, check that your users have appropriate access to the available functionality. Verify that your users can acknowledge alarms if necessary.

To adjust the global privileges for the elements previously listed for a more complex security architecture, adjust the [Privilege] parameters in the Citect.ini file. Click the relevant link in the previous table above for details.

## See Also

[Run the Setup Wizard](#)

[Privilege Parameters](#)

## Run the Setup Wizard

As with any Plant SCADA project, you need to run the Setup Wizard on any machine where a SxW template project will be run. See [Run the Setup Wizard](#) for more information.

## Use Instant Trending

Instant trending is natively supported by the Process Analyst. The steps to implement instant trending are as follows:

1. **Create a Process Analyst page.** Create a page that has a Process Analyst object on it. You may create these pages based on the provided tab style templates. For a fullscreen page, create the page based on the SinglePA template. For a pop-up page, create the page based on the PopPA template.
2. **Set the sample cache duration.** Samples collected for Instant Trends are kept in the cache up to a default number of samples and time period. You can adjust these settings using the parameters [Trend]InstantTrendSamples and [Trend]InstantTrendIdleTime.
3. **Call the Process Analyst Display.** At runtime, call your process analyst page either as a fullscreen page or

pop-up page using the Cicode function PageDisplay(pagename) or PagePopup(pagename) respectively.

4. **Add Variable Tags / Local Variables for Trending.** Use the Add Pens dialog provided with the Process Analyst to add pens to the process analyst chart as normal. In the dialog, you can now select any variable tags or local variables for trending.

## See Also

[Process Analyst Page Templates](#)

## Implement Audible Alarms

The SxW template project offers support for audible alarms. You can configure a project so that a selected wav file is sounded whenever an alarm of a particular priority is triggered. You can even assign different sounds to different alarm priorities, allowing the urgency of an alarm to be determined from its sound.

### **WARNING**

#### **LOSS OF CONTROL**

- Do not use Plant SCADA's alarming system as the primary alert mechanism for safety-related alarms or notifications (that is, those classified as critical to process safety, the protection of the plant and equipment, or the protection of human life).
- Do not use Plant SCADA's audible alarm functionality as a replacement for safety-related warning systems critical to process safety, the protection of the plant and equipment or the protection of human life.
- Safety-related alarms and notifications should be independent and separate from the process control system. They should be implemented in the form of a stand-alone physical alarm system driving individual discrete alarm annunciators.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

#### **To implement audible alarms in your project:**

Verify that the alarms you want to trigger an audible alert are assigned to a category and given a particular priority.

1. Adjust the parameter **[Alarm]Sound<n>**. This parameter determines the wav file used when an alarm sounds, based on the priority of the alarm (n). For example, [Alarm]Sound1 identifies the .wav file that will be sounded whenever a priority 1 alarm is triggered.
2. If necessary, adjust the parameter **[Alarm]Sound<n>Interval**. This parameter determines how long the selected wav file sounds, based on the priority of the alarm (n). For example, [Alarm]Sound1Interval sets the length of time a priority 1 alarm is sounded for.
3. Verify that your users have appropriate privileges associated with their login to silence alarms, otherwise they will not be able to silence an alarm. See [Creating a Privileged User](#) and the parameter **[Privilege]SilenceAlarms**.

## See Also

[Alarm Page Templates](#)

## Create Pages

If you created your project by selecting the **Create project based on starter project** check box in the **New Project** dialog, pages of Alarm, Summary, SOE, Disabled, Hardware, ProcessAnalyst and !ProcessAnalystPopup will already be created in your new project. You can accept those pages or modify them as you develop your project. See [Creating a New Project](#).

If you created your project from scratch, there are no predefined pages in your project. The navigation controls at runtime on the SxW template expect several default pages to exist in your project for basic functionality such as alarm and trend displays. You are recommended to add the following pages to your project based on the provided templates of your chosen size:

Template	Description
SinglePA or DoublePA	Process Analyst page for displaying trends.
!PopPA	Process Analyst pop-up page.
Data Browse Template	Tag Browse page.
Alarm	Active Alarms page, called from the Alarms toolbar.
Hardware	Hardware Alarms page, called from the Alarms toolbar.
Disabled	Disabled Alarms page, called from the Alarms toolbar.
Summary	Alarms Summary page, called from the Alarms toolbar using the Historical events icon.
SOE (Sequence of Events)	Sequence of events page, called from the Alarms toolbar using the Historical events icon.

The links between the pages listed and the controls that call them are defined by the [Page] parameter settings within the Citect.ini file. The relevant parameters are [\[Page\]ProcessAnalystPage](#), [\[Page\]ProcessAnalystPopupPage](#), [\[Page\]AlarmPage](#) and [\[Page\]HardwarePage](#), etc. If you want a button to call a page with a different name, adjust these parameter settings. For details see [Page Parameters](#).

## Creating New Pages

The normal template in the SxW template library is designed with minimal content to enable the presentation of customer-necessary information and plant mimics.

Pages are created based on this template within Graphics Builder by choosing the necessary template from the Use Template dialog (**File | New Page**). You can access the SxW templates by selecting **SxW\_style\_1** from the **Style** field.

## Creating Custom Menus for SxW Templates

The content of the menus on pages based on the SxW templates are configured using the **Visualization** activity in Plant SCADA Studio (see [Menu Configuration](#)).

**Note:** If your project is based on a SxW starter project, a basic pre-configured menu exists. The menu allows you to access the page templates included in the starter project and is displayed in the right-panel. Page specific menu items are displayed under the active page list. Options available vary according to the type of page.

The following examples demonstrate how you can add additional pages to the SxW menus.

### Example 1

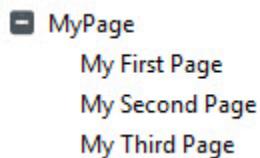
In this example, the pages MyFirstPage, MySecondPage and MyThirdPage will be added to the menu so they will be listed in the right-hand panel.

**To configure this custom menu:**

1. In the **Visualization** activity, select **Menu Configuration**.
2. Add a row to the Grid Editor.
3. In the row complete the fields in the table below.
4. Click **Save**
5. Repeat steps 1 to 4 for the remaining pages MySecondPage, and MyThirdPage.
6. When finished compile and run your project.

Field	Value
Level 1	MyPage
Level 2	MyFirstPage
Menu Command	PageDisplay("Name of Page") e.g PageDisplay("MyFirstPage")

The right-panel should look like the one below.



### Example 2

To separate the alarm pages from the other graphic pages, this example demonstrates how to create a new node, MyAlarm Pages, with MyActiveAlarmPage, MyDisabledPage, and MySOEPage.

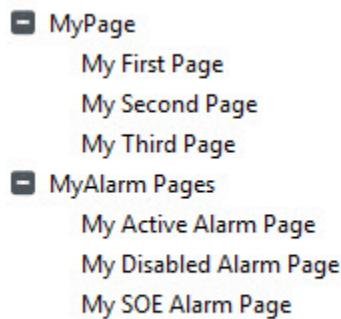
**To configure this custom menu:**

1. In the **Visualization** activity, select **Menu Configuration**.

2. Add a row to the Grid Editor.
3. In the row complete the fields in the table below.
4. Click **Save**.
5. Repeat steps 1 to 4 for the remaining pages MyDisabledPage, and MySOEPage.
6. When finished compile and run your project.

Field	Value
Level 1	MyAlarm Pages
Level 2	My Active Alarm Page
Menu Command	PageDisplay("Name of Page") e.g PageDisplay("MyActiveAlarmPage")

The right-panel should look like the one below:



### Example 3

In the next example, an icon will be added to appear next to the Active Alarm Page. The icon used is a standard symbol in the Include project.

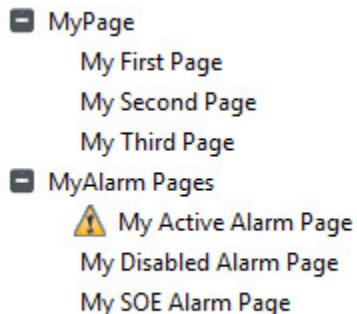
#### To configure this custom menu:

1. In the **Visualization** activity, select **Menu Configuration**.
2. Add a row to the Grid Editor.
3. In the row complete the fields in the table below.
4. When finished compile and run your project.

Field	Value
Level 1	MyAlarm Pages
Level 2	My Active Alarm Page
Menu Command	PageDisplay("Name of Page") e.g

Field	Value
	PageDisplay("MyActiveAlarmPage")
Symbol	icons_16x16.alam_act

The right-panel should look like the one below:




---

**Note:** The above examples are for basic illustration only. It is recommended you refer to the example project and the menu configuration properties prior to configuring your own menus.

---

## See Also

[Menu Configuration](#)

## Using Environment Variables in SxW Templates

Whereas parameters apply specified rules to every template page, environment variables apply rules to *specific* pages. In this way, you can use environment variables to specify functionality for specific pages.

For example, to display a specific page, specify the page name as the environment variable. To call a specific Cicode function, specify the function name (with a "?" prefix), space, then a list of comma separated arguments, like this: ?WinPrint LPT1,0,0,Trend.pa1. (The "?" indicates that the variable is to be interpreted as a function call rather than a page to display.)

Let's look at a more detailed example. Usually the standard **Print** button is used for printing a graphics page in WYSIWYG format. If you have a text/html file displayed on the page, or perhaps a trend, you can create your own print function (for example, PrintTrend) and specify it in the environment variable, as ?PrintTrend. To reduce the amount of ink used, specify the WinPrint function with a user-defined palette. In this case you would specify it as a parameter rather than as an environment variable so that it applies to every page.

---

**Note:** For pages created based on the Process Analyst templates, you can print the screen using the standard Print button, or print the chart using the Print button on the toolbar of the Process Analyst object. Give the user both alternatives as they both serve different purposes.

---

## Tab Style Templates Project

The Tab Style Templates project (named "Tab\_Style\_Include") is a system project that includes a set of templates

that use a tabbed ribbon menu.

When a new Plant SCADA project is created, the Tab\_Style\_Include project is automatically incorporated as an included project. This means the project's templates are available for implementation when creating your graphics pages in Graphics Builder.

As well as including a graphics page template for creating plant mimics, the project includes predefined Process Analyst and alarm templates (with or without the equipment tree-view panel), a set of file templates for displaying text, Rich Text Format and HTML files.

Each type of page type has the same tab-styled navigation and alarm menus, providing a consistent "look and feel" across an entire project. The project also supports multi-monitor displays, allowing you to simultaneously display several graphics pages across several computer screens.

---

**Note:** Avoid making modifications to the Tab Style Template project for use as a runtime project. A Plant SCADA upgrade will install a new version of the project, which will overwrite any changes you make to the project.

---

Use the following links for details about the pre-configured content included in the Tab Style Template project:

- Predefined templates (see [Use Pages and Templates](#)).
- Common toolbars (see [Common Navigation Functionality](#)).

Use the following links for details about the processes used to create a project based on Tab Style templates:

- Creating a new project (see [Create a New Project](#)).
- Creating pages (see [Create Pages](#)).
- Creating custom menus (see [Create Custom Tab Style Menus](#)).

Reference material is also included, describing the Citect.ini parameters and Cicode functions available to customize a project. See [Tab Style Template Reference](#).

## Use Pages and Templates

When you create a new project based on the Tab Style templates, the following templates are available for you to use as you create your graphics pages. These templates share [Common Navigation Functionality](#), such as a customizable tabbed menu, a navigation toolbar and an alarm toolbar. This section describes the templates and their buttons, menus and tools.

The Tab Style template project has the following templates:

### Normal

A template designed for creating user-necessary content and plant mimics. This page contains the custom tabbed menus, the standard navigation and alarm toolbars featured on tab style templates. If you are creating a project based on the tab style templates, use this template for your standard graphics pages.

The Normal template creates a page in display format to be shown with a title bar. Several sizes of templates are available including XGA (1024 x 768), SXGA (1280 x 1024), HD1080 (1920 x 1080) and WUXGA (1920 x 1200).

---

**Note:** Starting in version 7.20, you can customize the behavior of the standard window titlebar to lock down its functions to achieve the same result that used to be only possible by hiding it in the previous versions. This is achieved by [overriding the titlebar events](#) or setting the startup mode of the window.

---

## Blank

The blank template is included for users who want to create their pages from scratch. There are no differences between the Tab Style blank template and the template in the include project.

## Data Browse

The data browse template is included for users who want to create a page that displays a table of data retrieved from built-in data browse functions. The template includes the Equipment tree library control.

When using this template to create a new page within your project, configure the Data Browse pop up form. When configured the new page will display the appropriate data session at runtime.

Field	Description
Equipment Symbol Library (Optional)	The symbol library for displaying equipment symbol for each item in the equipment tree. The symbol is displayed based on the type of the equipment. It should have the same name as the equipment type assigned to an equipment record. If not specified, it defaults to a symbol library named "Equipment".
Equipment Default Symbol (Optional)	The default symbol to be displayed next to the item in the equipment tree if symbol corresponding to the equipment type of the item cannot be found from the supplied Equipment Symbol Library. The symbol should be specified in format of <library name>.<symbol name>
Browse Type	The type of project configuration to be browsed and displayed, including: AccumBrowse – for listing accumulators AlmBrowse – for listing alarm tags EquipBrowse - for listing equipment ServerBrowse – for listing servers TagBrowse – for listing variable tags TrnBrowse – for listing trend tags
Clusters (Optional)	A comma delimited string specifying only items from a subset of clusters to be included in the display. If not specified, items from all clusters will be included.
Filter (Optional)	A filter expression specifying the records to return during the browse. If not specified, all records will be returned. Please see the browse open Cicode function of the selected browse type for the details of filter expression supported.
Fields (Optional)	A comma delimited string specifying the fields (columns) to be returned during the browse. If it is not

Field	Description
	specified, all fields of the selected browse type will be displayed: Tag, Tag Item, Value, Comment, Cluster, Equipment, Item, Type, IODev, Addr, Arr_Size, Eng_Zero, Eng_Full, Eng_Units, Format, Deadband, Raw_Zero, Raw_Full, Scaled_Type, Ovr_Mode, Ctrl_Mode, Override, Valid, Status, Field, Num_Subs.
Sort By (Optional)	A comma delimited string specifying the order of sorting preferences. This is only applicable to TagBrowse. Please see the <i>Sort</i> parameter of the built-in Cicode function <a href="#">TagBrowseOpen</a> for details.
View Name (Optional)	<p>A name to represent a view setting about the placement and width of the columns displayed on the browse table. The view name works with the Save view and Restored saved view options at runtime. You can assign a unique view on different data browse pages or share the same view among multiple pages. When user selects the Save view command from the Action tab, the current column settings will be saved under the view name. Once a view is saved, when user re-enters the page, previous saved view settings will be retrieved automatically. The saved view can also be restored on demand by selecting the Restored saved view from the Action tab. User can reset (forget) the view to the default as how it was pre-configured by selecting the Restore view to default command under the Action tab.</p> <p>For system engineers, the information associated with a view name is saved under INI parameters:  [BrowseTableView] &lt;BrowseType&gt;.&lt;ViewName&gt;.Fields  &lt;BrowseType&gt;.&lt;ViewName&gt;.ColWidths  in comma delimited format. You can pre-configure these parameters in your project as the defaults.</p>
Privilege (Optional)	Only applicable to Browse Type of "TagBrowse", this specifies the privilege required by the logged on user to change the values / operating mode of variable tags displayed on the table. If no privilege is defined, no modification will be allowed.

Use the equipment tree to filter the data browse table at runtime. Items belonging to selected parent and child branches will be displayed with the exception of 'ServerBrowse' which is not associated with equipment.

Refer to Interactive Data Browse List for list display options available at runtime.

## Alarm

Templates are included for the following types of alarm displays:

- **Active Alarm Page (Alarm):** Used to create a page displaying active alarms that are unacknowledged or acknowledged and still in alarm state. To create a default active alarm page for your project, create a new page based on this template. By default, the name of the page is expected to be "Alarm". However, you can change the name of the page by adjusting the setting for the INI parameter [\[Page\]AlarmPage](#).
- **Hardware Alarm Page (Hardware):** Used to create a page displaying details of any system errors that are unacknowledged or acknowledged and still in alarm state, for example, if communications are lost, if Cicode can't execute, if a graphics page is not updating correctly, or if a server becomes inoperative, this page will alert you. To create a default hardware alarm page for your project, create a new page based on this template. By default, the name of the page is expected to be "Hardware". However, you can change the name of the page by adjusting the setting for the INI parameter [\[Page\]HardwarePage](#).
- **Disabled Alarm Page (Disabled):** Used to create a page displaying alarms that are presently disabled in the system. The alarms will appear on this page when they have been disabled from the normal alarm system due to a maintenance shutdown, nuisance tripping etc. To create a default disabled alarm page for your project, create a new page based on this template. By default, the name of the page is expected to be "Disabled". However, you can change the name of the page by adjusting the setting for the INI parameter [\[Page\]DisabledPage](#).
- **Sequence of Events Page (SOE):** Used to create a page displaying a snapshot of all state transitions of an alarm instance that have occurred. To create a default SOE page for your project, create a new page based on this template. By default, the name of the page is expected to be "SOE". However, you can change the name of the page by adjusting the setting for the INI parameter [\[Page\]SOEPage](#). Unlike the active, hardware, disabled pages that are refreshed automatically at runtime to display real time data, the SOE and Summary pages display a snapshot of events that have occurred, therefore is not updated automatically. To retrieve the latest events you can manually refresh the page.
- **Alarm Summary Page (Summary):** Used to create a page displaying a historical log of alarms that have occurred. This page can be used for troubleshooting purposes. To create a default alarm summary page for your project, create a new page based on this template. By default, the name of the page is expected to be "Summary". However, you can change the name of the page by adjusting the setting for the INI parameter [\[Page\]SummaryPage](#).

### CAUTION

#### CAUTION

Summary data is not automatically refreshed at runtime. Manually refresh the Summary page before using its data for analysis of plant state.

The alarm templates (except the hardware alarm template) display alarm lists using a "no draw" mode. The display of alarm fields is handled by a new set of template specific Cicode functions and genies to control each of the alarm fields' position and size. The alarm list is installed with a callback event function to redraw the alarm list upon data change. The "no draw" mode and data change callback event are supported by a modified form of the existing Cicode function [AlarmDsp\(\)](#).

## Process Analyst

The Process Analyst template is the default template for you to create trend pages. The Tab Style project includes templates for the three following types of trend display:

- **SinglePA** - A template that displays a single Process Analyst chart on the page. To create a default trend page for your project, please create a new page based on this template and name it "ProcessAnalyst". You can change the default page name by setting parameter [\[Page\]ProcessAnalystPage](#).
- **DoublePA** - A template that displays two Process Analyst charts on the page.
- **PopPA** - A template that displays a single Process Analyst chart on the page. The size of the page is reduced to make it suitable for pop-up pages. To create a default trend pop-up page for your project, please create a new page based on this template and name it "!ProcessAnalystPopup". You can change the default page name by setting parameter [\[Page\]ProcessAnalystPopupPage](#).

The functionality of traditional trends, double trends, and popup trends are largely incorporated into the Process Analyst on the new templates.

There is a key difference between the new PA templates and the existing trend templates, which is pen assignment. The trend templates can get pens assigned to the page at both design time and runtime, but every PA template needs the pens assigned at runtime only. This can be done by calling the Process Analyst Cicode Functions and pass the PAV file or individual trend pens to those functions.

There are two new items on the PA templates:

- The Show/Hide Trend Statistics button shows and hides trend statistics columns in the Process Analyst.
- The Std Deviation column works out the sample standard deviation value of the samples within the display area.

## SPC

The Tab Style project includes the following trend templates for different types of statistical process control:

- **SPCPareto**: Displays a Pareto chart for multiple variable tags
- **SPCCpk**: Displays an SPC mean chart and a Capability chart
- **SPCXRSChart**: Displays SPC mean, range and standard deviation charts
- **EventSPCXRS**: Displays SPC mean, range and standard deviation charts for Event SPC trend tag
- **MeanMeanChart**: Displays SPC mean charts for two SPC trend tags
- **RangeChart**: Displays SPC mean and range charts for an SPC trend tag
- **StandardChart**: Displays SPC mean and standard deviation charts for an SPC trend tag

## File

The tab style project includes the following templates for displaying different types of file:

- File: Displays a plain text file on the page. This template replicates the functionality of the original file template found in the standard Include project. The new template is compatible with any existing PageFile() or DspFile() Cicode functions. It can display up to 24 lines of text from a file at a time. Buttons are provided

for the user to browse to different rows and columns of the file.

- File\_RTF: Displays a rich text format (RTF) file on the page using the CiTextBox ActiveX control. It can also display plain text file.
- File\_HTML: Displays a hypertext markup language (HTML) file or an universal resource locator (URL) link on the page. It needs Microsoft Edge to be installed on the machine.

Several common features are presented on the templates:

The new file templates replicate the functionality of the original file template in the Include project. The new templates are compatible with any existing PageFile() or DspFile() Cicode functions. They can display up to 24 lines of text from a file at a time. Buttons are provided for the user to browse to different rows and columns of the file.

Several new features are added to the templates:

- When you create a page based on any of these templates in Graphics Builder, you can predefine the file for display at runtime. Clicking anywhere on the page will show a genie style dialog to allow you to enter a file path to assign a file to the page. A Plant SCADA path substitution string can be used in the file path. When the page is recalled at runtime, for example via PageDisplay(), the specified file will be automatically displayed.
- The "Open File" button displays an Open File dialog to select a different file to display.
- The "Refresh" button reopens the already opened file to get the latest contents.
- The full path name of the currently opened file is displayed.

## See Also

[Create a New Project](#)

[Create Pages](#)

## Common Navigation Functionality

The tab style page templates include common toolbars for easy navigation and feature access, as well as a consistent appearance. This helps ensure pages that are created based on these templates will look and operate in the same style.

The following three toolbars remain on screen during operation:

### Custom tabbed menus toolbar

The tab style templates provide a tabbed toolbar to present the custom menus defined in the menu configuration database. The tabbed menu toolbar consists of following components:

- **Tab Buttons**

Tab buttons are equivalent of selecting a menu option in a traditional menu system. Each tab contains a list of menu items (represented as icons) relating to that tab.

- **Menu Icons (Buttons)**

Each tab contains a row of menu icons, each of which either performs a function such as calling up a page or

running a Cicode expression.

- **Sub-menu Drop Down Lists**

Optionally, a drop-down list may be provided next to the menu icon for further options. You can also view it as a sub-menu to the top-level tab. In this case, the menu icon serves as a collection point for related sub-menu items.

- **Hidden Menu Items**

Where there are too many menu items to fit comfortably within the menu bar (either for the tabs or the menu icons), a pair of drop-down list buttons are supplied at either end of the menu bar which provide access to the menu items. The horizontal arrow buttons on the menu bar displays only those items not visible on either side of the screen. The vertical arrow button on the left side of the menu bar displays menu items (either tabs or the menu icons for the currently selected tab).

## Navigation toolbar

The Navigation toolbar includes clickable text and icons to quickly access the common pages and functions from every page in the project.

The following items are available from the toolbar:

Icon	Name	Description
	Goes back one page in the page history	Goes back to the page displayed before the current page.
	Goes forward one page in the page history	Goes forward to the page that was displayed prior to the back button being pressed. The arrow to the right of the button allows you to select from menu of recently visited pages.
	Displays the parent page	Changes the display to the "parent page" of the current page. You can assign a parent page to a graphics page by setting an environment variable in Graphics Builder. To do this, open the page you want to assign a parent to. Go to the properties dialog for the page ( <b>File   Properties</b> ), and click the <b>Environment</b> tab. Add a new variable called "ParentPage" with a value of the page name of the parent page.
	Displays the previous page	Moves backwards through pages in a browse sequence if a sequence is configured. To configure a browse sequence, go to the Page

Icon	Name	Description
		Properties dialog ( <b>File   Properties</b> ) for each page in the sequence and set the <b>Previous</b> field on the <b>General</b> tab accordingly.
	Displays the next page	Moves forwards through pages in a browse sequence if a sequence is configured. To configure a browse sequence, go to the Page Properties dialog ( <b>File   Properties</b> ) for each page in the sequence and set the <b>Next</b> field on the <b>General</b> tab accordingly.
	Displays the home page	Displays the "home" page. A home page act as a default page for the project. To set a home page, use the [Page]HomePage parameter.
	Displays the Page Select dialog	Displays a dialog box with a list of graphics pages defined in the project. The user can select a page name for display.
	Prints the current page	Prints the current graphic page to the selected printer using the built-in WinPrint() Cicode function. You can specify your own printing function for the page by setting an environment variable in Graphics Builder. To do this, open the page you want to assign a printer function to. Go to the properties dialog for the page ( <b>File   Properties</b> ), and click the Environment tab. Add a new variable called "PrintPage" with a value of Cicode command in the syntax of "?" followed by a valid Cicode function name and comma separated arguments, for example, ?MyPrintFunction "arg1","arg2".
	Login / Logout	Displays the standard Plant SCADA login prompt. The menu to the right offers extra options. The default options are Login, Logout,

Icon	Name	Description
		Change Password, Edit User (restricted by login) and Create User (restricted by login). Logged in user is indicated to the right of the button. You can override the default options by defining your own login menu in the Visualization activityCommand Bar option <b>menu configuration</b> . The Page field and Level 1 field of these menu items needs to be set to "Template" and "Login" respectively.

## Alarm toolbar

The alarms toolbar provides access to current alarm information and navigation buttons for single-click access to alarm pages. It also includes standard Plant SCADA prompt and current date and time.

The alarms toolbar contains the following features:

Icon	Name	Description
n/a	<b>Last Alarms</b>	The center panel of the toolbar displays the last alarms triggered within the system, providing the operator with an immediate visual cue to alarm occurrences.  You can modify the format of this list and control which alarms appear. For example, you can set the list to only display alarms of a particular type, category or priority.  For details, see the alarm parameters LastAlarmFmt, LastAlarmCategories, LastAlarmPriorities, and LastAlarmType .
	<b>Active Alarms</b>	Changes the display to the active alarms page. This button is animated and shows the count of unacknowledged alarms in the system. The page displayed by this button is determined by the parameter [Page]AlarmPage.

Icon	Name	Description
	<b>Historical Events</b>	<p>The arrow to the right allows you to change the display to the sequence of events or alarm summary page. When clicking on the historical events button again the sequence of events page will display by default. However, if there is no SOE page configured in the project, the alarm summary page will be displayed.</p> <p><b>Sequence of events</b> - Select the option Sequence of Events. The SOE page displays each state transition of an alarm instance as a separate row.</p> <p>This page is determined by the <a href="#">[Page]SOEPAGE</a>.</p> <p><b>Alarm Summary</b> - The alarm summary page provides an historical log of alarm occurrences.</p> <p>This page is determined by the <a href="#">[Page]SummaryPage</a>.</p>
	<b>Hardware Alarms</b>	Changes the display to the hardware alarms page. This is an animated button which will blink when an unacknowledged hardware alarm occurs. The page displayed by this button is determined by the parameter <a href="#">[Page]HardwarePage</a> .
	<b>Disabled Alarms</b>	Changes the display to the disabled alarms page. This button shows the count of disabled alarms in the system. The page displayed by this button is determined by the parameter <a href="#">[Page]DisabledPage</a> .
	<b>Alarm Silence</b>	Silences the audible alarm buzzer that sounds when there are pending unacknowledged alarms. See <a href="#">Implement Audible Alarms</a> .

## Common Alarm Page Template Functionality

The alarm page templates offered as part of the tab style page templates include common graphical user interface (GUI) features for easy operation and consistent appearance.

These include:

### Task Panels

The alarm page templates share panels to the left of the alarms list that support the following functionality:

- **Page Tasks**

Allows the user to navigate through the list of alarms a page at a time. The Print / Export hyperlink (not available in the hardware alarm page) allows the user to send the alarms on the list to a printer or a file. The output will be presented in a HTML table, and reflects the columns, filter and sort settings that are currently used on the display. There are options to output available alarms (starting from page 1) or specific pages of alarms starting from the current page of the alarm list. The blue box indicates the current page number of the alarm list.

- **Filter Tasks**

Allows the user to filter the current list based on date time, alarm name, plant area, alarm category and other criteria. The Reset filter hyperlink clears the effect of the filter criteria specified through the filter form. The blue box indicates the current filter setting. It may display one of the following states:

- No Filter - no filter is applied
- Filter Applied - filter criteria specified in the filter form is applied
- Custom Filter - filter criteria is applied via Cicode function AlarmSetInfo by the user.

- **Control Tasks**

Offers the option to:

- Acknowledge alarms on the current page (only available on Active Alarms and Hardware Alarms page)
- Disable alarms on the current page (only available on Active Alarms page)
- Enable alarms on the current page (only available on Disabled Alarms page)
- Silence alarm sound (not available on Hardware Alarms page)

This panel is not featured on the Alarm Summary page.

- **View Tasks**

Offers the option to:

- Add new column to the alarm list.
- Automatically fit the width of columns to show the most content.
- Reload the column settings according to the alarm format set in parameter [Format]FormatName.
- Save the current column settings to parameter [Format]FormatName, so that the same settings are used when this page is displayed in the future.

This panel is not featured on the Hardware Alarm page.

## Interactive Alarm List

The alarm and alarm equipment tree view page templates (except the Hardware Alarms template) feature an interactive alarm list that supports the following functionality:

- **Adjustable Columns**

The user can adjust the columns displayed on the alarm list. Options include:

- Reposition a column by drag-and-drop its column heading to the new position.
- Delete a column by drag-and-drop its column heading (vertically) beyond the column heading row.
- Insert or delete a column via a pop-up menu by right-clicking a column heading
- Adjust the size of a column by dragging the column separator that appears at the right edge of a column heading.

- **Selection of Alarms**

The user can select single or range of alarms on the alarm list so that the same operation can be applied to multiple selected alarms in one go. Alarm selection works as follows:

- To select a single alarm: Left-click an alarm that is not already selected to select it.
- To select a range of alarms: Select a single alarm to mark the start of the selection. Then hold the SHIFT key and left-click another alarm to extend the selection. You can adjust the range of the selection by SHIFT left-clicking another row of alarm.
- To deselect alarm(s): Left-click any alarm within a selection range to deselect the whole range.

- **Context Menu**

Right-click on any alarm row to bring up the context menu for alarm operations. Options available in the context menu will vary depending on whether single or multiple items are selected. If you right-click on an alarm that is not already selected, this will select it. Subsequent commands (selected from the context menu) will then only operate on this single alarm. On the other hand, if you right-click on any alarm within a selection range, this will not change the selection. Subsequent commands selected will operate on each alarm within the selection range. Several operation options are available in the context menu:

- Information: displays the detailed information of a single alarm that has the cursor focus (indicated by surrounded by a white box).
- Help: displays the Help page of a single alarm that has the cursor focus.
- Acknowledge: acknowledges selected alarms. This option is restricted by the privilege set by the parameter [\[Privilege\]AckAlarms](#). This is not available on the Alarm Summary page.
- Disable: disables selected alarms. This option is restricted by the privilege set by the parameter [\[Privilege\]DisableAlarms](#). It is not available on the Disabled Alarms page.
- Enable: enables selected alarms. This option is restricted by the privilege set by the parameter [\[Privilege\]DisableAlarms](#) It is not available on the Active Alarms page.
- Comment: Add a comment to an event. This option is only available when a single row is selected. This option is restricted by the privilege set by the parameter [\[Privilege\]AckAlarms](#). This option is available to Alarm Summary and SOE pages only.

**Note:** The Hardware Alarms template only allows a single alarm to be selected. Its context menu has only the Acknowledge option which is also restricted by the privilege set by parameter [\[Privilege\]AckAlarms](#).

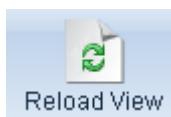
## Common Data Browse Template Functionality

The data browse page template offered as part of the tab style page templates include common graphical user interface (GUI) features for easy operation and consistent appearance.

These include:

### Data Browse Action Tab

When the user displays a Data Browse page based on the Data Browse template, an Action tab will be automatically added to the menu bar. The following options are available on the Action tab:

Icon	Name	Description
 Reload View	Refresh	Reload the column settings.
 Auto-f...	Auto-Fit Columns	Auto size column widths in the data browse list.
 Save View	Save View	Has the following options: Save View - Current column settings will be saved. When you re-enter the page, previous saved view settings will be retrieved automatically. Restore Saved View - Restores the saved view on demand. Reset View to Default - Reset the view to the default value.

### Interactive Data Browse List

The Data Browse template features an interactive list that supports the following functionality:

#### Adjustable Columns

The user can adjust the columns displayed on the data browse list. Options include:

- Reposition a column by drag-and-drop its column heading to the new position.
- Adjust the size of a column by dragging the column separator that appears at the right edge of a column heading.

#### Selection of data rows

The user can select single or multiple rows on the display list so that the same operation can be applied to multiple selected rows in one go. Row selection works as follows:

- To select a single data browse row: Left-click a row that is not already selected to select it.
- To select a continuous range of data browse rows: Select a single row to mark the start of the selection. Then hold the SHIFT key and left-click another row to extend the selection. Alternatively, you can select rows by dragging the mouse cursor from the starting to the ending row. Once rows are selected, you can extend its range by SHIFT left-clicking another row of alarm.
- To select disjointed rows, hold the CTRL key while selecting a row.
- To deselect alarm(s): Left-click any alarm within a selection range to deselect the whole range. Only the row that you have clicked will remain selected. Click the selected row again to deselect it.

### Context Menu

If the page is set to display tag browse data, right-click on any row will bring up the context menu for operations. Options available in the context menu will vary depending on whether single or multiple items are selected:

The commands will only work if:

- Tag field is included in the table display
- For system with multiple clusters, the cluster field will also need to be included

The items available on the menu are selection sensitive depending on whether single or multiple items are selected at the time. The following options are available only when single row is selected:

- Set Value

The following options are available:

- Set Override Mode
- Set Control Mode

## Alarm Equipment Tree-view Page Templates

Alarm page templates are included for the following types of alarm displays:

- **Active Alarm Page (`alarm_equip`):** Used to create a page displaying active alarms that are unacknowledged or acknowledged and still in alarm state. The template has the equipment tree-view panel displayed on the left hand side of the page. When configured the tree-view will display a hierarchy of equipment available to the currently logged on user. The template also includes vertical and horizontal scroll bars to allow users to view columns and rows off screen. To create a default active alarm page for your project, create a new page based on the `alarm_equip` template. By default, the name of the page is expected to be "Alarm". However, you can change the name of the page by adjusting the setting for the INI parameter [\[Page\]AlarmPage](#).
- **Disabled Alarm Page (`disabled_equip`):** Used to create a page displaying alarms that are presently disabled in the system. The alarms will appear on this page when they have been disabled from the normal alarm system due to a maintenance shutdown, nuisance tripping etc. The template has the equipment tree-view panel displayed on the left hand side of the page. When configured the tree-view will display a hierarchy of equipment available to the currently logged on user. To create a default disabled alarm page for your project, create a new page based on the `disabled_equip` template. By default, the name of the page is expected to be "Disabled". However, you can change the name of the page by adjusting the setting for the INI parameter [\[Page\]DisabledPage](#).

- **Sequence of Events Page (soe\_equip):** Used to create a page displaying a snapshot of every state transition of an alarm instance that have occurred. The template has the equipment tree-view panel displayed on the left hand side of the page. When configured the tree-view will display a hierarchy of equipment available to the currently logged on user. To create a default SOE alarm page for your project, create a new page based on the **soe\_equip** template. By default, the name of the page is expected to be "SOE". However, you can change the name of the page by adjusting the setting for the INI parameter [\[Page\]SOEPage](#). Unlike the active, hardware, and disabled alarm pages, that are refreshed automatically at runtime to display real time data, the SOE and summary pages display a snapshot of events that have occurred, therefore is not updated automatically. To retrieve the latest events you can manually refresh the page.
- **Alarm Summary Page (summary\_equip):** Used to create a page displaying a historical log of alarms that have occurred. This page can be used for troubleshooting purposes. The template has the equipment tree-view panel displayed on the left hand side of the page. When configured the tree-view will display a hierarchy of equipment available to the currently logged on user. To create a default summary alarm page for your project, create a new page based on the **summary\_equip** template. By default, the name of the page is expected to be "Summary". However, you can change the name of the page by adjusting the setting for the INI parameter [\[Page\]SummaryPage](#).

## CAUTION

### SUMMARY DATA NOT AUTOMATICALLY REFRESHED

Summary data is not automatically refreshed at runtime. Manually refresh the Summary page before using its data for analysis of plant state.

**Failure to follow these instructions can result in injury or equipment damage.**

The alarm equipment tree-view templates display alarm lists using a "no draw" mode. The display of alarm fields is handled by a new set of template specific Cicode functions and genies to control each of the alarm fields' position and size. The alarm list is installed with a callback event function to redraw the alarm list upon data change. The "no draw" mode and data change callback event are supported by a modified form of the existing Cicode function [AlarmDsp\(\)](#).

## Common Functionality

The alarm equipment tree-view page templates offered as part of the tab style page templates include common graphical user interface (GUI) features for easy operation and consistent appearance.

These include:

- Action Tab - Provides controls to carry out common tasks that apply to the whole of the alarm list.
- Interactive Alarm List - Provides controls to manipulate the columns displayed on the alarm list and select alarms.
- Equipment Tree-view filter - The tree-view can be used to filter the results of the list of alarms displayed.
- Column Sorting and Filtering Status Indicator - Sorting and filtering status indicator shown in header of column.

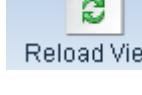
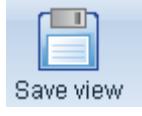
## See Also

[Action Tab](#)

[Interactive Alarm List](#)[Filtering Alarms Using the Equipment Tree-view](#)[Column Sorting and Filter Status Indicator](#)

## Action Tab

When the user displays an alarm page based on the alarm equipment tree-view templates, an Action tab will be automatically added to the menu bar. Options available on the Action tab vary according to the type of Alarm page.

Icon	Name	Description
 Ackn...	Acknowledge Page	Acknowledge all alarms on the current page. Available on Active Alarm page.
 Disab...	Disable Page	Disable all alarms on the current page. Available on Active Alarm page.
 Silence	Silence Alarm	Silence alarm sound. Available on Active Alarm page.
 Enable page	Enable Page	Enables all alarms on Page. Available on Disabled Alarm Page.
 Filter...	Set Filter	Opens the alarm filter form. Filter form varies according to alarm page open.
 Reset ...	Reset filter	Clears the effect of the filter criteria specified through the filter form.
 Reload View	Reload View	Reload the column settings according to the alarm format set in parameter [Format]FormatName.
 Save view	Save View	Save changes to the alarm display

Icon	Name	Description
	Add column	Add column to the alarm display list.
	Auto size columns	Auto size column widths in the alarm display list.
	Print/Export	Print or Export the current page

### Interactive Alarm List

The alarm equipment tree view page templates feature an interactive alarm list that supports the following functionality:

### Adjustable Columns

The user can adjust the columns displayed on the alarm list. Options include:

- Reposition a column by drag-and-drop its column heading to the new position.
- Delete a column by drag-and-drop its column heading (vertically) beyond the column heading row.
- Insert or delete a column via a pop-up menu by right-clicking a column heading
- Adjust the size of a column by dragging the column separator that appears at the right edge of a column heading.

### Dockable Equipment tree-view

- Adjust the width of the equipment tree-view by dragging the column separator that appears at the right edge of the column. When selected the column separator will change from blue to gray.
- To hide the equipment tree-view click the arrow icon on the column separator. On clicking it the equipment tree-view will collapse and be hidden from view. Click the arrow icon at the edge of the page to show the equipment tree-view on screen.

### Selection of Alarms

The user can select single or multiple alarms on the alarm list so that the same operation can be applied to multiple selected alarms in one go. If some of the selected alarms are off-screen, before the required operation is completed, a message dialog will popup, reminding the user that some of the selected records are not viewable. On clicking Continue or the Enter key the selected records on and off the screen will be acted on. On clicking Cancel or the ESC key the command is canceled. If the selected records are viewable on-screen the message dialog will not open.

Alarm selection works as follows:

- To select a single alarm: Left-click an alarm that is not already selected to select it.
- To select a continuous range of alarms: Select a single alarm to mark the start of the selection. Then hold the SHIFT key and left-click another alarm to extend the selection. Alternatively, you can select rows by dragging the mouse cursor from the starting to the ending row. Once rows are selected, you can extend the range of the selection by SHIFT left-clicking another row of alarm.
- To select disjointed rows, hold the CTRL key while selecting a row.
- To deselect alarm(s): Left-click any alarm within a selection range to deselect the whole range. Only the row that you have clicked will remain selected. Click the selected row again to deselect it.

## Sorting the Alarm List

Users can sort the alarm list display using most fields except for 'State' and 'Type'. To sort click on the header of the relevant column.

The default order of alarms is dependent on the [\[Alarm\]SummarySortMode](#) parameter and the type of alarm page.

Refer to the topic [Default Sort Order](#) for more information.

## Context Menu

Right-click on any alarm row to bring up the context menu for alarm operations. Options available in the context menu will vary depending on whether single or multiple items are selected. If you right-click on an alarm that is not already selected, this will select it. Subsequent commands (selected from the context menu) will then only operate on this single alarm. On the other hand, if you right-click on any alarm within a selection range, this will not change the selection. Subsequent commands selected will operate on each alarm within the selection range. Several operation options are available in the context menu:

- Information: displays the detailed information of a single alarm that has the cursor focus (indicated by surrounded by a white box).
- Help: displays the Help page of a single alarm that has the cursor focus.
- Acknowledge: acknowledges selected alarms. This option is restricted by the privilege set by the parameter [\[Privilege\]AckAlarms](#). This is not available on the Alarm Summary page.
- Disable: disables selected alarms. This option is restricted by the privilege set by the parameter [\[Privilege\]DisableAlarms](#). It is not available on the Disabled Alarms page.
- Enable: enables selected alarms. This option is restricted by the privilege set by the parameter [\[Privilege\]DisableAlarms](#). It is not available on the Active Alarms page.
- Comment: Add a comment to an event. This option is only available when a single row is selected. This option is restricted by the privilege set by the parameter [\[Privilege\]AckAlarms](#). This option is available to Alarm Summary and SOE pages only.

**Note:** The Hardware Alarms template only allows a single alarm to be selected. Its context menu has only the Acknowledge option which is also restricted by the privilege set by the parameter [\[Privilege\]AckAlarms](#).

## Column Sorting and Filter Status Indicator

When you specify a filter or sort columns via the provided UI the status is shown in the header of the columns. Whether you select equipment in the tree view panel, specify filters via the filter form or click on the column header to sort a column, one of the following symbols will appear on the filtered/sorted column(s).

Symbol	Description
	No sorting or filtering is applied
	Sorted in ascending order
	Sorted in descending order
	Filtered without explicit sorting order
	Filtered and sorted in ascending order
	Filtered and sorted in descending order

If an equivalent field to the column is being filtered the appropriate filter symbol will appear in the heading of the column. For example, date and time should be filtered together, thus when the date field is specified in the alarm filter form, the filter symbol will appear in the time column.

## Filtering Alarms Using the Equipment Tree-view

For alarm pages with the equipment tree view pane, you can filter the alarm display list to show alarms belonging to selected parent and child equipment branches. If no equipment is selected in the tree-view then all alarms are displayed.

To select equipment, check the box to the left of the relevant branches. If you select equipment with child branches, then the child equipment will be selected and grayed out.

**Note:** If the parent equipment branch is checked, child equipment cannot be deselected.

If the parent is not selected, you can change the checked states of the child equipment.

Next to each item in the tree, an alarm count is displayed. The count includes the total number of alarms belonging to the equipment and its child hierarchy. The type of alarms that are counted vary according to the type of alarms that are displayed on the list. For example, on the active alarms page, the count represents the active alarms associated to that equipment, while on the disabled alarms page, the count represents the disabled alarms associated to that equipment. Other points to consider with the equipment count include:

1. The count on the tree node represents the total number of alarms that are associated with that equipment.
2. For alarms that are not associated to equipment, the count is not available due to the nature of the equipment tree view.
3. The counts are not affected by the filter applied to the alarm list.
4. Due to point 2 and 3, the total sum of the counts from the equipment tree does not equal to the alarm count

on the alarm banner, and they never mean to be comparable. The alarm count on the banner represents the total number of (unfiltered) unacknowledged alarms in the system (subjected to the viewable areas of the currently logged on user).

## Create a New Project

Creating a project based on the Tab Style templates project is simple; by default, it is incorporated as an included project in new projects. This means whenever you start a new project, the tab style templates are ready to use as necessary.

There are two ways to create a project based on the tab style templates:

- Use the pre-defined starter project.
- Create a project from scratch.

### Using the Pre-defined Starter Project

**To base a project on an existing starter project:**

1. Choose **New Project** from the **File** menu, or click the **New Project** button.
2. Type a name for your project and choose a location for the files. Using spaces in the project name is not recommended. Although Plant SCADA does not stop you from using spaces, it can cause unpredictable results.
3. Click the **Create project based on starter project** check box.
4. Choose the project on which you want to base your new project.
5. Click **OK**.

Four pre-defined starter projects with page resolutions of XGA, SXGA, WUXGA and HD1080 (Full HD) are provided as part of the product installation. Each of the starter projects will be based on project "Tab\_Style\_1" and contain:

- A cluster named "Cluster1".
- A role named "Administrators" which is linked to the "BUILTIN\Administrators" Windows group and have global privilege of 8. This allow a user to log in Plant SCADA at runtime using a Windows user account who belongs to the Administrators group of the local machine.
- Pages of Alarm, Summary, Disabled, Hardware, ProcessAnalyst and !ProcessAnalystPopup based on the relevant templates found in the Tab\_Style\_Include project.

The newly created project will be immediately compilable, and will contain a basic level of built-in functions such as viewing alarms and trends.

## Create a Project from Scratch

**To configure a project based on the tab style templates from scratch:**

1. Create a privileged user; see [Create a Privileged User](#).

2. Run the Setup Wizard; see [Run the Setup Wizard](#).
3. Set up instant trending; see [Use Instant Trending](#).
4. Set up multiple monitor display; see [Display a Project on Multiple Monitors](#).
5. Implement audible alarms; see [Implement Audible Alarms](#).

Don't modify the tab style templates project for use as a runtime project; set it aside for use as a template for new projects.

If creating a project based on the tab style templates, it is not recommended to include pages based on templates that use a different style, including the earlier Include project. Doing so might affect functionality.

## See Also

[Create Pages](#)

## Create a Privileged User

Some tab style templates project content is restricted via a user login. Without a valid login, some project functionality will be disabled. For example, the Close window (the cross) button on title bar will not work if you log in as a user with restricted privileges.

By default, the following elements within the tab style templates project are restricted by global privileges.

Element	Global Privilege	Associated Parameter
Editing users	8	<a href="#">[Privilege]EditUser</a>
Project shutdown	8	<a href="#">[Privilege]Shutdown</a>
Acknowledge alarms	1	<a href="#">[Privilege]AckAlarms</a>
Disable alarms	8	<a href="#">[Privilege]DisableAlarms</a>
Silence alarms	0	<a href="#">[Privilege]SilenceAlarms</a>

When configuring a tab style templates project, check that your users have appropriate access to the available functionality. Verify that your users can acknowledge alarms if necessary.

To adjust the global privileges for the elements previously listed for a more complex security architecture, adjust the [Privilege] parameters in the Citect.ini file. Click the relevant link in the previous table above for details.

## See Also

[Run the Setup Wizard](#)

[Privilege Parameters](#)

## Run the Setup Wizard

As with any Plant SCADA project, you need to fill in the Setup Wizard on any machine where a tab style

templates project will be run. See [Run the Setup Wizard](#) for more information.

How you set up a computer using the Wizard is mostly up to you; however, due to some necessary settings, you need to run the Setup Wizard as a custom setup. The Wizard page you need to pay attention to is the Control Menu Page.

## Security Setup - Control Menu Page

All tab templates are designed to be displayed with the title bar. You need to select the "Show Title Bar" option when you select to display the page in fullscreen mode.

Window title bar controls can be secured from unauthorized access. See [Secure the Windows Title Bar](#).

## See Also

[Create a Privileged User](#)

[Use Instant Trending](#)

[Implement Audible Alarms](#)

## Use Instant Trending

Instant trending is natively supported by the Process Analyst. The steps to implement instant trending are as follows:

1. **Create a Process Analyst page.** Create a page that has a Process Analyst object on it. You may create these pages based on the provided tab style templates. For a fullscreen page, create the page based on the SinglePA template. For a pop-up page, create the page based on the PopPA template.
2. **Set the sample cache duration.** Samples collected for Instant Trends are kept in the cache up to a default number of samples and time period. You can adjust these settings using the parameters [Trend]InstantTrendSamples and [Trend]InstantTrendIdleTime.
3. **Call the Process Analyst Display.** At runtime, call your process analyst page either as a fullscreen page or pop-up page using the Cicode function PageDisplay(pagename) or PagePopup(pagename) respectively.
4. **Add Variable Tags / Local Variables for Trending.** Use the Add Pens dialog provided with the Process Analyst to add pens to the process analyst chart as normal. In the dialog, you can now select any variable tags or local variables for trending.

## See Also

[Use Pages and Templates](#)

## Display a Project on Multiple Monitors

For projects based on the tab style templates, multiple monitors will be supported through the core product. You can display a Plant SCADA window on each of the monitors on the machine by adjusting the following parameters:

- [\[MultiMonitors\]Monitors](#) - Adjust this parameter to indicate how many monitors your project will be running

on.

- [\[MultiMonitors\]StartupPage<n>](#) - This parameter determines which page will appear on each monitor at startup. You have to duplicate this parameter for each monitor. For example, StartupPage1 determines the page that appears on the first monitor at startup, StartupPage2 determines what appears on the second monitor, and so on. If a startup page is not defined for a particular monitor, the page specified in parameter [\[Page\]Startup](#) will be displayed.

## See Also

[MultiMonitors Parameters](#)

## Implement Audible Alarms

The tab style templates project offers support for audible alarms. You can configure a project so that a selected wav file is sounded whenever an alarm of a particular priority is triggered. You can even assign different sounds to different alarm priorities, allowing the urgency of an alarm to be determined from its sound.



### LOSS OF CONTROL

- Do not use Plant SCADA's alarming system as the primary alert mechanism for safety-related alarms or notifications (that is, those classified as critical to process safety, the protection of the plant and equipment, or the protection of human life).
- Do not use Plant SCADA's audible alarm functionality as a replacement for safety-related warning systems critical to process safety, the protection of the plant and equipment or the protection of human life.
- Safety-related alarms and notifications should be independent and separate from the process control system. They should be implemented in the form of a stand-alone physical alarm system driving individual discrete alarm annunciators.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

### To implement audible alarms in your project:

1. Verify that the alarms you want to trigger an audible alert are assigned to a category and given a particular priority.
2. Adjust the parameter [\[Alarm\]Soundn](#). This parameter determines the wav file used when an alarm sounds, based on the priority of the alarm (n). For example, [Alarm]Sound1 identifies the .wav file that will be sounded whenever a priority 1 alarm is triggered.
3. If necessary, adjust the parameter [\[Alarm\]SoundnInterval](#). This parameter determines how long the selected wav file sounds, based on the priority of the alarm (n). For example, [Alarm]Sound1Interval sets the length of time a priority 1 alarm is sounded for.
4. Verify that your users have appropriate privileges associated with their login to silence alarms, otherwise they will not be able to silence an alarm. See [Create a Privileged User](#) and the parameter [\[Privilege\]SilenceAlarms](#).

## See Also

[Use Pages and Templates](#)

## Create Pages

If you created your project by selecting the **Create project based on starter project** check box in the **New Project** dialog, Alarm, SOE, Disabled, Hardware, ProcessAnalyst, !ProcessAnalystPopup, ControllInhibit, ManualOverride, VariableTags and generic equipment pop-up pages would already have been created in your new project. You can accept those pages or modify them as you develop your project. See [Create a New Project](#).

If you created your project from scratch, there are no predefined pages in your project. The navigation controls on the tab template expect several default pages to exist in your project for basic functionality such as alarm and trend displays. You are recommended to add the following pages to your project based on the provided templates of your chosen size:

Template	Description
SinglePA or DoublePA	Process Analyst page for displaying trends.
!PopPA	Process Analyst pop-up page.
Alarm	Active Alarms page, called from the Alarms toolbar.
Hardware	Hardware Alarms page, called from the Alarms toolbar.
Disabled	Disabled Alarms page, called from the Alarms toolbar.
Summary	Alarms Summary page, called from the Alarms toolbar using the Historical events icon.
SOE (Sequence of Events)	Sequence of events page, called from the Alarms toolbar using the Historical events icon.

The links between the pages listed and the controls that call them are defined by the [Page] parameter settings within the Citect.ini file. The relevant parameters are [Page]ProcessAnalystPage, [Page]ProcessAnalystPopupPage, [Page]AlarmPage and [Page]HardwarePage, etc. If you want a button to call a page with a different name, adjust these parameter settings. For details see [Page Parameters](#).

## Creating new tab pages using the normal template

The normal template in the tab style template library is designed with minimal content to enable the presentation of customer-necessary information and plant mimics.

Pages are created based on this template within Graphics Builder by choosing the necessary template from the Use Template dialog (**File | New Page**). You can access the tab style templates by selecting **tab\_style\_1** from the **Style** field.

## Create Custom Tab Style Menus

The tabbed menus that appear on pages created by the Tab Style Starter Project are configured using the **Visualization** activity in Plant SCADA Studio (see [Menu Configuration](#)).

You can choose to use the default menus offered by the Tab Style templates; they can be turned on or off via the parameter [\[Page\]AddDefaultMenu](#).

Items in the default menus are assigned a sorting order of 100. When you configure to show the default menus and your own defined menus at the same time, the two will be merged and ordered accordingly.

The default menus contain the following items:

Tabs	Buttons
Pages	Automatically populate the non-system pages configured in the project and included project(s)
Alarms	Active Alarms
	Historical events (Alarm summary and SOE pages)
	Disabled Alarms
	Hardware Alarms
Trends	Process Analyst Calls Cicode function: PageProcessAnalyst("", "")
	Process Analyst Popup Calls Cicode function: ProcessAnalystPopup("", "")

**Note:** If you need to migrate from CSV\_Include project to the tab style templates, you can use the Project Migration Tool to migrate the CSV\_Include menu definition to the Menu Configuration database.

## See Also

[Configure Tab Style Popup Menus](#)

## Configure Tab Style Popup Menus

The Tab style templates menu bar features two pop up menus:

- **Login** dropdown - displays a drop-down menu with login options.

You can override the default menu by specifying your own menu item in the [Menu Configuration](#) view of the Visualization activity. The override menu item needs the **Page** field set to "Template". It also needs to have the **Level 1** field set to "Login".

- **Alarm silence** option button - clicking this button displays a context menu for selecting different silence options.

This button is only enabled if you have sufficient privileges (as specified in the parameter [\[Privilege\]SilenceAlarms](#)).

You can override the default menu by specifying your own menu item in the [Menu Configuration](#) view of the Visualization activity. The override menu item needs the **Page** field set to "Template". It also needs to have the **Level 1** field set to "Alarm Sound".

## See Also

[Create Custom Tab Style Menus](#)

## Using Environment Variables in Tab Style Templates

Whereas parameters apply specified rules to every template page, environment variables apply rules to *specific* pages. In this way, you can use environment variables to specify functionality for specific pages.

For example, to display a specific page, you'd specify the page name as the environment variable. To call a specific Cicode function, you'd specify the function name (with a "?" prefix), space, then a list of comma separated arguments, like this: ?WinPrint LPT1,0,0,Trend.pal. (The "?" indicates that the variable is to be interpreted as a function call rather than a page to display.)

Let's look at a more detailed example: usually the standard **Print** button is used for printing a graphics page in WYSIWYG format. If you have a text/html file displayed on the page, or perhaps a trend, you can create your own print function (for example, PrintTrend) and specify it in the environment variable, as ?PrintTrend. To reduce the amount of ink used, specify the WinPrint function with a user-defined palette. In this case you would specify it as a parameter rather than as an environment variable so that it applies to every page.

---

**Note:** For pages created based on the Process Analyst templates, you can print the screen using the standard Print button, or print the chart using the Print button on the toolbar of the Process Analyst object. Give the user both alternatives as they both serve different purposes.

---

## Tab Style Template Reference

Plant SCADA includes a set of parameters that support the Tab Style Templates. These include:

- Alarm parameters
- Alarm heading parameters
- Format parameters
- Page parameters
- Privilege parameters
- Custom alarm parameters
- Custom menu parameters.

For a full list of these parameters, see [Tab\\_Style Template Parameters](#) in the Parameters documentation.

There is also a set of Cicode functions that only apply to the Tab Style Templates. See [Tab Style Template Functions](#).

## Tab Style Template Functions

The list below contains Cicode functions that only apply to the Tab Style Templates:

### Alarm Functions

- [TabAlarm\\_GetAn](#)
- [TabAlarm\\_GetAckPriv](#)
- [TabAlarm\\_GetDisablePriv](#)

### Alarm Sound Functions

- [TabAlarmSnd\\_GetPriv](#)
- [TabAlarmSnd\\_SoundState](#)
- [TabAlarmSnd\\_Silence](#)
- [TabAlarmSnd\\_SilenceOnAck](#)
- [TabAlarmSnd\\_ShowContextMenu](#)

### Display Functions

- [DspFileGetAn](#)
- [DspRtfFileGetAn](#)
- [DspRtfFileGetName](#)
- [DspRtfFileSetName](#)

### Page Functions

- [PageDspConfigMenu](#)
  - [PageDspLoginMenu](#)
  - [PageGetConfigMenuHnd](#)
  - [PageHomeGetName](#)
  - [PageParent](#)
  - [PageParentGetName](#)
1. [PagePrint](#)
  2. [PagePrintGetName](#)

### **TabAlarm\_GetAn**

Gets the Animation Number (AN) of the alarm list on the provided alarm templates. The returned AN can be passed to the alarm family of Cicode functions such as `AlarmSetInfo()`, `AlarmGetInfo()` etc. to further manipulate the alarm list.

---

**Note:** The alarm list displayed by the alarm templates is instantiated using the new noDraw mode. This helps eliminate the need to have consecutive ANs for each row of the alarm records displayed on screen. Therefore, this alarm list is not compatible with any alarm Cicode functions that operates on current cursor position, such as

---

mode 0 of AlarmGetInfo(), AlarmSetInfo(), etc.

---

## Syntax

**TabAlarm\_GetAn(*iListID*)**

*iListID*

The list number of the alarm list used by the alarm templates:

0 - last alarm list (at alarm banner)

1 - main alarm list (only exist on alarm templates)

## Return Value

The AN of the alarm list, otherwise -1 is returned.

## See Also

[Tab Style Template Functions](#)

## TabAlarm\_GetAckPriv

Checks if currently logged on user has privilege to acknowledge alarms from the controls on the provided alarm templates.

---

**Note:** The privilege to acknowledge alarms is defined under INI / project parameter [\[Privilege\]AckAlarms](#).

---

## Syntax

**TabAlarm\_GetAckPriv()**

## Return Value

1 if the logged on user has privilege, otherwise 0 is returned.

## See Also

[Tab Style Template Functions](#)

## TabAlarm\_GetDisablePriv

Checks if currently logged on user has privilege to disable / enable alarms from the controls on the provided alarm templates.

---

**Note:** The privilege to disable / enable alarms is defined under INI / project parameter [\[Privilege\]DisableAlarms](#).

---

## Syntax

`TabAlarm_GetDisablePriv()`

## Return Value

1 if the logged on user has privilege, otherwise 0 is returned.

## See Also

[Tab Style Template Functions](#)

## TabAlarmSnd\_GetPriv

Checks if currently logged on user has privilege to change alarm sound settings.

**Note:** The privilege to change alarm sound settings is defined under INI / project parameter [\[Privilege\]SilenceAlarms](#).

## Syntax

`TabAlarmSnd_GetPriv()`

## Return Value

1 if the logged on user has privilege and alarm sound is available, otherwise 0 is returned.

## See Also

[Tab Style Template Functions](#)

## TabAlarmSnd\_SoundState

Gets the state of alarm sound.

## Syntax

`TabAlarmSnd_SoundState()`

## Return Value

A number that represents the state of alarm sound: 0 - No audible alarms exist; 1 - Audible alarms exist; 2 - Alarm sound is being silenced.

## See Also

[Tab Style Template Functions](#)

### TabAlarmSnd\_Silence

Silences audible alarms for a specified time period in seconds.

## Syntax

**TabAlarmSnd\_Silence([iSilenceTime])**

*iSilenceTime*

Number of seconds to silence audible alarms. If not specified, defaults to -1

-1 - silence audible alarms until every alarm has been acknowledged

0 - re-enable audible alarms

>0 - silence alarms for the specified number of seconds

## Return Value

Zero (0) if function is executed successfully, otherwise error code is returned.

## See Also

[Tab Style Template Functions](#)

### TabAlarmSnd\_SilenceOnAck

Silences alarm sound upon alarm acknowledgement. Call this function when you wish to integrate your alarm acknowledgement Cicode with the alarm sound.

---

**Note:** The silence period can be adjusted by INI / project parameter [\[Alarm\]SoundSilenceTimeoutOnAck](#).

Because the alarm silence is a result of alarm acknowledgment, the privilege to perform this function is controlled by the INI / project parameter [\[Privilege\]AckAlarms](#) instead of [\[Privilege\]SilenceAlarms](#).

---

## Syntax

**TabAlarmSnd\_SilenceOnAck()**

## Return Value

Zero (0) if function is executed successfully, otherwise error code is returned.

## See Also

[Tab Style Template Functions](#)

## **TabAlarmSnd\_ShowContextMenu**

Pops up the alarm sound context menu. The menu is expected to be defined in the Menu Configuration database under the page name of "Template" and Level 1 field of "Alarm Sound". If no menu is defined, a default menu will be shown.

## **Syntax**

**TabAlarmSnd\_ShowContextMenu()**

## **Return Value**

Zero (0) if function is executed successfully, otherwise error code is returned.

## **See Also**

[Tab Style Template Functions](#)

## **DspFileGetAn**

Gets the Animation Number (AN) on the provided file template where the text file is displayed. The returned AN can be passed to the DspFile family of Cicode functions such as DspFileGetInfo(), DspFileGetName(), DspFileSetName etc. to further manipulate the text file.

## **Syntax**

**DspFileGetAn([iAN])**

*iAN*

The AN where text file is displayed. Only specify this argument if you would like to override the value maintained by the template. Default value -1.

## **Return Value**

The AN where the text file is displayed, otherwise -1 is returned.

## **See Also**

[Tab Style Template Functions](#)

## **DspRtfFileGetAn**

Gets the Animation Number (AN) of the CiTextBox ActiveX control that displays rich text file / text file on the page based on the file\_rtf template.

## Syntax

**DspRtfFileGetAn([iAN])**

*iAN*

The AN where the CiTextBox ActiveX control is located. Only specify this argument if you would like to override the value maintained by the template. Default value -1.

## Return Value

The AN where the CiTextBox ActiveX control is located, otherwise -1 is returned.

## Related Functions

[DspRtfFileGetName](#), [DspRtfFileSetName](#)

## See Also

[Tab Style Template Functions](#)

## DspRtfFileGetName

Returns the name of the rich text / text file that is being displayed on the page based on the file\_rtf template.

## Syntax

**DspRtfFileGetName([iAN])**

*iAN*

The AN where the CiTextBox ActiveX control is located. Only specify this argument if you would like to override the value maintained by the template. Default value -1.

## Return Value

The full path file name of the file is being displayed, otherwise an empty string is returned.

## Related Functions

[DspRtfFileGetAn](#), [DspRtfFileSetName](#)

## See Also

[Tab Style Template Functions](#)



0 - no scrollbars

1 - horizontal scroll bar only

2 - vertical scroll bar only

You can use the combination of the above modes by adding the modes together.

## Return Value

Zero (0) if run successfully, otherwise error code is returned.

## Related Functions

[DspRtfFileGetAn](#), [DspRtfFileGetName](#)

## See Also

[Tab Style Template Functions](#)

## PageDspConfigMenu

Displays a pop-up menu. The contents of the menu are defined in the Menu configuration under the specified page name and menu name (Level 1) field.

---

**Note:** If localization syntax, that is, @ (<Name>) is used in the page name or menu name field of your menu configuration record. You need to pass the exact string to this function to retrieve the menu.

---

## Syntax

**PageDspConfigMenu(*sPageName*, *sMenuName* [, *iXPos* [, *iYPos*]])**

*sPageName*

The page name of the menu to be found in the menu configuration database

*sMenuName*

The menu name (Level 1) to be found in the menu configuration database

*iXPos*

The x position (relative to origin of the page) when the menu pops up. Do not specify or set this argument to -1 to show menu at mouse pointer.

*iYPos*

The y position (relative to origin of the page) when the menu pops up. Do not specify or set this argument to -1 to show menu at mouse pointer.

## Return Value

Zero (0) if function is executed successfully, otherwise error code is returned.

## See Also

[Tab Style Template Functions](#)

### PageDspLoginMenu

This Cicode function is associated with the Normal template. Pops up the login context menu. The menu is expected to be defined in the Menu Configuration database under the page name of "Template" and menu name (Level 1) field of Login. If no menu is defined, a default menu will be shown.

## Syntax

**PageDspLoginMenu()**

## Return Value

Zero (0) if function is executed successfully, otherwise error code is returned.

## See Also

[Tab Style Template Functions](#)

### PageGetConfigMenuHnd

Gets the menu configuration node handle for the specified page name and menu (Level 1) name. The returned handle can be passed to the menu family of Cicode functions such as MenuGetFirstChild(), MenuGetNextChild(), MenuNodeGetProperty(), etc. to browse the contents of the menu.

## Syntax

**PageGetConfigMenuHnd(*sPageName*, *sMenuName* [, *iIgnoreEmpty*])**

*sPageName*

The page name of the menu to be found in the menu configuration database

*sMenuName*

The menu name (Level 1) to be found in the menu configuration database

*iIgnoreEmpty*

Indicates whether to treat empty menu node as invalid (1), default is false (0)

## Return Value

The menu handle of the specified menu (Level 1) node, otherwise -1 is returned.

## Related Functions

[PageDspConfigMenu](#)

## See Also

[Tab Style Template Functions](#)

### PageHomeGetName

Returns the name of the home page of the running project.

## Syntax

`PageHomeGetName()`

## Return Value

Page name of the home page.

## See Also

[Tab Style Template Functions](#)

### PageParent

Displays the parent page of the currently displayed page based on page environment variable "ParentPage".

## Syntax

`PageParent()`

## Return Value

Zero (0) if function is executed successfully, otherwise error code is returned.

## Related Functions

[PageParentGetName](#)

## See Also

[Tab Style Template Functions](#)

## PageParentGetName

Returns the name of the parent page of the currently displayed page based on page environment variable "ParentPage".

### Syntax

**PageParentGetName()**

### Return Value

Page name of the parent page.

### Related Functions

[PageParent](#)

### See Also

[Tab Style Template Functions](#)

## PagePrint

Execute a custom Cicode function to print the currently displayed page. The custom Cicode function is either defined in page environment variable "PrintPage" or Citect.INI / project parameter [\[Page\]PrintPage](#). If custom Cicode function is not defined, a Select Printer dialog will pop-up and a screenshot of the currently displayed page will be sent to the selected printer. The print command is defined in the format of: ?Cicode\_fn comma\_separated\_argument\_list e.g. ?WinPrint LPT1;,0,0

### Syntax

**PagePrint()**

### Return Value

Zero (0) if function is executed successfully, otherwise error code is returned.

### Related Functions

[PagePrintGetName](#)

### See Also

[Tab Style Template Functions](#)

## PagePrintGetName

This Cicode function is associated with the Normal template. Returns the page name / Cicode command for printing the currently displayed page based on page environment variable PrintPage. If the environment variable is not defined, it returns the value of INI / project parameter [\[Page\]PrintPage](#).

## Syntax

**PagePrintGetName()**

## Return Value

Page name / Cicode command for printing the currently displayed page.

## Related Functions

[PagePrint](#)

## See Also

[Tab Style Template Functions](#)

# Library Controls Include Project

The Library Control Include project contains a set of Genies for common user interface controls such as an equipment tree view, data table, and vertical and horizontal scroll bars.

**Note:** The Library Control Include project is a system include project, and only appears in the Project list tree if the **System Projects** filter is applied in the Projects activity.

## Library Control Genies

Genie	Description	When to use
<a href="#">Alarm Table Library Control</a>	A version of the Table genie for displaying an alarm list.	Place on page to display an alarm list.
<a href="#">Calendar Library Control</a>	Displays a monthly calendar view.	Place on page to show calendar information or allow user to select a particular date for input.
<a href="#">Data Browse Table Library Control</a>	A version of the Table genie to display snapshot information retrieved via the built-in data browse family of Cicode functions, such as AlmBrowse, TagBrowse and	Place on page to display a snapshot of data retrieved from the built-in data browse functions.

Genie	Description	When to use
	TrnBrowse or other data browse-like data by providing custom initialization callback function to the genie.	
Equipment Tree Library Control	<p>A version of the Tree genie for displaying a hierarchy of equipment configured in the project. The EquipTree genie automatically loads the equipment defined in the project.</p> <p>For advanced usage, such as doing custom actions upon different events, you will need to write your own Cicode functions and assign them to the callback functions part of the genie form.</p>	Place on page to display an equipment tree.
Scroll Bar Horizontal Library Control	Displays a horizontal scroll bar for panning the displayed content horizontally to reveal the off-screen portion of the content. This genie can be used with any content by hooking up the call-back events of the genies with the appropriate Cicode functions (written by the users).	Place on page to view the off screen content
Scroll Bar Vertical Library Control	Displays a vertical scroll bar for panning the displayed content vertically to reveal the off-screen portion of the content. This genie can be used with any content by hooking up the call-back events of the genies with the appropriate Cicode functions (written by the users).	Place on page to view the off screen content
Slider Library Control	Displays a Slider for hiding and displaying the equipment tree-view panel on screen.	Place on page to hide the equipment tree-view panel
SQL Table Library Control	A version of the BrowseTable genie to display information retrieved using the built-in SQL functions as a disconnected recordset.	Place on page to display data retrieved from an SQL data source.

Genie	Description	When to use
Tab	<p>Displays a list of opened pages uniquely in the Activity Bar.</p> <p>See <a href="#">Use the Tabs Genie</a> for examples in how the control can be used.</p>	Place on page to display a list of opened pages that will allow users to navigate between pages easily.
Table Row Library Control	<p>Displays background highlight for a particular row when it is selected. This genie is to be used in conjunction with the Table genies. However, its use is optional.</p> <p>The position and size of the highlight is not automatically calculated at runtime. Locate and re-size the instance of this genie in the Graphics Builder at design time.</p>	Use when want to highlight the selected row with a blue background color.
Table Library Control	<p>Displays multiple rows of data on the screen in tabular layout. The genie is designed to display only a certain number of rows and columns of data on screen. You can view the off-screen portion of the table by using it with the scroll bar genies or via specific Cicode functions. This genie does not include the row highlight animation for row selection. However, you can find out which rows are selected via a set of specific Cicode functions.</p>	Place on page to display table like data such as alarms, or database records.
Tag Table Library Control	<p>A version of the Table genie for displaying a table of variable tags added to the genie using the LibTagTable_AddTag() Cicode function.</p> <p>See <a href="#">Using the Tag Table Genie</a> for examples in how the control can be used.</p>	Place on page to display a table of variable tags.
Tree Library Control	<p>Displays a collection of items in a parent-child hierarchical relationship. Displays only a certain number of items on screen. You can view the off-screen rows and</p>	Place on page to display hierarchical data, such as an equipment menu.

Genie	Description	When to use
	<p>columns of the table by associating it with the scroll bar genies or via specific Cicode functions.</p> <p>See <a href="#">Use the Tree Genie</a> for examples in how the control can be used.</p>	

## See Also

[Use the Library Controls](#)

## Use the Library Controls

### To use a Genie from the Library\_Controls include project:

1. In Graphics Builder, open the page to which you would like to add a Genie.
2. Click the Paste Genie button in the objects toolbox, or select **Paste Genie** from the **Edit** menu.
3. In the Paste Genie dialog, select the "lib\_controls" library.
4. Select the required Genie from the **Genie** list and click **OK**. You can also double-click the thumbnail of the Genie.

The Genie is pasted on the graphics page. A dialog will open prompting you to configure the properties of the Genie.

5. Click **Save**.

At runtime, the dialog values you entered into the prompt will replace each substitution in the Genie.

## See Also

[Library Controls Include Project](#)

[Genies](#)

## Alarm Table Library Control

Alarm table genie has the following parameters:

Parameter	Description
<b>Alarm Table Name</b>	Name of table (max characters). Used to identify the genie when on a graphics page.
<b>Width</b>	Pixel width of the table at runtime. <b>Note:</b> Leave the size of the genie at design time as 100

Parameter	Description
	pixels, as it will be automatically calculated at runtime.
<b>Number Of Rows</b>	Number of rows displayed on the page.
<b>Header Height</b>	Pixel Height of header row in table display.
<b>Row Height</b>	Pixel height of row in the table display.
<b>Header Padding</b>	The pixel padding between the bottom of the header text (column name) and the bottom of the header row.
<b>Row Padding</b>	The pixel padding between the bottom of the cell text and the bottom of each row.
<b>Column Padding</b>	The pixel padding on the left-hand-side and right-hand-side of the cell text relative to the width of each column.
<b>Refresh Rate</b>	The minimum (quickest possible) rate that the control will be redrawn.  The control is only redrawn as required upon changes to the layout, scrolling or other user initiated events.
<b>Scan Delay (Optional)</b>	The number of page scan delay before the genie is initialized at runtime. this can be set to different number on different control genies to stagger the initialization order. If not set defaults to 1.
<b>Last Alarm List</b>	True/False drop down box. Determines whether the alarm list is a normal list (which will be instantiated via the function AlarmDsp) or a last alarm list (which will be instantiated via AlarmDspLast).
<b>Horizontal Scrollbar (Optional)</b>	The name of the horizontal scroll bar genie on the page that allows the user to horizontally scroll the content displayed on the control genie If you do not require horizontal scrolling, leave this field blank.
<b>Vertical Scrollbar (Optional)</b>	The name of the horizontal scroll bar genie on the page that allows the user to vertically scroll the content displayed on the control genie If you do not require vertical scrolling, leave this field blank.
<b>Header Font</b>	The pre-configured font for header text .
<b>Auto Width</b>	True/False drop down box. False is selected by default.
<b>Format Category</b>	The alarm category that is used to determine the

Parameter	Description
	fields (columns) displayed. If this is not specified, the columns displayed on the table will be determined by the field format defined in alarm category 0.
<b>Cluster</b>	The cluster of the alarm list If this is not specified, alarms from all clusters configured in the project will be displayed.
<b>Allow Selection</b>	True/False drop down. True is selected by default.
<b>Alarm Type</b>	<p>The display type of the alarm list. See the <b>Type</b> parameter for the Cicode function <b>AlarmDsp</b> for details of the values you can use and what each represents.</p> <p><b>Note:</b> If you set the alarm type to a hardware alarm (types 5 - 9), the following limitations will apply:</p> <ul style="list-style-type: none"> <li>• The format used to display the hardware alarms will be defined by the <b>Alarm Format</b> property for alarm category 255.</li> <li>• The <b>Cluster</b> field is not used for hardware alarms.</li> <li>• You may need to specify off-screen animation numbers (ANs) for each table row. If you are using a Plant SCADA template project (SxW or Tab Style templates), a set of ANs starting at 21 is already reserved for this purpose. If you are not using a template project, you will need to manually add these ANs.</li> </ul>
<b>AN</b>	The Animation Number (AN) of a user created Cicode / Animation Number object where the alarm list will be instantiated (displayed). If this is less than 1, the alarm list will be displayed at a random AN. The AN allows the user to interact with the alarm list at runtime using built-in Cicode functions such as <b>AlarmSetInfo()</b> , <b>AlarmDspPrev()</b> and <b>AlarmDspNext()</b> .
<b>Initialize (Optional)</b>	<p>Callback function that is called when the control initializes. The callback function should return either TRUE (1) for success or FALSE (0) if unsuccessful. By default, if the user specified function does not initialize, the control genie will retry the function a few times before giving up. This callback only supports the #Name keyword which represents the name assigned to the control genie.</p> <p><b>Note:</b> The initialization callback function may be called</p>

Parameter	Description
	multiple times whenever the table is re-initialized.
<b>Left Mouse (Optional)</b>	<p>Callback function that is called when the user clicks the left mouse button No particular return value is expected from this function, and it supports the following keywords in its argument list:</p> <p>#Name = name assigned to the genie</p> <p>#Row = row number of the cell (with respect to the underlying data) that is clicked</p> <p>#Col = column number of the cell that is clicked</p> <p>#RowDsp = row number of the cell relative to the display that is clicked</p> <p>#ColDsp = column number of the cell relative to the display that is clicked</p> <p>#AN= the animation number where the alarm is displayed</p>
<b>Right Mouse (Optional)</b>	Callback function that is called when the user clicks the right mouse button It uses the same specification as that of the Left mouse callback .
<b>Double Click (Optional)</b>	Callback function that is called when the user double-click the left mouse button It uses the same specification as that of the Left mouse callback.
<b>Table Reload (Optional)</b>	<p>Callback function that is called when the control is redrawn No particular return value is expected from this function, and it supports the following keywords in its argument list:</p> <p>#Name = name assigned to the genie</p>

**Note:** The table can be up to 62 columns (although the latter columns may not have the column separators). For tables that return more columns than the limit, the first 62 columns will be displayed.

You can filter the alarm list at runtime by using the [Alarm Filter Functions](#).

## See Also

[Use the Library Controls](#)

## Use the Alarm Table Genie

In the example below, a simple initialization function is used to set the columns (fields) and then filter the display list when the alarm table initializes.

## Example

The Cicode sample below will be used in the following example.

```
INT FUNCTION MyAlmTable_Init(STRING sTable, STRING sEquipment)
    // Set up columns
    LibTable_AddColumn(sTable, "Name", 150);
    LibTable_AddColumn(sTable, "Description", 150, "Desc");
    LibTable_AddColumn(sTable, "State", 150);
    // Set up filter
    INT nAN = LibAlmTable_GetAN(sTable);
    INT hEdit = AlarmFilterEditOpen(nAN);
    AlarmFilterEditSet(hEdit, "Equipment=" + sEquipment);
    AlarmFilterEditCommit(hEdit);
    AlarmFilterEditClose(hEdit);
    RETURN TRUE;
END
```

### To set columns and filter via table initialization callback event:

1. Open the Cicode Editor and create a new Cicode file (for example, \_MyAlmTable\_Ex.ci).
2. From the Graphics Builder create a new page (for example, the Normal Tab Style Page).
3. When created, from the **Object toolbox** select the **Paste Genie** icon.
4. In the **Paste Genie** window select the **lib\_controls** library
5. Select the Alarm Table genie. The Alarm Table genie is pasted onto the graphics page.
6. In the Alarm Table Genie property dialog, configure the fields according to the table below. Click **OK** to save the changes.
7. Save the graphics page and compile and run the project.

Parameter	Description
Table Name	Table 1
Width	400 ( all columns within the table are displayed. If left at 100 some columns may not be displayed)
Number of rows	10
Header Height	18
Row Height	15
Header Padding	5
Row Padding	4
Column Padding	5
Horizontal Scrollbar	
Vertical Scrollbar	

Parameter	Description
Initialize	MyAlmTable_Init("#Name","Plant*")

At runtime the table will look like the following:

Alarm Table		
Name	Description	State
Plant Mixer Pe...	HIGH	HIGH
Plant Bottler Pe...	HIGH	HIGH
Plant Bottler Pre...	HIGH	HIGH
Plant Mixer Predict...	HIGH	HIGH
Mixer Chocolate F...	FAULT	OFF
Mixer Malt Feeder ...	FAULT	OFF
Mixer Full Milk Out...		OFF

## See Also

[Use the Library Controls](#)

## Calendar Library Control

The following parameters are available for the Calendar library control:

Parameter	Description
<b>Calendar Name</b>	A unique name used to identify the Genie within a page.
<b>Width</b>	The pixel width of the control at runtime. <b>Note:</b> Please do not re-size the Genie at design time, as its width will be automatically calculated at runtime.
<b>Height</b>	The pixel height of the control at runtime. <b>Note:</b> Please do not re-size the Genie at design time, as its height will be automatically calculated at runtime.
<b>Header Height</b>	Pixel height of header row (that shows the week day initials) in calendar display.
<b>Label Height</b>	Pixel height of the label row (that shows the month).
<b>Header Padding</b>	The pixel padding between the bottom of the header text and the bottom of the header row.
<b>Label Padding</b>	The pixel padding between the bottom of the label text and the bottom of the label row.

Parameter	Description
<b>Top Padding</b>	The pixel padding on top of the calendar display.
<b>Left Padding</b>	The pixel padding on left of the calendar display.
<b>Side Padding</b>	The pixel padding on both sides of the calendar display.
<b>Header Font</b>	The pre-configured font for header text.
<b>Date Font</b>	The pre-configured font for the month text.
<b>Day Font</b>	The pre-configured font for the day displays.
<b>Next Month Font</b>	The pre-configured font for the next month text.
<b>Selection Font</b>	The pre-configured font for the selected day.
<b>Single selection</b>	Whether allow single day or multiple days to be selected.
<b>Refresh Rate</b>	The minimum (quickest possible) rate that the control will be drawn. The control is only redrawn as required upon changes to the layout or other user initiated events.
<b>Initialize (Optional)</b>	<p>Callback function that is called when the control initializes. The callback function should return either TRUE (1) for success or FALSE (0) if unsuccessful. By default, if the user specified function does not initialize, the control Genie will retry the function a few time before giving up. This callback only supports the #Name keyword which represents the name assigned to the control Genie.</p> <p><b>Note:</b> The initialization callback function may be called multiple times whenever the table is re-initialized.</p>
<b>Left Mouse (Optional)</b>	<p>Callback function that is called when the user clicks the left mouse button. No particular return value is expected from this function, and it supports the following keywords in its argument list:</p> <ul style="list-style-type: none"> <li>#Name = name assigned to the Genie</li> <li>#Date = the date selected as a timestamp value. Users may wish to convert the result using TimestampToStr to the desired format and locale.</li> </ul>
<b>Right Mouse (Optional)</b>	Callback function that is called when the user clicks the right mouse button. It uses the same specification as that of the Left mouse callback.

Parameter	Description
<b>Double Click (Optional)</b>	Callback function that is called when the user double-click the left mouse button. It uses the same specification as that of the Left mouse callback.
<b>Table Reload (Optional)</b>	Callback function that is called when the control is redrawn. No particular return value is expected from this function, and it supports the following keywords in its argument list: #Name = name assigned to the Genie.

## See Also

[Use the Library Controls](#)

## Data Browse Table Library Control

Browse table genie has the following parameters:

Parameter	Description
<b>Browse Table Name</b>	Name of table. Used to identify the genie when on a graphics page
<b>Width</b>	Pixel width of the table at runtime. <b>Note:</b> Leave the size of the genie at design time as 100 pixels, as it will be automatically calculated at runtime.
<b>Number of rows</b>	Number of rows displayed on the page.
<b>Header Height</b>	Pixel Height of header row in table display.
<b>Row Height</b>	Pixel height of row in the table display.
<b>Header Padding</b>	The pixel padding between the bottom of the header text (column name) and the bottom of the header row
<b>Row Padding</b>	The pixel padding between the bottom of the cell text and the bottom of each row.
<b>Column Padding</b>	The pixel padding on the left-hand-side and right-hand-side of the cell text relative to the width of each column.
<b>Refresh Rate</b>	The minimum (quickest possible) rate that the control will be redrawn. The control is only redrawn as required upon changes

Parameter	Description
	to the layout, scrolling or other user initiated events.
<b>Scan Delay (Optional)</b>	The number of page scan delay before the genie is initialized at runtime. this can be set to different number on different control genies to stagger the initialization order. If not set defaults to 1.
<b>Browse Type</b>	<p>The type of project configuration to be browsed and displayed, including:</p> <ul style="list-style-type: none"> <li>AccumBrowse – for listing accumulators</li> <li>AlmBrowse – for listing alarm tags</li> <li>EquipBrowse – for listing equipment</li> <li>ServerBrowse – for listing servers</li> <li>TagBrowse – for listing variable tags</li> <li>TrnBrowse – for listing trend tags</li> </ul>
<b>Filter (Optional)</b>	A filter expression specifying the records to return during the browse If it is not specified, all records will be returned. Please see the browse open Cicode function of the selected browse type for the details of filter expression supported.
<b>Fields (Optional)</b>	A comma delimited string specifying the fields (columns) to be returned during the browse. If it is not specified, all fields of the selected browse type will be displayed: Tag, Tag Item, Value, Comment, Cluster, Equipment, Item, Type, IODev, Addr, Arr_Size, Eng_Zero, Eng_Full, Eng_Units, Format, Deadband, Raw_Zero, Raw_Full, Scaled_Type, Ovr_Mode, Ctrl_Mode, Override, Valid, Status, Field, Num_Subs.
<b>Sort By(Optional)</b>	A comma delimited string specifying the order of sorting preferences. This is only applicable to TagBrowse. Please see the Sort parameter of the built-in Cicode function TagBrowseOpen for details.
<b>Horizontal Scrollbar (Optional)</b>	The name of the horizontal scroll bar genie on the page that allows the user to horizontally scroll the content displayed on the control genie If you will not require horizontal scrolling, leave this field blank.
<b>Vertical Scrollbar (Optional)</b>	The name of the horizontal scroll bar genie on the page that allows the user to vertically scroll the content displayed on the control genie If you will not require vertical scrolling, leave this field blank.

Parameter	Description
<b>Header Font</b>	The pre-configured font for header text .
<b>Cell Font</b>	The pre-configured font for cell text.
<b>Selection Font</b>	The pre-configured font for cell text when it is selected .
<b>Auto Width</b>	True/False drop down box. False is selected by default. Determines whether the column width will be automatically adjusted to show the entire cell content when the control is redrawn.
<b>Allow Selection</b>	True/False drop down. True is selected by default. Determines whether allow the rows displayed on the control are selectable .
<b>Clusters (Optional)</b>	A comma delimited string specifying only items from a subset of clusters to be included in the display. If it is not specified, items from all clusters will be included.
<b>Initialize (Optional)</b>	<p>Callback function that is called when the control initializes. The callback function should return either TRUE (1) for success or FALSE (0) if unsuccessful. By default, if the user specified function does not initialize, the control genie will retry the function a few times before giving up. This callback only supports the #Name keyword which represents the name assigned to the control genie.</p> <p><b>Note:</b> The initialization callback function may be called multiple times whenever the table is re-initialized.</p>
<b>Left Mouse (Optional)</b>	<p>Callback function that is called when the user clicks the left mouse button. No particular return value is expected from this function, and it supports the following keywords in its argument list:</p> <ul style="list-style-type: none"> <li>#Name = name assigned to the genie</li> <li>#Row = row number of the cell (w.r.t. the underlying data) that is clicked</li> <li>#Col = column number of the cell that is clicked</li> <li>#RowDsp = row number of the cell relative to the display that is clicked</li> <li>#ColDsp = column number of the cell relative to the display that is clicked</li> </ul>
<b>Right Mouse (Optional)</b>	Callback function that is called when the user clicks the right mouse button. It uses the same specification

Parameter	Description
	as that of the Left mouse callback .
<b>Double Click (Optional)</b>	Callback function that is called when the user double-click the left mouse button. It uses the same specification as that of the Left mouse callback.
<b>Table Reload (Optional)</b>	Callback function that is called when the control is redrawn No particular return value is expected from this function, and it supports the following keywords in its argument list: <code>#Name = name assigned to the genie</code>

**Note:** The Table can be up to 62 columns (although the latter columns may not have the column separators). For tables that return more columns than the limit, the first 62 columns will be displayed.

## See Also

[Use the Library Controls](#)

## Equipment Tree Library Control

The Equipment Tree Genie (named "equiptree") allows you to create an in-memory menu hierarchy. It has the following parameters:

**Note:** For the Genie to work, a report server needs to be defined in your project and accessible at runtime.

Parameter	Description
<b>Tree Name</b>	Name of the tree. Used to identify the Genie when on a graphics page. Each tree on a page must have a unique name.
<b>Width</b>	Pixel width of the table at runtime. <b>Note:</b> Leave the size of the Genie at design time as 100 pixels, as it will be automatically calculated at runtime.
<b>Menu Name</b>	Name of the menu that stores the content that will be used by the tree. By default, "_Equip" is used. This specifies that all of the equipment hierarchy will be used (based on what is available to the current user). If you apply a filter expression to the tree using the <b>Filter</b> parameter (see below), you need to enter a unique menu name, as all EquipTree controls share the same equipment tree if they are assigned to the same menu name.

Parameter	Description
<b>Number of items</b>	Number of items displayed on the page.
<b>Item Height</b>	Pixel height of item in the tree display.
<b>Show CheckBox</b>	True/False drop down box. False is selected by default. Determines whether to show the checkboxes next to each of the tree items.
<b>Force Child</b>	True/False drop down box. False is selected by default. Determines whether to force all children tree items to follow the checked status of their parent when the latter is checked / unchecked.
<b>Window Inst</b>	True/False drop down box. True is selected by default. Determines whether create a new instance of the equipment hierarchy on each window where the equipment tree is displayed.
<b>Refresh Rate</b>	The minimum (quickest possible) rate that the control will be redrawn.  The control is only redrawn as required upon changes to the layout, scrolling or other user initiated events.
<b>Expand Level</b>	The minimum expand level of the tree hierarchy. Set to 0 to allow the tree be collapsed to its root items.
<b>Horizontal Scrollbar (Optional)</b>	The name of the horizontal scroll bar Genie on the page that allows the user to horizontally scroll the content displayed on the control Genie If you will not require horizontal scrolling, leave this field blank.
<b>Vertical Scrollbar (Optional)</b>	The name of the horizontal scroll bar Genie on the page that allows the user to vertically scroll the content displayed on the control Genie If you will not require vertical scrolling, leave this field blank.
<b>Label Font</b>	The font used for displaying the tree item label.
<b>Item Font</b>	The font used for displaying the optional dynamic text next to the tree item.
<b>Symbol Library (Optional)</b>	The name of the symbol library that contains the symbols for displayed next to the tree items.
<b>Refresh Rate</b>	The minimum rate that the control will be redrawn.  <b>Note:</b> The control is only redrawn as required upon changes to the layout, scrolling or other user initialized events.

Parameter	Description
<b>Scan Delay</b>	The number of page scan delay before the Genie is initialized at runtime. This can be set to different number on different control Genies to stagger the initialization order.
<b>Expand Level</b>	The minimum expand level of the tree hierarchy. Set this to 0 to allow the tree to be collapsed to its root items.
<b>Filter (Optional)</b>	<p>A filter expression specifying the equipment records that will be available to the tree. If it is not specified, all equipment available to the currently logged on user will be returned.</p> <p>If you apply a filter, you need to enter a unique name in the <b>Menu Name</b> parameter (see above). A unique menu name will allow you to determine which filter settings apply to a particular menu.</p>
<b>Initialize</b>	Callback function that is called when the control initializes. The callback function should return either TRUE (1) for success or FALSE (0) for unsuccessful. By default, if the user specified function does not initialize, the control Genie will retry the function a few times before giving up. This callback only supports the #Name keyword which represents the name assigned to the control Genie.
<b>Row Item</b>	<p>Callback function that is called when the table initializes. The callback function should return a string value. The value returned will be displayed next to the tree item label. It supports the following keyword in its argument list:</p> <ul style="list-style-type: none"> <li>#Name - name assigned to the Genie.</li> <li>#Ref – the complete path of the tree item.</li> <li>#Label – the label of the tree item.</li> <li>#ItemDsp- the position of the item as displayed on screen.</li> <li>#IsChecked - the checked status of the tree item.</li> <li>#IsParent - whether the item is expanded/collapsed.</li> <li>#ItemDsp - the position of the item as displayed on screen.</li> <li>#Node - the handle to the internal menu node object that represents the tree item.</li> <li>#Level - the level of the tree item.</li> </ul>

Parameter	Description
<b>Selected</b>	Callback function that is called when a tree item is selected. No particular return value is expected from this function, and it supports the same set of keywords as that of the Row Item function in its argument list. Callback function that is called when the check box of a tree item is clicked. No particular return value is expected from this function, and it supports the same set of keywords as that of the Row Item function in its argument list.
<b>Checked</b>	Callback function that is called when a tree item is selected. No particular return value is expected from this function, and it supports the same set of keywords as that of the Row Item function in its argument list. Callback function that is called when the check box of a tree item is clicked. No particular return value is expected from this function, and it supports the same set of keywords as that of the Row Item function in its argument list.
<b>Left Mouse</b>	Callback function that is called when the user clicks the left mouse button. No particular return value is expected from this function, and it supports the following keywords in its argument list:  #Node - the handle to the internal menu node object that represents the tree item.  #Level - the level of the tree item.
<b>Right Mouse</b>	Callback function that is called when the user clicks the right mouse button. It uses the same specification as that of the Left mouse callback .
<b>Double Click</b>	Callback function that is called when the user double-click the left mouse button. It uses the same specification as that of the Left mouse callback.
<b>Tree Reload</b>	Callback function that is called when the control is redrawn. No particular return value is expected from this function, and it supports the following keywords in its argument list:  #Name = name assigned to the Genie.
<b>Tree Name</b>	Name of table (max characters). Used to identify the Genie when on a graphics page.
<b>Width</b>	Pixel width of the table at runtime.  <b>Note:</b> Leave the size of the Genie at design time as

Parameter	Description
	100 pixels, as it will be automatically calculated at runtime.

**Note:** The project will compile and run when configured with longer equipment names, a runtime error may occur when you attempt to interact with the tree display based on the 'EquipTree' Genie. When this happens, an alarm icon will appear on the tree view, and a "Too few arguments for function or argument string too long" hardware alarm will be reported. This usually indicates either the equipment name is too long or the name of the tree Genie is too long.

## See Also

[Use the Library Controls](#)

## PageTabs Library Control

PageTabs genie has the following parameters:

**Note:** The PageTabs genie is specifically designed for displaying page data. The genie internally binds page data to the tab display. You should not add tabs directly. Manipulating the tab data using the LibTabs Cicode function may result in unexpected behavior.

Parameter	Description
<b>Page Tabs Name</b>	Name of page tabs control (max characters 254). Used to identify the genie when on a graphics page.
<b>Width</b>	Pixel width of the tabs control at runtime. Note: Leave the size of the genie at design time as 100 pixels, as it will be automatically calculated at runtime.
<b>Tab height</b>	Pixel height of the tabs control at runtime.
<b>Vertical Padding</b>	The vertical distance (in pixels) between the tab elements and the bottom of the tabs control.
<b>Tab Padding</b>	The horizontal distance (in pixels) between the edges and elements of a tab
<b>Tab Min Width</b>	The minimum width allowed for a tab
<b>Tab Max Width</b>	The maximum width allowed for a tab
<b>Auto Width</b>	True/False drop down box. False is selected by default. Determines whether the column width will be automatically adjusted to show full cell content when the control is redrawn.
<b>Icon Add Padding</b>	Additional vertical padding for the display of icon

Parameter	Description
	relative to the text. Positive padding will move the icon further upward, and vice versa.
<b>Refresh Rate</b>	<p>The minimum (quickest possible) rate that the control will be redrawn.</p> <p>The control is only redrawn as required upon changes to the layout, scrolling or other user initiated events.</p>
<b>Scan Delay (Optional)</b>	The number of page scan delay before the genie is initialized at runtime. this can be set to different number on different control genies to stagger the initialization order. If not set defaults to 1
<b>Page Type</b>	The type of page data to be displayed in the tabs: PageList GenericMenu PageMenu WindowMenu
<b>Menu (Page) Name</b>	Name of page with menu items to display. this is only applicable when Page type is set to "PageMenu"
<b>Default Font</b>	The pre-configured font for tab text.
<b>Selection Font</b>	The pre-configured font for a tab when it is selected.
<b>Tab Line Style</b>	<p>The display style of the separator between tabs, options are:</p> <p>Dark Light None</p>
<b>Tab Select Style</b>	<p>The display style for highlighting of selected tab:</p> <p>Dark Light None</p>
<b>Allow Drag Sort</b>	Indicates whether to allow individual tabs to be repositioned by drag and drop.
<b>Allow Drag Delete</b>	Indicates whether to allow individual tabs to be deleted by dragging it vertically away from the tabs control.
<b>Tab Select Icon</b>	The icon to be displayed when a tab is selected. An icon needs to be pre-configured as a symbol in your project. It is specified by the syntax of <symbol library>.<symbol>.

Parameter	Description
<b>Initialize (Optional)</b>	<p>Callback function that is called when the control initializes. The callback function should return either TRUE (1) for success or FALSE (0) if unsuccessful. By default, if the user specified function does not initialize, the control genie will retry the function a few times before giving up. This callback only supports the #Name keyword which represents the name assigned to the control genie.</p> <p><b>Note:</b> The initialization callback function may be called multiple times whenever the tab is re-initialized.</p>
<b>Right Mouse (Optional)</b>	<p>Callback function that is called when the user clicks the right mouse button. No particular return value is expected from this function, and it supports the following keywords in its argument list:</p> <ul style="list-style-type: none"> <li>#Name = name assigned to the genie</li> <li>#Ref = reference (assigned when tab is added) of the cell that is clicked</li> <li>#TabDsp = tab number of the cell relative to the display that is clicked</li> <li>#Icon = icon number indicating the different areas with a tab is clicked: <ul style="list-style-type: none"> <li>&lt;=0 - text on tab is clicked</li> <li>1 - icon to left hand side of text is clicked</li> <li>2 - icon to right hand side of text is clicked</li> <li>3 - selection icon (only shows up when a tab selected) is clicked</li> </ul> </li> </ul>
<b>Double Click (Optional)</b>	<p>Callback function that is called when the user double-click the left mouse button. It uses the same specification as that of the Left mouse callback.</p>
<b>Tabs Reload (Optional)</b>	<p>Callback function that is called when the control is redrawn. No particular return value is expected from this function, and it supports the following keywords in its argument list:</p> <ul style="list-style-type: none"> <li>#Name = name assigned to the genie</li> </ul>

## See Also

[Use the Library Controls](#)

## Scroll Bar Horizontal Library Control

The horizontal scroll bar genie has the following parameters:

Parameter	Description
<b>Scrollbar Name</b>	A unique name used to identify the genie within a page.
<b>Width</b>	The pixel width of the control at runtime. Leave the size of the genie at design time as 100 pixels, as it will be automatically calculated at runtime.
<b>Update on Drag</b>	True/False (True selected by default). Determines whether the position of the scroll bar and the contents of the associated control genie is updated while the thumb is being dragged
<b>Auto Hide</b>	Determines whether the scroll bar will be automatically hidden when the contents of the associated control are fully displayed on screen
<b>Allow Double Click</b>	Determines whether the scroll bar will response to mouse double-click event
<b>Hidden When</b>	A user specified expression to explicitly hide the scroll bar If this is not specified, it is defaulted to FALSE (not hidden).
<b>Initialize</b>	Callback function that is called when the control initializes. The callback function should return either TRUE (1) for success or FALSE (0) if unsuccessful. By default, if the user specified function does not initialize, the control genie will retry the function a few time before giving up. This callback only supports the #Name keyword which represents the name assigned to the control genie.
<b>Callback</b>	Callback function to be called when the position of the scroll bar is updated. No particular return value is expected from this function, and it supports the following keywords in its argument list: #Name = name assigned to the genie #Pos = position of the scroll bar #View = number of items for the scroll bar view (that constitutes one page of items) #Max = maximum (total) number of items for the scroll bar

Parameter	Description
<b>Right Mouse</b>	Callback function to be called when the user clicks the right mouse button on the scroll bar. Supports the same set of substitutions as Callback.

## See Also

[Use the Library Controls](#)

## Scroll Bar Vertical Library Control

The vertical scroll bar genie has the following parameters:

Parameter	Description
<b>Scrollbar Name</b>	A unique name used to identify the genie within a page.
<b>Height</b>	The pixel height of the control at runtime. Leave the height of the genie at design time as 100 pixels, as it will be automatically calculated at runtime.
<b>Update on Drag</b>	True/False (True selected by default). Determines whether the position of the scroll bar and the contents of the associated control genie is updated while the thumb is being dragged
<b>Auto Hide</b>	Determines whether the scroll bar will be automatically hidden when the contents of the associated control are fully displayed on screen
<b>Allow Double Click</b>	Determines whether the scroll bar will response to mouse double-click event
<b>Hidden When</b>	A user specified expression to explicitly hide the scroll bar If this is not specified, it is defaulted to FALSE (not hidden).
<b>Initialize</b>	Callback function that is called when the control initializes. The callback function should return either TRUE (1) for success or FALSE (0) if unsuccessful. By default, if the user specified function does not initialize, the control genie will retry the function a few time before giving up. This callback only supports the #Name keyword which represents the name assigned to the control genie.
<b>Callback</b>	Callback function to be called when the position of the

Parameter	Description
	scroll bar is updated. No particular return value is expected from this function, and it supports the following keywords in its argument list: <code>#Name</code> = name assigned to the genie <code>#Pos</code> = position of the scroll bar <code>#View</code> = number of items for the scroll bar view (that constitutes one page of items) <code>#Max</code> = maximum (total) number of items for the scroll bar
<b>Right Mouse</b>	Callback function to be called when the user click the right mouse button on the scroll bar. Supports the same set of substitutions as Callback.

## See Also

[Use the Library Controls](#)

## Slider Library Control

The Slider genie has the following parameters:

Parameter	Description
<b>Slider Name</b>	A unique name used to identify the control genie within a page.
<b>Length</b>	The pixel height of the control at runtime. Leave the size of the genie at design time as 100 pixels, as it will be automatically calculated at runtime.
<b>Minimum Offset</b>	The min. offset (in pixels) the slider can be dragged to. The acceptable range is -4000 to 4000.
<b>Maximum Offset</b>	The max. offset (in pixels) the slider can be dragged to. The acceptable range is -4000 to 4000.
<b>Docked Offset (Optional)</b>	The offset (in pixels) the slider is moved to when it is docked. If this is not set, it is defaulted to -4000. The acceptable range is -4000 to 4000.
<b>Dockable</b>	Whether the middle thumb button will appear at runtime, so that the user can click on it to move the slider to the docked offset.

Parameter	Description
<b>Update on Drag</b>	Whether the position of the scroll bar and the contents of the associated control genie is updated while the thumb is being dragged.
<b>Hidden</b>	Determines whether to hide the genie
<b>Scan Delay (Optional)</b>	The number of page scan delay before the genie is initialized at runtime. This can be set to different number on different control genies to stacker their initialization order. If this is not set, it defaults to 0.
<b>Initialize</b>	Callback function that is called when the control initializes. The callback function should return either TRUE (1) for success or FALSE (0) for unsuccessful. By default, if the user specified function can not initialize successfully, the control genie will retry the function a few times before giving up. This callback only supports the #Name keyword which represents the name assigned to the control genie.
<b>Callback</b>	Callback function to be called when the position of the slider is updated. No particular return value is expected from this function, and it supports the following keywords in its argument list:  #Name = name assigned to the genie #Offset = Offset of the slider position relative to its initial position
<b>Right Mouse</b>	Callback function to be called when the user click the right mouse button on the scroll bar

## See Also

[Use the Library Controls](#)

## SQL Table Library Control

Tag table genie has the following parameters:

Parameter	Description
<b>SQL Table Name</b>	Name of table (max characters). Used to identify the genie when on a graphics page
<b>Width</b>	Pixel width of the table at runtime. <b>Note:</b> Leave the size of the genie at design time as 100

Parameter	Description
	pixels, as it will be automatically calculated at runtime.
<b>Number of rows</b>	Number of rows displayed on the page.
<b>Header Height</b>	Pixel Height of header row in table display.
<b>Row Height</b>	Pixel height of row in the table display.
<b>Header Padding</b>	The pixel padding between the bottom of the header text (column name) and the bottom of the header row
<b>Row Padding</b>	The pixel padding between the bottom of the cell text and the bottom of each row.
<b>Column Padding</b>	The pixel padding on the left-hand-side and right-hand-side of the cell text relative to the width of each column.
<b>Refresh Rate</b>	The minimum (quickest possible) rate that the control will be redrawn.  The control is only redrawn as required upon changes to the layout, scrolling or other user initiated events.
<b>Scan Delay (Optional)</b>	The number of page scan delay before the genie is initialized at runtime. this can be set to different number on different control genies to stagger the initialization order. If not set defaults to 1.
<b>Connection Str</b>	The ADO connection string to the database source. See the sConnect parameter of the built-in Cicode function SQLCreate for details.
<b>SQL Selection Query</b>	The SQL query statement to return a table of records (recordset)
<b>Horizontal Scrollbar (Optional)</b>	The name of the horizontal scroll bar genie on the page that allows the user to horizontally scroll the content displayed on the control genie If you will not require horizontal scrolling, leave this field blank.
<b>Vertical Scrollbar (Optional)</b>	The name of the horizontal scroll bar genie on the page that allows the user to vertically scroll the content displayed on the control genie If you will not require vertical scrolling, leave this field blank.
<b>Header Font</b>	The pre-configured font for header text.
<b>Cell Font</b>	The pre-configured font for cell text.

Parameter	Description
<b>Selection Font</b>	The pre-configured font for cell text when it is selected.
<b>Auto Width</b>	True/False drop down box. False is selected by default. Determines whether the column width will be automatically adjusted to show full cell content when the control is redrawn.
<b>Allow Selection</b>	True/False drop down. True is selected by default. Determines whether allow the rows displayed on the control are selectable .
<b>Initialize (Optional)</b>	<p>Callback function that is called when the control initializes. The callback function should return either TRUE (1) for success or FALSE (0) if unsuccessful. By default, if the user specified function does not initialize, the control genie will retry the function a few times before giving up.</p> <p>This callback only supports the #Name keyword which represents the name assigned to the control genie.</p> <p><b>Note:</b> The initialization callback function may be called multiple times whenever the table is re-initialized.</p>
<b>Left Mouse (Optional)</b>	<p>Callback function that is called when the user clicks the left mouse button. No particular return value is expected from this function, and it supports the following keywords in its argument list:</p> <ul style="list-style-type: none"> <li>#Name = name assigned to the genie</li> <li>#Row = row number of the cell (w.r.t. the underlying data) that is clicked</li> <li>#Col = column number of the cell that is clicked</li> <li>#RowDsp = row number of the cell relative to the display that is clicked</li> <li>#ColDsp = column number of the cell relative to the display that is clicked</li> </ul>
<b>Right Mouse (Optional)</b>	Callback function that is called when the user clicks the right mouse button. It uses the same specification as that of the Left mouse callback.
<b>Double Click (Optional)</b>	Callback function that is called when the user double-click the left mouse button. It uses the same specification as that of the Left mouse callback.
<b>Table Reload (Optional)</b>	Callback function that is called when the control is redrawn. No particular return value is expected from

Parameter	Description
	this function, and it supports the following keywords in its argument list: #Name = name assigned to the genie

**Note:** The Table can be up to 62 columns (although the latter columns may not have the column separators). For tables that return more columns than the limit, the first 62 columns will be displayed.

## See Also

[Use the Library Controls](#)

## Tabs Library Control

Tabs genie has the following parameters:

**Note:** All fields are mandatory unless specified otherwise.

Parameter	Description
<b>Tabs Name</b>	Name of tabs control (max characters). Used to identify the genie when on a graphics page.
<b>Width</b>	Pixel width of the tabs control at runtime. Note: Leave the size of the genie at design time as 100 pixels, as it will be automatically calculated at runtime.
<b>Tab height</b>	Pixel height of the tabs control at runtime.
<b>Vertical Padding</b>	The vertical distance (in pixels) between the tab elements and the bottom of the tabs control.
<b>Tab Padding</b>	The horizontal distance (in pixels) between the edges and elements of a tab
<b>Tab Min Width</b>	The minimum width allowed for a tab
<b>Tab Max Width</b>	The maximum width allowed for a tab
<b>Auto Width</b>	True/False drop down box. False is selected by default. Determines whether the column width will be automatically adjusted to show full cell content when the control is redrawn.
<b>Icon Add Padding</b>	Additional vertical padding for the display of icon relative to the text. Positive padding will move the icon further upward, and vice versa.
<b>Refresh Rate</b>	The minimum (quickest possible) rate that the control

Parameter	Description
	<p>will be redrawn.</p> <p>The control is only redrawn as required upon changes to the layout, scrolling or other user initiated events.</p>
<b>Scan Delay (Optional)</b>	<p>The number of page scan delay before the genie is initialized at runtime. this can be set to different number on different control genies to stagger the initialization order. If not set defaults to 1</p>
<b>Default Font</b>	<p>The pre-configured font for tab text.</p>
<b>Selection Font</b>	<p>The pre-configured font for a tab when it is selected.</p>
<b>Tab Line Style</b>	<p>The display style of the separator between tabs, options are:</p> <ul style="list-style-type: none"> <li>Dark</li> <li>Light</li> <li>None</li> </ul>
<b>Tab Select Style</b>	<p>The display style for highlighting of selected tab:</p> <ul style="list-style-type: none"> <li>Dark</li> <li>Light</li> <li>None</li> </ul>
<b>Allow Drag Sort</b>	<p>Indicates whether to allow individual tabs to be repositioned by drag and drop.</p>
<b>Allow Drag Delete</b>	<p>Indicates whether to allow individual tabs to be deleted by dragging it vertically away from the tabs control.</p>
<b>Tab Select Icon</b>	<p>The icon to be displayed when a tab is selected. An icon needs to be pre-configured as a symbol in your project. It is specified by the syntax of &lt;symbol library&gt;.&lt;symbol&gt;.</p>
<b>Initialize (Optional)</b>	<p>Callback function that is called when the control initializes The callback function should return either TRUE (1) for success or FALSE (0) if unsuccessful. By default, if the user specified function does not initialize, the control genie will retry the function a few time before giving up. This callback only supports the #Name keyword which represents the name assigned to the control genie.</p> <p><b>Note:</b> The initialization callback function may be called multiple times whenever the tab is re-initialized.</p>
<b>Left Mouse (Optional)</b>	<p>Callback function that is called when the user clicks</p>

Parameter	Description
	<p>the left mouse button No particular return value is expected from this function, and it supports the following keywords in its argument list:</p> <ul style="list-style-type: none"> <li>#Tab= tab ordinal number of the cell that is clicked</li> <li>#Name = name assigned to the genie</li> <li>#Ref = reference (assigned when tab is added) of the cell that is clicked</li> <li>#TabDsp = tab number of the cell relative to the display that is clicked</li> <li>#Icon =Icon number indicating the different areas within a tab when clicked:           <ul style="list-style-type: none"> <li>&lt;=0 - text on tab is clicked</li> <li>1- icon to left hand side of text is clicked</li> <li>2- icon to right hand side of text is clicked</li> <li>3- selection icon (only shows up when a tab selected) is clicked</li> </ul> </li> </ul>
<b>Right Mouse (Optional)</b>	Callback function that is called when the user clicks the right mouse button. It uses the same specification as that of the Left mouse callback.
<b>Double Click (Optional)</b>	Callback function that is called when the user double-click the left mouse button. It uses the same specification as that of the Left mouse callback.
<b>Tabs Reload (Optional)</b>	<p>Callback function that is called when the control is redrawn. No particular return value is expected from this function, and it supports the following keywords in its argument list:</p> <ul style="list-style-type: none"> <li>#Name = name assigned to the genie</li> </ul>

## See Also

[Use the Library Controls](#)

## Use the Tabs Genie

There are two ways to populate data to the tabs:

1. Set cell value directly to the table.
2. Binding a Cicode function to return the value for the individual tabs.

## Example 1

### Set create tabs statically on initialization

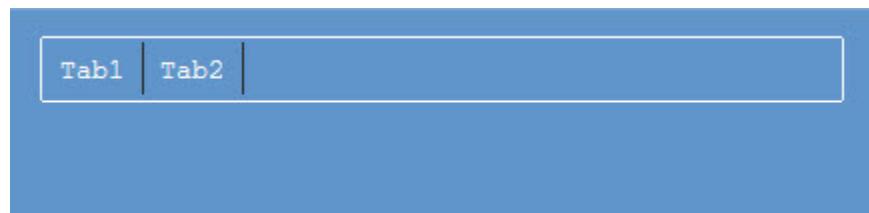
The Cicode sample below will be used in the following example. Open the Cicode Editor and create a new Cicode file, for example: \_MyTab\_Ex1.ci

```
//Callback function to respond to the initialization event of the tab genie
INT FUNCTION MyTabs_Init(STRING sTabs)
// Pre-populate with 2 auto-width tabs
LibTabs_AddTab(sTabs, "Tab1", 0);
LibTabs_AddTab(sTabs, "Tab2", 0);
RETURN TRUE;
END
```

1. From Graphics Builder create a new page (for example, a Normal Tab Style Page).
2. When created, from the **Object Toolbox** select the **Paste Genie** icon.
3. In the **Paste Genie** window, select the **lib\_controls library**.
4. Select the **Tabs** genie. The genie is pasted onto the graphics page.
5. In the Tab Genie dialog, configure the fields according to the table below. Click **OK** to save the changes.
6. Save the graphics page, and compile and run the project.

Parameter	Description
Tab BarName	MyTabs1
Width	400
Height	30
Initialize	MyTabs_Init("#Name")

At runtime the tab layout should look like the following:



## Example 2

### Binding a Cicode function to return the value for the individual tabs

This is similar to setting a formula in Excel. With binding, cell values are not stored in the tabs control. the control will call the provided Cicode function when it needs the data.

Using the Cicode sample below, open the Cicode Editor and create a new Cicode file. For example: 'MyTabs\_Ex2.ci'

```
INT FUNCTION MyTabs_InitDyn(STRING sTabs)
// Bind callback function for data display
```

```
LibTabs_SetDataTask(sTabs, "Value", "MyTabs_GetCellValue", "^#Name^", #Tab");

// Set the number of tabs for the control
LibTabs_SetCount(sTabs, 3);

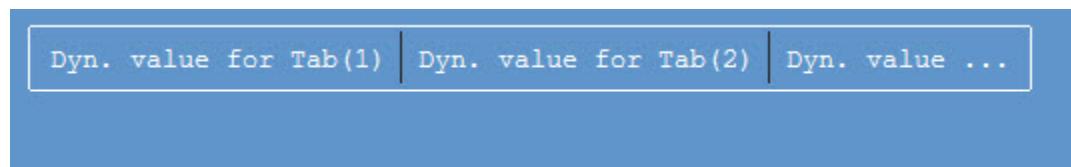
RETURN TRUE;
END

//Function to return the value for a data cell
STRING FUNCTION MyTabs_GetCellValue(STRING sTabs, INT nTab)
RETURN "Dyn. value for Tab(" + nTab:# + ")";
END
```

1. As in example 1, add a tabs library control to the graphics page.
2. In the Tabs Genie dialog, configure the fields according to the table below.
3. Save the graphics page, and compile and run the project.

Parameter	Description
Tabs Name	MyTabs2
Width	400
Height	30
Can Remove Tab	TRUE
Initialize	MyTabs_Initdyn("#Name")

At runtime the tab layout should look like the following:



## See Also

[Use the Library Controls](#)

## Table Library Control

Table genie has the following parameters:

---

**Note:** All fields are mandatory unless specified otherwise.

---

Parameter	Description
<b>Table Name</b>	Name of table (max characters). Used to identify the genie when on a graphics page
<b>Width</b>	Pixel width of the table at runtime. <b>Note:</b> Leave the size of the genie at design time as 100 pixels, as it will be automatically calculated at runtime.
<b>Number of rows</b>	Number of rows displayed on the page.
<b>Header Height</b>	Pixel Height of header row in table display.
<b>Row Height</b>	Pixel height of row in the table display.
<b>Header Padding</b>	The pixel padding between the bottom of the header text (column name) and the bottom of the header row
<b>Row Padding</b>	The pixel padding between the bottom of the cell text and the bottom of each row.
<b>Column Padding</b>	The pixel padding on the left-hand-side and right-hand-side of the cell text relative to the width of each column.
<b>Refresh Rate</b>	The minimum (quickest possible) rate that the control will be redrawn. The control is only redrawn as required upon changes to the layout, scrolling or other user initiated events.
<b>Horizontal Scrollbar (Optional)</b>	The name of the horizontal scroll bar genie on the page that allows the user to horizontally scroll the content displayed on the control genie If you will not require horizontal scrolling, leave this field blank.
<b>Vertical Scrollbar (Optional)</b>	The name of the horizontal scroll bar genie on the page that allows the user to vertically scroll the content displayed on the control genie If you will not require vertical scrolling, leave this field blank.
<b>Header Font</b>	The pre-configured font for header text.
<b>Cell Font</b>	The pre-configured font for cell text.
<b>Selection Font</b>	The pre-configured font for cell text when it is selected.
<b>Auto Width</b>	True/False drop down box. False is selected by default. Determines whether the column width will be automatically adjusted to show complete cell content when the control is redrawn.

Parameter	Description
<b>Allow Selection</b>	True/False drop down. True is selected by default. Determines whether allow the rows displayed on the control are selectable.
<b>Scan Delay (Optional)</b>	The number of page scan delay before the genie is initialized at runtime. this can be set to different number on different control genies to stagger the initialization order. If not set defaults to 1.
<b>Initialize (Optional)</b>	<p>Callback function that is called when the control initializes. The callback function should return either TRUE (1) for success or FALSE (0) if unsuccessful. By default, if the user specified function does not initialize, the control genie will retry the function a few time before giving up. This callback only supports the #Name keyword which represents the name assigned to the control genie.</p> <p><b>Note:</b> The initialization callback function may be called multiple times whenever the table is re-initialized.</p>
<b>Left Mouse (Optional)</b>	<p>Callback function that is called when the user clicks the left mouse button. No particular return value is expected from this function, and it supports the following keywords in its argument list:</p> <ul style="list-style-type: none"> <li>#Name = name assigned to the genie</li> <li>#Row = row number of the cell (w.r.t. the underlying data) that is clicked</li> <li>#Col = column number of the cell that is clicked</li> <li>#RowDsp = row number of the cell relative to the display that is clicked</li> <li>#ColDsp = column number of the cell relative to the display that is clicked</li> </ul>
<b>Right Mouse (Optional)</b>	Callback function that is called when the user clicks the right mouse button. It uses the same specification as that of the Left mouse callback .
<b>Double Click (Optional)</b>	Callback function that is called when the user double-click the left mouse button. It uses the same specification as that of the Left mouse callback.
<b>Table Reload (Optional)</b>	<p>Callback function that is called when the control is redrawn. No particular return value is expected from this function, and it supports the following keywords in its argument list:</p> <ul style="list-style-type: none"> <li>#Name = name assigned to the genie</li> </ul>

**Note:** The Table can be up to 62 columns (although the latter columns may not have the column separators). For tables that return more columns than the limit, the first 62 columns will be displayed. The Table can be up to 100 rows.

## See Also

[Using the Tag Table Genie](#)

[Use the Library Controls](#)

## Use the Table Genie

There are two ways to populate data to the table:

1. Set cell value directly to the table.
2. Binding a Cicode function to return the value for the individual cells of the table.

### Example 1

#### Set cell value directly to the table

In this case, the value is stored with the table. You will need to set a new value to the individual cell of the table if required.

The Cicode sample below will be used in the following example. Open the Cicode Editor and create a new cicode file, for example, \_MyTable\_Ex1.ci

```
//Callback function to respond to the initialization event of the table genie
INT FUNCTION MyTable_Init(STRING sTable)
    // Add your columns
    LibTable_AddColumn(sTable, "Column1", 150);
    LibTable_AddColumn(sTable, "Column2", 100);

    // Add data directly to the table
    MyTable_AddRow(sTable, "Value for Cell(1,1)", "Value for Cell(1,2)");
    MyTable_AddRow(sTable, "Value for Cell(2,1)", "Value for Cell(2,2)");
    MyTable_AddRow(sTable, "Value for Cell(3,1)", "Value for Cell(3,2)");
    RETURN TRUE;
END
//?Function to add a new row of data to the table
INT FUNCTION MyTable_AddRow(STRING sTable, STRING sCol1Value, STRING sCol2Value)
    // Add data to the table, and the data is stored in the table
    INT nRow = LibTable_SetCellValue(sTable, -1, 1, sCol1Value);
    RETURN LibTable_SetCellValue(sTable, nRow, 2, sCol2Value);
END
```

1. From the Graphics Builder create a new page (E.g. the Normal Tab Style Page)
2. When created, from the **Object toolbox** select -> The **Paste Genie** icon.
3. In the **Paste Genie** window select the **lib\_controls** library
4. Select the Table genie. The Table genie is pasted onto the graphics page.
5. In the Table Genie property dialog, configure the fields according to the table below. Click OK to save your changes.

6. Save the graphics page and compile and run the project.

Parameter	Description
Table Name	Table 1
Width	400 (all columns within the table are displayed. If left at 100 some columns may not be displayed)
Number of rows	10
Header Height	18
Row Height	15
Header Padding	5
Row Padding	4
Column Padding	5
Horizontal Scrollbar	
Vertical Scrollbar	
Initialize	MyTable_Init("#Name")

At runtime the table will look like the following:

Table with Stored Value		
Column1 Value for Cell(1,1) Value for Cell(2,1) Value for Cell(3,1)	Column2 Value for Ce... Value for Ce... Value for Ce...	

## Example 2

### Binding a Cicode function to return the value for the individual cells of the table

This is similar to setting a formula in Excel. With binding, cell values are not stored in the table. The table will call the provided Cicode function when it needs data.

The Cicode sample below will be used in the following example. Open the Cicode Editor and create a new cicode file, for example, \_MyTable\_Ex2.ci

```
INT FUNCTION MyTable_InitDyn(STRING sTable)
    // Add your columns
```

```

LibTable_AddColumn(sTable, "Column1", 150);
LibTable_AddColumn(sTable, "Column2", 100);

// Bind callback function for data display
LibTable_SetDataTask(sTable, "Value", "MyTable_GetCellValue", "^^#Name^", #Row, #Col");

// Set the number of rows for the table
LibTable_SetPropertyInt(sTable, "RowCount", 9);

RETURN TRUE;
END
//Function to return the value for a data cell
STRING FUNCTION MyTable_GetCellValue(STRING sTable, INT nRow, INT nCol)
IF (nRow > LibTableGetPropertyInt(sTable, "RowCount")) THEN
    RETURN "";
END
RETURN "Dyn. value for Cell(" + nRow:# + "," + nCol:# + ")";
END

```

1. As in example 1, add a table library control to the graphics page.
2. In the Table Genie property dialog, configure the fields according to the table below. Click **OK** to save the changes.
3. Compile and then run the project.

Parameter	Description
Table Name	MyTable2
Width	400 (all columns within the table are displayed. If left at 100 some columns may not be displayed)
Number of rows	10
Header Height	18
Row Height	15
Header Padding	5
Row Padding	4
Column Padding	5
Horizontal Scrollbar	
Vertical Scrollbar	
Initialize	MyTable_InitDyn("#Name")

At runtime the table will look like the following:

**Table with Dynamic Data**

Column1	Column2	
Dyn. value for Cell(1,1)	Dyn. value f...	
Dyn. value for Cell(2,1)	Dyn. value f...	
Dyn. value for Cell(3,1)	Dyn. value f...	
Dyn. value for Cell(4,1)	Dyn. value f...	
Dyn. value for Cell(5,1)	Dyn. value f...	
Dyn. value for Cell(6,1)	Dyn. value f...	
Dyn. value for Cell(7,1)	Dyn. value f...	
Dyn. value for Cell(8,1)	Dyn. value f...	
Dyn. value for Cell(9,1)	Dyn. value f...	

**See Also**[Use the Library Controls](#)**Table Row Library Control**

Table Row genie has the following parameters:

Parameter	Description
<b>Table Name</b>	The name of the table (or its derived) genies that already exists on the page.
<b>Row</b>	Number of row as it appears on screen

**See Also**[Use the Library Controls](#)**Tag Table Library Control**

Tag table genie has the following parameters:

Parameter	Description
<b>Tag Table Name</b>	Name of table (max characters). Used to identify the genie when on a graphics page.
<b>Width</b>	Pixel width of the table at runtime. Note: Leave the size of the genie at design time as 100 pixels, as it will be automatically calculated at runtime.
<b>Number of rows</b>	Number of rows displayed on the page.

Parameter	Description
<b>Header Height</b>	Pixel Height of header row in table display.
<b>Row Height</b>	Pixel height of row in the table display.
<b>Header Padding</b>	The pixel padding between the bottom of the header text (column name) and the bottom of the header row
<b>Row Padding</b>	The pixel padding between the bottom of the cell text and the bottom of each row.
<b>Column Padding</b>	The pixel padding on the left-hand-side and right-hand-side of the cell text relative to the width of each column.
<b>Refresh Rate</b>	<p>The minimum (quickest possible) rate that the control will be redrawn.</p> <p>The control is only redrawn as required upon changes to the layout, scrolling or other user initiated events.</p>
<b>Scan Delay (Optional)</b>	The number of page scan delay before the genie is initialized at runtime. this can be set to different number on different control genies to stagger the initialization order. If not set defaults to 1.
<b>Horizontal Scrollbar (Optional)</b>	The name of the horizontal scroll bar genie on the page that allows the user to horizontally scroll the content displayed on the control genie If you will not require horizontal scrolling, leave this field blank.
<b>Vertical Scrollbar (Optional)</b>	The name of the horizontal scroll bar genie on the page that allows the user to vertically scroll the content displayed on the control genie If you will not require vertical scrolling, leave this field blank.
<b>Header Font</b>	The pre-configured font for header text.
<b>Cell Font</b>	The pre-configured font for cell text.
<b>Selection Font</b>	The pre-configured font for cell text when it is selected.
<b>Auto Width</b>	True/False drop down box. False is selected by default. Determines whether the column width will be automatically adjusted to show full cell content when the control is redrawn.
<b>Allow Selection</b>	True/False drop down. True is selected by default. Determines whether allow the rows displayed on the control are selectable.

Parameter	Description
<b>Initialize (Optional)</b>	<p>Callback function that is called when the control initializes. The callback function should return either TRUE (1) for success or FALSE (0) if unsuccessful. By default, if the user specified function does not initialize, the control genie will retry the function a few times before giving up. This callback only supports the #Name keyword which represents the name assigned to the control genie.</p> <p><b>Note:</b> The initialization callback function may be called multiple times whenever the table is re-initialized.</p>
<b>Left Mouse (Optional)</b>	<p>Callback function that is called when the user clicks the left mouse button. No particular return value is expected from this function, and it supports the following keywords in its argument list:</p> <ul style="list-style-type: none"> <li>#Name = name assigned to the genie</li> <li>#Row = row number of the cell (w.r.t. the underlying data) that is clicked</li> <li>#Col = column number of the cell that is clicked</li> <li>#RowDsp = row number of the cell relative to the display that is clicked</li> <li>#ColDsp = column number of the cell relative to the display that is clicked</li> </ul>
<b>Right Mouse (Optional)</b>	<p>Callback function that is called when the user clicks the right mouse button. It uses the same specification as that of the Left mouse callback.</p>
<b>Double Click (Optional)</b>	<p>Callback function that is called when the user double-clicks the left mouse button. It uses the same specification as that of the Left mouse callback.</p>
<b>Table Reload (Optional)</b>	<p>Callback function that is called when the control is redrawn. No particular return value is expected from this function, and it supports the following keywords in its argument list:</p> <ul style="list-style-type: none"> <li>#Name = name assigned to the genie</li> </ul>

**Note:** The Table can be up to 62 columns (although the latter columns may not have the column separators).

## See Also

[Use the Library Controls](#)

## Using the Tag Table Genie

Use the Tag Table Genie on a page when you want to be able to view variable tags where the value of each tag changes constantly within your project.

The table does not automatically populate with variable tags unless a Cicode function is called to add the tags to it. The Cicode function you use to add a variable tag to the table is:

```
LibTagTable_AddTag(STRING sTable, STRING sTag, STRING sName = "", STRING sFormat = "", STRING sFormatActive = "", INT iArrayIndex = 0)
```

Where:

- *sTable* = the name of the Tag Table Genie where values will be inserted (required)
- *sTag* = the actual tag itself (required)
- *sName* = the name of the tag (optional)
- *sFormat* = A replacement value to insert into the table if the tag's value is empty or equal to zero (and *sFormat* and *sFormatActive* are both not empty). For example, "bad tag". (optional)
- *sFormatActive* = A replacement value to insert into the table if the tag value is not empty or not equal to zero (and *sFormat* and *sFormatActive* are both not empty). For example, "good tag". (optional)
- *iArrayIndex* = The index of the element in the tag to insert the value for if an array tag is provided in the *sTag* parameter (optional)

For example:

```
LibTagTable_AddTag("TestTable", "ModnetIntArray[1]", "ModnetIntArray[1]", "Array","",1)
```

## Example 1

### Populating the table with a button

1. From the **Object toolbox** select -> The **Paste Genie** icon.
2. In the **Paste Genie** window select the **lib\_controls** library.
3. Select the Tag Table genie. The tag Table genie is pasted onto the graphics page.
4. In the Tag Table Genie properties dialog, configure the fields according to the table below. Click **OK** to save the changes.

Parameter	Description
TagTable Name	Tags
Width	300 ( all columns within the table are displayed. If left at 100 the value column is not displayed)
Number of rows	20
Header Height	18
Row Height	15
Header Padding	5

Parameter	Description
Row Padding	4
Column Padding	5
Horizontal Scrollbar	
Vertical Scrollbar	Verbar (name of horizontal scroll bar placed on current page)

Paste the Vertical Scrollbar genie onto the graphics page.

1. Position where the width of the table might end. In the 'Library\_Control - Scrollbar' dialog:
  - **Scrollbar Name:** Verbar
  - **Height:** 100
  - **Update on Drag:** True (default)
  - **Allow double-click:** True (default)
  - **Auto Hide:** False (default)
  - **Hidden When:** False (default)
2. Click **OK** to save the changes
3. From the Object toolbox select the 'Button' object
4. Draw a button onto the page and configure as follows:
  - **Action:** Up
  - **Up Command:** LibTagTable\_AddTag("VariableTags","Bit\_1")
5. Click **OK** to save the changes
6. Compile and Run the project.
7. At runtime open the example page and click on the 'VariableTag' button
8. The table will refresh and the Bit\_1 tag and corresponding value will be displayed.

Variations of the example include:

- Adding a second button for another Tag e.g Bit\_2
- Configuring the button to accept multiple tags( though space is limited). The Cicode for the up command could be :
 

```
LibTagTable_AddTag("VariableTags","Bit_1");
LibTagTable_AddTag("VariableTags","Bit_2");
LibTagTable_AddTag("VariableTags","Bit_4")
```
- Configuring a button to launch a form field where you can enter the name of the tag: The Cicode for the up command could be:
 

```
LibTagTable_AddTag("Variabletags", Input("Add Tag","Tag name:", ""));
```

**Example 2:****Initialized Cicode function in table configuration**

Using the Cicode sample below, open the Cicode Editor and create a function.

Example: 'MyTagTableInitialize' initialization function:

```
INT FUNCTION MyTagTableInitialize(STRING sTable)
    INT hSession;
    STRING sCluster;
    STRING sTag;
    INT nError;
    // Add all variable tags to the tag table
    hSession = TagBrowseOpen("", "Cluster,Tag", "Cluster:A,Tag:A", "");
    nError = TagBrowseFirst(hSession);
    WHILE (nError = 0) DO
        sCluster = TagBrowseGetField(hSession, "Cluster");
        sTag = TagBrowseGetField(hSession, "Tag");
        LibTagTable_AddTag(sTable, sCluster + "." + sTag);
        nError = TagBrowseNext(hSession);
    END
    IsError();
    TagBrowseClose(hSession);
    // Tell the tag table initialization is
    RETURN TRUE;
END
```

As in example 1, add the Tag Table Genie and Vertical scroll bar to the graphics page.

In the Tag Table Genie property dialog, configure the following fields:

Parameter	Description
TagTable Name	Tags
Width	300 ( all columns within the table are displayed. If left at 100 the value column is not displayed)
Number of rows	20
Header Height	18
Row Height	15
Header Padding	5
Row Padding	4
Column Padding	5
Horizontal Scrollbar	
Vertical Scrollbar	Verbar (name of horizontal scroll bar placed on current page)

Parameter	Description
Initialize	MyTagTableInitialize("#Name")

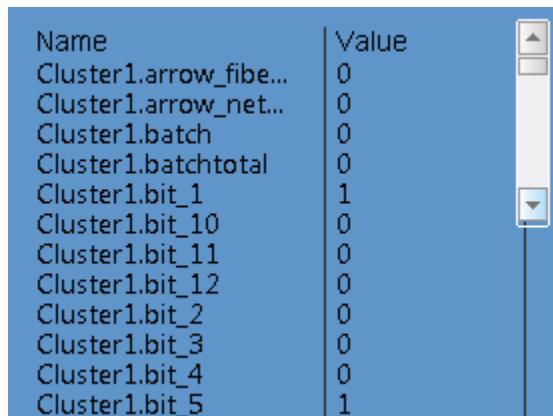
Paste the Vertical Scrollbar genie onto the graphics page. Position where the width of the table might end.

In the 'Library\_Control - Scrollbar' dialog, configure the following fields.

Parameter	Description
Scrollbar Name	Verbar
Height	100
Update on Drag	True (default)
Allow double-click	True (default)
Auto Hide	False (default)
Hidden When	False (default)

Compile and then run the project.

At runtime the table will look like the following:



A screenshot of a scrollable table control. The table has two columns: 'Name' and 'Value'. The 'Name' column lists various tags from a cluster, including arrow\_fibre, arrow\_net, batch, batchtotal, bit\_1 through bit\_12, and bit\_3 through bit\_5. The 'Value' column shows binary values (0 or 1) corresponding to each tag. A vertical scrollbar is visible on the right side of the table, indicating it can be scrolled vertically.

Name	Value
Cluster1.arrow_fibre	0
Cluster1.arrow_net	0
Cluster1.batch	0
Cluster1.batchtotal	0
Cluster1.bit_1	1
Cluster1.bit_10	0
Cluster1.bit_11	0
Cluster1.bit_12	0
Cluster1.bit_2	0
Cluster1.bit_3	0
Cluster1.bit_4	0
Cluster1.bit_5	1

## See Also

[Use the Library Controls](#)

## Tree Library Control

Tree Genie has the following parameters:

Parameter	Description
<b>Tree Name</b>	Name of the tree. Used to identify the Genie when on a graphics page.
<b>Width</b>	<p>Pixel width of the table at runtime.</p> <p><b>Note:</b> Leave the size of the Genie at design time as 100 pixels, as it will be automatically calculated at runtime.</p>
<b>Menu (Page) Name</b>	Name of page with menu items to display. If not specified it will default to the menu configuration of the current window.
<b>Number of items</b>	Number of items displayed on the page.
<b>Item Height</b>	Pixel height of item in the tree display.
<b>Show CheckBox</b>	True/False drop down box. False is selected by default. Determines whether to show the checkboxes next to each of the tree items.
<b>Force Child</b>	True/False drop down box. False is selected by default. Determines whether to force all children tree items to follow the checked status of their parent when the latter is checked / unchecked.
<b>Window Inst</b>	True/False drop down box. True is selected by default. Determines whether create a new instance of the equipment hierarchy on each window where the equipment tree is displayed.
<b>Refresh Rate</b>	<p>The minimum (quickest possible) rate that the control will be redrawn.</p> <p>The control is only redrawn as required upon changes to the layout, scrolling or other user initiated events</p>
<b>Expand Level</b>	The minimum expand level of the tree hierarchy. Set to 0 to allow the tree to be collapsed to its root items.
<b>Horizontal Scrollbar (Optional)</b>	The name of the horizontal scroll bar Genie on the page that allows the user to horizontally scroll the content displayed on the control Genie If you will not require horizontal scrolling, leave this field blank.
<b>Vertical Scrollbar (Optional)</b>	The name of the horizontal scroll bar Genie on the page that allows the user to vertically scroll the content displayed on the control Genie If you will not require vertical scrolling, leave this field blank.

Parameter	Description
<b>Label Font</b>	The font used for displaying the tree item label.
<b>Item Font</b>	The font used for displaying the optional dynamic text next to the tree item.
<b>Symbol Library (Optional)</b>	The name of the symbol library that contains the symbols for displayed next to the tree items.
<b>Refresh Rate</b>	<p>The minimum rate that the control will be redrawn.</p> <p><b>Note:</b> The control is only redrawn as required upon changes to the layout, scrolling or other user initialized events.</p>
<b>Scan Delay</b>	The number of page scan delay before the Genie is initialized at runtime. This can be set to different number on different control Genies to stagger the initialization order.
<b>Expand Level</b>	The minimum expand level of the tree hierarchy. Set this to 0 to allow the tree to be collapsed to its root items.
<b>Initialize</b>	Callback function that is called when the control initializes. The callback function should return either TRUE (1) for success or FALSE (0) for unsuccessful. By default, if the user specified function cannot initialize successfully, the control Genie will retry the function a few time before giving up. This callback only supports the #Name keyword which represents the name assigned to the control Genie.
<b>Row Item</b>	<p>Callback function that is called when the table initializes. The callback function should return a string value. The value returned will be displayed next to the tree item label. It supports the following keyword in its argument list:</p> <ul style="list-style-type: none"> <li>#Name - Name assigned to the Genie</li> <li>#Ref – the complete path of the tree item</li> <li>#Label – the label of the tree item</li> <li>#ItemDsp- The position of the item as displayed on screen</li> <li>#IsChecked - The checked status of the tree item</li> <li>#IsParent - Whether the item is expanded/collapsed</li> <li>#ItemDsp - The position of the item as displayed on screen</li> </ul>

Parameter	Description
	#Node - The handle to the internal menu node object that represents the tree item #Level - The level of the tree item
<b>Selected</b>	Callback function that is called when a tree item is selected. No particular return value is expected from this function, and it supports the same set of keywords as that of the Row Item function in its argument list. Callback function that is called when the check box of a tree item is clicked. No particular return value is expected from this function, and it supports the same set of keywords as that of the Row Item function in its argument list.
<b>Checked</b>	Callback function that is called when a tree item is selected. No particular return value is expected from this function, and it supports the same set of keywords as that of the Row Item function in its argument list. Callback function that is called when the check box of a tree item is clicked. No particular return value is expected from this function, and it supports the same set of keywords as that of the Row Item function in its argument list.
<b>Left Mouse</b>	Callback function that is called when the user clicks the left mouse button. No particular return value is expected from this function, and it supports the following keywords in its argument list: #Node - The handle to the internal menu node object that represents the tree item #Level - The level of the tree item
<b>Right Mouse</b>	Callback function that is called when the user clicks the right mouse button. It uses the same specification as that of the Left mouse callback .
<b>Double Click</b>	Callback function that is called when the user double-click the left mouse button. It uses the same specification as that of the Left mouse callback.
<b>Tree Reload</b>	Callback function that is called when the control is redrawn. No particular return value is expected from this function, and it supports the following keywords in its argument list: #Name = name assigned to the Genie

## See Also

[Use the Tree Genie](#)

[Use the Library Controls](#)

### Use the Tree Genie

Use the [Tree Library Control](#) on a page when you want to be able to view hierarchical data such as a menu.

**Note:** Configure the [Equipment Tree Library Control](#) to view equipment in your system.

The example below illustrates how the Genie can be used to view menu items within your system for a particular page. A basic example, the implementation of the tree Genie can also be used to display dynamic generated menu items using Cicode.

#### Example 1: Displaying Pre-configured Menu Items

Using the Example project, if you would like to display the menu items for the "\_plant" page, complete the following:

1. From the **Object toolbox** select the **Paste Genie** icon.
2. In the **Paste Genie** window, select the **lib\_controls** library
3. Select the Tree Genie. The Tree Genie is pasted onto the graphics page.
4. In the Tree Genie property dialog, configure the fields according to the table below. Click **OK** to save the changes.

Parameter	Description
Tree Name	PlantMenu
Width	200
Number of rows	10
Row Height	15
Show CheckBox	False (default)
Force Child	False (default)
Window Inst	True (default)
Expand Level	0
Menu (Page) Name	_Plant
Selected	MenuNodeRunCommand(#Node)

When you compile and then run the project, the tree (menu) will look like the following in the runtime environment.



### Example 2: Displaying Dynamic Menu Items

This is the more versatile usage of the menu tree where you can create an in-memory menu hierarchy and populate it with your own data, and use the Tree control to display it on the screen. The EquipTree Genie is an example of such usage.

Using the Cicode sample below, open the Cicode Editor and create a callback function.

Example: 'Mytree' initialization callback function.

```
INT FUNCTION MyTreeInitialize(STRING sTreeName)
    // get the menu name for my instance of tree control
    STRING sMenu = LibTreeGetProperty(sTreeName, "Menu");
    // create a in-memory menu hierarchy
    INT hMenu = MenuGetPageNode(sMenu, 1);
    INT hNode;
    // populate the menu hierarchy (only once)
    IF (hMenu > -1) AND (MenuGetFirstChild(hMenu) < 0) THEN
        // add level 1 item
        hNode = MenuNodeAddChild(hMenu, "Item 1", "");
        // add level 2 (sub) items
        MenuNodeAddChild(hNode, "Sub-item 1", "");
        MenuNodeAddChild(hNode, "Sub-item 2", "");
        // add level 1 item
        hNode = MenuNodeAddChild(hMenu, "Item 2", "");
        // add level 2 (sub) items
        MenuNodeAddChild(hNode, "Sub-item 1", "");
        MenuNodeAddChild(hNode, "Sub-item 2", "");
        // add level 1 item
        MenuNodeAddChild(hMenu, "Item 3", "");
    END
    // return TRUE (1) to signal initialization completes
    RETURN 1;
END
```

As in example 1, paste the Tree Genie onto a graphics page.

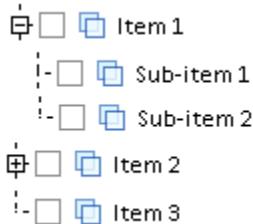
In the Tree Genie property dialog configure the following fields:

Parameter	Description
Tree Name	TreeMenu
Width	200
Number of rows	10
Show CheckBox	True
Force Child	False (default)
Window Inst	True (default)
Expand Level	0
Menu (Page) Name	MyMenu
Initialize	MyTreeInitialize("#Name")
Selected	MenuNodeRunCommand(#Node)

Assign the name of the in-memory menu hierarchy to the Menu name field of the Genie by assigning this name to the "Menu (Page) Name" field of the Tree Genie at configuration time.

Click **OK** to save the changes and save the page.

Compile and then run the project. At runtime the tree (menu) will look like the following:



## See Also

[Use the Library Controls](#)

## Include Project Specifications

This section of the help defines specifications for the predefined components included in the Plant SCADA Include project.

- [Predefined Commands](#)
- [Predefined Character Sets](#)
- [Predefined Fonts](#)
- [Predefined Devices](#)

- Predefined Cicode Files
- Predefined Color Names and Codes
- Predefined Keyboard Key Codes
- Predefined Labels.

**Note:** The Include project is a system include project, and only appears in the Project List tree in Plant SCADA Studio if the **System Projects** filter is applied in the Projects activity. See the System Projects described in [Project Types](#).

## Predefined Commands

This section describes the system keyboard commands that are predefined in the Include Project. (System keyboard commands operate on any graphics page displayed on the computer screen).

- [System Keyboard Commands Database](#)
- [Predefined Keyboard Keys](#).

### System Keyboard Commands Database

The table below gives the key sequences associated with commands and their function.

Key Sequence	Command	Description
*BS	KeyBS()	Backspace over the current key
DOWN	KeyDown()	Move the cursor down
PGDN	PagePrev()	Display the previous page
PGUP	PageNext()	Display the next page
RIGHT	KeyRight()	Move the cursor right
UP	KeyUp()	Move the cursor up

Usually you can override a predefined command by configuring a new command in your project with the same key sequence.

The only command that you cannot override is the \*BS command, as this sequence is a hotkey used to remove the last key from the key command line.

**Note:** Do not modify the Include Project. Your changes to the Include project will be lost when you reinstall Plant SCADA or upgrade to a new version.

## Predefined Keyboard Keys

The keyboard keys described below are predefined in the Include Project. You can use these keys in any key sequence field; for example, to define the keyboard commands for an object.

The table below defines the key names, codes, and description.

Key Name	Key Code	Description
BS	KEY_BACKSPACE	BackSpace key
DOWN	KEY_DOWN	Cursor down
DOWN_SHIFT	KEY_DOWN_SHIFT	Shift down key
ENTER	KEY_ENTER	Enter key
LBUTTON_DBL	KEY_LBUTTON_DBL	Left mouse button double click
LBUTTON_DN	KEY_LBUTTON_DN	Left mouse button down
LBUTTON_UP	KEY_LBUTTON_UP	Left mouse button up
LBUTTON_CMD_DN	KEY_LBTN_CMD_DN	Left mouse button down (command cursor)
LBUTTON_CMD_DNC	KEY_LBTN_CMD_DNC	Ctrl left mouse button down (command cursor)
LBUTTON_CMD_DNS	KEY_LBTN_CMD_DNS	Shift left mouse button down (command cursor)
LBUTTON_CMD_UP	KEY_LBTN_CMD_UP	Left mouse button up (command cursor)
LBUTTON_CMD_UPC	KEY_LBTN_CMD_UPC	Ctrl left mouse button up (command cursor)
LBUTTON_CMD_UPS	KEY_LBTN_CMD_UPS	Shift left mouse button up (command cursor)
LEFT	KEY_LEFT	Cursor left
MBUTTON_DN	KEY_MBUTTON_DN	Middle mouse button down
MBUTTON_UP	KEY_MBUTTON_UP	Middle mouse button up
PGDN	KEY_PGDN	Page down key
PGUP	KEY_PGUP	Page up key
RBUTTON_DN	KEY_RBUTTON_DN	Right mouse button down
RBUTTON_UP	KEY_RBUTTON_UP	Right mouse button up
RBUTTON_CMD_DN	KEY_RBTN_CMD_DN	Right mouse button down (command cursor)
RBUTTON_CMD_UP	KEY_RBTN_CMD_UP	Right mouse button up (command

Key Name	Key Code	Description
		cursor)
RIGHT	KEY_RIGHT	Cursor right
UP	KEY_UP	Cursor up
UP_SHIFT	KEY_UP_SHIFT	Shift up key

**Note:** Do not modify the Include Project. Changes to the Include project are lost when you reinstall or upgrade Plant SCADA.

---

## Predefined Character Sets

The following character sets are predefined as labels in the Include Project:

Label	Value	Description
DEFAULT_CHARSET	1	Use the default Windows character set
SHIFTJIS_CHARSET	128	Japanese character set
HANGEUL_CHARSET	129	Korean character set
GB2312_CHARSET	134	Chinese character set
CHINESEBIG5_CHARSET	136	Chinese character set
JOHAB_CHARSET	130	
HEBREW_CHARSET	177	
ARABIC_CHARSET	178	
GREEK_CHARSET	161	
TURKISH_CHARSET	162	
VIETNAMESE_CHARSET	163	
THAI_CHARSET	222	
EASTEUROPE_CHARSET	238	
RUSSIAN_CHARSET	204	
BALTIC_CHARSET	186	

**Note:** Do not modify the Include Project. Changes to the Include project are lost when you reinstall or upgrade Plant SCADA.

---

## Predefined Fonts

The following fonts are predefined in the Include Project:

Font Name	Font Type	Size	Color
AlmAccOffFont	Arial	10	White
AlmAccOnFont	Arial	10	Cyan
AlmDisabledFont	Arial	10	White
AlmUnAccOffFont	Arial	10	Brown
AlmUnAccOnFont	Arial	10	Yellow
ButtonFont	Arial	10	Black
Casanova	Arial	-10	Black
ControlLimits	Times New Roman	14	Black
DefaultFont	Courier New	14	White
DisabledFont	Arial	10	White
FontOP	Courier New	14	Light Cyan
FontPV	Courier New	14	Light Green
FontSP	Courier New	14	Light Red
FontTune	Courier New	14	Yellow
GraphBigFont	Arial	60	Black
GraphColour	Arial	32	Blue
GraphColourBig	Arial	60	Red
GraphColourSmall	Courier New	20	Black
GraphFont	Arial	32	Black
GraphSmallFont	Courier New	20	Black
HardwareFont	Arial	10	Light_Red
Pen1SpcFont	Courier	10	White
Pen1TrendFont	Courier New	14	Light_Green
Pen2SpcFont	Courier New	14	Light_Green

Font Name	Font Type	Size	Color
Pen2TrendFont	Courier New	14	Yellow
Pen3SpcFont	Courier New	14	Light_Cyan
Pen3TrendFont	Courier New	14	Light_Red
Pen4SpcFont	Courier New	14	Light_Blue
Pen4TrendFont	Courier New	14	Light_Cyan
Pen5TrendFont	Courier New	14	Light_Magenta
Pen6TrendFont	Courier New	14	White
Pen7TrendFont	Courier New	14	Light_Blue
Pen8TrendFont	Courier New	14	Gray
PromptFont	Arial	10	White
SpcFont	Courier New	14	White
TextFont	Arial	10	White
TimeFont	Arial	10	Black
TrendFont	Courier New	14	White
TrendHistFont	Courier New	14	Yellow
TrendSHistFont	Arial	-10	Magenta
TrendSFont	Arial	-10	Black
UnacceptedFont	Arial	10	Yellow
Vanuatu	Arial	-9	Black
System	Arial	10	Black
TrendSCentreFont	Arial	-10	Yellow
PopFont	Arial	9	Black

You can override a predefined font by adding a new font with the same name to your project.

---

**Note:** Do not modify the Include Project. Changes to the Include project are lost when you reinstall or upgrade Plant SCADA.

---

## Predefined Devices

This section describes devices that are predefined in the Include Project.

### Devices database

The table below shows the devices supported by Plant SCADA.

Device Name	Type	Description
ASCII_DEV	0	Ascii Device number
PRINTER_DEV	1	Printer Device number
dBASE_DEV	2	dBASE device number
SQL_DEV	4	SQL device number
AlarmDisk	0 (ASCII File)	Default alarm log file
AlarmPrint	0 (ASCII File)	Default alarm print device
KeyDisk	0 (ASCII File)	Default keyboard log file
KeyPrint	1 (Printer)	Default keyboard printer log
Printer1	1 (Printer) LPT1:	Printer 1 device
Printer2	1 (Printer) LPT2:	Printer 2 device
SummaryPrint	0 (ASCII File)	Default alarm summary printer device
SummaryDisk	0 (ASCII File)	Default alarm summary log file
_Trend	3 (dBASE)	Trend RDB device
Scratch	0 (ASCII File)	Device for DevModify function

**Note:** Do not modify the Include Project. Changes to the Include project are lost when you reinstall or upgrade Plant SCADA.

## Predefined Cicode Files

The following Cicode files are part of the Include Project:

File Name	Description
citect.ci	General utility functions
debug.ci	User Cicode debugging functions

File Name	Description
export.ci	Information functions
graph.ci	Trend data export functions
info.ci	Information functions
numpad.ci	Number entry keypad functions
page.ci	Graphics page utility functions
pareto.ci	Functions for the Pareto charts
spc.ci	Default SPC functions
spcplus.ci	SPC functions - extension
statpop.ci	Trend statistic functions
tag.ci	Functions for Tag assignment and manipulation
trend.ci	Default trend functions
trninfo.ci	Functions to gather trend information
zoom.ci	Trend zoom functions

**Note:** Do not modify the Include Project. Changes to the Include project are lost when you reinstall or upgrade Plant SCADA.

## Predefined Color Names and Codes

Sixteen standard colors are available for use with your Plant SCADA system. They have been predefined in the Include Project. refer to these colors by name which make them more readily understandable, wherever you would use the code value:

Color Label	Code
Black	0x000000
Blue	0x000080
Green	0x008000
Cyan	0x008080
Red	0x800000
Magenta	0x800080
Brown	0x808000

Color Label	Code
Grey	0xBFBFBF
Dark_Grey	0x7F7F7F
Light_Blue	0x0000FF
Light_Green	0x00FF00
Light_Cyan	0x00FFFF
Light_Red	0xFF0000
Light_Magenta	0xFF00FF
Yellow	0xFFFF00
White	0xFFFFFFFF
TRANSPARENT	0XFF000000

**Note:** Do not modify the Include Project. Changes to the Include project are lost when you reinstall or upgrade Plant SCADA.

## Predefined Keyboard Key Codes

The following meaningful key code labels are predefined in the Plant SCADA Include Project. They can be entered as key codes when you define your keyboard keys, so you don't need to remember the hex value associated with each key.

Key Code (Plant SCADA label)	Key Code (Hex Value)	Key Description
KEY_LBUTTON	0x0001	Left Mouse Button
KEY_RBUTTON	0x0002	Right Mouse Button
KEY_MBUTTON	0x0004	Middle Mouse Button
KEY_LBUTTON_UP	0x0201	Left Mouse Button Up
KEY_RBUTTON_UP	0x0202	Right Mouse Button Up
KEY_MBUTTON_UP	0x0204	Middle Mouse Button Up
KEY_LBUTTON_DBL	0x0401	Left Mouse Button Double Click
KEY_RBUTTON_DBL	0x0402	Right Mouse Button Double Click
KEY_MBUTTON_DBL	0x0403	Middle Mouse Button Double Click
KEY_LBUTTON_DN	0x0801	Left Mouse Button Down

<b>Key Code (Plant SCADA label)</b>	<b>Key Code (Hex Value)</b>	<b>Key Description</b>
KEY_RBUTTON_DN	0x0802	Right Mouse Button Down
KEY_MBUTTON_DN	0x0804	Middle Mouse Button Down
KEY_LBTN_CMD_UP	0x0601	Left Mouse Button Up (Command Cursor)
KEY_RBTN_CMD_UP	0x0602	Right Mouse Button Up (Command Cursor)
KEY_MBTN_CMD_UP	0x0604	Middle Mouse Button Up (Command Cursor)
KEY_LBTN_CMD_DN	0x0605	Left Mouse Button Down (Command Cursor)
KEY_RBTN_CMD_DN	0x0606	Right Mouse Button Down (Command Cursor)
KEY_MBTN_CMD_DN	0x0608	Middle Mouse Button Down (Command Cursor)
KEY_LBTN_CMD_UPS	0x1601	Shift Left Mouse Button Up (Command Cursor)
KEY_RBTN_CMD_UPS	0x1602	Shift Right Mouse Button Up (Command Cursor)
KEY_MBTN_CMD_UPS	0x1604	Shift Middle Mouse Button Up (Command Cursor)
KEY_LBTN_CMD_DNS	0x1605	Shift Left Mouse Button Down (Command Cursor)
KEY_RBTN_CMD_DNS	0x1606	Shift Right Mouse Button Down (Command Cursor)
KEY_MBTN_CMD_DNS	0x1608	Shift Middle Mouse Button Down (Command Cursor)
KEY_LBTN_CMD_UPC	0x2601	Ctrl Left Mouse Button Up (Command Cursor)
KEY_RBTN_CMD_UPC	0x2602	Ctrl Right Mouse Button Up (Command Cursor)
KEY_MBTN_CMD_UPC	0x2604	Ctrl Middle Mouse Button Up (Command Cursor)
KEY_LBTN_CMD_DNC	0x2605	Ctrl Left Mouse Button Down

Key Code (Plant SCADA label)	Key Code (Hex Value)	Key Description
		(Command Cursor)
KEY_RBTN_CMD_DNC	0x2606	Ctrl Right Mouse Button Down (Command Cursor)
KEY_MBTN_CMD_DNC	0x2608	Ctrl Middle Mouse Button Down (Command Cursor)
KEY_BACKSPACE	0x0008	Backspace
KEY_TAB	0x0009	Tab
KEY_LF	0x000A	Line Feed
KEY_VT	0x000B	Vertical Tab
KEY_FF	0x000C	Form Feed
KEY_RETURN	0x000D	Return
KEY_ENTER	0x000D	Enter (same key as above)
KEY_ESCAPE	0x001B	Escape
KEY_ESC	0x001B	Escape (same key as above)
KEY_DELETE	0x012E	Delete
KEY_PGUP	0x0121	PageUp
KEY_PGDN	0x0122	PageDown
KEY_END	0x0123	End
KEY_HOME	0x0124	Home
KEY_LEFT	0x0125	Cursor Left
KEY_UP	0x0126	Cursor Up
KEY_RIGHT	0x0127	Cursor Right
KEY_DOWN	0x0128	Cursor Down
KEY_LEFT_SHIFT	0x1125	Shift Left
KEY_UP_SHIFT	0x1126	Shift Up
KEY_RIGHT_SHIFT	0x1127	Shift Right
KEY_DOWN_SHIFT	0x1128	Shift Down

<b>Key Code (Plant SCADA label)</b>	<b>Key Code (Hex Value)</b>	<b>Key Description</b>
KEY_INSERT	0x012D	Insert
KEY_HELP	0x012F	Help
KEY_F1	0x0170	F1
KEY_F2	0x0171	F2
KEY_F3	0x0172	F3
KEY_F4	0x0173	F4
KEY_F5	0x0174	F5
KEY_F6	0x0175	F6
KEY_F7	0x0176	F7
KEY_F8	0x0177	F8
KEY_F9	0x0178	F9
KEY_F10	0x0179	F10
KEY_F11	0x017A	F11
KEY_F12	0x017B	F12
KEY_F13	0x017C	F13
KEY_F14	0x017D	F14
KEY_F15	0x017E	F15
KEY_F16	0x017F	F16
KEY_F1_SHIFT	0x1170	Shift F1
KEY_F2_SHIFT	0x1171	Shift F2
KEY_F3_SHIFT	0x1172	Shift F3
KEY_F4_SHIFT	0x1173	Shift F4
KEY_F5_SHIFT	0x1174	Shift F5
KEY_F6_SHIFT	0x1175	Shift F6
KEY_F7_SHIFT	0x1176	Shift F7
KEY_F8_SHIFT	0x1177	Shift F8

Key Code (Plant SCADA label)	Key Code (Hex Value)	Key Description
KEY_F9_SHIFT	0x1178	Shift F9
KEY_F10_SHIFT	0x1179	Shift 10
KEY_F11_SHIFT	0x117A	Shift F11
KEY_F12_SHIFT	0x117B	Shift F12
KEY_F13_SHIFT	0x117C	Shift F13
KEY_F14_SHIFT	0x117D	Shift F14
KEY_F15_SHIFT	0x117E	Shift F15
KEY_F16_SHIFT	0x117F	Shift F16
KEY_F1_CTRL	0x2170	Ctrl F1
KEY_F2_CTRL	0x2171	Ctrl F2
KEY_F3_CTRL	0x2172	Ctrl F3
KEY_F4_CTRL	0x2173	Ctrl F4
KEY_F5_CTRL	0x2174	Ctrl F5
KEY_F6_CTRL	0x2175	Ctrl F6
KEY_F7_CTRL	0x2176	Ctrl F7
KEY_F8_CTRL	0x2177	Ctrl F8
KEY_F9_CTRL	0x2178	Ctrl F9
KEY_F10_CTRL	0x2179	Ctrl F10
KEY_F11_CTRL	0x217A	Ctrl F11
KEY_F12_CTRL	0x217B	Ctrl F12
KEY_F13_CTRL	0x217C	Ctrl F13
KEY_F14_CTRL	0x217D	Ctrl F14
KEY_F15_CTRL	0x217E	Ctrl F15
KEY_F16_CTRL	0x217F	Ctrl F16
KEY_A_SHIFT	0x1041	Shift A
KEY_B_SHIFT	0x1042	Shift B

<b>Key Code (Plant SCADA label)</b>	<b>Key Code (Hex Value)</b>	<b>Key Description</b>
KEY_C_SHIFT	0x1043	Shift C
KEY_D_SHIFT	0x1044	Shift D
KEY_E_SHIFT	0x1045	Shift E
KEY_F_SHIFT	0x1046	Shift F
KEY_G_SHIFT	0x1047	Shift G
KEY_H_SHIFT	0x1048	Shift H
KEY_I_SHIFT	0x1049	Shift I
KEY_J_SHIFT	0x104A	Shift J
KEY_K_SHIFT	0x104B	Shift K
KEY_L_SHIFT	0x104C	Shift L
KEY_M_SHIFT	0x104D	Shift M
KEY_N_SHIFT	0x104E	Shift N
KEY_O_SHIFT	0x104F	Shift O
KEY_P_SHIFT	0x1050	Shift P
KEY_Q_SHIFT	0x1051	Shift Q
KEY_R_SHIFT	0x1052	Shift R
KEY_S_SHIFT	0x1053	Shift S
KEY_T_SHIFT	0x1054	Shift T
KEY_U_SHIFT	0x1055	Shift U
KEY_V_SHIFT	0x1056	Shift V
KEY_W_SHIFT	0x1057	Shift W
KEY_X_SHIFT	0x1058	Shift X
KEY_Y_SHIFT	0x1059	Shift Y
KEY_Z_SHIFT	0x105A	Shift Z
KEY_A_CTRL	0x2041	Ctrl A
KEY_B_CTRL	0x2042	Ctrl B

Key Code (Plant SCADA label)	Key Code (Hex Value)	Key Description
KEY_C_CTRL	0x2043	Ctrl C
KEY_D_CTRL	0x2044	Ctrl D
KEY_E_CTRL	0x2045	Ctrl E
KEY_F_CTRL	0x2046	Ctrl F
KEY_G_CTRL	0x2047	Ctrl G
KEY_H_CTRL	0x2048	Ctrl H
KEY_I_CTRL	0x2049	Ctrl I
KEY_K_CTRL	0x204B	Ctrl K
KEY_L_CTRL	0x204C	Ctrl L
KEY_M_CTRL	0x204D	Ctrl M
KEY_N_CTRL	0x204E	Ctrl N
KEY_O_CTRL	0x204F	Ctrl O
KEY_P_CTRL	0x2050	Ctrl P
KEY_Q_CTRL	0x2051	Ctrl Q
KEY_R_CTRL	0x2052	Ctrl R
KEY_S_CTRL	0x2053	Ctrl S
KEY_T_CTRL	0x2054	Ctrl T
KEY_U_CTRL	0x2055	Ctrl U
KEY_V_CTRL	0x2056	Ctrl V
KEY_W_CTRL	0x2057	Ctrl W
KEY_X_CTRL	0x2058	Ctrl X
KEY_Y_CTRL	0x2059	Ctrl Y
KEY_Z_CTRL	0x205A	Ctrl Z
KEY_A_ALT	0x4041	Alt A
KEY_B_ALT	0x4042	Alt B
KEY_C_ALT	0x4043	Alt C

Key Code (Plant SCADA label)	Key Code (Hex Value)	Key Description
KEY_D_ALT	0x4044	Alt D
KEY_E_ALT	0x4045	Alt E
KEY_F_ALT	0x4046	Alt F
KEY_G_ALT	0x4047	Alt G
KEY_H_ALT	0x4048	Alt H
KEY_I_ALT	0x4049	Alt I
KEY_J_ALT	0x404A	Alt J
KEY_K_ALT	0x404B	Alt K
KEY_L_ALT	0x404C	Alt L
KEY_M_ALT	0x404D	Alt M
KEY_N_ALT	0x404E	Alt N
KEY_O_ALT	0x404F	Alt O
KEY_P_ALT	0x4050	Alt P
KEY_Q_ALT	0x4051	Alt Q
KEY_R_ALT	0x4052	Alt R
KEY_S_ALT	0x4053	Alt S
KEY_T_ALT	0x4054	Alt T
KEY_U_ALT	0x4055	Alt U
KEY_V_ALT	0x4056	Alt V
KEY_W_ALT	0x4057	Alt W
KEY_X_ALT	0x4058	Alt X
KEY_Y_ALT	0x4059	Alt Y
KEY_Z_ALT	0x405A	Alt Z

To define a key with:

1. The Shift key, add 0x1000 to the value of the key.
2. The Ctrl key, add 0x2000 to the value of the key.
3. The Alt key, add 0x4000 to the value of the key.

The above key definitions are standard IBM-compatible keys.

**Note:** Do not modify the Include Project. Changes to the Include project are lost when you reinstall or upgrade Plant SCADA.

## Predefined Labels

The table below defines the labels predefined in the Include project's labels database (labels.dbf).

Name	Expression	Comment
__DATE__	\$1	Date of compilation
__DB__	\$4	Compiler database name
__FIELD__	\$6	Compiler field name
__FILE__	\$2	Compiler file name
__LINE__	\$3	Compiler line number
__RECORD__	\$5	Compiler record number
__TIME__	\$0	Time of compilation
_BLANK_		NULL Definition
AlarmFirstCatRec(hCat,nType,hArea=-1)	_AlarmQueryFirstRec(hCat,nType,hArea,0)	Get Alarm Cat Rec with Area
AlarmFirstPriRec(hPri,nType,hArea=-1)	_AlarmQueryFirstRec(hPri,nType,hArea,1)	Get Alarm Pri Rec with Area
AlarmNextCatRec(hRec,hCat,nType,hArea=-1)	_AlarmQueryNextRec(hRec,hCat,nType,hArea,0)	Get Alarm Cat Rec with Area
AlarmNextPriRec(hRec,hPri,nType,hArea=-1)	_AlarmQueryNextRec(hRec,hPri,nType,hArea,1)	Get Alarm Pri Rec with Area
ANIMATE	2	Display mode 2
ANM_ARRAY	16	Animated symbols in array mode
ANSI_CHARSET	0	ANSI character set
Arg1	GetGlbStr(0)	keyboard argument 1
Arg2	GetGlbStr(1)	keyboard argument 2
Arg3	GetGlbStr(2)	keyboard argument 3
Arg4	GetGlbStr(3)	keyboard argument 4

Name	Expression	Comment
Arg5	GetGlbStr(4)	keyboard argument 5
Arg6	GetGlbStr(5)	keyboard argument 6
Arg7	GetGlbStr(6)	keyboard argument 7
Arg8	GetGlbStr(7)	keyboard argument 8
ArgValue1	StrToValue(Arg1)	Get the value of argument 1
Assert(arg)	IF NOT (arg) THEN _Assert (#arg, __FILE__, __LINE__); END	Process an assertion
BAD_HANDLE	-1	Bad Handle
BORDER	2	Border Only
BORDER_3D	1	3D Transparent Button
_CreateControlObject(sCls,sName,x1,y1,x2,y2,sEventCls)	_CreateControlObject(sCls,sName,x1,y1,x2,y2,sEventCls)	_CreateControlObject default event class
DateDay(time)	_TimeSub(time,3)	Get days from time
DateDayMonth(time)	_TimeSub(time,10)	Get the last day of the month
DateMonth(time)	_TimeSub(time,5)	Get month from time
DateWeekDay(time)	_TimeSub(time,4)	Get weekday from time
DELETE_ANM	000	Delete animation
DevFirst(hDev)	DevSeek(hDev,0)	DevSeek with Offset=0
DspButton(hAn,UK=0,sText,hFont=0,nW=0,nH=0,DK=0,RK=0,nM=0)	_DspButton(hAn,UK,sText,hFont,nW,nH,DK,RK,nM)	Display button
DspButtonFn(hAn,UF=0,sText,hFont=0,nW=0,nH=0,DF=0,RF=0,nM=0)	_DspButtonFn(hAn,UF,sText,hFont,nW,nH,DF,RF,nM)	Display a button
DspSymAnmEx(hAn,mode,s1,s2=0,s3=0,s4=0,s5=0,s6=0,s7=0,s8=0,s9=0)	_DspSymAnm(hAn,s1,s2,s3,s4,s5,s6,s7,s8,mode,s9)	DspSymAnm with mode
EVEN_P	2	Even Parity
Exec(sText,mode=1)	_Exec(sText,mode)	Exec program, default to normal

Name	Expression	Comment
FALSE	0	Boolean False
FlashColourState()	StrToInt(PageInfo(18))	Flashing Color State as a boolean
GetBlueValue(PackedRGB)	((PackedRGB) / 65536) BITAND 255	Get the blue component of a packed RGB color
GetGreenValue(PackedRGB)	((PackedRGB) / 256) BITAND 255	Get the green component of a packed RGB color
GetRedValue(PackedRGB)	((PackedRGB) BITAND 255)	Get the red component of a packed RGB color
GetVar(sTag,sField)	\$7	Get variable field data
GetVarDef(sTag,sField,sDefault)	\$10	Get variable field data if defined
GetVarStr(sTag,sField)	\$8	Get variable field data as str
GetVarStrDef(sTag,sField,sDefault)	\$11	Get variable field data as a str if defined
GRAY_ALL	3	Gray the entire button
GRAY_HIDE	4	Hide object when grayed
GRAY_PART	2	Sink and gray the text / symbol
GRAY_SUNK	1	Sink the text / symbol
IFDEF(sTag,sTrue,sFalse)	\$9	Inline IF defined macro
IFDEFDIGALM(sTag,sTrue,sFalse)	\$12	Inline IF defined macro
IFDEFADVALM(sTag,sTrue,sFalse)	\$13	Inline IF defined macro
IFDEFANAALM(sTag,sTrue,sFalse)	\$14	Inline IF defined macro
InAnimationCycle()	StrToInt(PageInfo(19))	In Animation Cycle as a boolean
InCommunicationsCycle()	StrToInt(PageInfo(20))	In Communications Cycle as a boolean
KeyDown()	KeyMove(4)	Move Cursor down
KeyLeft()	KeyMove(1)	Move Cursor left
KeyReplayAll()	_KeyReplay(0)	Key Replay All
KeyRight()	KeyMove(2)	Move Cursor right

Name	Expression	Comment
KeyUp()	KeyMove(3)	Move Cursor up
MakeCitectColour(Red, Green, Blue)	MakeColour(Red, Green, Blue)	Alias for <a href="#">MakeColour</a>
NONE	0	No Parity
NORMAL	0	Normal Button
ObjectByName(sName)	ObjectByNameEx(WinNumber(),sName)	Get ActiveX object on current page by name
ODD_P	1	Odd Parity
OVERLAP	1	Display mode 1
Print(sText,nMode=0)	DevPrint(DevCurr(),sText,nMode)	Print output to device
PrintLn(sText)	DevPrint(DevCurr(),sText,1)	Print output to device, newline
Pulse(arg)	arg = TRUE; Sleep(2); arg = FALSE;	Pulse the variable
RAboveUCL	8192	
RBelowLCL	16384	
ROutsideCL	4096	
Shutdown(sDest="",sProject="",nMode=1)	_Shutdown(sDest,sProject,nMode)	Shutdown macro
SQLNoFields(hSQL)	SQLNumFields(hSQL)	Get the number of fields
SOFT	0	Display mode 0
TARGET	3	Screen Target
TestRandomWave(p=60,lo=0,hi=100,off=0)	_Wave(4,p,lo,hi,off)	Test random wave
TestSawWave(p=60,lo=0,hi=100,off=0)	_Wave(3,p,lo,hi,off)	Test Saw wave
TestSinWave(p=60,lo=0,hi=100,off=0)	_Wave(0,p,lo,hi,off)	Test sin wave
TestSquareWave(p=60,lo=0,hi=100,off=0)	_Wave(1,p,lo,hi,off)	Test square wave
TestTriangWave(p=60,lo=0,hi=100,o	_Wave(2,p,lo,hi,off)	Test Triag wave

Name	Expression	Comment
ff=0)		
TimeHour(time)	_TimeSub(time,0)	Get hours from time
TimeMidNight(time)	_TimeSub(time, 7)	Extract time at midnight
TimeMin(time)	_TimeSub(time,1)	Get minutes from time
TimeSec(time)	_TimeSub(time,2)	Get seconds from time
TimeSecond(time)	_TimeSub(time, 2)	Get seconds from time
TimeYearDay(time)	_TimeSub(time, 8)	
Toggle(arg)	arg = NOT arg;	Toggle the variable
TRN_EVENT	2	Event trend
TRN_PERIODIC	1	Periodic trend
TRN_PERIODIC_EVENT	3	Periodic Event trend
TRUE	1	Boolean True
UnitControl(IODev,Type,Data)	IODeviceControl(IODev,Type,Data)	
UnitInfo(IODev,Type)	IODeviceInfo(IODev,Type)	
UnitStats()	IODeviceStats()	
VarToArrayIndex(sTag)	\$16	Get the array index of a tag as a value
VarToStr(sTag)	\$15	Convert a tag reference to a string
WRITE_ON_DRAG	1	Write mode for slider
WRITE_ON_DROP	0	Write mode for slider
XAboveUCL	4	
XBelowLCL	8	
XDownTrend	64	
XErratic	512	
XFreak	1	
XGradualDown	256	

Name	Expression	Comment
XGradualUp	128	
XMixture	2048	
XOutsideCL	2	
XOutsideWL	16	
XStratification	1024	
XUpTrend	32	

**Note:** Do not modify the Include Project. Changes to the Include project are lost when you reinstall or upgrade Plant SCADA.

---

# Extensibility

In this section you will find information about the following:

- CtAPI
- Graphics Builder Automation Interface
- Using External Databases
- Exchanging Data with Other Applications
- Integration with Historian
- Using Microsoft Excel to Edit DBF Tables
- Using an OPC AE Server
- Using an OPC DA Server.

## Components

You can also extend Plant SCADA using the following languages and libraries:

- **Cicode**

Cicode is a programming language designed for use in Plant SCADA to monitor and control plant equipment. It is a structured language similar to Visual Basic or C.

For more information, see [Cicode Reference](#).

- **.Net Libraries**

You can incorporate .Net objects into a project, allowing you to extend Plant SCADA with components that have been developed externally.

To add .Net objects to your project, see [Cicode Reference](#).

- **ActiveX Objects**

You can incorporate ActiveX objects into a project, allowing you to use extend Plant SCADA with components that have been developed externally.

For example, you can:

- Add an ActiveX object to a graphics page (see [Add an ActiveX Object](#))
- Call an ActiveX object using Cicode (see [ActiveX Functions](#) in the Cicode Reference).

---

**Note:** ActiveX objects are not supported on a 64-bit process, such as an alarm server operating in Extended Memory mode. If a call to an ActiveX Cicode function occurs from a 64-bit process, an error code will be returned, a hardware alarm will be raised and the Cicode thread will stop.

---

- **VBA**

You can use VBA to extend Plant SCADA. VBA is a Visual Basic for Applications (VBA) and VBScript-compatible Basic scripting language. Plant SCADA has embedded support for VBA.

For more information, see [VBA Programming Reference](#).

---

**Note:** VBA is not supported on a 64-bit process, such as an alarm server operating in Extended Memory

---

---

mode. If a call to a VBA Cicode function occurs from a 64-bit process, an error code will be returned, a hardware alarm will be raised and the Cicode thread will stop.

---

## CtAPI

CtAPI allows access to Plant SCADA's tag data via a DLL interface. It allows third party developers to create applications in C (or other languages) to read and write to I/O devices, to use data query, and to call Cicode functions.

### Building Applications

To build CtAPI client applications, you need runtime binaries, a DLL helper static library (CtAPI.lib) and a header file (CtAPI.h) that defines data structures and functions. Configure your project to use these files by following these steps:

1. Check that the files necessary to build your application that calls CtAPI functions are ready on your development environment as per your target platform (x86 or x64), by deploying these files into your project location:
  - All the runtime binaries packaged in the files **CtAPI.Win32.Redist.zip** and **CtAPI.x64.Redist.zip** (see *Running CtAPI Applications* below).
  - Development specific files in [bin] directory for x86 platform or in [bin]\Bin (x64)\ directory for x64 platform:
    - CtAPI.lib
    - CtAPI.
2. Add CtAPI.lib on your imported static library (dependency) list, to load up CtAPI.dll and its satellite binaries at application startup.
3. Include CtAPI.h on your source files that use CtAPI functions.

---

**Note:** This procedure may differ based on your development environment. Contact your system administrator or system provider if more detailed information is required.

---

### Running CtAPI Applications

To run a CtAPI application, the program needs to be able to locate CtAPI.dll and its dependencies. There are several ways to achieve this:

- If you are running your application on a computer where Plant SCADA is installed, you can copy your program and its dependencies to the Plant SCADA Bin directory.
- You can copy the set of required binaries into the same directory as your application. These binaries are packaged in the files **CtAPI.Win32.Redist.zip** and **CtAPI.x64.Redist.zip**, which can be found in the following directory on the Plant SCADA installation media:  
  \AVEVA Plant SCADA2023\Extras\CtAPI  
You will need to use this approach if you want to use CtAPI on a computer that does not have Plant SCADA installed.
- If the above options are not possible (for example, you are using CtAPI from VBA in a Microsoft™ Office

application such as Excel™), then you can add the Plant SCADA Bin directory to the %PATH% environment variable. This allows your application to locate CtAPI.dll, however, certain dependencies will not be located this way because Windows will not use the path to locate .NET managed assemblies.

To resolve this, call the CtAPI function [ctSetManagedBinDirectory](#) with the path to the Plant SCADA Bin directory before your call to [ctOpen](#). If Plant SCADA is not installed on the computer running the application, then extract the binaries from the above zip file to another directory and use that directory in place of the Bin directory.

---

**Note:** You can encrypt the connection between a remote CtAPI client and Plant SCADA. For more information, see [Secure the Connection from a Remote CtAPI Application](#).

---

The following needs to be installed on remote CtAPI clients:

- .NET 4.8
- Microsoft Visual C++ 2015-2022 Redistributable (x86)

Or:

Microsoft Visual C++ 2015-2022 Redistributable (x64)

---

**Note:** This API has not been designed to be called from Plant SCADA Cicode DLL functions or from a Plant SCADA protocol driver. If you attempt this, it may cause a deadlock condition to occur. If you need to call the CtAPI from a protocol driver, create a new Win32 thread to call the API. You cannot call the CtAPI from the Cicode DLL interface.

---

## See Also

[I/O Point Count](#)

[CtAPI Synchronous Operation](#)

[Reading Data Using the CtAPI Functions](#)

[Secure the Connection from a Remote CtAPI Application](#)

[Error Codes](#)

[Debug Tracing](#)

[CtAPI Functions Reference](#)

## I/O Point Count

Physical I/O Device tags read or written to, using the CtAPI are counted as dynamic Plant SCADA points. If the point limit is exceeded by making calls to this interface, those calls will not succeed and Plant SCADA will not be allocated any more dynamic points.

---

**Note:** Plant SCADA licensing works on the basis of how many points you use. Every tag in your system has the potential to add to your point count. It is important to remember this and plan your system properly; otherwise you may exceed your point limit.

---

The point limit is the maximum number of I/O Device addresses (variable tags) that can be read, and is specified by your Plant SCADA license. Plant SCADA counts I/O Device addresses dynamically at runtime. This includes tags used by alarms, trends, reports, events, pages, in Super Genies, use of the [TagRead\(\)](#) and [TagWrite\(\)](#) Cicode functions, or read or write using DDE, ODBC, or the CtAPI.

It does not count any points statically at compile time.

When your system is running, any new use of tags through Super Genies, DDE, ODBC, or CtAPI can potentially add to your dynamic point count.

## See Also

[CtAPI Synchronous Operation](#)

## CtAPI Synchronous Operation

The Plant SCADA CTAPI supports both synchronous and asynchronous (or overlapped) operations. The [ctCiCode](#), [ctListRead](#), and [ctListWrite](#) functions can be performed either synchronously or asynchronously. The [ctTagRead](#) and [ctTagWrite](#) functions can be performed synchronously only.

When a function is executed synchronously, it does not return until the operation has been completed. This means that the execution of the calling thread can be blocked for an indefinite period while it waits for a time-consuming operation to finish. A function called for an overlapped operation can return immediately, even though the operation has not been completed. This enables a time-consuming I/O operation to be executed in the background while the calling thread is free to perform other tasks. For example, a single thread can perform simultaneous operations on different handles, or even simultaneous read and write operations on the same handle. To synchronize its execution with the completion of the overlapped operation, the calling thread uses the [ctGetOverlappedResult](#) function or one of the wait functions to determine when the overlapped operation has been completed. You can also use the [ctHasOverlappedIoCompleted](#) macro to poll for completion.

To call a function to perform an overlapped operation, the calling thread needs to specify a pointer to a **CTOVERLAPPED** structure. If this pointer is NULL, the function return value may incorrectly indicate that the operation completed. The **CTOVERLAPPED** structure needs to contain a handle to a manual-reset, not an auto-reset event object. The system sets the state of the event object to non-signaled when a call to the I/O function returns before the operation has been completed. The system sets the state of the event object to signaled when the operation has been completed.

When a function is called to perform an overlapped operation, it is possible that the operation will be completed before the function returns. When this happens, the results are handled as if the operation had been performed synchronously. If the operation was not completed, however, the function's return value is FALSE, and the [GetLastError\(\)](#) function returns ERROR\_IO\_PENDING.

A thread can manage overlapped operations by either of two methods:

- Use the [ctGetOverlappedResult](#) function to wait for the overlapped operation to be completed.
- Specify a handle to the **CTOVERLAPPED** structure's manual-reset event object in one of the wait functions and then call [ctGetOverlappedResult](#) after the wait function returns. The [ctGetOverlappedResult](#) function returns the results of the completed overlapped operation, and for functions in which such information is appropriate, it reports the actual number of bytes that were transferred.

When performing multiple simultaneous overlapped operations, the calling thread needs to specify a **CTOVERLAPPED** structure with a different manual-reset event object for each operation. To wait for any one of the overlapped operations to be completed, the thread specifies the manual-reset event handles as wait criteria in one of the multiple-object wait functions. The return value of the multiple-object wait function indicates which manual-reset event object was signaled, so the thread can determine which overlapped operation caused the wait operation to be completed.

You can cancel a pending asynchronous operation using the `ctCancelIO` function. Pending asynchronous operations are canceled when you call `ctClose`.

**Note:** Due to a session isolation feature introduced in an earlier version of Windows, CTAPI applications running as a service may be unable to connect. To overcome this, in the CTAPI application, specify an IP Address (which can be 127.0.0.1) and a valid username and password.

## See Also

[Reading Data Using the CtAPI Functions](#)

# Reading Data Using the CtAPI Functions

## I/O tags interface

The Plant SCADA I/O Server is designed on a client read on demand basis. The Plant SCADA I/O Server will read I/O tags from the I/O Devices when requested to by a Client. This reduces the load on the I/O Devices and increases the overall system performance.

The client interface to the real time data is more complex as the client needs to wait for a physical I/O cycle to complete before the data can be used. The client needs to request the data it requires from the I/O Server and then wait up to several seconds while the I/O Server reads the requested data. This design is reflected in the operation of the CTAPI interface. Using CTAPI to read a tag can take several seconds to complete. It is up the caller to allow for this in their design in calling this interface.

If you need to use a polling type of service, use the `ctList` functions.

## The Tag functions

The simplest way to read data is via the `ctTagRead` function. This function reads the value of a single variable, and the result is returned as a formatted engineering string.

## List functions

The List functions provide a higher level of performance for reading data than the tag based interface. The List functions also provide support for overlapped operations.

The List functions allow a group of tags to be defined and then read as a single request. They provide a simple tag based interface to data which is provided in formatted engineering data. You may create several lists and control each individually.

Tags can be added to, or deleted from lists dynamically, even if a read operation is pending on the list.

## Array support

Arrays are supported in the tag functions `ctTagWrite()`, and `ctTagRead()`. These functions can take the singular tag name as "PV123", or use the array syntax as "Recipe[10]". When the array syntax is used in the "Recipe[10]" example, the single value can be read or written to, not the entire array.

## Bit shifting when reading digital arrays

When digital types are read, Plant SCADA may adjust the starting position of the first point. This is done to improve the performance of the digital read. For example, if you start reading an array of digital values, Plant SCADA may read several digitals before the start of the array, and the data will be offset. When Plant SCADA shifts the bits, extra data will be read from the I/O Device. Plant SCADA may shift the data up to 15 bits, resulting in an extra 2 bytes of buffer space necessary for reads. Therefore, always use digital buffers which contain 2 bytes extra.

## See Also

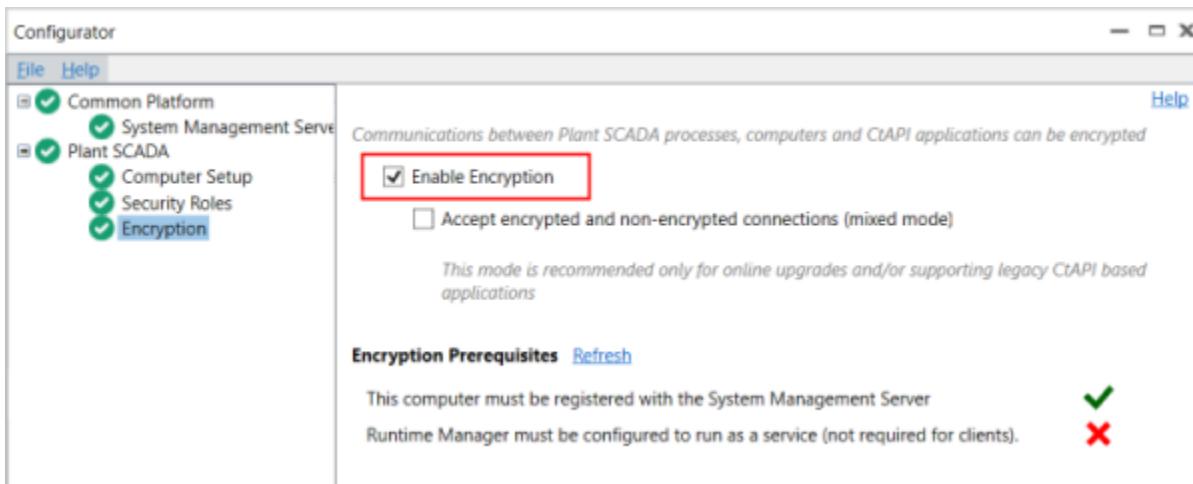
[CtAPI Functions Reference](#)

## Secure the Connection from a Remote CtAPI Application

You can encrypt the connection between a remote computer running a CtAPI application and Plant SCADA's servers.

### To enable encryption for a remote CtAPI client application:

1. Copy the set of required binaries into the same directory as the CtAPI client application (see *Running CtAPI Applications* in the topic [CtAPI](#)).
2. If Plant SCADA is not installed on the computer where the CtAPI application is hosted, install a runtime-only client (see [Install a Runtime-only Client](#)).
3. Use Configurator to connect the computer to the System Management Server you use to manage the required certificates for encryption (see [Connect a Computer to a System Management Server](#)).
4. On Configurator's **Encryption** page, select **Enable Encryption**. Do not select **Accept encrypted and non-encrypted connections (mixed mode)**.



**Note:** Runtime Manager does not need to be configured to run as a service as it is not required to run a CtAPI application.

5. Confirm that the Windows user account under which the CtAPI application will run is part of the SCADA.RuntimeUsers Windows group on the CtAPI server machine, or a member of a Windows Group that is associated with the SCADA.RuntimeUsers group. For more information, see [Security Roles](#).

6. Within the `ctOpen` command, set the `sComputer` argument to the computer name of the remote CTAPI server. Do not use an IP Address for `sComputer`. If you do, encryption will not be possible as the certificate is bound to the computer name.

**Note:** Using a hostname to connect to a CTAPI server running Citect SCADA 8.20 or earlier will not work if Windows is configured to return an IPv6 address for that hostname, as 8.20 and earlier do not support IPv6 network addresses. To avoid this, configure Windows to return an IPv4 address in preference to an IPv6 address.

## See Also

[CtAPI Functions Reference](#)

[Error Codes](#)

## Error Codes

The error codes returns from the CtAPI functions are the Microsoft WIN 32 error codes. These error codes are documented in the Microsoft SDKs. Where the error code is a Plant SCADA special error code, the error code is added to the value `-ERROR_USER_DEFINED_BASE`.

**Note:** If a CtAPI function returns the error 233, it typically means the connection to the client is not established. However, it may also mean the client has not logged in correctly. Check for both scenarios.

## Example

```
int bRet = ctTagWrite(hCTAPI, "SP123", "12.34");
if (bRet == 0) {
    dwStatus = GetLastError();
    if (dwStatus < ERROR_USER_DEFINED_BASE) {
        // Microsoft error codes see ERROR.H
    } else {
        short status;
        // status is thePlant SCADA error codes, see Plant SCADA help
        status = dwStatus - ERROR_USER_DEFINE_BASE;
    }
}
```

The following defines have been declared to make this checking easier:

```
IsPlantSCADAError(dwStatus) // test if Plant SCADA error
WIN32_TO_CT_ERROR(dwStatus) // Convert to Plant SCADA status
```

For example:

```
if (IsPlantSCADAError(dwStatus)) {
    short status;
    // status is the Plant SCADA error codes, see Plant SCADA help
    status = WIN32_TO_CT_ERROR(dwStatus);
}
```

If the connection is lost between your application and Plant SCADA, close the connection and reopen. An inoperative connection will be shown by the returning of a Microsoft error code. If a Plant SCADA status error is returned, the connection has not been lost. The command requested is invalid and the connection does not have to be closed and reopened.

```
int bRet = ctTagWrite(hCTAPI, "SP123", "12.34");
if (bRet == 0) {
    dwStatus = GetLastError();
    if (dwStatus < ERROR_USER_DEFINED_BASE) {
        ctClose(hCTAPI);
        hCTAPI = ctOpen(NULL, NULL, NULL, 0);
        while (hCTAPI == NULL) {
            Sleep(2000); // wait a while
            hCTAPI = ctOpen(NULL, NULL, NULL, 0);
        }
    }
}
```

When the connection between your application and Plant SCADA is lost, any pending overlapped commands will time out and be canceled by CtAPI. You need to destroy handles which are associated with the connection.

In Version 5.10, the CT\_OPEN\_RECONNECT mode was added to ctOpen(). When this mode is enabled, CtAPI will attempt to re-establish the connection to Plant SCADA if a communication interruption occurs. Handles created with the connection will remain valid when the connection is re-created. While the connection is down, functions will be ineffective and will report errors.

## See Also

[Debug Tracing](#)

## Debug Tracing

Debug tracing of the CtAPI has been added to the kernel. You may enable the debug trace with the command CtAPI 1 in the main kernel window. CtAPI 0 will disable the debug tracing. You may also enable the debug tracking by setting the CITECT.INI parameter:

```
[CTAPI]
Debug=1
```

The debug tracing will display each client CtAPI traffic to Plant SCADA. This tracing may slow down the performance of Plant SCADA and the CtAPI client if a large amount of communication is occurring.

The debug trace is displayed in the main Plant SCADA kernel window and is logged to the syslog.dat file.

## See Also

[\[CtAPI\]Debug](#)

## CtAPI Functions Reference

The CtAPI functions allow access to Plant SCADA I/O variable tags via a DLL interface. This allows third-party developers to create applications in C or other languages to read and write to the I/O Devices.

Function	Argument(s)	Type	Description
<a href="#">ctCancelIO</a>	hCTAPI, pctOverlapped	Boolean	Cancels a pending overlapped I/O operation.
<a href="#">ctCiCode</a>	hCTAPI, sCmd, hWin, nMode, sResult, dwLength, pctOverlapped	DWORD	Executes a Cicode function.
<a href="#">ctClientCreate</a>	()	n/a	Initializes the resources for a new CtAPI client instance
<a href="#">ctClientDestroy</a>	hCTAPI	Boolean	The handle to the CTAPI as returned from <a href="#">ctOpen()</a> .
<a href="#">ctClose</a>	hCTAPI	Boolean	Closes a connection to the Plant SCADA API.
<a href="#">ctCloseEx</a>	hCTAPI,bDestroy	Handle	The handle to the CTAPI as returned from <a href="#">ctOpen()</a> .
<a href="#">ctEngToRaw</a>	pResult, dValue, pScale, dwMode	Boolean	Converts the engineering scale variable into raw I/O Device scale.
<a href="#">ctFindClose</a>	hnd	Boolean	Closes a search initiated by <a href="#">ctFindFirst()</a> .
<a href="#">ctFindFirst</a>	hCTAPI, szTableName, szFilter, pObjHnd, dwFlags	Handle	Searches for the first object in the specified database which satisfies the filter string.
<a href="#">ctFindFirstEx</a>	hCTAPI, szTableName, szFilter, szCluster, pObjHnd, dwFlags	Handle	Searches for the first object in the specified database which satisfies the filter string specified by cluster.
<a href="#">ctFindNext</a>	hnd, pObjHnd	Boolean	Retrieves the next object in a search initiated by <a href="#">ctFindFirst()</a> .
<a href="#">ctFindNumRecords</a>	hnd	Boolean	Gets the number of records for a given browsing session.
<a href="#">ctFindPrev</a>	hnd, pObjHnd	Boolean	Retrieves the previous

Function	Argument(s)	Type	Description
			object in a search initiated by ctFindFirst().
ctFindScroll	hnd, dwMode, dwOffset, pObjHnd	Handle	Scrolls to the necessary object in a search initiated by ctFindFirst().
ctGetOverlappedResult	hCTAPI, lpctOverlapped, pBytes, bWait	Boolean	Returns the results of an overlapped operation.
ctGetProperty	hnd, szName, pData, dwBufferLength, dwResultLength, dwType	Boolean	Retrieves an object property.
ctHasOverlappedIoCompleted	lpctOverlapped	Boolean	Checks for the completion of an outstanding I/O operation.
ctListAdd	hList, sTag	Handle	Adds a tag to the list.
ctListAddEx	hList, sTag, bRaw, nPollPeriodMS, dDeadband	Handle	Adds a tag to the list with a specified poll period.
ctListData	hTag, pBuffer, dwLength, dwMode	Boolean	Gets the value of a tag on the list.
ctListDelete	hTag	Boolean	Frees a tag created with ctListAdd().
ctListEvent	hCTAPI, dwMode	Handle	Returns the elements in the list which have changed state since they were last read using the ctListRead() function.
ctListFree	hList	Boolean	Frees a list created with ctListNew().
ctListItem	hTag, dwItem, pBuffer, dwLength, dwMode	Boolean	Gets the tag element item data.
ctListNew	hCTAPI, dwMode	Handle	Creates a new list.
ctListRead	hList, pctOverlapped	Boolean	Reads every tag on the list.
ctListWrite	hTag, sValue, pctOverlapped	Boolean	Writes to a single tag on the list.

Function	Argument(s)	Type	Description
<a href="#">ctOpen</a>	sComputer, sUser, sPassword, nMode	Handle	Opens a connection to the Plant SCADA API.
<a href="#">ctOpenEx</a>	sComputer, sUser, sPassword, nMode, hCTAPI	Handle	Establishes the connection to the CtAPI server using the given client instance.
<a href="#">ctRawToEng</a>	pResult, dValue, pScale, dwMode	Boolean	Converts the raw I/O Device scale variable into engineering scale.
<a href="#">ctTagGetProperty</a>	hCTAPI, szTagName, szProperty, pData, dwBufferLength, dwType	Boolean	Gets the given property of the given tag.
<a href="#">ctTagRead</a>	hCTAPI, sTag, sValue, dwLength	Boolean	Reads the current value from the given I/O Device variable tag.
<a href="#">ctTagReadEx</a>	hCTAPI, sTag, sValue, dwLength, pctTagvalueItems	Boolean	Performs the same as <a href="#">ctTagRead</a> , but with an additional new argument
<a href="#">ctTagWrite</a>	hCTAPI, sTag, sValue	Boolean	Writes the given value to the I/O Device variable tag.
<a href="#">ctTagWriteEx</a>	hCTAPI, sTag, sValue, pctOverlapped	Boolean	Performs the same as <a href="#">ctTagWrite</a> , but with an additional new argument.

## ctCancelIO

Cancels a pending overlapped I/O operation. When the I/O command is canceled, the event will be signaled to show that the command has completed. The status will be set to the Plant SCADA error **ERROR CANCELED**. If the command completes before you can cancel it, [ctCancelIO\(\)](#) will return **FALSE**, and [GetLastError\(\)](#) will return **GENERIC\_CANNOT\_CANCEL**. The status of the overlapped operation will be the completion status of the command.

The CTAPI interface will automatically cancel any pending I/O commands when you call [ctClose\(\)](#).

## Syntax

**ctCancelIO(hCTAPI, pctOverlapped)**

*hCTAPI*

Type: Handle

Input/output: Input

Description: The handle to the CTAPI as returned from [ctOpen\(\)](#).

*pctOverlapped*

Type: CTOVERLAPPED\*

Input/output: Input

Description: Pointer to the overlapped I/O operation to cancel. If you specify NULL, any pending overlapped I/O operations on the interface will be canceled.

## Return Value

If the function succeeds, the return value is TRUE. If the function does not succeed, the return value is FALSE. To get extended error information, call [GetLastError](#).

## Related Functions

[ctOpen](#), [ctClose](#)

## Example

```
char sVersion[128];
CTOVERLAPPED ctOverlapped;
ctOverlapped.hEvent = CreateEvent(NULL, TRUE, TRUE, NULL);
ctCiCode(hCTAPI, "Version(0)", 0, 0, sVersion, sizeof(sVersion),
&ctOverlapped);
ctCancelIO(hCTAPI, &ctOverlapped);
```

## ctCiCode

Executes a Cicode function on the connected Plant SCADA computer. This allows you to control Plant SCADA or to get information returned from Cicode functions. You may call either built in or user defined Cicode functions. Cancels a pending overlapped I/O operation.

The function name and arguments to that function are passed as a single string. Standard Plant SCADA conversion is applied to convert the data from string type into the type expected by the function. When passing strings put the strings between the Plant SCADA string delimiters.

Functions which expect pointers or arrays are not supported. Functions which expect pointers are functions which update the arguments. This includes functions DspGetMouse(), DspAnGetPos(), StrWord(), and so on. Functions which expect arrays to be passed or returned are not supported, for example TableMath(), TrnSetTable(), TrnGetTable(). You may work around these limitations by calling a Cicode wrapper function which in turn calls the function you require.

If the Cicode function you are calling takes a long time to execute, is pre-empt or blocks, then the result of the function cannot be returned in the sResult argument. The Cicode function will, however, execute correctly.

## Syntax

**ctCiCode(*hCTAPI*, *sCmd*, *hWin*, *nMode*, *sResult*, *dwLength*, *pctOverlapped*)**

*hCTAPI*

Type: Handle

Input/output: Input

Description: The handle to the CTAPI as returned from [ctOpen\(\)](#).

*sCmd*

Type: String

Input/output: Input

Description: The command to execute.

*vhWin*

Type: Dword

Input/output: Input

Description: The Plant SCADA window to execute the function. This is a logical Plant SCADA window (0, 1, 2, 3 etc.) not a Windows Handle.

*nMode*

Type: Dword

Input/output: Input

Description: The mode of the Cicode call. Set this to 0 (zero).

*sResult*

Type: LPSTR

Input/output: Output

Description: The buffer to put the result of the function call, which is returned as a string. This may be NULL if you do not require the result of the function.

*dwLength*

Type: Dword

Input/output: Input

Description: The length of the sResult buffer. If the result of the Cicode function is longer than this number, then the result is not returned and the function call does not succeed, however the Cicode function is still executed. If the sResult is NULL then this length needs to be 0.

*pctOverlapped*

Type: CTOVERLAPPED\*

Input/output: Input

Description: CTOVERLAPPED structure. This structure is used to control the overlapped notification. Set to NULL if you want a synchronous function call.

## Return Value

Type: Dword. TRUE if successful, otherwise FALSE. Use [GetLastError\(\)](#) to get extended error information.

## Related Functions

[ctOpen](#)

## Example

```
char sName[32];
ctCicode(hCTAPI, "AlarmAck(0,)", 0, 0, NULL, 0, NULL);
```

```
ctCicode(hCTAPI, "PageInfo(0)", 0, 0, sName, sizeof(sName), NULL);
/* to call the Prompt function with the string "Hello", the C code would be:
*/
ctCicode(hCTAPI, "Prompt(\"Hello\")", 0, 0, NULL, 0, NULL);
/* If the string does not contain any delimiters (for example spaces or commas) you may
omit the string delimiters. For example to display a page called "Menu" the C code would
be:
*/
ctCicode(hCTAPI, "PageDisplay(Menu)", 0, 0, NULL, 0, NULL);
```

## ctClientCreate

ctClientCreate initializes the resources for a new CtAPI client instance. Once you have called ctClientCreate, you can pass the handle returned to [ctOpenEx](#) to establish communication with the CtAPI server.

Consider a situation where you try to communicate to the CtAPI server and the server takes a long time to respond (or doesn't respond at all). If you just call , you haven't been given a handle to the CtAPI instance, so you can't cancel the ctOpen by calling [ctCancelIO](#). But if you use ctClientCreate and then call ctOpenEx, you can use the handle returned by ctClientCreate to cancel the ctOpenEx.

## Syntax

**ctClientCreate()**

## Return Value

If the function succeeds, the return value specifies a handle. If the function does not succeed, the return value is NULL. Use [GetLastError\(\)](#) to get extended error information.

## Related Functions

[ctOpen](#), [ctOpenEx](#), [ctClose](#), [ctCloseEx](#), [ctClientDestroy](#)

## Example

```
DWORD dwStatus = 0;
HANDLE hCtapi = ctClientCreate();
if (hCtapi == NULL) {
    dwStatus = GetLastError(); // An error has occurred, trap it.
} else {
    if (TRUE == ctOpenEx(NULL, NULL, NULL, 0, hCtapi)) {
        ctTagWrite(hCtapi, "Fred", "1.5");
        if (FALSE == ctCloseEx(hCtapi, FALSE)) {
            dwStatus = GetLastError(); // An error has occurred, trap it.
        }
    } else {
        dwStatus = GetLastError(); // An error has occurred, trap it.
    }
    if (FALSE == ctClientDestroy(hCtapi)) {
        dwStatus = GetLastError(); // An error has occurred, trap it
    }
}
```

```
}
```

## ctClientDestroy

Cleans up the resources of the given CtAPI instance. Unlike [ctClose](#), ctClientDestroy does not close the connection to the CtAPI server.

You need to call [ctCloseEx](#) with *bDestroy* equal to FALSE before calling ctClientDestroy.

### Syntax

**ctClientDestroy(*hCTAPI*)**

*hCTAPI*

Type: Handle

Input/output: Input

Description: The handle to the CTAPI as returned from [ctOpen\(\)](#).

### Return Value

TRUE if successful, otherwise FALSE. Use [GetLastError\(\)](#) to get extended error information.

### Related Functions

[ctCloseEx](#), [ctClose](#), [ctClientCreate](#), [ctOpen](#), [ctOpenEx](#)

### Example

See [ctClientCreate](#) for an example.

## ctClose

Closes the connection between the application and the CtAPI. When called, any pending commands will be canceled. You need to free any handles allocated before calling ctClose. These handles are not freed when ctClose is called. Call this function from an application on shutdown or when a major error occurs on the connection.

### Syntax

**ctClose(*hCTAPI*)**

*hCTAPI*

Type: Handle

Input/output: Input

Description: The handle to the CTAPI as returned from [ctOpen](#).

## Return Value

TRUE if successful, otherwise FALSE. Use GetLastError() to get extended error information.

## Related Functions

[ctOpen](#)

## Example

See the example for [ctOpen](#).

## ctCloseEx

Closes the connection to the CtAPI server for the given CtAPI instance. It closes the connection the same way as does the [ctClose](#) method, but provides an option for whether or not to destroy the CtAPI instance within the [ctCloseEx](#) function call. [ctClose](#) always destroys the CtAPI instance within its function call.

For example, consider a situation where when we try to close the connection to the CtAPI server and it takes a long time to respond (or doesn't at all). If you call [ctClose](#), you can't cancel the [ctClose](#) by calling [ctCancelIO](#) because you cannot guarantee that the CtAPI instance is not in the process of being destroyed. But if you call [ctCloseEx](#) with the option of not destroying the CtAPI instance, you can call [ctCancelIO](#) to cancel the [ctCloseEx](#).

When you call [ctCloseEx](#) with *bDestroy* equal to FALSE, you need to then call [ctClientDestroy](#) afterwards to free the CtAPI client instance.

## Syntax

**ctCloseEx(*hCTAPI*,*bDestroy*);**

*hCTAPI*

Type: Handle

Input/output: Input

Description: The handle to the CTAPI as returned from [ctOpen\(\)](#).

*bDestroy*

Type: boolean

Input/output: Input

Description: If TRUE will destroy the CtAPI instance within the [ctCloseEx](#) function call. Default is FALSE.

## Return Value

TRUE if successful, otherwise FALSE. Use GetLastError() to get extended error information.

## Related Functions

[ctClientDestroy](#), [ctClose](#), [ctClientCreate](#), [ctOpen](#), [ctOpenEx](#)

## Example

See [ctClientCreate](#) for an example.

## ctEngToRaw

Converts the engineering scale variable into raw I/O Device scale. This is not necessary for the Tag functions as Plant SCADA will do the scaling. Scaling is not necessary for digitals, strings or if no scaling occurs between the values in the I/O Device and the Engineering values. You need to know the scaling for each variables as specified in the Plant SCADA Variable Tags table.

## Syntax

**ctEngToRaw(*pResult*, *dValue*, *pScale*, *dwMode*)**

*pResult*

Type: Double

Input/output: Output

Description: The resulting raw scaled variable.

*dValue*

Type: Double

Input/output: Input

Description: The engineering value to scale.

*pScale*

Type: CTSCALE\*

Input/output: Input

Description: The scaling properties of the variable.

*dwMode*

Type: Dword

Input/output: Input

Description: The mode of the scaling:

**CT\_SCALE\_RANGE\_CHECK:** Range check the result. If the variable is out of range then generate an error. The *pResult* still contains the raw scaled value.

**CT\_SCALE\_CLAMP\_LIMIT:** Clamp limit to maximum or minimum scales. If the result is out of scale then set result to minimum or maximum scale (which ever is closest). No error is generated if the scale is clamped. Cannot be used with CT\_SCALE\_RANGE\_CHECK or CT\_SCALE\_NOISE\_FACTOR options.

**CT\_SCALE\_NOISE\_FACTOR:** Allow noise factor for range check on limits. If the variable is our of range by less than 0.1 % then a range error is not generated.

## Return Value

TRUE if successful, otherwise FALSE. Use [GetLastError\(\)](#) to get extended error information.

## Related Functions

[ctOpen](#), [ctRawToEng](#), [ctTagRead](#)

## Example

```
CTSCALE Scale = { 0.0, 32000.0, 0.0, 100.0};  
double dSetPoint = 42.23;  
double dRawValue;  
ctEngToRaw(&dRawValue, dSetPoint, &Scale, CT_SCALE_RANGE_CHECK);
```

## ctFindClose

Closes a search initiated by [ctFindFirst](#).

## Syntax

```
ctFindClose(hnd)  
hnd  
Type: Handle  
Input/output: Input  
Description: Handle to the search, as returned by ctFindFirst\(\).
```

## Return Value

If the function succeeds, the return value is non-zero. If the function does not succeed, the return value is zero.  
To get extended error information, call [GetLastError\(\)](#).

## Related Functions

[ctOpen](#), [ctFindNext](#), [ctFindPrev](#), [ctFindScroll](#), [ctGetProperty](#)

## Example

See [ctFindFirst](#)

## ctFindFirst

Searches for the first object in the specified table, device, trend, tag, or alarm data which satisfies the filter string. A handle to the found object is returned via pObjHnd. The object handle is used to retrieve the object properties. To find the next object, call the [ctFindNext](#) function with the returned search handle.

If you experience server performance problems when using [ctFindFirst\(\)](#) refer to [CPULoadCount](#) and [CpuLoadSleepMS](#).

## Syntax

**ctFindFirst(*hCTAPI*, *szTableName*, *szFilter*, *pObjHnd*,*dwFlags*)**

*hCTAPI*

Type: Handle

Input/output: Input

Description: The handle to the CTAPI as returned from ctOpen().

*szTableName*

Type: LPCTSTR

Input/output: Input

Description: The table, device, trend, or alarm data to be searched. The following tables and fields can be searched:

**Trend** - Trend Tags

CLUSTER, TAG, RAW\_ZERO, RAW\_FULL, ENG\_ZERO, ENG\_FULL, ENG\_UNITS, COMMENT, SAMPLEPER, TYPE, ACQERROR, HISTORIAN, EQUIPMENT, EXPRESSION, ITEM, FORMAT, STORMETHOD

**DigAlm** - Digital Alarm Tags

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, ALMCOMMENT, EQUIPMENT, ACQERROR, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**AnaAlm** - Analog Alarm Tags

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, VALUE, HIGH, LOW, HIGHHIGH, LOWLOW, DEADBAND, RATE, DEVIATION, ALMCOMMENT, EQUIPMENT, ACQERROR, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**AdvAlm** - Advanced Alarm Tags

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, ALMCOMMENT, EQUIPMENT, ACQERROR, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**HResAlm** - Time-Stamped Alarm Tags

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, MILLISEC, DATE, AREA, ALMCOMMENT, EQUIPMENT, ACQERROR, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**ArgDigAlm** - Argyle Digital (Multi-digital) Alarm Tags

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, ALMCOMMENT, PRIORITY, STATE\_DESC, OLD\_DESC, EQUIPMENT, ACQERROR, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**ArgAnaAlm** - Argyle Analog Alarm Tags

CLUSTER, TAG, NAME, HELP, ALMCOMMENT, CATEGORY, STATE, TIME, DATE, AREA, VALUE, PRIORITY, HIGH, LOW, HIGHHIGH, LOWLOW, DEADBAND, RATE, DEVIATION, EQUIPMENT, ACQERROR, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**TsDigAlm** - Time-Stamped Digital Alarm Tags

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, ALMCOMMENT, EQUIPMENT, ACQERROR, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**TsAnaAlm** - Time-Stamped Analog Alarm Tags

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, VALUE, HIGH, LOW, HIGHHIGH, LOWLOW, DEADBAND, RATE, DEVIATION, ALMCOMMENT, EQUIPMENT, ACQERROR, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**ArgDigAlmStateDesc** - Argyle Digital (Multi-digital) Alarm Tag State Descriptions

CLUSTER, TAG, STATE\_DESC0, STATE\_DESC1, STATE\_DESC2, STATE\_DESC3, STATE\_DESC4, STATE\_DESC5, STATE\_DESC6, STATE\_DESC7, EQUIPMENT, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**DoublePointAlarm** - Double Point Status Alarm

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, ALMCOMMENT, PRIORITY, STATE\_DESC, OLD\_DESC, EQUIPMENT, ACQERROR, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**DoublePointAlarmStateName** - Double Point Status Alarm State Name

CLUSTER, TAG, STATE\_DESC0, STATE\_DESC1, STATE\_DESC2, STATE\_DESC3, STATE\_DESC4, STATE\_DESC5, STATE\_DESC6, STATE\_DESC7, EQUIPMENT, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**Alarm** - Alarm Tags

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, ALMCOMMENT, VALUE, HIGH, LOW, HIGHHIGH, LOWLOW, DEADBAND, RATE, DEVIATION, PRIORITY, STATE\_DESC, OLD\_DESC, ALARMTYPE, EQUIPMENT, ACQERROR, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**AlarmSummary** - Alarm Summary

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, TIME, DATE, AREA, VALUE, HIGH, LOW, HIGHHIGH, LOWLOW, DEADBAND, RATE, DEVIATION, PRIORITY, STATE\_DESC, OLD\_DESC, ALARMTYPE, ONDATE, ONDATEEXT, ONTIME, ONMILLI, OFFDATE, OFFDATEEXT, OFFTIME, OFFMILLI, DELTATIME, ACKDATE, ACKDATEEXT, ACKTIME, ALMCOMMENT, USERNAME, FULLNAME, USERDESC, SUMSTATE, SUMDESC, NATIVE\_SUMDESC, COMMENT, NATIVE\_COMMENT, EQUIPMENT, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**AlarmSOE** - Alarm Sequence Of Events

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, TIME, DATE, AREA, VALUE, HIGH, LOW, HIGHHIGH, LOWLOW, DEADBAND, RATE, DEVIATION, PRIORITY, STATE\_DESC, OLD\_DESC, ALARMTYPE, ONDATE, ONDATEEXT, ONTIME, ONMILLI, OFFDATE, OFFDATEEXT, OFFTIME, OFFMILLI, DELTATIME, ACKDATE, ACKDATEEXT, ACKTIME, ALMCOMMENT, USERNAME, FULLNAME, USERDESC, SUMSTATE, SUMDESC, NATIVE\_SUMDESC, COMMENT, NATIVE\_COMMENT, EQUIPMENT, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8, USERNAME, USERLOCATION

**Accum** - Accumulators

PRIV, AREA, CLUSTER, NAME, TRIGGER, VALUE, RUNNING, STARTS, TOTALISER

**Equip** – Equipment

CLUSTER, NAME, IODEVICE, AREA, LOCATION, SCHEDULED, DEFSTATE, COMMENT, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8, COMPOSITE, PARENT, PAGE, HELP, REFERENCE, DEVSCHEDE, SCHEDID

**EquipState** – Equipment State

CLUSTER, NAME, EQUIPMENT, PERIOD, DELAY, PRIORITY, DESCRIPTION, DRMODE

**EquipRef** – Equipment Reference

CLUSTER, EQUIP, REFCLUST, REFEQUIP, REFITEM, CATEGORY, ORDER, COMMENT, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**Tag - Variable Tags**

TAG, RAW\_ZERO, RAW\_FULL, ENG\_ZERO, ENG\_FULL, ENG\_UNITS, COMMENT, TYPE, CLUSTER, FULLNAME, IODEV, ADDR, DEADBAND, LOG\_UNIT, NET\_UNIT, EQUIPMENT, ITEM, HISTORIAN, OVR\_MODE, CTRL\_MODE, NUM\_SUBS, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8, ARR\_SIZE, BIT\_WIDTH, FORMAT

**Cluster - Clusters**

NAME, COMMENT, ACTIVE

**Server**

NAME, TYPE, COMMENT, CLUSTER, MODE, NETADDR, PORT, LEGACYPORT

**Schedule\_Config - Schedule Configuration**

ID, EQUIPMENT, STATE, START, OCCURRENCESTART, END, OCCURRENCEEND, DESC, MODIFIEDTIME, FREQ, INTERVAL, WEEKDAY, WEEKDAYMASK, MAXREC, RECOUNTIL, DAYORD, DAY, MONTH, SPECIALINC, GROUPIDS, RESOURCETYPE, SOURCEVALUE, SOURCEID

**Schedule\_Runtime - Schedule Runtime**

EQUIPMENT, STATE, START, END, INHERITED, DESC, RESOURCETYPE, SOURCEVALUE, SOURCEID

**Schedule\_Special\_Group - Schedule Special Group**

NAME, ID, RESOURCE, CHILDRESOURCES

**Schedule\_SpecialItem - Schedule Special Item**

ID, NAME, DAY, GROUPNAME, GROUPID

---

**Note:** The migration tool in Plant SCADA 2023 converts memory PLC variables to local variable tags which are in a separate table to the variable tags. Calling ctFindFirst with *szTableName* "Tag" will not return the local variable tags. In order to return the local variable tags you need to call ctFindFirst with the *szTableName* of "LocalTag". Local variables do not have clusters and have only one pair of zero/full scales (as opposed to raw and engineering scales for variable tags).

---

**LocalTag - The field names for local variable tags are:**

NAME, TYPE, ASIZE (array size), ZERO, FULL, UNITS, COMMENT, FORMAT.

The array size field is only available for local tags.

**szFilter**

Type: LPCTSTR

Input/output: Input

Description: Filter criteria. This is a string based on the following format:

"PropertyName1=FilterCriteria1;PropertyName2=FilterCriteria2"

The wildcard \* may be used as part of the filter criteria to match multiple entries. Use an empty string, or "\*" as the filter string to match every entry.

**pObjHnd**

Type: HANDLE

Input/output: Output

Description: The pointer to the found object handle. This is used to retrieve the properties.

**dwFlags**

This argument is no longer used, pass in a value of 0 for this argument.

**To search a table:**

- In *szTableName* specify the name of the table.

**To search a device:**

- In *szTableName* specify the name as defined in the Plant SCADA devices settings, for example "RECIPES" for the Example project.

**To search trend data:**

- In *szTableName* specify the trend using the following format (including the quotation marks):  
'TRNQUERY,Endtime,EndtimeMs,Period,NumSamples,Tagname,Displaymode,Datemode'  
See [TrnQuery](#) for syntax details.

**To search alarm data:**

- In *szTableName* specify the alarm data using the following format (including the quotation marks):  
'ALMQUERY,Database,TagName,Starttime,StarttimeMs,Endtime,EndtimeMs,Period'  
See [AlmQuery](#) for syntax details.

**Return Value**

If the function succeeds, the return value is a search handle used in a subsequent call to `ctFindNext()` or `ctFindClose()`. If the function does not succeed, the return value is NULL. To get extended error information, call `GetLastError()`

**Related Functions**

[ctOpen](#), [ctFindNext](#), [ctFindClose](#), [ctGetProperty](#), [ctFindFirst](#)

**Example**

```
HANDLE hSearch;
HANDLE hObject;
HANDLE hFind;
// Search the Tag table
hSearch = ctFindFirst(hCTAPI, "Tag", NULL, &hObject, 0);
if (hSearch == NULL) {
    // no tags found
} else {
    do {
        char sName[32];
        // Get the tag name
        ctGetProperty(hObject, "Tag", sName, sizeof(sName), NULL,
        DBTYPE_STR);
    } while (ctFindNext(hSearch, &hObject));
    ctFindClose(hSearch);
}
```

```
// Get Historical Trend data via CTAPI
// Get 100 samples of the CPU trend at 2 second
hFind = ctFindFirst(hCTAPI, "CTAPITrend(\"10:15:00 \", \"11/8/1998\", 2, 100, 0,
\"CPU\")", &hObject, 0);
while (hFind) {
    char sTime[32], sDate[32], sValue[32];
    ctGetProperty(hObject, "TIME", sTime, sizeof(sTime), NULL, DBTYPE_STR);
    ctGetProperty(hObject, "DATE", sDate, sizeof(sDate), NULL, DBTYPE_STR);
    ctGetProperty(hObject, "CPU", sValue, sizeof(sValue), NULL, DBTYPE_STR);
    // do something with the trend data.
    if (!ctFindNext(hFind, &hObject)) {
        ctFindClose(hFind);
        hFind = NULL;
        break;
    }
}
```

## ctFindFirstEx

Performs the same as [ctFindFirst](#), but with an additional new argument. Searches for the first object in the specified table, device, trend, or alarm data which satisfies the filter string. A handle to the found object is returned via *pObjHnd*. The object handle is used to retrieve the object properties. To find the next object, call the [ctFindNext](#) function with the returned search handle.

If you experience server performance problems when using [ctFindFirst\(\)](#) refer to [CPULoadCount](#) and [CpuLoadSleepMS](#).

If [ctFindFirst](#) is called instead of [ctFindFirstEx](#), the *szCluster* defaults to NULL.

**ctFindFirstEx(*hCTAPI*, *szTableName*, *szFilter*, *szCluster*, *pObjHnd*, *dwFlags*)**

*hCTAPI*

Type: Handle

Input/output: Input

Description: The handle to the CTAPI as returned from [ctOpen\(\)](#).

*szTableName*

Type: LPCTSTR

Input/output: Input

Description: The table, device, trend, or alarm data to be searched. The following tables and fields can be searched:

**Trend** - Trend Tags

CLUSTER, TAG, RAW\_ZERO, RAW\_FULL, ENG\_ZERO, ENG\_FULL, ENG\_UNITS, COMMENT, SAMPLEPER, TYPE, ACQERROR, HISTORIAN, EQUIPMENT, EXPRESSION, ITEM, FORMAT, STORMETHOD

**DigAlm** - Digital Alarm Tags

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, ALMCOMMENT, EQUIPMENT, ACQERROR, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**AnaAlm** - Analog Alarm Tags

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, VALUE, HIGH, LOW, HIGHHIGH, LOWLOW, DEADBAND, RATE, DEVIATION, ALMCOMMENT, EQUIPMENT, ACQERROR, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**AdvAlm** - Advanced Alarm Tags

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, ALMCOMMENT, EQUIPMENT, ACQERROR, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**HResAlm** - Time-Stamped Alarm Tags

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, MILLISEC, DATE, AREA, ALMCOMMENT, EQUIPMENT, ACQERROR, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**ArgDigAlm** - Argyle Digital (Multi-digital) Alarm Tags

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, ALMCOMMENT, PRIORITY, STATE\_DESC, OLD\_DESC, EQUIPMENT, ACQERROR, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**ArgAnaAlm** - Argyle Analog Alarm Tags

CLUSTER, TAG, NAME, HELP, ALMCOMMENT, CATEGORY, STATE, TIME, DATE, AREA, VALUE, PRIORITY, HIGH, LOW, HIGHHIGH, LOWLOW, DEADBAND, RATE, DEVIATION, EQUIPMENT, ACQERROR, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**TsDigAlm** - Time-Stamped Digital Alarm Tags

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, ALMCOMMENT, EQUIPMENT, ACQERROR, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**TsAnaAlm** - Time-Stamped Analog Alarm Tags

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, VALUE, HIGH, LOW, HIGHHIGH, LOWLOW, DEADBAND, RATE, DEVIATION, ALMCOMMENT, EQUIPMENT, ACQERROR, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**ArgDigAlmStateDesc** - Argyle Digital (Multi-digital) Alarm Tag State Descriptions

CLUSTER, TAG, STATE\_DESC0, STATE\_DESC1, STATE\_DESC2, STATE\_DESC3, STATE\_DESC4, STATE\_DESC5, STATE\_DESC6, STATE\_DESC7, EQUIPMENT, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**DoublePointAlarm** - Double Point Status Alarm

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, ALMCOMMENT, PRIORITY, STATE\_DESC, OLD\_DESC, EQUIPMENT, ACQERROR, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**DoublePointAlarmStateName** - Double Point Status Alarm State Name

CLUSTER, TAG, STATE\_DESC0, STATE\_DESC1, STATE\_DESC2, STATE\_DESC3, STATE\_DESC4, STATE\_DESC5, STATE\_DESC6, STATE\_DESC7, EQUIPMENT, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**Alarm** - Alarm Tags

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, ALMCOMMENT, VALUE, HIGH, LOW, HIGHHIGH, LOWLOW, DEADBAND, RATE, DEVIATION, PRIORITY, STATE\_DESC, OLD\_DESC, ALARMTYPE, EQUIPMENT, ACQERROR, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**AlarmSummary** - Alarm Summary

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, TIME, DATE, AREA, VALUE, HIGH, LOW, HIGHHIGH, LOWLOW,

DEADBAND, RATE, DEVIATION, PRIORITY, STATE\_DESC, OLD\_DESC, ALARMTYPE, ONDATE, ONDATEEXT, ONTIME,  
ONMILLI, OFFDATE, OFFDATEEXT, OFFTIME, OFFMILLI, DELTATIME, ACKDATE, ACKDATEEXT, ACKTIME,  
ALMCOMMENT, USERNAME, FULLNAME, USERDESC, SUMSTATE, SUMDESC, NATIVE\_SUMDESC, COMMENT,  
NATIVE\_COMMENT, EQUIPMENT, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5,  
CUSTOM6, CUSTOM7, CUSTOM8

**AlarmSOE** - Alarm Sequence Of Events

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, TIME, DATE, AREA, VALUE, HIGH, LOW, HIGHHIGH, LOWLOW,  
DEADBAND, RATE, DEVIATION, PRIORITY, STATE\_DESC, OLD\_DESC, ALARMTYPE, ONDATE, ONDATEEXT, ONTIME,  
ONMILLI, OFFDATE, OFFDATEEXT, OFFTIME, OFFMILLI, DELTATIME, ACKDATE, ACKDATEEXT, ACKTIME,  
ALMCOMMENT, USERNAME, FULLNAME, USERDESC, SUMSTATE, SUMDESC, NATIVE\_SUMDESC, COMMENT,  
NATIVE\_COMMENT, EQUIPMENT, HISTORIAN, ITEM, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5,  
CUSTOM6, CUSTOM7, CUSTOM8, USERNAME, USERLOCATION

**Accum** - Accumulators

PRIV, AREA, CLUSTER, NAME, TRIGGER, VALUE, RUNNING, STARTS, TOTALISER

**Equip** – Equipment

CLUSTER, NAME, IODEVICE, AREA, LOCATION, SCHEDULED, DEFSTATE, COMMENT, CUSTOM1, CUSTOM2,  
CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8, COMPOSITE, PARENT, PAGE, HELP,  
REFERENCE, DEVSCHEID, SCHEDID

**EquipState** – Equipment State

CLUSTER, NAME, EQUIPMENT, PERIOD, DELAY, PRIORITY, DESCRIPTION, DRMODE

**EquipRef** – Equipment Reference

CLUSTER, EQUIP, REFCLUST, REFEQUIP, REFITEM, CATEGORY, ORDER, COMMENT, CUSTOM1, CUSTOM2,  
CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8

**Tag** - Variable Tags

TAG, RAW\_ZERO, RAW\_FULL, ENG\_ZERO, ENG\_FULL, ENG\_UNITS, COMMENT, TYPE, CLUSTER, FULLNAME,  
IODEV, ADDR, DEADBAND, LOG\_UNIT, NET\_UNIT, EQUIPMENT, ITEM, HISTORIAN, OVR\_MODE, CTRL\_MODE,  
NUM\_SUBS, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8,  
ARR\_SIZE, BIT\_WIDTH, FORMAT

**Cluster** - Clusters

NAME, COMMENT, ACTIVE

**Server**

NAME, TYPE, COMMENT, CLUSTER, MODE, NETADDR, PORT, LEGACYPOR

**Schedule\_Config** - Schedule Configuration

ID, EQUIPMENT, STATE, START, OCCURRENCESTART, END, OCCURRENCEEND, DESC, MODIFIEDTIME, FREQ,  
INTERVAL, WEEKDAY, WEEKDAYMASK, MAXREC, RECUNTIL, DAYORD, DAY, MONTH, SPECIALINC, GROUPIDS,  
RESOURCETYPE, SOURCEVALUE, SOURCEID

**Schedule\_Runtime** - Schedule Runtime

EQUIPMENT, STATE, START, END, INHERITED, DESC, RESOURCETYPE, SOURCEVALUE, SOURCEID

**Schedule\_Special\_Group** - Schedule Special Group

NAME, ID, RESOURCE, CHILDRESOURCES

**Schedule\_SpecialItem** - Schedule Special Item

ID, NAME, DAY, GROUPNAME, GROUPID

---

**Note:** The migration tool in Plant SCADA 2023 converts memory PLC variables to local variable tags which are in a separate table to the variable tags. Calling `ctFindFirst` with `szTableName` "Tag" will not return the local variable tags. In order to return the local variable tags you need to call `ctFindFirst` with the `szTableName` of "LocalTag". Local variables do not have clusters and have only one pair of zero/full scales (as opposed to raw and engineering scales for variable tags).

**LocalTag** - The field names for local variable tags are:

NAME, TYPE, ASIZE (array size), ZERO, FULL, UNITS, COMMENT, FORMAT.

The array size field is only available for local tags.

*szFilter*

Type: LPCTSTR

Input/output: Input

Description: Filter criteria. This is a string based on the following format:

"PropertyName1=FilterCriteria1;PropertyName2=FilterCriteria2"\.

Use "\*" as the filter to achieve the same result.

*szCluster*

Type: LPCTSTR

Input/output: Input

Description: Specifies on which cluster the `ctFindFirst` function will be performed. This can be left empty (null) for a single cluster system. In this case, `ctFindFirst` will be performed on the active cluster. In a system with multiple clusters, you need to specify a cluster.

*pObjHnd*

Type: HANDLE

Input/output: Output

Description: The pointer to the found object handle. This is used to retrieve the properties.

*dwFlags*

This argument is no longer used, pass in a value of 0 for this argument.

### To search a table:

- In `szTableName` specify the name of the table.

### To search a device:

- In `szTableName` specify the name as defined in the Plant SCADA Devices form, for example "RECIPES" for the Example project.

### To search trend data:

- In `szTableName` specify the trend using the following format (including the quotation marks):  
'TRNQUERY,Endtime,EndtimeMs,Period,NumSamples,Tagname,Displaymode,Datemode'  
See [TrnQuery](#) for syntax details.

### To search alarm data:

- In `szTableName` specify the alarm data using the following format (including the quotation marks):

'ALMQUERY,Database,TagName,Starttime,StarttimeMs,Endtime,EndtimeMs,Period'

See [AlmQuery](#) for syntax details.

## Return Value

If the function succeeds, the return value is a search handle used in a subsequent call to [ctFindNext\(\)](#) or [ctFindClose\(\)](#). If the function does not succeed, the return value is NULL. To get extended error information, call [GetLastError\(\)](#).

## Related Functions

[ctOpen](#), [ctFindNext](#), [ctFindClose](#), [ctGetProperty](#), [ctFindFirst](#)

## ctFindNext

Retrieves the next object in the search initiated by [ctFindFirst](#).

## Syntax

**ctFindNext(*hnd*, *pObjHnd*)**

*hnd*

Type: Handle

Input/output: Input

Description: Handle to the search, as returned by [ctFindFirst\(\)](#).

*pObjHnd*

Type: HANDLE

Input/output: Output

Description: The pointer to the found object handle. This is used to retrieve the properties.

## Return Value

If the function succeeds, the return value is TRUE (1). If the function does not succeed, the return value is FALSE (0). To get extended error information, call [GetLastError\(\)](#). If you reach the end of the search, the [GetLastError\(\)](#) function returns an Object Not Found error. Once past the end of the search, you cannot scroll the search using [ctFindNext\(\)](#) or [ctFindPrev\(\)](#) commands. You need to reset the search pointer by creating a new search using [ctFindFirst\(\)](#), or by using the [ctFindScroll\(\)](#) function to move the pointer to a valid position.

## Related Functions

[ctOpen](#), [ctFindFirst](#), [ctFindPrev](#), [ctFindClose](#), [ctGetProperty](#)

## Example

See [ctFindFirst](#).

## ctFindNumRecords

Gets the number of records for a given browsing session.

### Syntax

**ctFindNumRecords(*hnd*)**

*hnd*

Type: Handle

Input/output: Input

Description: Handle to the search, as returned by [ctFindFirst\(\)](#).

### Return Value

If the function succeeds, the return value is the number of records (INT) or -1 if unsuccessful.

### Related Functions

[ctOpen](#), [ctFindFirst](#), [ctFindPrev](#), [ctFindClose](#), [ctGetProperty](#)

## ctFindPrev

Retrieves the previous object in the search initiated by [ctFindFirst](#).

### Syntax

**ctFindPrev(*hnd*, *pObjHnd*)**

*hnd*

Type: Handle

Input/output: Input

Description: Handle to the search, as returned by [ctFindFirst\(\)](#).

*pObjHnd*

Type: HANDLE

Input/output: Output

Description: The pointer to the found object handle. This is used to retrieve the properties.

### Return Value

If the function succeeds, the return value is TRUE (1). If the function does not succeed, the return value is FALSE (0). To get extended error information, call [GetLastError](#)(). If you reach the end of the search, the [GetLastError](#)() function returns an Object Not Found error. Once past the end of the search, you cannot scroll the search using [ctFindNext\(\)](#) or [ctFindPrev\(\)](#) commands. You need to reset the search pointer by creating a new search using [ctFindFirst\(\)](#), or by using the [ctFindScroll\(\)](#) function to move the pointer to a valid position.

## Related Functions

[ctOpen](#), [ctFindFirst](#), [ctFindNext](#), [ctFindClose](#), [ctGetProperty](#)

## Example

See [ctFindFirst](#)

## ctFindScroll

Scrolls to the necessary object in the search initiated by [ctFindFirst](#).

To find the current scroll pointer, you can scroll relative (dwMode = CT\_FIND\_SCROLL\_RELATIVE) with an offset of 0. To find the number of records returned in a search, scroll to the end of the search.

## Syntax

**ctFindScroll(hnd, dwMode, dwOffset, pObjHnd)**

*hnd*

Type: Handle

Input/output: Input

Description: Handle to the search, as returned by [ctFindFirst\(\)](#).

*dwMode*

Type: DWORD

Input/output:

Description: Mode of the scroll. The following modes are supported:

**CT\_FIND\_SCROLL\_NEXT:** Scroll to the next record. The dwOffset parameter is ignored.

**CT\_FIND\_SCROLL\_PREV:** Scroll to the previous record. The dwOffset parameter is ignored.

**CT\_FIND\_SCROLL\_FIRST:** Scroll to the first record. The dwOffset parameter is ignored.

**CT\_FIND\_SCROLL\_LAST:** Scroll to the last record. The dwOffset parameter is ignored.

**CT\_FIND\_SCROLL\_ABSOLUTE:** Scroll to absolute record number. The record number is specified in the dwOffset parameter. The record number is from 1 to the maximum number of records returned in the search.

**CT\_FIND\_SCROLL\_RELATIVE:** Scroll relative records. The number of records to scroll is specified by the dwOffset parameter. If the offset is positive, this function will scroll to the next record, if negative, it will scroll to the previous record. If 0 (zero), no scrolling occurs.

*dwOffset*

Type: LONG

Input/output: Input

Description: Offset of the scroll. The meaning of this parameter depends on the dwMode of the scrolling operation.

*pObjHnd*

Type: HANDLE

Input/output: Output

Description: The pointer to the found object handle. This is used to retrieve the properties.

*pObjHnd*

Type: HANDLE

Input/output: Output

Description: The pointer to the found object handle. This is used to retrieve the properties.

## Return Value

If the function succeeds, the return value is non-zero. If the function does not succeed, the return value is zero. To get extended error information, call **GetLastError()**. If you reach the end of the search, the **GetLastError()** function returns an Object Not Found error. The return value is the current record number in the search. Record numbers start at 1 (for the first record) and increment until the end of the search has been reached. Remember, 0 (zero) is not a valid record number - it signifies that the function was not successful.

## Related Functions

[ctOpen](#), [ctFindFirst](#), [ctFindNext](#), [ctFindPrev](#), [ctFindClose](#), [ctGetProperty](#)

## Example

```
HANDLE hSearch;
HANDLE hObject;
DWORD dwNoRecords;
// Search the Tag table
hSearch = ctFindFirst(hCTAPI, "Tag", NULL, &hObject, 0);
// Count number of records
dwNoRecords = ctFindScroll(hSearch, CT_FIND_SCROLL_LAST, 0, &hObject);
// scroll back to beginning
ctFindScroll(hSearch, CT_FIND_SCROLL_FIRST, 0, &hObject);
do {
    char sName[32];
    // Get the tag name
    ctGetProperty(hObject, "Tag", sName, sizeof(sName), NULL, DBTYPE_STR);
} while (ctFindScroll(hSearch, CT_FIND_SCROLL_NEXT, 0, &hObject));
ctFindClose(hSearch);
```

## ctGetOverlappedResult

Returns the results of an overlapped operation. The results reported by the **ctGetOverlappedResult()** function are those of the specified handle's last CTOVERLAPPED operation to which the specified **CTOVERLAPPED** structure was provided, and for which the operation's results were pending. A pending operation is indicated when the function that started the operation returns FALSE, and the **GetLastError** function returns ERROR\_IO\_PENDING. When an I/O operation is pending, the function that started the operation resets the **hEvent** member of the **CTOVERLAPPED** structure to the non-signaled state. Then when the pending operation has been completed, the system sets the event object to the signaled state.

If the *bWait* parameter is TRUE, **ctGetOverlappedResult()** determines whether the pending operation has been completed by waiting for the event object to be in the signaled state.

Specify a manual-reset event object in the CTOVERLAPPED structure. If an auto-reset event object is used, the event handle needs to not be specified in any other wait operation in the interval between starting the

CTOVERLAPPED operation and the call to **ctGetOverlappedResult()**. For example, the event object is sometimes specified in one of the wait functions to wait for the operation's completion. When the wait function returns, the system sets an auto-reset event's state to non-signaled, and a subsequent call to **ctGetOverlappedResult()** with the bWait parameter set to TRUE causes the function to be blocked indefinitely.

## Syntax

**ctGetOverlappedResult(*hCTAPI*, *lpctOverlapped*, *pBytes*, *bWait*)**

*hCTAPI*

Type: Handle

Input/output: Input

Description: The handle to the CTAPI as returned from [ctOpen\(\)](#).

*lpctOverlapped*

Type: CTOVERLAPPED\*

Input/output: Input

Description: Address of the CTOVERLAPPED structure which was used when an overlapped operation was started.

*pBytes*

Type: DWORD\*

Input/output: Input

Description: Address of actual bytes transferred. For the CTAPI this value is undefined.

*bWait*

Type: BOOL

Input/output: Input

Description: Specifies whether the function waits for the pending overlapped operation to be completed. If TRUE, the function does not return until the operation has been completed. If FALSE and the operation is still pending, the function returns FALSE and the GetLastError function returns ERROR\_IO\_INCOMPLETE.

## Return Value

If the function succeeds, the return value is TRUE. If the function does not succeed, the return value is FALSE. Use [GetLastError\(\)](#) to get extended error information.

## Related Functions

[ctOpen](#), [ctHasOverlappedIoCompleted](#)

## Example

```
DWORD Bytes;
char sVersion[128];
CTOVERLAPPED ctOverlapped;
ctOverlapped.hEvent = CreateEvent(NULL, TRUE, TRUE, NULL);
ctCicode(hCTAPI, "Version(0)", 0, 0, sVersion, sizeof(sVersion), &ctOverlapped);
...
// do something else.
```

```
///  
// wait for the ctCicode to complete  
ctGetOverlappedResult(hCTAPI, &ctOverlapped, &Bytes, TRUE);
```

## ctGetProperty

Retrieves an object property or meta data for an object. Use this function in conjunction with the `ctFindFirst()` and `ctFindNext()` functions. i.e. First, you find an object, then you retrieve its properties.

To retrieve property meta data such as type, size and so on, use the following syntax for the `szName` argument:

- `object.fields.count` - the number of fields in the record
- `object.fields(n).name` - the name of the nth field of the record
- `object.fields(n).type` - the type of the nth field of the record
- `object.fields(n).actualsize` - the actual size of the nth field of the record

## Syntax

**ctGetProperty(*hnd*, *szName*, *pData*, *dwBufferLength*, *dwResultLength*, *dwType*)**

*hnd*

Type: Handle

Input/output: Input

Description: Handle to the search, as returned by [ctFindFirst\(\)](#).

*szName*

Type: LPCTSTR\*

Input/output: Input

Description: The name of the property to be retrieved. The following properties are supported:

**Name** - The name of the tag.

**FullName** - The full name of the tag in the form cluster.tagname.

**Network** - The unique I/O Device Number.

**BitWidth** - Width of the data type in bits. for example digital will be 1, integer 16, long 32, etc.

**UnitType** - The protocol specific unit type.

**UnitAddress** - The protocol specific unit address.

**UnitCount** - The protocol specific unit count.

**RawType** - The raw data type of the point. The following types are returned: 0 (Digital), 1 (Integer), 2 (Real), 3 (BCD), 4 (Long), 5 (Long BCD), 6 (Long Real), 7 (String), 8 (Byte), 9 (Void), 10 (Unsigned integer).

**Raw\_Zero** - Raw zero scale.

**Raw\_Full** - Raw full scale.

**Eng\_Zero** - Engineering zero scale.

**Eng\_Full** - Engineering full scale.

**Equipment** – The associated equipment name

**Name** - The name of the equipment.

**Cluster** - The name of the cluster that this equipment runs on.

**Type** - The specific type of equipment in the system.

**IoDevice** - The I/O Device used to communicate with this piece of equipment.

**Area** - The area number or label to which this equipment belongs.

**Location** - A string describing the location of the equipment.

**Comment** - Comment.

**Page** - The name of the page on which this equipment appears.

**Help** - The help context string.

**Parent** - The name of the parent equipment derived from the name of the equipment.

**Composite** – The equipment specific composite name

**Custom1 .. Custom8** - User-defined strings.

*pData*

Type: VOID\*

Input/output: Output

Description: The result buffer to store the read data. The data is raw binary data, no data conversion or scaling is performed. If this buffer is not large enough to receive the data, the data will be truncated, and the function will return false.

*dwBufferLength*

Type: DWORD

Input/output: Input

Description: Length of result buffer. If the result buffer is not large enough to receive the data, the data will be truncated, and the function will return false.

*dwResultLength*

Type: DWORD\*

Input/output: Output

Description: Length of returned result. You can pass NULL if you want to ignore this parameter

*dwType*

Type: DWORD

Input/output: Input

Description: The desired return type as follows:

Value	Meaning
DBTYPE_UI1	UCHAR
DBTYPE_I1	1 byte INT
DBTYPE_I2	2 byte INT
DBTYPE_I4	4 byte INT
DBTYPE_R4	4 byte REAL
DBTYPE_R8	8 byte REAL
DBTYPE_BOOL	BOOLEAN

DBTYPE_BYTES	Byte stream
DBTYPE_STR	NULL Terminated STRING

## Return Value

If the function succeeds, the return value is non-zero. If the function does not succeed, the return value is zero. To get extended error information, call GetLastError().

## Related Functions

[ctOpen](#), [ctFindFirst](#), [ctFindNext](#), [ctFindPrev](#), [ctFindClose](#)

## Example

```
// get the property of the TAG field
ctGetProperty(hObject, "TAG", sName, sizeof(sName), NULL, DBTYPE_STR);
// Use the meta property fields to enumerate the entire row of data
// first get number of fields in the row
ctGetProperty(hObject, "object.fields.count", &dwFields, sizeof(dwFields),
NULL, DBTYPE_I4);
for (i = 0; i < dwFields; i++) {
    sprintf(sObject, "object.fields(%d).name", i + 1);
    // get name of field
    if (ctGetProperty(hObject, sObject, sName, sizeof(sName), NULL, DBTYPE_STR)) {
        // get value of field
        if (ctGetProperty(hObject, sName, sData, sizeof(sData),
NULL, DBTYPE_STR)) {
            printf("%8.8s ", sData);
        }
    }
}
```

Also see [ctFindFirst\(\)](#).

## ctHasOverlappedIoCompleted

Provides a high performance test operation that can be used to poll for the completion of an outstanding I/O operation.

## Syntax

**ctHasOverlappedIoCompleted(*lpctOverlapped*)**

*lpctOverlapped*

Type: CTOVERLAPPED\*

Input/output: Input

Description: Address of the CTOVERLAPPED structure which was used when an overlapped operation was started.

## Return Value

TRUE if the I/O operation has completed, and FALSE otherwise.

## Return Value

[ctOpen](#), [ctGetOverlappedResult](#)

## ctListAdd

Adds a tag or tag element to the list. Once the tag has been added to the list, it may be read using [ctListRead\(\)](#) and written to using [ctListWrite\(\)](#). If a read is already pending, the tag will not be read until the next time [ctListRead\(\)](#) is called. [ctListWrite\(\)](#) may be called immediately after the [ctListAdd\(\)](#) function has completed.

## Syntax

**ctListAdd(*hList*, *sTag*)**

*hList*

Type: HANDLE

Input/output: Input

Description: The handle to the list, as returned from [ctListNew\(\)](#).

*sTag*

Type: LPCSTR

Input/output: Input

Description: The tag or tag name and element name, separated by a dot to be added to the list. If the element name is not specified, it will be resolved at runtime as for an unqualified tag reference.

Variable tags can be specified as a string in multiple forms. Refer to [Tag Names](#) for more information.

## Return Value

If the function succeeds, the return value specifies a handle. If the function does not succeed, the return value is NULL. To get extended error information, call [GetLastError\(\)](#)

If a tag not currently defined in your system is specified using this function then the return value will specify a valid handle. Calling [ctListData](#) will allow identification of the true state of the tag. Passing an empty tag to this function will result in the function exiting immediately and returning NULL.

## Related Functions

[ctOpen](#), [ctListNew](#), [ctListFree](#), [ctListRead](#), [ctListWrite](#), , [ctListAddEx](#)

## Example

```
HANDLE hCTAPI;  
HANDLE hList;  
HANDLE hTagOne;
```

```
HANDLE hTagOneField;
HANDLE hTagOneControlMode;
HANDLE hTagOneStatus;
char sProcessValue[20];
char sProcessValueField[20];
char sProcessValueControlMode[20];
char sProcessValueStatus[20];
hCTAPI = ctOpen(NULL, NULL, NULL, 0);
hList = ctListNew(hCTAPI, 0);
hTagOne = ctListAdd(hList, "TagOne");
hTagOneField = ctListAdd(hList, "TagOne.Field");
hTagOneControlMode = ctListAdd(hList, "TagOne.ControlMode");
hTagOneStatus = ctListAdd(hList, "TagOne.Status");
ctListRead(hList, NULL);
ctListData(hTagOne, sProcessValue, sizeof(sProcessValue), 0);
ctListData(hTagOneField, sProcessValueField, sizeof(sProcessValueField) , 0);
ctListData(hTagOneControlMode, sProcessValueControlMode, sizeof(sProcessValueControlMode) , 0);
ctListData(hTagOneStatus, sProcessValueStatus, sizeof(sProcessValueStatus) , 0);
ctListFree(hList);
```

## ctListAddEx

Performs the same as `ctListAdd`, but with 2 additional new arguments. Adds a tag, or tag element, to the list. Once the tag has been added to the list, it may be read using `ctListRead()` and written to using `ctListWrite()`. If a read is already pending, the tag will not be read until the next time `ctListRead()` is called. `ctListWrite()` may be called immediately after the `ctListAdd()` function has completed.

If `ctListAdd` is called instead of `ctListAddEx`, The poll period of the subscription for the tag defaults to 500 milliseconds, and the `bRaw` flag defaults to the engineering value of FALSE.

## Syntax

**ctListAddEx(*hList*, *sTag*, *bRaw*, *nPollPeriodMS*, *dDeadband*)**

*hList*

Type: HANDLE

Input/output: Input

Description: The handle to the list, as returned from `ctListNew()`.

*sTag*

Type: LPCSTR

Input/output: Input

Description: The tag or tag name and element name, separated by a dot to be added to the list. If the element name is not specified, it will be resolved at runtime as for an unqualified tag reference.

Variable tags can be specified as a string in multiple forms. Refer to [Tag Names](#) for more information.

*bRaw*

Type: BOOL

Input/output: Input

Description: Specifies whether to subscribe to the given tag in the list using raw mode if TRUE or engineering mode if FALSE.

*nPollPeriodMS*

Type: INTEGER

Input/output: Input

Description: Dictates the poll period used in the subscription made for the tag (in milliseconds).

*dDeadband*

Type: DOUBLE

Input/output: Input

Description: Percentage of the variable tag's engineering range that a tag needs to change by in order for an update to be sent through the system. A value of -1.0 indicates that the default deadband specified by the tag definition is to be used.

## Return Value

If the function succeeds, the return value specifies a handle. If the function does not succeed, the return value is NULL. To get extended error information, call GetLastError()

If a tag not currently defined in your system is specified using this function then the return value will specify a valid handle. Calling ctListData will allow identification of the true state of the tag. Passing an empty tag to this function will result in the function exiting immediately and returning NULL.

## Related Functions

[ctOpen](#), [ctListNew](#), [ctListFree](#), [ctListRead](#), [ctListWrite](#), [ctListData](#), [ctListItem](#)

## Example

See [ctListNew](#)

## ctListData

Gets the value of a tag on the list. Call this function after [ctListRead\(\)](#) has completed for the added tag. You may call [ctListData\(\)](#) while subsequent [ctListRead\(\)](#) functions are pending, and the last data read will be returned. If you wish to get the value of a specific quality part of a tag element item data use [ctListItem](#) which includes the same parameters with the addition of the *dwItem* parameter.

## Syntax

**ctListData(*hTag*, *pBuffer*, *dwLength*, *dwMode*)**

*hTag*

Type: HANDLE

Input/output: Input

Description: The handle to the tag, as returned from [ctListAdd\(\)](#).

*pBuffer*

Type: VOID\*

Input/output: Input

Description: Pointer to a buffer to return the data. The data is returned scaled and as a formatted string.

*dwLength*

Type: Dword

Input/output: Input

Description: Length (in bytes) of the raw data buffer.

*dwMode*

Type: DWORD

Input/output: Input

Description: Mode of the data. The following modes are supported:

**0 (zero)** - The value is scaled using the scale specified in the Plant SCADA project, and formatted using the format specified in the Plant SCADA project.

To choose the format of the value returned, you can use the Citect INI parameter [\[CtAPI\]RoundToFormat](#).

The *dwMode* argument no longer supports option FMT\_NO\_SCALE which allowed you to dynamically get the raw value or the engineering value of a tag in the list. If you wish to get the raw value of a tag, add it to the list with this mode by calling `ctListAddEx` and specifying `bRaw = TRUE`.

## Return Value

If the function succeeds, the return value is TRUE. If the function does not succeed, the return value is FALSE. To get extended error information, call `GetLastError()`.

If an error occurred when reading the data from the I/O Device, the return value will be FALSE and `GetLastError()` will return the associated Plant SCADA error code.

## Related Functions

[ctListItem](#), [ctOpen](#), [ctListNew](#), [ctListFree](#), [ctListAdd](#), [ctListRead](#), [ctListWrite](#)

## Example

See [ctListItem](#).

## ctListDelete

Frees a tag created with [ctListAdd](#). Your program is permitted to call `ctListDelete()` while a read or write is pending on another thread. The `ctListWrite()` and `ctListRead()` will return once the tag has been deleted.

## Syntax

**ctListDelete(*hTag*)**

*hTag*

Type: HANDLE

Input/output: Input

Description: The handle to the tag, as returned from [ctListAdd\(\)](#).

## Return Value

If the function succeeds, the return value is TRUE. If the function does not succeed, the return value is FALSE. To get extended error information, call GetLastError().

## Related Functions

[ctOpen](#), [ctListNew](#), [ctListFree](#), [ctListAdd](#), [ctListRead](#), [ctListWrite](#), [ctListData](#), [ctListItem](#)

## Example

```
HANDLE hList;
HANDLE hTagOne;
HANDLE hTagTwo;
hList = ctListNew(hCTAPI, 0);
hTagOne = ctListAdd(hList, "TagOne");
hTagTwo = ctListAdd(hList, "TagTwo");
ctListRead(hList, NULL); // read TagOne and TagTwo
ctListData(hList, hTagOne, sBufOne, sizeof(sBufOne), 0);
ctListData(hList, hTagTwo, sBufTwo, sizeof(sBufTwo) , 0);
ctListDelete(hTagOne); // delete TagOne;
ctListRead(hList, NULL); // read TagTwo only
ctListData(hList, hTagTwo, sBufTwo, sizeof(sBufTwo) , 0);
```

## ctListEvent

Returns the elements in the list which have changed state since they were last read using the [ctListRead\(\)](#) function. You need to have created the list with CT\_LIST\_EVENT mode in the [ctListNew\(\)](#) function.

## Syntax

**ctListEvent(*hCTAPI*, *dwMode*)**

*hCTAPI*

Type: Handle

Input/output: Input

Description: The handle to the CTAPI as returned from [ctListNew\(\)](#).

*dwMode*

Type: Dword

Input/output: Input

Description: The mode of the list event. You need to use the same mode for each call to [ctListEvent\(\)](#) until NULL is returned before changing mode. The following modes are supported:

**CT\_LIST\_EVENT\_NEW** - Gets notifications when tags are added to the list. When this mode is used, you will get an event message when new tags added to the list.

**CT\_LIST\_EVENT\_STATUS** - Gets notifications for status changes. Tags will change status when the I/O Device goes offline. When this mode is used, you will get a notification when the tag goes into #COM and another one when it goes out of #COM. You can verify that the tag is in #COM when an error is returned from [ctListData\(\)](#) for that tag.

## Return Value

If the function succeeds, the return value specifies a handle to a tag which has changed state since the last time ctListRead was called. If the function does not succeed or there are no changes, the return value is NULL. To get extended error information, call GetLastError().

## Related Functions

[ctListAdd](#), [ctListDelete](#), [ctListWrite](#), [ctListData](#), [ctListItem](#)

## Example

```
HANDLE hList; HANDLE hTag[100];
hList = ctListNew(hCTAPI, CT_LIST_EVENT);
hTagArray[0] = ctListAdd(hList, "TagOne");
hTagArry[1] = ctListAdd(hList, "TagTwo");
and so on...
while (TRUE) {
    ctListRead(hList, NULL);
    hTag = ctListEvent(hList, 0);
    while (hTag != NULL) {
        // hTag has changed state, do whatever you need
        hTag = ctListEvent(hList, 0);
    }
}
```

## ctListFree

Frees a list created with ctListNew. Every tag added to the list is freed, you do not have to call ctListDelete() for each tag. not call ctListFree() while a read operation is pending. Wait for the read to complete before freeing the list.

## Syntax

**ctListFree(*hList*)**  
*hList*  
Type: HANDLE  
Input/output: Input  
Description: The handle to the list, as returned from ctListNew().

## Return Value

If the function succeeds, the return value is TRUE. If the function does not succeed, the return value is FALSE. To get extended error information, call GetLastError().

## Related Functions

[ctOpen](#), [ctListNew](#), [ctListAdd](#), [ctListDelete](#), [ctListEvent](#), [ctListRead](#), [ctListWrite](#), [ctListData](#)

## Example

See [ctListNew](#).

## ctListNew

Creates a new list. The CTAPI provides two methods to read data from I/O Devices. Each level varies in its complexity and performance. The simplest way to read data is via the `ctTagRead()` function. This function reads the value of a single variable, and the result is returned as a formatted engineering string.

The List functions provide a higher level of performance for reading data than the tag based interface. The List functions also provide support for overlapped operations.

The list functions allow a group of tags to be defined and then read as a single request. They provide a simple tag based interface to data which is provided in formatted engineering data. You can create several lists and control each individually.

Tags can be added to, or deleted from lists dynamically, even if a read operation is pending on the list.

## Syntax

`ctListNew(hCTAPI, dwMode)`

*hCTAPI*

Type: Handle

Input/output: Input

Description: The handle to the CTAPI as returned from [ctOpen\(\)](#).

*dwMode*

Type: DWORD

Input/output: Input

Description: The mode of the list creation. The following modes are supported:

**CT\_LIST\_EVENT** - Creates the list in event mode. This mode allows you to use the [ctListEvent\(\)](#) function.

**CT\_LIST\_LIGHTWEIGHT\_MODE** - Setting this mode for a list means any tag updates will use a "lightweight" version of the tag value that does not include a quality timestamp or value timestamp.

These flags can be used in combination with each other.

## Return Value

If the function succeeds, the return value specifies a handle. If the function does not succeed, the return value is NULL. To get extended error information, call `GetLastError()`.

## Related Functions

[ctOpen](#), [ctListFree](#), [ctListAdd](#), [ctListDelete](#), [ctListEvent](#), [ctListRead](#), [ctListWrite](#), [ctListData](#)

## Example

```
HANDLE hList;
HANDLE hTagOne;
HANDLE hTagTwo;
hList = ctListNew(hCTAPI, 0);
hTagOne = ctListAdd(hList, "TagOne");
hTagTwo = ctListAdd(hList, "TagTwo");
while (you want the data) {
    ctListRead(hList, NULL);
    ctListData(hTagOne, sBufOne, sizeof(sBufOne), 0);
    ctListData(hTagTwo, sBufTwo, sizeof(sBufTwo), 0);
}
ctListFree(hList);
```

## ctListItem

Gets the tag element item data. For specific quality part values please refer to The Quality Tag Element.

## Syntax

**ctListItem(*hTag*, *dwItem*, *pBuffer*, *dwLength*, *dwMode*)**

*hTag*

Type: HANDLE

Input/output: Input

Description: The handle to the tag, as returned from [ctListAdd\(\)](#).

*dwitem*

Type: DWORD

Input/output: Input

Description: The tag element item:

CT\_LIST\_VALUE - value.

CT\_LIST\_TIMESTAMP – timestamp.

CT\_LIST\_VALUE\_TIMESTAMP – value timestamp.

CT\_LIST\_QUALITY\_TIMESTAMP quality timestamp.

CT\_LIST\_QUALITY\_GENERAL - quality general.

CT\_LIST\_QUALITY\_SUBSTATUS - quality substatus.

CT\_LIST\_QUALITY\_LIMIT - quality limit .

CT\_LIST\_QUALITY\_EXTENDED\_SUBSTATUS - quality extended substatus.

CT\_LIST\_QUALITY\_DATASOURCE\_ERROR - quality datasource error.

CT\_LIST\_QUALITY\_OVERRIDE - quality override flag.

CT\_LIST\_QUALITY\_CONTROL\_MODE - quality control mode flag.

*pBuffer*

Type: VOID\*

Input/output: Input

Description: Pointer to a buffer to return the data. The data is returned scaled and as a formatted string.

*dwLength*

Type: Dword

Input/output: Input

Description: Length (in bytes) of the raw data buffer.

*dwMode*

Type: DWORD

Input/output: Input

Description: Mode of the data. The following modes are supported:

**0 (zero)** - The value is scaled using the scale specified in the Plant SCADA project, and formatted using the format specified in the Plant SCADA project.

**FMT\_NO\_FORMAT** - The value is not formatted to the format specified in the Plant SCADA project. A default format is used. If there is a scale specified in the Plant SCADA project, it will be used to scale the value.

The dwMode argument no longer supports option FMT\_NO\_SCALE which allowed you to dynamically get the raw value or the engineering value of a tag in the list. If you wish to get the raw value of a tag, add it to the list with this mode by calling ctListAddEx and specifying bRaw = TRUE..

## Return Value

If the function succeeds, the return value is TRUE. If the function does not succeed, the return value is FALSE. To get extended error information, call GetLastError().

If an error occurred when reading the data from the I/O Device, the return value will be FALSE and GetLastError() will return the associated Plant SCADA error code.

## Related Functions

[ctOpen](#), [ctListNew](#), [ctListFree](#), [ctListAdd](#), [ctListRead](#), [ctListData](#), [ctListWrite](#)

## Example

```
HANDLE hCTAPI;
HANDLE hList;
HANDLE hTagOne;
HANDLE hTagOneField;
HANDLE hTagOneControlMode;
HANDLE hTagOneStatus;
char sProcessValue[20];
char sProcessValueField[20];
char sProcessValueControlMode[20];
char sProcessValueStatus[20];
char sProcessValueFieldQualityGeneral[20];
char sProcessValueFieldQualitySubstatus[20];
char sProcessValueFieldQualityLimit[20];
char sProcessValueFieldQualityExtendedSubstatus[20];
char sProcessValueFieldQualityOverride[20];
char sProcessValueFieldQualityControlMode[20];
char sProcessValueFieldQualityDatasourceError[20];
char sProcessValueFieldTimestamp[20];
```

```
char sProcessValueFieldValueTimestamp[20];
char sProcessValueFieldQualityTimestamp[20];
hCTAPI = ctOpen(NULL, NULL, NULL, 0);
hList = ctListNew(hCTAPI, 0);
hTagOne = ctListAdd(hList, "TagOne");
hTagOneField = ctListAdd(hList, "TagOne.Field");
hTagOneControlMode = ctListAdd(hList, "TagOne.ControlMode");
hTagOneStatus = ctListAdd(hList, "TagOne.Status");
ctListRead(hList, NULL);
ctListData(hTagOne, sProcessValue, sizeof(sProcessValue), 0);
ctListData(hTagOneField, sProcessValueField, sizeof(sProcessValueField) , 0);
ctListData(hTagOneControlMode, sProcessValueControlMode, sizeof(sProcessValueControlMode) , 0);
ctListData(hTagOneStatus, sProcessValueStatus, sizeof(sProcessValueStatus) , 0);
ctListItem(hTagOneField, CT_LIST_VALUE, sProcessValueField, sizeof(sProcessValueField), 0);
ctListItem(hTagOneField, CT_LIST_TIMESTAMP, sProcessValueFieldTimestamp,
sizeof(sProcessValueFieldTimestamp), 0);
ctListItem(hTagOneField, CT_LIST_VALUE_TIMESTAMP, sProcessValueFieldValueTimestamp,
sizeof(sProcessValueFieldValueTimestamp), 0);
ctListItem(hTagOneField, CT_LIST_QUALITY_TIMESTAMP, sProcessValueFieldQualityTimestamp,
sizeof(sProcessValueFieldQualityTimestamp), 0);
ctListItem(hTagOneField, CT_LIST_QUALITY_GENERAL, sProcessValueFieldQualityGeneral,
sizeof(sProcessValueFieldQualityGeneral), 0);
ctListItem(hTagOneField, CT_LIST_QUALITY_SUBSTATUS, sProcessValueFieldQualitySubstatus,
sizeof(sProcessValueFieldQualitySubstatus), 0);
ctListItem(hTagOneField, CT_LIST_QUALITY_LIMIT, sProcessValueFieldQualityLimit,
sizeof(sProcessValueFieldQualityLimit), 0);
ctListItem(hTagOneField, CT_LIST_QUALITY_EXTENDED_SUBSTATUS,
sProcessValueFieldQualityExtendedSubstatus,
sizeof(sProcessValueFieldQualityExtendedSubstatus), 0);
ctListItem(hTagOneField, CT_LIST_QUALITY_OVERRIDE, sProcessValueFieldQualityOverride,
sizeof(sProcessValueFieldQualityOverride), 0);
ctListItem(hTagOneField, CT_LIST_QUALITY_CONTROL_MODE,
sProcessValueFieldQualityControlMode, sizeof(sProcessValueFieldQualityControlMode), 0);
ctListItem(hTagOneField, CT_LIST_QUALITY_DATASOURCE_ERROR,
sProcessValueFieldQualityDatasourceError, sizeof(sProcessValueFieldQualityDatasourceError),
0);
ctListFree(hList);
```

## ctListRead

Reads the tags on the list. This function will read tags which are attached to the list. Once the data has been read from the I/O Devices, you may call `ctListData()` to get the values of the tags. If the read does not succeed, `ctListData()` will return an error for the tags that cannot be read.

While `ctListRead()` is pending you are allowed to add and delete tags from the list. If you delete a tag from the list while `ctListRead()` is pending, it may still be read one more time. The next time `ctListRead()` is called, the tag will not be read. If you add a tag to the list while `ctListRead()` is pending, the tag will not be read until the next time `ctListRead()` is called. You may call `ctListData()` for this tag as soon as you have added it. In this case `ctListData()` will not succeed, and `GetLastError()` will return `GENERIC_INVALID_DATA`.

You can only have 1 pending read command on each list. If you call `ctListRead()` again for the same list, the function will not succeed.

Before freeing the list, check that there are no reads still pending. wait for the any current `ctListRead()` to return

and then delete the list.

## Syntax

**ctListRead(*hList*, *pctOverlapped*)**

*hList*

Type: HANDLE

Input/output: Input

Description: The handle to the list, as returned from ctListNew().

*pctOverlapped*

Type: CTOVERLAPPED\*

Input/output: Input

Description: CTOVERLAPPED structure. This structure is used to control the overlapped notification. Set to NULL if you want a synchronous function call.

## Return Value

If the function succeeds, the return value is TRUE. If the function does not succeed, the return value is FALSE. To get extended error information, call GetLastError().

If an error occurred when reading any of the list data from the I/O Device, the return value will be FALSE and GetLastError() will return the associated Plant SCADA error code. As a list can contain tags from many data sources, some tags may be read correctly while other tags may not. If any tag read does not succeed, ctListRead() will still return TRUE, the other tags will contain valid data. You can call ctListData() to retrieve the value of each tag and the individual error status for each tag on the list.

## Related Functions

[ctOpen](#), [ctListNew](#), [ctListFree](#), [ctListAdd](#), [ctListWrite](#), [ctListData](#), [ctListItem](#)

## Example

See [ctListNew](#)

To read the Paging Alarm property using ctListRead:

```
HANDLE hList;
HANDLE hAlarmOne;
HANDLE hAlarmTwo;
hList = ctListNew(hCTAPI, 0);
hTagOne = ctListAdd(hList, "AlarmOne.Paging");
hTagTwo = ctListAdd(hList, "AlarmTwo.Paging");
while (you want the data) {
    ctListRead(hList, NULL);
    ctListData(hAlarmOne, sBufOne, sizeof(sBufOne), 0);
    ctListData(hAlarmTwo, sBufTwo, sizeof(sBufTwo), 0);
}
ctListFree(hList);
```

## ctListWrite

Writes to a single tag on the list.

### Syntax

**ctListWrite(*hTag*, *sValue*, *pctOverlapped*)**

*hTag*

Type: HANDLE

Input/output: Input

Description: The handle to the tag, as returned from ()�.

*sValue*

Type: LPCSTR

Input/output: Input

Description: The value to write to the tag as a string. The value will be converted and scaled into the correct format to write to the tag.

*pctOverlapped*

Type: CTOVERLAPPED\*

Input/output: Input

Description: CTOVERLAPPED structure. This structure is used to control the overlapped notification. Set to NULL if you want a synchronous function call.

### Return Value

If the function succeeds, the return value is TRUE. If the function does not succeed, the return value is FALSE. To get extended error information, call GetLastError().

### Related Functions

[ctOpen](#), [ctListNew](#), [ctListFree](#), [ctListDelete](#), [ctListRead](#), [ctListData](#), [ctListItem](#)

### Example

```
HANDLE hTagOne;
HANDLE hList;
hList = ctListNew(hCTAPI, 0);
hTagOne = ctListAdd(hList, "TagOne");
ctListWrite(hTagOne, "1.23", NULL); // write to TagOne
```

## ctOpen

Opens a connection to the Plant SCADA API. The CTAPI.DLL is initialized and a connection is made to Plant SCADA. If Plant SCADA is not running when this function is called, the function will exit and report an error. This function needs to be called before any other CTAPI function to initialize the connection to Plant SCADA.

If you use the CT\_OPEN\_RECONNECT mode, and the connection is lost, the CTAPI will attempt to reconnect to

Plant SCADA. When the connection has been re-established, you can continue to use the CTAPI. However, while the connection is down, every function will return errors. If a connection cannot be created the first time `ctOpen()` is called, a valid handle is still returned; however `GetLastError()` will indicate an error.

If you do not use the `CT_OPEN_RECONNECT` mode, and the connection to Plant SCADA is lost, you need to free handles returned from the CTAPI and call `ctClose()` to free the connection. You need to then call `ctOpen()` to re-establish the connection and re-create any handles.

---

**Note:** To use CtAPI on a remote computer without installing Plant SCADA, you will need several CtAPI binaries. These binaries are packaged in the files `CtApi.Win32.Redist.zip` and `CtApi.x64.Redist.zip`, which can be found in the `<Plant SCADA_Disk>\Plant SCADA <VersionNumber>\Extras\CtAPI` directory of the Plant SCADA installation media. See *Running CtAPI Applications* in the topic [CtAPI](#).

If calling this function from a remote computer, a valid username and a non-blank password needs to be used.

To establish encrypted communications for a remote connection, see [Secure the Connection from a Remote CtAPI Application](#).

## Syntax

**ctOpen(sComputer, sUser, sPassword, nMode)**

*sComputer*

Type: LPCSTR

Input/output: Input

Description: The computer you want to communicate with via CTAPI. For a local connection, specify NULL as the computer name. The Windows Computer Name is the name as specified in the Identification tab, under the Network section of the Windows Control Panel.

---

**Note:** To use a local connection when the CtAPI process is running as a Windows service, you need to add the user account as a member to the **Runtime Users** security role in Configurator (see [Security Roles](#)). You also need to check that the **Enable Encryption** option is selected on Configurator's Encryption page.

---

**Note:** Using a hostname to connect to a CTAPI server running Plant SCADA 8.20 or earlier will not work if Windows is configured to return an IPv6 address for that hostname, as 8.20 and earlier do not support IPv6 network addresses. To avoid this, use an IPv4 address directly, or if encryption is enabled, configure Windows to return an IPv4 address in preference to an IPv6 address.

*sUser*

Type: LPCSTR

Input/output: Input

Description: A user name as defined in the Plant SCADA project running on the computer to which you want to connect, or a Windows user name if `nMode` is set to `CT_OPEN_WINDOWSUSER`.

This argument is only necessary if you are calling this function from a remote computer. On a local computer, it is optional.

*sPassword*

Type: LPCSTR

Input/output: Input

Description: The `sUser` password defined in the Plant SCADA project running on the computer to which you want to connect, or the password for the Windows user name if `nMode` is set to `CT_OPEN_WINDOWSUSER`.

This argument is only necessary if you are calling this function from a remote computer. You need to use a non-blank password. On a local computer, it is optional.

*nMode*

Type: DWORD

Input/output: Input

Description: The mode of the CtAPI call. Set this to 0 (zero). The following modes are supported:

**CT\_OPEN\_RECONNECT** (0x00000002) — Reopen connection on error or communication interruption. If the connection to Plant SCADA is lost CTAPI will continue to retry to connect to Plant SCADA.**CT\_OPEN\_READ\_ONLY** (0x00000004) — Open the CTAPI in read only mode. This allows read only access to data - you cannot write to any variable in Plant SCADA or call any Cicode function.**CT\_OPEN\_BATCH** (0x00000008) — Disables the display of message boxes when an error occurs.**CT\_OPEN\_EXTENDED** (0x00000010) — Allows for encrypted communication. This parameter is implicitly set when encryption is enabled (with the Accept encrypted and non-encrypted connections option not selected) via the Configurator.**CT\_OPEN\_WINDOWSUSER** (0x00000020) — Indicates that sUser and sPassword are Windows user credentials. If sUser and sPassword are blank, then the credentials of the Windows user that is launching the application will be used.

To use Windows credentials to run CtAPI operations, you need to confirm the following:

- 1) Your server needs to be running Plant SCADA 2020 R2 (or later).
- 2) You need to use the **CT\_OPEN\_WINDOWSUSER** mode, and either:

**CT\_OPEN\_EXTENDED** mode is used in the CtOpen call.

Or:

The encryption mode on the CtAPI client is set to "Encrypted".

**Return Value**

If the function succeeds, the return value specifies a handle. If the function does not succeed, the return value is NULL. Use GetLastError() to get extended error information.

**Related Functions**[ctCiCode](#), [ctClose](#), [ctEngToRaw](#), [ctGetOverlappedResult](#), [ctHasOverlappedIoCompleted](#), [ctRawToEng](#), [ctTagRead](#), [ctTagWrite](#)**Examples**

```
HANDLE hCTAPI;
hCTAPI = ctOpen(NULL, NULL, NULL, 0);
if (hCTAPI == NULL) {
    dwStatus = GetLastError(); // get error
} else {
    ctTagWrite(hCTAPI, "SP123", "1.23");
    ctClose(hCTAPI);
}
// example of open for remote TCP/IP connection.
hCTAPI = ctOpen("203.19.130.2", "ENGINEER", "PASSWORD", 0);
.
```

```
STRING sComputer, sUser, sPassword;
HANDLE hCTAPI;
// get input for sComputer, sUser, sPassword variables from UI, or 3rd party APIs
hCTAPI = ctOpen(sComputer, sUser, sPassword, CT_OPEN_WINDOWSUSER);
```

## ctOpenEx

Establishes the connection to the CtAPI server using the given client instance. Create the client instance prior to calling `ctOpenEx`, using the function `ctClientCreate`.

`ctOpenEx` provides exactly the same connection functionality as `ctOpen`, the only difference being that `ctOpen` also creates the CtAPI client instance. See [ctOpen](#) for details on the connection mechanism and the parameters involved.

## Syntax

`ctOpenEx(sComputer, sUser, sPassword, nMode, hCTAPI);`

*sComputer*

Type: LPCSTR

Input/output: Input

Description: The computer you want to communicate with via CtAPI. For a local connection, specify NULL as the computer name. The Windows Computer Name is the name as specified in the Identification tab, under the Network section of the Windows Control Panel.

---

**Note:** To use a local connection when the CtAP process is running as a Windows service, you need to:

- (i) Add the user account as a member to the **Runtime Users** security role in Configurator (see [Security Roles](#))
  - (ii) Check that the **Enable Encryption** option is selected on the Configurator, Encryption page.
- 

**Note:** Using a hostname to connect to a CtAPI server running Plant SCADA 8.20 or earlier will not work if Windows is configured to return an IPv6 address for that hostname, as 8.20 and earlier do not support IPv6 network addresses. To avoid this, use an IPv4 address directly, or if encryption is enabled, configure Windows to return an IPv4 address in preference to an IPv6 address.

---

*sUser*

Type: LPCSTR

Input/output: Input

Description: A user name as defined in the Plant SCADA project running on the computer to which you want to connect, or a Windows user name if *nMode* is set to `CT_OPEN_WINDOWSUSER`.

This argument is only necessary if you are calling this function from a remote computer. On a local computer, it is optional.

*sPassword*

Type: LPCSTR

Input/output: Input

Description: The *sUser* password defined in the Plant SCADA project running on the computer to which you want to connect, or the password for the Windows user name if *nMode* is set to `CT_OPEN_WINDOWSUSER`.

This argument is only necessary if you are calling this function from a remote computer. You need to use a non-blank password. On a local computer, it is optional.

*nMode*

Type: Dword

Input/output: Input

Description: The mode of the Cicode call.

**CT\_OPEN\_WINDOWSUSER** — Indicates that *sUser* and *sPassword* are Windows user credentials. If *sUser* and *sPassword* are blank, then the credentials of the Windows user that is launching the application will be used.

To use Windows credentials to run CtAPI operations, you need to confirm the following:

- Your server needs to be running Plant SCADA 2020 R2 (or later).
- You need to use the CT\_OPEN\_WINDOWSUSER mode, and either:
  - CT\_OPEN\_EXTENDED mode is used in the CtOpen call.
  - Or:
  - The encryption mode on the CtAPI client is set to "Encrypted" or "Mixed Mode".

*hCTAPI*

Type: Handle

Input/output: Input

Description: The handle to the CTAPI as returned from [ctOpen\(\)](#).

## Return Value

TRUE if successful, otherwise FALSE. Use [GetLastError\(\)](#) to get extended error information.

## Related Functions

[ctClientCreate](#), [ctOpen](#), [ctClose](#), [ctCloseEx](#), [ctClientDestroy](#)

## Example

See [ctClientCreate](#).

## ctRawToEng

Converts the raw I/O Device scale variable into Engineering scale. This is not necessary for the Tag functions as Plant SCADA will do the scaling. Scaling is not necessary for digitals, strings or if no scaling occurs between the values in the I/O Device and the Engineering values. You need to know the scaling for each variables as specified in the Plant SCADA Variable Tags table.

## Syntax

**ctRawToEng(*pResult*, *dValue*, *pScale*, *dwMode*)**

*pResult*

Type: Double

Input/output: Output

Description: The resulting raw scaled variable.

*dValue*

Type: Double

Input/output: Input

Description: The engineering value to scale.

*pScale*

Type: CTSCALE\*

Input/output: Input

Description: The scaling properties of the variable.

*dwMode*

Type: Dword

Input/output: Input

Description: The mode of the scaling. The following modes are supported:

**CT\_SCALE\_RANGE\_CHECK** - Range check the result. If the variable is out of range then generate an error. The pResult still contains the raw scaled value.

**CT\_SCALE\_CLAMP\_LIMIT** - Clamp limit to max or minimum scales. If the result is out of scale then set result to minimum or maximum scale (which ever is closest). No error is generated if the scale is clamped. Cannot be used with CT\_SCALE\_RANGE\_CHECK or CT\_SCALE\_NOISE\_FACTOR options.

**CT\_SCALE\_NOISE\_FACTOR** - Allow noise factor for range check on limits. If the variable is our of range by less than 0.1 % then a range error is not generated.

## Return Value

TRUE if successful, otherwise FALSE. Use GetLastError() to get extended error information.

## Related Functions

[ctOpen](#), [ctEngToRaw](#)

## Example

```
// SP123 is type INTEGER and has raw scale 0 to 32000 and Eng scale
// 0 to 100
HANDLE hList = ctListNew(s_hCTAPI, 0);
HANDLE hTag = ctListAddEx(hList, "SP123", TRUE, 500, -1);
CTSCALE Scale = { 0.0, 32000.0, 0.0, 100.0};
CHAR valueBuf[256] = {0};
double dRawValue = 0.0;
double dSetPoint = 0.0;
ctListRead(hList, NULL);
ctListData(hTag, valueBuf, sizeof(valueBuf), 0);
dRawValue = strtod(valueBuf, NULL);
ctEngToRaw(&dSetPoint, dRawValue, &Scale, CT_SCALE_RANGE_CHECK);
// dSetPoint now contains the Engineering scaled setpoint.
```

## ctSetManagedBinDirectory

Allows a CTAPI consumer to specify from where it will load certain CTAPI dependencies (.NET managed dependencies).

Typically, this would be the Plant SCADA Bin directory (for example, C:\Program Files (x86)\AVEVA Plant SCADA\Bin). This is only required when the dependencies are not located in the same directory as the consuming application, see *Running CTAPI Applications* in the topic [CtAPI](#) for more details.

This function takes a single string containing a path. Calling this multiple times is safe. If a different path is specified it will replace the previous path; if the same path is specified then nothing will happen.

If called with a NULL or empty string, then any previous path will be removed and the default search order will be used.

## Syntax

**ctSetManagedBinDirectory(*sPath*)**

*sPath*

Type: LPCSTR

Input/output: Input

Description: A path representing the location of the CTAPI managed dependencies.

## Return Value

The function will return TRUE if successful.

The function will return FALSE if passed an invalid path, a path that does not exist, or a path that specifies a file instead of a directory.

## Related Functions

This function should be called before [ctOpen](#).

## ctTagGetProperty

Gets the given property of the given tag.

## Syntax

**ctTagGetProperty(*hCTAPI*, *szTagName*, *szProperty*, *pData*, *dwBufferLength*, *dwType*)**

*hCTAPI*

Type: Handle

Input/output: Input

Description: The handle to the CTAPI as returned from [ctOpen\(\)](#).

*szTagName*

Type: LPCSTR

Input/output: Input

Description: The name of the tag. To specify cluster add "ClusterName." in front of the tag. For example Cluster1.Tag1 (note the period at the end of the cluster name).

Variable tags can be specified as a string in multiple forms. Refer to [Tag Names](#) for more information.

*szProperty*

Type: LPCSTR

Input/output: Input

Description: The property to read. Property names are case sensitive. Supported properties are:

**ArraySize:** Array size of the associated tag. Returns 1 for non-array types.

**DataBitWidth:** Number of bits used to store the value.

**Description:** Tag description.

**EngUnitsHigh:** Maximum scaled value.

**EngUnitsLow:** Minimum scaled value.

**Format:** Format bit string. The format information is stored in the integer as follows:

\* Bits 0-7 - format width

\* Bits 8-15 - number of decimal places

\* Bits 16 - zero-padded

\* Bit 17- left-justified

\* Bit 18 - display engineering units

\* Bit 20 - exponential (scientific) notation

**FormatDecPlaces:** Number of decimal places for default format.

**FormatWidth:** Number of characters used in default format.

**RangeHigh:** Maximum unscaled value.

**RangeLow:** Minimum unscaled value.

**Type:** Type of tag as a number:

\* 0 = Digital

\* 1 = Byte

\* 2 = Integer16

\* 3 = UInteger16

\* 4 = Long

\* 5 = Real

\* 6 = String

\* 7 = ULONG

\* 8 = Undefined

**Units:** Engineering Units for example %, mm, Volts.

*pData*

Type: VOID\*

Input/output: Output

Description: The output data buffer for the property value retrieved.

*dwBufferLength*

Type: DWORD

Input/output: Input

Description: The length of the output data buffer in bytes.

*dwType*

Type: DWORD

Input/output: Input

Description: The type of data to return.

Value	Meaning
DBTYPE_U11	UCHAR
DBTYPE_I1	1 byte INT
DBTYPE_I2	2 byte INT
DBTYPE_I4	4 byte INT
DBTYPE_R4	4 byte REAL
DBTYPE_R8	8 byte REAL
DBTYPE_BOOL	BOOLEAN
DBTYPE_BYTES	Byte Stream
DBTYPE_STR	NULL terminated STRING

## Return Value

If the function succeeds, the return value is non-zero. If the function does not succeed, the return value is zero. To get extended error information, call GetLastError().

## ctTagRead

Reads the value, quality and timestamp, not only a value. The data will be returned in string format and scaled using the Plant SCADA scales.

The function will request the given tag from the Plant SCADA I/O Server. If the tag is in the I/O Servers device cache the data will be returned from the cache. If the tag is not in the device cache then the tag will be read from the I/O Device. The time taken to complete this function will be dependent on the performance of the I/O Device. The calling thread is blocked until the read is completed.

## Syntax

**ctTagRead(*hCTAPI*, *sTag*, *sValue*, *dwLength*)**

*hCTAPI*

Type: Handle

Input/output: Input

Description: The handle to the CTAPI as returned from ctOpen().

*sTag*

Type: LPCSTR

Input/output: Input

Description: The tag name or tag name and element name, separated by a dot. If the element name is not

specified, it will be resolved at runtime as for an unqualified tag reference. You may use the array syntax [] to select an element of an array.

Variable tags can be specified as a string in multiple forms. Refer to Tag Names for more information.

#### sValue

Type: LPCSTR

Input/output: Output

Description: The buffer to store the read data. The data is returned in string format.

#### dwLength

Type: Dword

Input/output: Input

Description: The length of the read buffer. If the data is bigger than the dwLength, the function will not succeed.

## Return Value

TRUE if successful, otherwise FALSE. Use GetLastError() to get extended error information.

## Related Functions

[ctOpen](#), [ctTagWrite](#), [ctTagWriteEx](#)

## Example

```
HANDLE hCTAPI = ctOpen(NULL, NULL, NULL, 0);
...
char sProcessValue[20];
char sProcessValueField[20];
char sProcessValueControlMode[20];
char sProcessValueStatus[20];
ctTagRead(hCTAPI, "PV123", sProcessValue, sizeof(sProcessValue));
ctTagRead(hCTAPI, "PV123.Field", sProcessValueField, sizeof(sProcessValueField));
ctTagRead(hCTAPI, "PV123.Field.V", sProcessValueField, sizeof(sProcessValueField));
ctTagRead(hCTAPI, "PV123.ControlMode", sProcessValueControlMode,
sizeof(sProcessValueControlMode));
ctTagRead(hCTAPI, "PV123.Status", sProcessValueStatus, sizeof(sProcessValueStatus));
```

## ctTagReadEx

Performs the same as ctTagRead, but with an additional new argument. Reads the value, quality and timestamp, not only a value. The data will be returned in string format and scaled using the Plant SCADA scales.

The function will request the given tag from the Plant SCADA I/O Server. If the tag is in the I/O Servers device cache the data will be returned from the cache. If the tag is not in the device cache then the tag will be read from the I/O Device. The time taken to execute this function will be dependent on the performance of the I/O Device. The calling thread is blocked until the read is finished.

## Syntax

`ctTagReadEx(hCTAPI, sTag, sValue, dwLength, pctTagvalueItems)`

*hCTAPI*

Type: Handle

Input/output: Input

Description: The handle to the CTAPI as returned from [ctOpen\(\)](#).

*sTag*

Type: LPCSTR

Input/output: Input

Description: The tag name or tag name and element name, separated by a dot. If the element name is not specified, it will be resolved at runtime as for an unqualified tag reference. You may use the array syntax [] to select an element of an array.

Variable tags can be specified as a string in multiple forms. Refer to [Tag Names](#) for more information.

*sValue*

Type: LPCSTR

Input/output: Output

Description: The tag element value.

*dwLength*

Type: Dword

Input/output: Input

Description: The length of the read buffer. If the data is bigger than the dwLength, the function will not succeed.

*pctTagvalueItems*

Type: CT\_TAGVALUE\_ITEMS

Input/output: Input

Description: The pointer to CT\_TAGVALUE\_ITEMS structure containing quality and timestamp data or NULL. For specific quality part values, refer to [The Quality Item](#).

## Return Value

TRUE if successful, otherwise FALSE. Use [GetLastError\(\)](#) to get extended error information.

## Related Functions

[ctOpen](#), [ctTagWrite](#), [ctTagWriteEx](#)

## Example

```
HANDLE hCTAPI = ctOpen(NULL, NULL, NULL, 0);
char sProcessValue[20];
char sProcessValueField[20];
char sProcessValueControlMode[20];
char sProcessValueStatus[20];
CT_TAGVALUE_ITEMS ctTagvalueItems = {0};
ctTagvalueItems.dwLength = sizeof(CT_TAGVALUE_ITEMS);
```

```
ctTagReadEx(hCTAPI, "PV123", sProcessValue, sizeof(sProcessValue), NULL);
ctTagReadEx(hCTAPI, "PV123", sProcessValue, sizeof(sProcessValue), &ctTagvalueItems);
ctTagReadEx(hCTAPI, "PV123.Field", sProcessValueField, sizeof(sProcessValueField),
&ctTagvalueItems);
ctTagReadEx(hCTAPI, "PV123.ControlMode", sProcessValueControlMode,
sizeof(sProcessValueControlMode), &ctTagvalueItems);
ctTagReadEx(hCTAPI, "PV123.Status", sProcessValueStatus, sizeof(sProcessValueStatus),
&ctTagvalueItems);
```

## ctTagWrite

Writes to the given Plant SCADA I/O Device variable tag. The value, quality and timestamp, not only a value, is converted into the correct data type, then scaled and then written to the tag. If writing to an array element only a single element of the array is written to. This function will generate a write request to the I/O Server. The time taken to complete this function will be dependent on the performance of the I/O Device. The calling thread is blocked until the write is completed. Writing operation will succeed only for those tag elements which have read/write access.

## Syntax

**ctTagWrite(*hCTAPI*, *sTag*, *sValue*)**

*hCTAPI*

Type: Handle

Input/output: Input

Description: The handle to the CTAPI as returned from [ctOpen\(\)](#).

*sTag*

Type: LPCSTR

Input/output: Input

Description: The tag name or tag name and element name, separated by a dot. If the element name is not specified, it will be resolved at runtime as for an unqualified tag reference. You may use the array syntax [] to select an element of an array.

Variable tags can be specified as a string in multiple forms. Refer to [Tag Names](#) for more information.

*sValue*

Type: LPCSTR

Input/output: Input

Description: The value to write to the tag as a string.

## Return Value

TRUE if successful, otherwise FALSE. Use [GetLastError\(\)](#) to get extended error information.

## Related Functions

[ctOpen](#), [ctTagWrite](#), [ctTagRead](#)

## Example

```
HANDLE hCTAPI = ctOpen(NULL, NULL, NULL, 0);
ctTagWrite (hCTAPI, "PV123", "123.12");
ctTagWrite (hCTAPI, "PV123.Field", "123.12");
ctTagWrite (hCTAPI, "PV123.Field.V", "123.12");
ctTagWrite (hCTAPI, "PV123.ControlMode", "1");
ctTagWrite (hCTAPI, "PV123.Status", "0");
```

## ctTagWriteEx

Performs the same as ctTagWrite, but with an additional new argument. Writes to the given Plant SCADA I/O Device variable tag. The value, quality and timestamp, not only a value, is converted into the correct data type, then scaled and then written to the tag. If writing to an array element only a single element of the array is written to. This function will generate a write request to the I/O Server. The time taken to complete this function will be dependent on the performance of the I/O Device.

If the value of pctOverlapped is NULL, the function behaves the same as ctTagWrite, and the calling thread is blocked until the write is completed. If the value of pctOverlapped is not NULL, the write is completed asynchronously and the calling thread is not blocked.

## Syntax

**ctTagWriteEx(*hCTAPI*, *sTag*, *sValue*, *pctOverlapped*)**

*hCTAPI*

Type: Handle

Input/output: Input

Description: The handle to the CTAPI as returned from ctOpen().

*sTag*

Type: LPCSTR

Input/output: Input

Description: The tag name or tag name and element name, separated by a dot to write to. If the element name is not specified, it will be resolved at runtime as for an unqualified tag reference. You may use the array syntax [] to select an element of an array.

Variable tags can be specified as a string in multiple forms. Refer to [Tag Names](#) for more information.

*sValue*

Type: LPSTR

Input/output: Input

Description: The value to write to the tag as a string.

*pctOverlapped*

Type: CTOVERLAPPED\*

Input/output: Input

Description: Passes in an overlapped structure so ctTagWriteEx can complete asynchronously. If the pctOverlapped structure is NULL, the function will block, completing synchronously.

## Return Value

TRUE if successful, otherwise FALSE. Use GetLastError() to get extended error information.

## Related Functions

[ctOpen](#), [ctTagRead](#)

## AlmQuery

Provides an interface into the alarm summary archive from external applications, replacing the old [CtAPIAlarm](#) query. AlmQuery performs significantly better than CtAPIAlarm.

---

**Note:** The AlmQuery function has been re-implemented to return results based on the sequence of events (SOE) page. The format of the result has not been modified and is compatible with 7.20. The re-implementation includes:

- *TableName* : No longer supports ArgAnaAlm.
- Alarm tags are unique (enforced by the compiler), so the comment on *alarmTag* is no longer necessary.
- The comment associated with the sample is coming from the user comments on the SOE page.

AlmQuery is performed through the same mechanism as CtAPIAlarm. To establish the query and return the first record, you call [ctFindFirst](#). Then, to browse the remaining records, you call [ctFindNext](#). To access the data of the current record, [ctGetProperty](#) is called for each field of the record.

[ctFindFirst](#) is called with the following parameters:

- *hCtapi*: Handle to a valid CtAPI client instance.
- *szTableName*: Command string for the almquery, see below.
- *szFilter*: Not used for Almquery. Just pass in NULL.
- *hObject*: Handle to the first record retrieved for the query.
- *dwFlags*: Not used for Almquery. Just pass in 0.

The *szTableName* is the command string for the query and contains the parameters for the query.

## Syntax

**'ALMQUERY,Database,TagName,Starttime,StarttimeMs,Endtime,EndtimeMs,Period'**

---

**Note:** Arguments need to be comma-separated. Spaces between arguments are supported but not necessary. We recommend no spaces between arguments as they require more processing and take up more space in the query string.

**Database:**

The Alarm database that the alarm is in (alarm type). The following databases are supported: DigAlm (Digital), AnaAlm (Analog), AdvAlm (Advanced), HResAlm (Time Stamped), ArgDigAlm (Multi-Digital), TsDigAlm (Time Stamped Digital), TsAnaAlm (Timestamped Analog).

**TagName:**

The Alarm tag as a string.

**Starttime:**

The start time of the alarm query in seconds since 1970 as an integer in UTC time.

**StarttimeMs:**

The millisecond portion of the start time as an integer. It is expected to be a number between 0 and 999.

**Endtime:**

The end time of the alarm query in seconds since 1970 as an integer in UTC time.

**EndtimeMs:**

Millisecond portion of the end time as an integer. It is expected to be a number between 0 and 999.

**Period:**

Time period in seconds between the samples returned as a floating point value. The only decimal separator supported is the `.'.

## Return Value

The maximum number of samples returned is the time range divided by the period, plus 3 (one for the sample exactly on the end time, and two for the previous and next samples).

---

**Note:** Divide the period evenly into the time range, otherwise one extra sample may be returned.

The AlmQuery does not return interpolated samples in periods where there were no alarm samples. However, to stay within the allowable number of samples, the raw alarm samples will be compressed when more than one sample occurs in one period.

When this compression occurs, the returned sample is flagged as a multiple sample. The timestamp is then an average of the samples within the period. The value and comment returned reflects that of the last sample in the period.

The following properties are returned for each data record of the query.

- **DateTime:** The time of the alarm sample in seconds since 1970 as an integer. This Time is in UTC (Universal Time Coordinates).
- **MSeconds:** The millisecond component of the time of the trend sample as an integer. This value is in between 0 and 999.
- **Comment:** The comment associated with the alarm sample as a string. Corresponds to the latest user comment associated with the most recent event on the alarm within the current period.
- **Value:** The alarm value of the sample as an unsigned integer. See below for a detailed description of the alarm value. The alarm value contains information describing the state of the alarm at the time of the sample:

**bGood (Bit 0)**- Future use only, intended to show when the quality of the alarm data goes bad. At the moment every sample has this bit set to 1 to say the sample is good.

**bDisabled (Bit 1)**- 1 if the alarm is disabled at the sample's time, 0 otherwise.

**bMultiple (Bit 2)**- 1 if the alarm sample is based on multiple raw samples, 0 if it is based on only 1 raw sample.

**bOn (Bit 3)**- 1 if the alarm is on at the sample's time, 0 otherwise.

**bAck (Bit 4)**- 1 if the alarm is acknowledged at the sample's time, 0 otherwise.

**state (Bits 5 - 7)**- Contains the state information of the alarm at the sample's time.

The alarm state represents the different states of the different alarm types. The state only contains relevant information if the alarm is on.

For analog, and time-stamped analog alarm the state can be as follows:

- **Deviation High (1)**- The alarm has deviated above the Setpoint by more than the specified threshold.
- **Deviation Low (2)**- The alarm has deviated below the Setpoint by more than the specified threshold.
- **Rate of Change (3)**- The alarm has changed at a faster rate than expected.
- **Low (4)**- The alarm has entered the low alarm range of values.
- **High (5)**- The alarm has entered the high alarm range of values.
- **Low Low (6)**- The alarm has entered the low low alarm range of values.
- **High High (7)**- The alarm has entered the high high alarm range of values.

For Multi-Digital Alarms the state can be as follows:

- **000 (0)**- Digital tags for the alarm are off.
- **00A (1)**- Tag A is on, B and C are off.
- **0B0 (2)**- Tag B is on, A and C are off.
- **0BA (3)**- Tags B and A are on, C is off.
- **C00 (4)**- Tag C is on, B and C are off.
- **C0A (5)**- Tag C and A are on, B is off.
- **CBO (6)**- Tag C and B are on, A is off.
- **CBA (7)**-Digital tags for the alarm are on.

For the rest of the alarm types ignore the state information.

## TrnQuery

Provides a powerful interface into the trend archive from external applications, replacing the old [CtAPITrend](#) query. TrnQuery performs significantly better than CtAPITrend.

TrnQuery is performed through the same mechanism as CtAPITrend. To establish the query and return the first record, you call [ctFindFirst](#). Then, to browse the remaining records, you call [ctFindNext](#). To access the data of the current record, [ctGetProperty](#) is called for each field of the record.

[ctFindFirst](#) is called with the following parameters:

- *hCtapi*: handle to a valid Ctapi client instance.
- *szTableName*: command string for the Trnquery, see below.
- *szFilter*: Not used for Trnquery. Just pass in NULL.
- *hObject*: handle to the first record retrieved for the query.
- *dwFlags*: Not used for Trnquery. Just pass in 0.

The *szTableName* is the command string for the query. It contains the parameters for the query.

## Syntax

**TRNQUERY,Endtime,EndtimeMs,Period,NumSamples,Tagname,Displaymode,Datamode,InstantTrend,SamplePeriod'**

**Note:** Arguments needs to be comma-separated. Spaces between arguments are supported but not necessary. We recommend no spaces between arguments as they require more processing and take up more space in the

---

query string.

***Endtime:***

End time of the trend query in seconds since 1970 as an integer. This time is expected to be a UTC time (Universal Time Coordinates).

***EndtimeMs:***

Millisecond portion of the end time as an integer, expected to be a number between 0 and 999.

***Period:***

Time period in seconds between the samples returned as a floating point value. The only decimal separator supported is the `.'

***NumSamples:***

Number of samples requested as an integer. The start time of the request is calculated by multiplying the Period by NumSamples - 1, then subtracting this from the EndTime.

The actual maximum amount of samples returned is actually NumSamples + 2. This is because we return the previous and next samples before and after the requested range. This is useful as it tells you where the next data is before and after where you requested it.

***TagName:***

The name of the trend tag as a string. This query only supports the retrieval of trend data for one trend at a time.

***DisplayMode:***

Specifies the different options for formatting and calculating the samples of the query as an unsigned integer.

See [Display Mode](#) for information.

***DataMode:***

Mode of this request as an integer. 1 if you want the timestamps to be returned with their full precision and accuracy. Mode 1 does not interpolate samples where there were no values. 0 if you want the timestamps to be calculated, one per period. Mode 0 does interpolate samples, where there was no values.

***InstantTrend:***

An integer specifying whether the query is for an instant trend. 1 if for an instant trend. 0 if not.

***SamplePeriod:***

An integer specifying the requested sample period in milliseconds for the instant trend's tag value.

## Return Value

See [Returned Data](#) for return values.

## Display Mode

The data returned can vary drastically depending on the display mode of the TrnQuery. The display mode is split into the following mutually exclusive options:

**Ordering Trend sample options**

- 0 - Order returned samples from oldest to newest
- 2 - Order returned samples from newest to oldest. This mode is not supported when the Raw data option has been specified.

### Condense method options

- 0 - Set the condense method to use the mean of the samples.
- 4 - Set the condense method to use the minimum of the samples.
- 8 - Set the condense method to use the maximum of the samples.
- 12 - Set the condense method to use the newest of the samples.

### Timestamp options

- 0 - Returns the mean timestamp for all of the samples in the sample period.
- 33554432 - Returns the actual timestamp of the minimum/maximum/newest sample (depending on the current condense method setting). A condense method of "mean" will still return the mean timestamp for all of the samples in the sample period.

## Example

The following table represents samples every ten minutes across an hour.

0:10:00	40
0:20:00	70
0:30:00	60
0:40:00	20
0:50:00	10
1:00:00	50

Based on the current condense method setting (see above), the following would be returned when the timestamp option is set to 33554432.

- 0 (mean) — the timestamp = **0:35:00** ( $10 + 20 + 30 + 40 + 50 + 60 / 6 = 210 / 6 = 35$ )
- 4 (minimum) — value = 10; timestamp = **0:50:00**.
- 8 (maximum) — value = 70, timestamp = **0:20:00**.
- 12 (newest) — value = 50, timestamp = **1:00:00**.

### Stretch method options

- 0 - Set the stretch method to step.
- 128 - Set the stretch method to use a ratio.
- 256 - Set the stretch method to use raw samples (no interpolation).

### Gap Fill Constant option

- n - the number of missed samples that the user wants to gap fill) x 4096.

### Last valid value options

- 0 - If we are leaving the value given with a bad quality sample as 0.
- 2097152 - If we are to set the value of a bad quality sample to the last valid value (zero if there is no last valid value).

### Raw data options

- 0 - If we are not returning raw data, that is we are using the condense and stretch modes to compress and interpolate the data.
- 4194304 - If we are to return totally raw data, that is no compression or interpolation. This mode is only supported if we have specified the DataMode of the query = 1. When using this mode, more samples than the maximum specified above will be returned if there are more raw samples than the maximum in the time range.

## Returned Data

The following properties are returned for each data record of the query.

- *DateTime*: Time of the trend sample in seconds since 1970 as an integer in UTC (Universal Time Coordinates).
- *MSeconds*: Millisecond component of the time of the trend sample as an integer. This value is inbetween 0 and 999.
- *Value*: Trend value of the sample as a double.
- *Quality*: The quality information associated with the trend sample as an unsigned integer. The Quality property contains specific information in each bit in the unsigned integer.

To interpret the quality code, converted to its underlying binary value (for example, 0001 0001 0011) and read the bits from right to left. Each bit represents the following:

#### Value Type (Bits 0 - 3)

- *ValueType\_None* (0): There is no value in the given sample. Ignore the sample value, time and quality.
- *ValueType\_Interpolated* (1): The value has been interpolated from data around it.
- *ValueType\_SingleRaw* (2): The value is based on one raw sample.
- *ValueType\_MultipleRaw* (3): The value has been calculated from multiple raw samples.

#### Value Quality (Bits 4 - 7)

- *ValueQuality\_Bad* (0): Ignore the value of the sample as there was no raw data to base it on.
- *ValueQuality\_Good* (1): The value of the sample is valid, and is based on some raw data.

#### Last Value Quality (Bits 8 - 11)

- *LastValueQuality\_Bad* (0): The value of the sample should be ignored as there was no raw data to base it on.
- *LastValueQuality\_Good* (1): The value quality of the last raw sample in the period was good.
- *LastValueQuality\_NotAvailable* (2): The value quality of the last raw sample in the period was Not Available.
- *LastValueQuality\_Gated* (3): The value quality of the last raw sample in the period was Gated.

#### Partial Flag (Bit 12)

When the Partial Flag is set to 1 it indicates that the sample may change the next time it is read. This occurs when you get samples right at the current time, and a sample returned is not necessarily complete because more samples may be acquired in this period.

## Quality examples

If the returned quality code is **275**, the binary conversion is **0001 0001 0011**.

Reading from right to left, the quality code includes the following information:

- Bit 0: ValueType\_None (0): There is no value in the given sample. Ignore the sample value, time and quality.
- Bit 1: ValueType\_Interpolated (1): The value has been interpolated from data around it.
- Bit 4: ValueQuality\_Bad (0): Ignore the value of the sample as there was no raw data to base it on.
- Bit 8: LastValueQuality\_Bad (0): The value of the sample should be ignored as there was no raw data to base it on.

If the returned quality code is **4609**, the binary conversion is **0001 0010 0000 0001**.

Reading from right to left, the quality code includes the following information:

- Bit 0: ValueType\_None (0): There is no value in the given sample. Ignore the sample value, time and quality.
- Bit 9: LastValueQuality\_Good (1): The value quality of the last raw sample in the period was good.
- Bit 12: The Partial Flag is set to 1 indicating that the sample may change the next time it is read.

## CtAPIAlarm

Provides an interface into the alarm summary archive from external applications. For performance improvements, use the [AlmQuery](#) function instead.

To establish the query and return the first record, you call [ctFindFirst](#). Then, to browse the remaining records, you call [ctFindNext](#). To access the data of the current record, [ctGetProperty](#) is called for each field of the record.

[ctFindFirst](#) is called with the following parameters:

- hCtapi: Handle to a valid CtAPI client instance.
- szTableName: Command string for the almquery, see below.
- szFilter: Not used for Almquery. Just pass in NULL.
- hObject: Handle to the first record retrieved for the query.
- dwFlags: Not used for Almquery. Just pass in 0.

The szTableName is the command string for the query and contains the parameters for the query.

## Syntax

**CTAPIAlarm**(*Category,Type,Area*)

---

**Note:** Arguments needs to be comma-separated. Spaces between arguments are supported but not necessary. We recommend no spaces between arguments as they require more processing and take up more space in the query string.

---

**Category:**

The alarm category or group number to match. Set Category to 0 (zero) to match every alarm categorie.

**Type:**

The type of alarms to find:

**Non-hardware alarms**

1. Unacknowledged alarms, ON and OFF.
2. Acknowledged ON alarms.
3. Disabled alarms.
4. Configured alarms, i.e. Types 0 to 3, plus acknowledged OFF alarms.

If you do not specify a Type, the default is 0.

**Area:**

The area in which to search for alarms. If you do not specify an area, or if you set Area to -1, only the current area will be searched.

To simplify the passing of this argument, you could first pass the CTAPIAlarm() function as a string, then use the string as the szTableName argument (without quotation marks).

## CtAPITrend

Provides an interface into the trend archive from external applications, replacing the old [CtAPITrend](#) query. For performance improvements, use the TrnQuery function instead.

To establish the query and return the first record, you call [ctFindFirst](#). Then, to browse the remaining records, you call [ctFindNext](#). To access the data of the current record, [ctGetProperty](#) is called for each field of the record.

[ctFindFirst](#) is called with the following parameters:

- *hCtapi*: handle to a valid Ctapi client instance.
- *szTableName*: command string for the Trnquery, see below.
- *szFilter*: Not used for Trnquery. Just pass in NULL.
- *hObject*: handle to the first record retrieved for the query.
- *dwFlags*: Not used for Trnquery. Just pass in 0.

The *szTableName* is the command string for the query. It contains the parameters for the query.

## Syntax

**CTAPITrend(*sTime*,*sDate*,*Period*,*Length*,*Mode*,*Tag*)**

---

**Note:** Arguments needs to be comma-separated. Spaces between arguments are supported but not necessary. We recommend no spaces between arguments as they require more processing and take up more space in the query string.

---

***sTime*:**

The starting time for the trend. Set the time to an empty string to search the latest trend samples.

***sDate*:**

The date of the trend.

**Period:**

The period (in seconds) that you want to search (this period can differ from the actual trend period).

The Period argument used in the CTAPITrend() function needs to be 0 (zero) when this function is used as an argument to ctFindFirst() for an EVENT trend query.

**Length:**

The length of the data table, i.e. the number of rows of samples to be searched.

**Mode:**

The format mode to be used:

**Periodic trends**

- 1 - Search the Date and Time, followed by the tags.
- 2 - Search the Time only, followed by the tags.
- 3 - Ignore any invalid or gated values. (This mode is only supported for periodic trends.)

**Event trends**

- 1 - Search the Time, Date, and Event Number, followed by the tags.
- 2 - Search the Time and Event Number, followed by the tags.

**Tag:**

The trend tag name for the data to be searched.

To simplify the passing of this argument, you could first pass the CTAPITrend() function as a string, then use the string as the szTableName argument (without quotation marks).

## Graphics Builder Automation Interface

The Plant SCADA Graphics Builder now offers support for "automation," an OLE service that allows applications to expose their functionality, or to control the functionality of other applications on the same computer or across a network. As a result, applications can be integrated and automated with programming code.

The two key elements of automation are:

- Applications or software components, called **automation servers**, that can be controlled because their functionality has been exposed and made accessible to other applications. Examples of Microsoft Automation servers are Microsoft Office applications and Microsoft Project. These Automation servers expose their functionality through object models.
- Other applications or development tools, called **automation controllers**, that can control OLE Automation servers through programming code, by accessing the functionality exposed by the Automation servers. Examples of Microsoft Automation controllers are Microsoft Visual Basic, Microsoft Visual C++, and Microsoft Visual Basic for Applications (which is built into Microsoft Access, Microsoft Excel, and Microsoft Project).

*Automation* is the umbrella term for the process by which an automation controller sends instructions to an automation server (using the functionality exposed by the automation server), where they are run.

The Plant SCADA Graphics Builder automation interface enables the Plant SCADA Graphics Builder to act as an automation server as it exposes many Graphics Builder functions.

The interface supports a simple object model: functions are on the root level. Names are structured and contain a group identifier and a function name; for example, DrawLine, DrawRectangle, PositionAt, PositionRotate, ProjectSelect, ProjectUpgrade. These functions can be called from a Visual Basic (VB) program.

**Note:** In the VB development environment, the reference GraphicsBuilder Type Library needs to have previously been selected. If it has not, choose **References** from the Project menu in the VB and check the Graphics Builder Type Library.

## Example

The following sample VB code allows you to create a new Plant SCADA page, place a Genie at a specific location, set one of its parameter, draw a line, and then save the page with the name "TEST". This example is compatible with the ExampleSA project.

```
Dim GraphicsBuilder As IGraphicsBuilder2
Set GraphicsBuilder = New GraphicsBuilder.GraphicsBuilder
With GraphicsBuilder
    .Visible = True
    .PageNew "sa_include", "situational_awareness", "pagecontent", 0, False, True
    .LibraryObjectPlace "sa_include", "sa_filter", "item_priority_hd1080", 0, True
    .PositionAt 300, 500
    .LibraryObjectPutProperty "AsmPriority", "1"
    .DrawLine 100, 100, 300, 300
    .AttributeLineColour = 120
    .PageSaveAs "ExampleSA", "TEST_AUTOMATION"
    .PageClose
End With
Set GraphicsBuilder = Nothing
```

## See Also

[Error Handling](#)  
[Automation Events](#)  
[Function Categories](#)

## Error Handling

Functions, when called from VB, throw an exception on error. The following table lists the possible HRESULT errors that may be encountered:

C++ define	Hex value	Hex codes in VB	Description
S_OK	0	No exception	Successful execution
E_INVALIDARG	80070057	5	One or more arguments are out of range
E_HANDLE	80070006	80070006	No active object (page or graphical object)
E_POINTER	80004003	80004003	Missing or broken link encountered
E_ABORT	80000007	11F	Enumeration terminated

C++ define	Hex value	Hex codes in VB	Description
			or function manually canceled (for example ProjectUpdate)
E_FAIL	80004005	80004005	Every other error

The following VB code can be used to process the error code:

```
On Error Resume Next
Err.Clear
GraphicsBuilder.LibObjectName Project, File, Page, Type
If Err.Number <> 0 Then
    Debug.Print "Error occurred in LibObjectName"
End If
```

Note the following points:

- VB sets the Err variable only in the erroneous case. It will not be set to 0 if the function succeeds.
- When VB handles an exception, it ignores the functions parameters. Hence when a function like ProjectNext does not succeed, the returned string is undefined and not an empty string.

The functions in the groups Page Functions, Options Functions, Object Drawing and Property Functions, Text Property Functions and the individual functions LibSelectionHooksEnabled, SelectionEventEnabled, BrokenLinkCancelEnabled and Visible are treated as variables in VB.

When calling these functions from C++, you need to use a "put\_" or "get\_" prefix, for example, "put\_Visible(TRUE)", "get\_Visible(bValue)" to set or fetch the values, except if the Attribute is read-only. In this case the function is the same in C++; for example, PageName.

To evaluate the correct function name for C++ reference the Type library CTDRAW32.TLB, which can be found in Plant SCADA's BIN directory. You can use Microsoft's Visual Studio Tool OLE / COM Object Viewer (select menu File | View Typelib...) to look at a type library.

## See Also

[Automation Events](#)

## Automation Events

The graphics builder also provides event-based notification of actions, which an Automation client can intercept and react to accordingly. The following example creates a form, creates a graphics builder automation object with event capability and performs actions on two events that the graphics builder might generate, pasting a symbol and saving a page.

To enable this:

- The Graphics Builder object needs to be declared "WithEvents"
- The event handler subroutine needs to have the correct name and signature. Note how the event handler function names are gb, the graphics builder object, followed by \_<eventName> e.g gb\_PasteSymbol. This is consistent with standard Visual Basic event handling subroutine naming.

For details, see the individual event subroutine description.

```
Private WithEvents gb As GraphicsBuilder.GraphicsBuilder
Public Sub Form_Load()
    Set gb = New GraphicsBuilder.GraphicsBuilder
    gb.LibrarySelectionHooksEnabled = True
    gb.Visible = True
End Sub
Public Sub gb_PasteSymbol()
    MsgBox ("PasteSymbol")
End Sub
Private Sub gb_PageSaved(ByVal Project As String, ByVal Page As String,
    ByVal LastPage As Boolean)
    MsgBox "PageSaved: " + Project + "." + Page + "--"
End Sub
```

## See Also

[Error Handling](#)

## Function Categories

This table lists the Plant SCADA functions exposed through the Graphics Builder automation interface, grouped into the following categories:

<a href="#">Alarm Indicator Functions</a>	Allows you to define the look of the Alarm Indicator attached to a genie. Related to the Alarm Indicator Properties Tab part of the Genie Properties dialog.
<a href="#">Arrange and Position Functions</a>	Allow you to modify the position of a selected object in three dimensions (X,Y and Z order).
<a href="#">Composite Genie Functions</a>	Allow you to place or locate Composite Genies.
<a href="#">Dynamic Properties Functions</a>	Allow you to modify the dynamic properties of the graphics objects in your project (movement, scaling, rotation, sliders, dynamic color fill).
<a href="#">Event Functions</a>	Allow you to use the automation dispatch mechanism to fire events in specific situations.
<a href="#">Library Object Functions</a>	Allow you to use and manipulate the objects stored in libraries in your project. This includes such objects as Genies, Super Genies, Symbols, and so on.
<a href="#">Metadata Functions</a>	Allow you to configure the metadata of the current object on a page.
<a href="#">Miscellaneous Functions</a>	Used for special interactions with the Graphics Builder, for example an external drag-and-drop action could be performed by requesting the active window handle.

Object Drawing and Property Functions	Allow you to draw objects and manipulate the properties of objects.
Options Functions	Relate to the options found under the Graphics Builder 's Tools menu.
Page Functions	Allow you to manipulate the pages in your project (for example open, close, save, delete), and select objects on those pages. This includes templates, symbols, Genies, Super Genies.
Page Properties Functions	Allow you to manipulate the properties of the pages in your project.
Project Functions	These functions operate on the project level. Some are actually initiated within Plant SCADA Studio.
Specific Functions	Currently include only the Visible function.
Text Property Functions	Allow you to read and modify the properties of the text objects in your project.

For details and a VB example on handling return and error values, see [Error Handling](#).

## See Also

[Automation Events](#)

## Alarm Indicator Functions

The following functions set and define the look of the Alarm indicator belonging to an object or a Genie.

<a href="#">AlarmIndicatorGetBorderIsInside</a>	Retrieves the status for the 'Is Inside Flag' for the alarm indicator.
<a href="#">AlarmIndicatorGetBorderPadding</a>	Retrieves the size of the border padding for the alarm indicator.
<a href="#">AlarmIndicatorGetBorderWidth</a>	Retrieves the width of the Alarm Indicator border.
<a href="#">AlarmIndicatorGetEnabled</a>	Retrieves the state of the Alarm Indicator option for an object or genie.
<a href="#">AlarmIndicatorGetEquipHierarchy</a>	Retrieves the location in the equipment hierarchy that will be used to determine the highest priority alarm for the alarm indicator.
<a href="#">AlarmIndicatorGetExpr</a>	Retrieves the Equipment string or expression belonging to the Alarm Indicator.

<a href="#">AlarmIndicatorGetFlagEnabled</a>	Retrieves the status of the Alarm Indicator flag.
<a href="#">AlarmIndicatorGetFlagLocation</a>	Retrieves the position of the flag for the Alarm Indicator.
<a href="#">AlarmIndicatorGetFrameColor</a>	Retrieves the color of the frame of the Alarm indicator.
<a href="#">AlarmIndicatorGetInclude</a>	Retrieves the status of the Include Equipment reference option for the Alarm indicator.
<a href="#">AlarmIndicatorSetBorderIsInside</a>	Sets the Alarm Indicator 'Is Inside flag'. If set to True the border is located within the extent of the object group or Genie.
<a href="#">AlarmIndicatorSetBorderPadding</a>	Sets the size of the border padding in pixels for the Alarm Indicator.
<a href="#">AlarmIndicatorSetBorderWidth</a>	Sets the width in pixels of the Alarm Indicator border.
<a href="#">AlarmIndicatorSetEnabled</a>	Sets the status of the Alarm Indicator option for an object or genie.
<a href="#">AlarmIndicatorSetEquipHierarchy</a>	Set the location in the equipment hierarchy that will be used to determine the highest priority alarm for the Alarm Indicator.
<a href="#">AlarmIndicatorSetExpr</a>	Set the Equipment string or expression that will be used to determine the highest priority alarm for the Alarm Indicator.
<a href="#">AlarmIndicatorSetFlagEnabled</a>	Sets the Alarm Indicator enabled flag. When set to True a flag is applied to the alarm indicator.
<a href="#">AlarmIndicatorSetFlagLocation</a>	Sets the Alarm Indicator flag location. Flag can be positioned at any of the 7 points around the border.
<a href="#">AlarmIndicatorSetFrameColor</a>	Sets the color of the frame for the Alarm Indicator in RGB hex or decimal values.
<a href="#">AlarmIndicatorSetInclude</a>	Sets the status of the "Include Equipment reference count" option for the Alarm indicator. When set to true the count will include equipment references when determining the highest priority alarm for an alarm indicator.

For details and a VB example on handling return and error values, see [Error Handling](#).

## See Also

[Automation Events](#)

## Function Categories

### AlarmIndicatorGetBorderIsInside

Retrieves the setting for the 'Is Inside Flag' for the alarm indicator.

## Syntax

### AlarmIndicatorGetBorderIsInside

## Return Value

True or False.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AlarmIndicatorGetBorderPadding](#), [AlarmIndicatorGetBorderWidth](#), [AlarmIndicatorGetEnabled](#),  
[AlarmIndicatorGetEquipHierarchy](#), [AlarmIndicatorGetExpr](#), [AlarmIndicatorGetFlagEnabled](#),  
[AlarmIndicatorGetFlagLocation](#), [AlarmIndicatorGetFrameColor](#), [AlarmIndicatorGetInclude](#),  
[AlarmIndicatorSetBorderIsInside](#), [AlarmIndicatorSetBorderPadding](#), [AlarmIndicatorSetBorderWidth](#),  
[AlarmIndicatorSetEnabled](#), [AlarmIndicatorSetEquipHierarchy](#), [AlarmIndicatorSetExpr](#),  
[AlarmIndicatorSetFlagEnabled](#), [AlarmIndicatorSetFlagLocation](#), [AlarmIndicatorSetFrameColor](#),  
[AlarmIndicatorSetInclude](#)

## Example

```
bBorderIsInsideGet = gb2.AlarmIndicatorGetBorderIsInside;
```

### AlarmIndicatorGetBorderPadding

Retrieves the size of the border padding for the alarm indicator.

## Syntax

### AlarmIndicatorGetBorderPadding

## Return Value

0 to 100 px

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AlarmIndicatorGetBorderIsInside](#), [AlarmIndicatorGetBorderWidth](#), [AlarmIndicatorGetEnabled](#),

[AlarmIndicatorGetEquipHierarchy](#), [AlarmIndicatorGetExpr](#), [AlarmIndicatorGetFlagEnabled](#),  
[AlarmIndicatorGetFlagLocation](#), [AlarmIndicatorGetFrameColor](#), [AlarmIndicatorGetInclude](#),  
[AlarmIndicatorSetBorderIsInside](#), [AlarmIndicatorSetBorderPadding](#), [AlarmIndicatorSetBorderWidth](#),  
[AlarmIndicatorSetEnabled](#), [AlarmIndicatorSetEquipHierarchy](#), [AlarmIndicatorSetExpr](#),  
[AlarmIndicatorSetFlagEnabled](#), [AlarmIndicatorSetFlagLocation](#), [AlarmIndicatorSetFrameColor](#),  
[AlarmIndicatorSetInclude](#)

## Example

```
iBorderPaddingGet = gb2.AlarmIndicatorGetBorderPadding;
```

### AlarmIndicatorGetWidth

Retrieves the width of the Alarm Indicator border.

## Syntax

### AlarmIndicatorGetWidth

## Return Value

1-20 px

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AlarmIndicatorGetBorderIsInside](#), [AlarmIndicatorGetBorderPadding](#), [AlarmIndicatorGetEnabled](#),  
[AlarmIndicatorGetEquipHierarchy](#), [AlarmIndicatorGetExpr](#), [AlarmIndicatorGetFlagEnabled](#),  
[AlarmIndicatorGetFlagLocation](#), [AlarmIndicatorGetFrameColor](#), [AlarmIndicatorGetInclude](#),  
[AlarmIndicatorSetBorderIsInside](#), [AlarmIndicatorSetBorderPadding](#), [AlarmIndicatorSetBorderWidth](#),  
[AlarmIndicatorSetEnabled](#), [AlarmIndicatorSetEquipHierarchy](#), [AlarmIndicatorSetExpr](#),  
[AlarmIndicatorSetFlagEnabled](#), [AlarmIndicatorSetFlagLocation](#), [AlarmIndicatorSetFrameColor](#),  
[AlarmIndicatorSetInclude](#)

## Example

```
iBorderWidthGet = gb2.AlarmIndicatorGetWidth;
```

### AlarmIndicatorGetEnabled

Retrieves the state of the Alarm Indicator option for an object or genie.

## Syntax

### AlarmIndicatorGetEnabled

## Return Value

True or False

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AlarmIndicatorGetBorderIsInside](#), [AlarmIndicatorGetBorderPadding](#), [AlarmIndicatorGetBorderWidth](#),  
[AlarmIndicatorGetEquipHierarchy](#), [AlarmIndicatorGetExpr](#), [AlarmIndicatorGetFlagEnabled](#),  
[AlarmIndicatorGetFlagLocation](#), [AlarmIndicatorGetFrameColor](#), [AlarmIndicatorGetInclude](#),  
[AlarmIndicatorSetBorderIsInside](#), [AlarmIndicatorSetBorderPadding](#), [AlarmIndicatorSetBorderWidth](#),  
[AlarmIndicatorSetEnabled](#), [AlarmIndicatorSetEquipHierarchy](#), [AlarmIndicatorSetExpr](#),  
[AlarmIndicatorSetFlagEnabled](#), [AlarmIndicatorSetFlagLocation](#), [AlarmIndicatorSetFrameColor](#),  
[AlarmIndicatorSetInclude](#)

## Example

```
bEnableGet = gb2.AlarmIndicatorGetEnabled;
```

### AlarmIndicatorGetEquipHierarchy

Retrieves the location in the equipment hierarchy that will be used to determine the highest priority alarm for the alarm indicator.

## Syntax

**AlarmIndicatorGetEquipHierarchy**

## Return Value

0, 1 or 2

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AlarmIndicatorGetBorderIsInside](#), [AlarmIndicatorGetBorderPadding](#), [AlarmIndicatorGetBorderWidth](#),  
[AlarmIndicatorGetEnabled](#), [AlarmIndicatorGetExpr](#), [AlarmIndicatorGetFlagEnabled](#),  
[AlarmIndicatorGetFlagLocation](#), [AlarmIndicatorGetFrameColor](#), [AlarmIndicatorGetInclude](#),  
[AlarmIndicatorSetBorderIsInside](#), [AlarmIndicatorSetBorderPadding](#), [AlarmIndicatorSetBorderWidth](#),  
[AlarmIndicatorSetEnabled](#), [AlarmIndicatorSetEquipHierarchy](#), [AlarmIndicatorSetExpr](#),  
[AlarmIndicatorSetFlagEnabled](#), [AlarmIndicatorSetFlagLocation](#), [AlarmIndicatorSetFrameColor](#),  
[AlarmIndicatorSetInclude](#)

## Example

```
iEquipmentHierarchyGet = gb2.AlarmIndicatorGetEquipHierarchy;
```

### AlarmIndicatorGetExpr

Retrieves the equipment string or expression belonging to the Alarm Indicator.

## Syntax

```
AlarmIndicatorGetExpr
```

## Return Value

Equipment string or expression

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AlarmIndicatorGetBorderIsInside](#), [AlarmIndicatorGetBorderPadding](#), [AlarmIndicatorGetBorderWidth](#),  
[AlarmIndicatorGetEnabled](#), [AlarmIndicatorGetEquipHierarchy](#), [AlarmIndicatorGetFlagEnabled](#),  
[AlarmIndicatorGetFlagLocation](#), [AlarmIndicatorGetFrameColor](#), [AlarmIndicatorGetInclude](#),  
[AlarmIndicatorSetBorderIsInside](#), [AlarmIndicatorSetBorderPadding](#), [AlarmIndicatorSetBorderWidth](#),  
[AlarmIndicatorSetEnabled](#), [AlarmIndicatorSetEquipHierarchy](#), [AlarmIndicatorSetExpr](#),  
[AlarmIndicatorSetFlagEnabled](#), [AlarmIndicatorSetFlagLocation](#), [AlarmIndicatorSetFrameColor](#),  
[AlarmIndicatorSetInclude](#)

## Example

```
sExprGet = gb2.AlarmIndicatorGetExpr;
```

### AlarmIndicatorGetFlagEnabled

Retrieves the status of the Alarm Indicator flag.

## Syntax

```
AlarmIndicatorGetFlagEnabled
```

## Return Value

True or False

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AlarmIndicatorGetBorderIsInside](#), [AlarmIndicatorGetBorderPadding](#), [AlarmIndicatorGetBorderWidth](#),  
[AlarmIndicatorGetEnabled](#), [AlarmIndicatorGetEquipHierarchy](#), [AlarmIndicatorGetExpr](#),  
[AlarmIndicatorGetFlagLocation](#), [AlarmIndicatorGetFrameColor](#), [AlarmIndicatorGetInclude](#),  
[AlarmIndicatorSetBorderIsInside](#), [AlarmIndicatorSetBorderPadding](#), [AlarmIndicatorSetBorderWidth](#),  
[AlarmIndicatorSetEnabled](#), [AlarmIndicatorSetEquipHierarchy](#), [AlarmIndicatorSetExpr](#),  
[AlarmIndicatorSetFlagEnabled](#), [AlarmIndicatorSetFlagLocation](#), [AlarmIndicatorSetFrameColor](#),  
[AlarmIndicatorSetInclude](#)

## Example

```
bFlagEnabledGet = gb2.AlarmIndicatorGetFlagEnabled;
```

### AlarmIndicatorGetFlagLocation

Retrieves the position of the flag for the Alarm Indicator.

## Syntax

**AlarmIndicatorGetFlagLocation**

## Return Value

0 to 7

The position is outlined below:

- 0 - Top Left
- 1 - Center Left
- 2 - Bottom Left
- 3 - Top Center
- 4 - Bottom Center
- 5 - Top Right
- 6 - Middle Right
- 7 - Bottom Right

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AlarmIndicatorGetBorderIsInside](#), [AlarmIndicatorGetBorderPadding](#), [AlarmIndicatorGetBorderWidth](#),  
[AlarmIndicatorGetEnabled](#), [AlarmIndicatorGetEquipHierarchy](#), [AlarmIndicatorGetExpr](#),  
[AlarmIndicatorGetFlagEnabled](#), [AlarmIndicatorGetFrameColor](#), [AlarmIndicatorGetInclude](#),  
[AlarmIndicatorSetBorderIsInside](#), [AlarmIndicatorSetBorderPadding](#), [AlarmIndicatorSetBorderWidth](#),  
[AlarmIndicatorSetEnabled](#), [AlarmIndicatorSetEquipHierarchy](#), [AlarmIndicatorSetExpr](#),

[AlarmIndicatorSetFlagEnabled](#), [AlarmIndicatorSetFlagLocation](#), [AlarmIndicatorSetFrameColor](#),  
[AlarmIndicatorSetInclude](#)

## Example

```
iFlagLocationGet= gb2.AlarmIndicatorGetFlagLocation;
```

### AlarmIndicatorGetFrameColor

Retrieves the color of the frame of the Alarm indicator.

## Syntax

**AlarmIndicatorGetFrameColor**

## Return Value

RGB value of the frame color or -1 for transparency

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AlarmIndicatorGetBorderIsInside](#), [AlarmIndicatorGetBorderPadding](#), [AlarmIndicatorGetBorderWidth](#),  
[AlarmIndicatorGetEnabled](#), [AlarmIndicatorGetEquipHierarchy](#), [AlarmIndicatorGetExpr](#),  
[AlarmIndicatorGetFlagEnabled](#), [AlarmIndicatorGetFlagLocation](#), [AlarmIndicatorGetInclude](#),  
[AlarmIndicatorSetBorderIsInside](#), [AlarmIndicatorSetBorderPadding](#), [AlarmIndicatorSetBorderWidth](#),  
[AlarmIndicatorSetEnabled](#), [AlarmIndicatorSetEquipHierarchy](#), [AlarmIndicatorSetExpr](#),  
[AlarmIndicatorSetFlagEnabled](#), [AlarmIndicatorSetFlagLocation](#), [AlarmIndicatorSetFrameColor](#),  
[AlarmIndicatorSetInclude](#)

## Example

```
iFrameColorGet = gb2.AlarmIndicatorGetFrameColour;
```

### AlarmIndicatorGetInclude

Retrieves the status of the Include Equipment reference option for the Alarm indicator.

## Syntax

**AlarmIndicatorGetInclude**

## Return Value

True or False

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AlarmIndicatorGetBorderIsInside](#), [AlarmIndicatorGetBorderPadding](#), [AlarmIndicatorGetBorderWidth](#),  
[AlarmIndicatorGetEnabled](#), [AlarmIndicatorGetEquipHierarchy](#), [AlarmIndicatorGetExpr](#),  
[AlarmIndicatorGetFlagEnabled](#), [AlarmIndicatorGetFlagLocation](#), [AlarmIndicatorGetFrameColor](#),  
[AlarmIndicatorSetBorderIsInside](#), [AlarmIndicatorSetBorderPadding](#), [AlarmIndicatorSetBorderWidth](#),  
[AlarmIndicatorSetEnabled](#), [AlarmIndicatorSetEquipHierarchy](#), [AlarmIndicatorSetExpr](#),  
[AlarmIndicatorSetFlagEnabled](#), [AlarmIndicatorSetFlagLocation](#), [AlarmIndicatorSetFrameColor](#),  
[AlarmIndicatorSetInclude](#)

## Example

```
bIncludeGet = gb2.AlarmIndicatorGetInclude;
```

### AlarmIndicatorSetBorderIsInside

Sets the Alarm Indicator 'Is Inside flag'.

## Syntax

**AlarmIndicatorSetBorderIsInside(IndicatorIsInside)**

*IndicatorIsInside:*

True or false. If set to True the border is located within the extent of the object group or Genie. This means no additional space will be required to support a border. If false there will be no inside border.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AlarmIndicatorGetBorderIsInside](#), [AlarmIndicatorGetBorderPadding](#), [AlarmIndicatorGetBorderWidth](#),  
[AlarmIndicatorGetEnabled](#), [AlarmIndicatorGetEquipHierarchy](#), [AlarmIndicatorGetExpr](#),  
[AlarmIndicatorGetFlagEnabled](#), [AlarmIndicatorGetFlagLocation](#), [AlarmIndicatorGetFrameColor](#),  
[AlarmIndicatorGetInclude](#), [AlarmIndicatorSetBorderPadding](#), [AlarmIndicatorSetBorderWidth](#),  
[AlarmIndicatorSetEnabled](#), [AlarmIndicatorSetEquipHierarchy](#), [AlarmIndicatorSetExpr](#),  
[AlarmIndicatorSetFlagEnabled](#), [AlarmIndicatorSetFlagLocation](#), [AlarmIndicatorSetFrameColor](#),  
[AlarmIndicatorSetInclude](#)

## Example

```
gb2.AlarmIndicatorSetEnabled = true;
gb2.AlarmIndicatorSetExpr = "C1.Motor1";
gb2.AlarmIndicatorSetEquipHierarchy = 0;
gb2.AlarmIndicatorSetBorderIsInside = false;
gb2.AlarmIndicatorSetBorderWidth = 2;
gb2.AlarmIndicatorSetFrameColour = 0x80AAD5;
gb2.AlarmIndicatorSetBorderPadding = 5;
```

```
gb2.AlarmIndicatorSetInclude = true;  
gb2.AlarmIndicatorSetFlagEnabled = false;  
gb2.AlarmIndicatorSetFlagLocation = 0;
```

## AlarmIndicatorSetBorderPadding

Set the size of the border padding for the Alarm Indicator.

## Syntax

**AlarmIndicatorSetBorderPadding(*BorderPadding*)**

*BorderPadding*:

Amount of space between the extent of the object group or Genie and the inside edge of the alarm border (in pixels). Set the value to be between 1 and 100 px.

## Return Value

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AlarmIndicatorGetBorderIsInside](#), [AlarmIndicatorGetBorderPadding](#), [AlarmIndicatorGetBorderWidth](#),  
[AlarmIndicatorGetEnabled](#), [AlarmIndicatorGetEquipHierarchy](#), [AlarmIndicatorGetExpr](#),  
[AlarmIndicatorGetFlagEnabled](#), [AlarmIndicatorGetFlagLocation](#), [AlarmIndicatorGetFrameColor](#),  
[AlarmIndicatorGetInclude](#), [AlarmIndicatorSetBorderIsInside](#), [AlarmIndicatorSetBorderWidth](#),  
[AlarmIndicatorSetEnabled](#), [AlarmIndicatorSetEquipHierarchy](#), [AlarmIndicatorSetExpr](#),  
[AlarmIndicatorSetFlagEnabled](#), [AlarmIndicatorSetFlagLocation](#), [AlarmIndicatorSetFrameColor](#),  
[AlarmIndicatorSetInclude](#)

## Example

```
gb2.AlarmIndicatorSetEnabled = true;  
gb2.AlarmIndicatorSetExpr = "C1.Motor1";  
gb2.AlarmIndicatorSetEquipHierarchy = 0;  
gb2.AlarmIndicatorSetBorderIsInside = false;  
gb2.AlarmIndicatorSetBorderWidth = 2;  
gb2.AlarmIndicatorSetFrameColour = 0x80AAD5;  
gb2.AlarmIndicatorSetBorderPadding = 5;  
gb2.AlarmIndicatorSetInclude = true;  
gb2.AlarmIndicatorSetFlagEnabled = false;  
gb2.AlarmIndicatorSetFlagLocation = 0;
```

## AlarmIndicatorSetBorderWidth

Set the size of the Alarm Indicator border .

## Syntax

**AlarmIndicatorSetBorderWidth(*BorderPadding*)**

*BorderPadding*:

The width of the alarm border (in pixels) between 1 to 20 px.

## Return Value

1 - 20

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AlarmIndicatorGetBorderIsInside](#), [AlarmIndicatorGetBorderPadding](#), [AlarmIndicatorGetBorderWidth](#),  
[AlarmIndicatorGetEnabled](#), [AlarmIndicatorGetEquipHierarchy](#), [AlarmIndicatorGetExpr](#),  
[AlarmIndicatorGetFlagEnabled](#), [AlarmIndicatorGetFlagLocation](#), [AlarmIndicatorGetFrameColor](#),  
[AlarmIndicatorGetInclude](#), [AlarmIndicatorSetBorderIsInside](#), [AlarmIndicatorSetBorderPadding](#),  
[AlarmIndicatorSetEnabled](#), [AlarmIndicatorSetEquipHierarchy](#), [AlarmIndicatorSetExpr](#),  
[AlarmIndicatorSetFlagEnabled](#), [AlarmIndicatorSetFlagLocation](#), [AlarmIndicatorSetFrameColor](#),  
[AlarmIndicatorSetInclude](#)

## Example

```
gb2.AlarmIndicatorSetEnabled = true;
gb2.AlarmIndicatorSetExpr = "C1.Motor1";
gb2.AlarmIndicatorSetEquipHierarchy = 0;
gb2.AlarmIndicatorSetBorderIsInside = false;
gb2.AlarmIndicatorSetBorderWidth = 2;
gb2.AlarmIndicatorSetFrameColour = 0x80AAD5;
gb2.AlarmIndicatorSetBorderPadding = 5;
gb2.AlarmIndicatorSetInclude = true;
gb2.AlarmIndicatorSetFlagEnabled = false;
gb2.AlarmIndicatorSetFlagLocation = 0;
```

## AlarmIndicatorSetEnabled

Sets the status of the Alarm Indicator option for an object or genie.

## Syntax

**AlarmIndicatorSetEnabled(*IndicatorEnabled*)**

*IndicatorEnabled*:

True or False. When set to True an alarm indicator will be applied to an object or genie.

## Return Value

True or False

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AlarmIndicatorGetBorderIsInside](#), [AlarmIndicatorGetBorderPadding](#), [AlarmIndicatorGetBorderWidth](#),  
[AlarmIndicatorGetEnabled](#), [AlarmIndicatorGetEquipHierarchy](#), [AlarmIndicatorGetExpr](#),  
[AlarmIndicatorGetFlagEnabled](#), [AlarmIndicatorGetFlagLocation](#), [AlarmIndicatorGetFrameColor](#),  
[AlarmIndicatorGetInclude](#), [AlarmIndicatorSetBorderIsInside](#), [AlarmIndicatorSetBorderPadding](#),  
[AlarmIndicatorSetBorderWidth](#), [AlarmIndicatorSetEquipHierarchy](#), [AlarmIndicatorSetExpr](#),  
[AlarmIndicatorSetFlagEnabled](#), [AlarmIndicatorSetFlagLocation](#), [AlarmIndicatorSetFrameColor](#),  
[AlarmIndicatorSetInclude](#)

## Example

```
gb2.AlarmIndicatorSetEnabled = true;
gb2.AlarmIndicatorSetExpr = "C1.Motor1";
gb2.AlarmIndicatorSetEquipHierarchy = 0;
gb2.AlarmIndicatorSetBorderIsInside = false;
gb2.AlarmIndicatorSetBorderWidth = 2;
gb2.AlarmIndicatorSetFrameColour = 0x80AAD5;
gb2.AlarmIndicatorSetBorderPadding = 5;
gb2.AlarmIndicatorSetInclude = true;
gb2.AlarmIndicatorSetFlagEnabled = false;
gb2.AlarmIndicatorSetFlagLocation = 0;
```

## AlarmIndicatorSetEquipHierarchy

Set the location in the equipment hierarchy that will be used to determine the highest priority alarm for the alarm indicator.

## Syntax

**AlarmIndicatorSetEquipHierarchy(*EquipmentHierarchy*)**

*EquipmentHierarchy*:

Locations you can set include:

0 = Equipment and Children

1= Equipment Only

2= Children Only.

## Return Value

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AlarmIndicatorGetBorderIsInside](#), [AlarmIndicatorGetBorderPadding](#), [AlarmIndicatorGetBorderWidth](#),  
[AlarmIndicatorGetEnabled](#), [AlarmIndicatorGetEquipHierarchy](#), [AlarmIndicatorGetExpr](#),  
[AlarmIndicatorGetFlagEnabled](#), [AlarmIndicatorGetFlagLocation](#), [AlarmIndicatorGetFrameColor](#),  
[AlarmIndicatorGetInclude](#), [AlarmIndicatorSetBorderIsInside](#), [AlarmIndicatorSetBorderPadding](#),  
[AlarmIndicatorSetBorderWidth](#), [AlarmIndicatorSetEnabled](#), [AlarmIndicatorSetExpr](#),  
[AlarmIndicatorSetFlagEnabled](#), [AlarmIndicatorSetFlagLocation](#), [AlarmIndicatorSetFrameColor](#),  
[AlarmIndicatorSetInclude](#)

## Example

```
gb2.AlarmIndicatorSetEnabled = true;
gb2.AlarmIndicatorSetExpr = "C1.Motor1";
gb2.AlarmIndicatorSetEquipHierarchy = 0;
gb2.AlarmIndicatorSetBorderIsInside = false;
gb2.AlarmIndicatorSetBorderWidth = 2;
gb2.AlarmIndicatorSetFrameColour = 0x80AAD5;
gb2.AlarmIndicatorSetBorderPadding = 5;
gb2.AlarmIndicatorSetInclude = true;
gb2.AlarmIndicatorSetFlagEnabled = false;
gb2.AlarmIndicatorSetFlagLocation = 0;
```

## AlarmIndicatorSetExpr

Sets the equipment string or expression that will be used to determine the highest priority alarm for the alarm indicator.

## Syntax

**AlarmIndicatorSetExpr(*Expression*)**

*Expression*

Enter the Equipment string that will be used to determine the highest priority alarm for the alarm indicator.

## Return Value

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AlarmIndicatorGetBorderIsInside](#), [AlarmIndicatorGetBorderPadding](#), [AlarmIndicatorGetBorderWidth](#),  
[AlarmIndicatorGetEnabled](#), [AlarmIndicatorGetEquipHierarchy](#), [AlarmIndicatorGetExpr](#),  
[AlarmIndicatorGetFlagEnabled](#), [AlarmIndicatorGetFlagLocation](#), [AlarmIndicatorGetFrameColor](#),  
[AlarmIndicatorGetInclude](#), [AlarmIndicatorSetBorderIsInside](#), [AlarmIndicatorSetBorderPadding](#),  
[AlarmIndicatorSetBorderWidth](#), [AlarmIndicatorSetEnabled](#), [AlarmIndicatorSetEquipHierarchy](#),  
[AlarmIndicatorSetFlagEnabled](#), [AlarmIndicatorSetFlagLocation](#), [AlarmIndicatorSetFrameColor](#),  
[AlarmIndicatorSetInclude](#)

## Example

```
gb2.AlarmIndicatorSetEnabled = true;
gb2.AlarmIndicatorSetExpr = "C1.Motor1";
gb2.AlarmIndicatorSetEquipHierarchy = 0;
gb2.AlarmIndicatorSetBorderIsInside = false;
gb2.AlarmIndicatorSetBorderWidth = 2;
gb2.AlarmIndicatorSetFrameColour = 0x80AAD5;
gb2.AlarmIndicatorSetBorderPadding = 5;
gb2.AlarmIndicatorSetInclude = true;
gb2.AlarmIndicatorSetFlagEnabled = false;
gb2.AlarmIndicatorSetFlagLocation = 0;
```

## AlarmIndicatorSetFlagEnabled

Sets the Alarm Indicator flag.

## Syntax

**AlarmIndicatorSetFlagEnabled(*FlagEnabled*)**

*FlagEnabled*:

When set to True a flag is applied to the alarm indicator. When set to false no flag is applied.

## Return Value

True or False

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AlarmIndicatorGetBorderIsInside](#), [AlarmIndicatorGetBorderPadding](#), [AlarmIndicatorGetBorderWidth](#),  
[AlarmIndicatorGetEnabled](#), [AlarmIndicatorGetEquipHierarchy](#), [AlarmIndicatorGetExpr](#),  
[AlarmIndicatorGetFlagEnabled](#), [AlarmIndicatorGetFlagLocation](#), [AlarmIndicatorGetFrameColor](#),  
[AlarmIndicatorGetInclude](#), [AlarmIndicatorSetBorderIsInside](#), [AlarmIndicatorSetBorderPadding](#),  
[AlarmIndicatorSetBorderWidth](#), [AlarmIndicatorSetEnabled](#), [AlarmIndicatorSetEquipHierarchy](#),  
[AlarmIndicatorSetExpr](#), [AlarmIndicatorSetFlagLocation](#), [AlarmIndicatorSetFrameColor](#), [AlarmIndicatorSetInclude](#)

## Example

```
gb2.AlarmIndicatorSetEnabled = true;
gb2.AlarmIndicatorSetExpr = "C1.Motor1";
gb2.AlarmIndicatorSetEquipHierarchy = 0;
gb2.AlarmIndicatorSetBorderIsInside = false;
gb2.AlarmIndicatorSetBorderWidth = 2;
gb2.AlarmIndicatorSetFrameColour = 0x80AAD5;
gb2.AlarmIndicatorSetBorderPadding = 5;
gb2.AlarmIndicatorSetInclude = true;
gb2.AlarmIndicatorSetFlagEnabled = true;
```

```
gb2.AlarmIndicatorSetFlagLocation = 3;
```

## AlarmIndicatorSetFlagLocation

Sets the position of the Alarm Indicator Flag.

## Syntax

**AlarmIndicatorSetFlagLocation(*FlagLocation*)**

*FlagLocation*:

Flag can be positioned at the following points around the border:

- 0 - Top Left
- 1 - Center Left
- 2 - Bottom Left
- 3 - Top Center
- 4 - Bottom Center
- 5 - Top Right
- 6 - Center Right
- 7 - Bottom Right

## Return Value

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AlarmIndicatorGetBorderIsInside](#), [AlarmIndicatorGetBorderPadding](#), [AlarmIndicatorGetBorderWidth](#),  
[AlarmIndicatorGetEnabled](#), [AlarmIndicatorGetEquipHierarchy](#), [AlarmIndicatorGetExpr](#),  
[AlarmIndicatorGetFlagEnabled](#), [AlarmIndicatorGetFlagLocation](#), [AlarmIndicatorGetFrameColor](#),  
[AlarmIndicatorGetInclude](#), [AlarmIndicatorSetBorderIsInside](#), [AlarmIndicatorSetBorderPadding](#),  
[AlarmIndicatorSetBorderWidth](#), [AlarmIndicatorSetEnabled](#), [AlarmIndicatorSetEquipHierarchy](#),  
[AlarmIndicatorSetExpr](#), [AlarmIndicatorSetFlagEnabled](#), [AlarmIndicatorSetFrameColor](#), [AlarmIndicatorSetInclude](#)

## Example

```
gb2.AlarmIndicatorSetEnabled = true; gb2.AlarmIndicatorSetExpr = "C1.Motor1";
gb2.AlarmIndicatorSetEquipHierarchy = 0; gb2.AlarmIndicatorSetBorderIsInside = false;
gb2.AlarmIndicatorSetBorderWidth = 2; gb2.AlarmIndicatorSetFrameColour = 0x80AAD5;
gb2.AlarmIndicatorSetBorderPadding = 5; gb2.AlarmIndicatorSetInclude = true;
gb2.AlarmIndicatorSetFlagEnabled = true; gb2.AlarmIndicatorSetFlagLocation = 3;
```

## AlarmIndicatorSetFrameColor

Sets the color of the frame for the Alarm Indicator.

---

**Note:** Colors are referenced by using hex codes for red, green and blue (RGB) values. You can view RGB values (in decimal) of a selected color by choosing **Tools|Edit Favorite Colors** in Graphics Builder. If, for example, the color you want to use has the values R = 128, G = 170, B = 213, you can convert these to their hex equivalents (R = 0x80, G = 0xAA, B = 0xD5).

---

## Syntax

**AlarmIndicatorSetFrameColor(*FrameColor*)**

*FrameColor*:

RGB value and -1 for transparency

## Return Value

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AlarmIndicatorGetBorderIsInside](#), [AlarmIndicatorGetBorderPadding](#), [AlarmIndicatorGetBorderWidth](#),  
[AlarmIndicatorGetEnabled](#), [AlarmIndicatorGetEquipHierarchy](#), [AlarmIndicatorGetExpr](#),  
[AlarmIndicatorGetFlagEnabled](#), [AlarmIndicatorGetFlagLocation](#), [AlarmIndicatorGetFrameColor](#),  
[AlarmIndicatorGetInclude](#), [AlarmIndicatorSetBorderIsInside](#), [AlarmIndicatorSetBorderPadding](#),  
[AlarmIndicatorSetBorderWidth](#), [AlarmIndicatorSetEnabled](#), [AlarmIndicatorSetEquipHierarchy](#),  
[AlarmIndicatorSetExpr](#), [AlarmIndicatorSetFlagEnabled](#), [AlarmIndicatorSetFlagLocation](#), [AlarmIndicatorSetInclude](#)

## Example

```
gb2.AlarmIndicatorSetEnabled = true;
gb2.AlarmIndicatorSetExpr = "C1.Motor1";
gb2.AlarmIndicatorSetEquipHierarchy = 0;
gb2.AlarmIndicatorSetBorderIsInside = false;
gb2.AlarmIndicatorSetBorderWidth = 2;
gb2.AlarmIndicatorSetFrameColour = 0x80AAD5;
gb2.AlarmIndicatorSetBorderPadding = 5;
gb2.AlarmIndicatorSetInclude = true;
gb2.AlarmIndicatorSetFlagEnabled = true;
gb2.AlarmIndicatorSetFlagLocation = 3;
```

## AlarmIndicatorSetInclude

Sets the status of the "Include Equipment references in count" option for the Alarm indicator.

## Syntax

**AlarmIndicatorSetInclude(*IndicatorInclude*)**

*IndicatorInclude*:

True or False. When set to true the count will include equipment references when determining the highest priority alarm for an alarm indicator.

## Return Value

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AlarmIndicatorGetBorderIsInside](#), [AlarmIndicatorGetBorderPadding](#), [AlarmIndicatorGetBorderWidth](#),  
[AlarmIndicatorGetEnabled](#), [AlarmIndicatorGetEquipHierarchy](#), [AlarmIndicatorGetExpr](#),  
[AlarmIndicatorGetFlagEnabled](#), [AlarmIndicatorGetFlagLocation](#), [AlarmIndicatorGetFrameColor](#),  
[AlarmIndicatorGetInclude](#), [AlarmIndicatorSetBorderIsInside](#), [AlarmIndicatorSetBorderPadding](#),  
[AlarmIndicatorSetBorderWidth](#), [AlarmIndicatorSetEnabled](#), [AlarmIndicatorSetEquipHierarchy](#),  
[AlarmIndicatorSetExpr](#), [AlarmIndicatorSetFlagEnabled](#), [AlarmIndicatorSetFlagLocation](#),  
[AlarmIndicatorSetFrameColor](#)

## Example

```
gb2.AlarmIndicatorSetEnabled = true;
gb2.AlarmIndicatorSetExpr = "C1.Motor1";
gb2.AlarmIndicatorSetEquipHierarchy = 0;
gb2.AlarmIndicatorSetBorderIsInside = false;
gb2.AlarmIndicatorSetBorderWidth = 2;
gb2.AlarmIndicatorSetFrameColour = 0x80AAD5;
gb2.AlarmIndicatorSetBorderPadding = 5;
gb2.AlarmIndicatorSetInclude = true;
gb2.AlarmIndicatorSetFlagEnabled = false;
gb2.AlarmIndicatorSetFlagLocation = 0;
```

## Arrange and Position Functions

The following functions modify the position of a selected object in three dimensions (X, Y and Z order).

<a href="#">PositionAt</a>	Positions the active object at the specified location.
<a href="#">PositionBringForwards</a>	Moves the last object addressed one step forward in the layering of objects on a page, creating the appearance of moving forward.
<a href="#">PositionBringToFront</a>	Positions the last object addressed as the closest layer on a graphics page, giving it the appearance of being in front of the other objects.
<a href="#">PositionMirrorHorizontal</a>	Turns the last object addressed into a mirror image of itself across a horizontal axis.
<a href="#">PositionMirrorVertical</a>	Turns the last object addressed into a mirror image of itself across a vertical axis.
<a href="#">PositionRotate</a>	Rotates the last object addressed by 90 degrees clockwise.

PositionSendBackwards	Moves the last object addressed one step backwards in the layering of objects on a page, creating the appearance of moving backwards.
PositionSendToBack	Positions the last object addressed as the lowest layer on a graphics page, giving it the appearance of being behind the other objects.

For details and a VB example on handling return and error values, see [Error Handling](#).

## See Also

[Automation Events](#)

[Function Categories](#)

## PositionAt

Positions the active object at the specified location. The destination coordinates is adjusted if OptionSnapToGrid or OptionSnapToGuidelines are set to TRUE.

## Syntax

**PositionAt(*XPosition*, *YPosition*)**

*XPosition*:

Absolute X position in pixels from the left side of the page.

*YPosition*:

Absolute Y position in pixels from the top of the page.

## Return Value

0 (zero) if successful; otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PositionRotate](#), [PositionMirrorVertical](#), [PositionMirrorHorizontal](#), [PositionSendToBack](#), [PositionBringToFront](#), [PositionBringForwards](#), [PositionSendBackwards](#)

## Example

```
GraphicsBuilder.LibraryObjectPlace "include", "agitator", "agit_1_Pos1_g", 2, True
GraphicsBuilder.PositionAt "200,200"
```

## PositionBringForwards

Moves the last object addressed one step forward in the layering of objects on a page, creating the appearance of moving forward.

## Syntax

**PositionBringForwards**

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PositionAt](#), [PositionRotate](#), [PositionSendToBack](#), [PositionBringToFront](#)

## Example

```
' Moves an object forward in the layering of objects on a graphics page
GraphicsBuilder.LibraryObjectPlace "include", "agitator", "agit_1_Pos1_g", 2, True
GraphicsBuilder.PositionAt 200, 200
GraphicsBuilder.PositionBringForwards
```

## PositionBringToFront

Positions the last object addressed as the closest layer on a graphics page, giving it the appearance of being in front of other objects.

## Syntax

**PositionBringToFront**

## Return Value

0 (zero) if successful; otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PositionAt](#), [PositionRotate](#), [PositionSendToBack](#), [PositionBringForwards](#), [PositionSendBackwards](#)

## Example

```
' Places an object in front of other objects on a graphics page
GraphicsBuilder.LibraryObjectPlace "include", "agitator", "agit_1_Pos1_g", 2, True
GraphicsBuilder.PositionAt 200, 200
GraphicsBuilder.PositionBringToFront
```

## PositionMirrorHorizontal

Turns the last object addressed into a mirror image of itself across a horizontal axis.

## Syntax

**PositionMirrorHorizontal**

## Return Value

0 (zero) if successful; otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PositionAt](#), [PositionRotate](#), [PositionMirrorVertical](#), [PositionSendToBack](#), [PositionBringToFront](#),  
[PositionBringForwards](#), [PositionSendBackwards](#)

## Example

```
' Mirrors an object across a horizontal access
GraphicsBuilder.LibraryObjectPlace "include", "agitator", "agit_1_Pos1_g", 2, True
GraphicsBuilder.PositionAt 200, 200
GraphicsBuilder.PositionMirrorHorizontal
```

## PositionMirrorVertical

Turns the last object addressed into a mirror image of itself across a vertical axis.

## Syntax

**PositionMirrorVertical**

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PositionAt](#), [PositionRotate](#), [PositionMirrorHorizontal](#), [PositionSendToBack](#), [PositionBringToFront](#),  
[PositionBringForwards](#), [PositionSendBackwards](#)

## Example

```
' Mirrors an object across a vertical accessGraphicsBuilder.LibraryObjectPlace "include",
"agitator", "agit_1_Pos1_g", 2, True
GraphicsBuilder.PositionAt 200, 200
GraphicsBuilder.PositionMirrorVertical
```

## PositionRotate

Rotates the last object addressed by 90 degrees clockwise.

## Syntax

**PositionRotate**

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PositionAt](#), [PositionMirrorVertical](#), [PositionMirrorHorizontal](#), [PositionSendToBack](#), [PositionBringToFront](#),  
[PositionBringForwards](#), [PositionSendBackwards](#)

## Example

```
' Rotates an object 90 degrees
GraphicsBuilder.LibraryObjectPlace "include", "agitator", "agit_1_Pos1_g", 2, True
GraphicsBuilder.PositionAt 200, 200
GraphicsBuilder.PositionRotate
```

## PositionSendBackwards

Moves the last object addressed one step backwards in the layering of objects on a page, creating the appearance of moving backwards.

## Syntax

**PositionSendBackwards**

## Return Value

0 (zero) if successful; otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PositionAt](#), [PositionMirrorVertical](#), [PositionMirrorHorizontal](#), [PositionSendToBack](#), [PositionBringToFront](#),  
[PositionBringForwards](#),

## Example

```
' Moves an object backwards in the layering of objects on a graphics page
GraphicsBuilder.LibraryObjectPlace "include", "agitator", "agit_1_Pos1_g", 2, True
GraphicsBuilder.PositionAt 200, 200
GraphicsBuilder.PositionSendBackwards
```

## PositionSendToBack

Positions the last object addressed as the lowest layer on a graphics page, giving it the appearance of being behind other objects.

## Syntax

**PositionSendToBack**

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PositionAt](#), [PositionMirrorHorizontal](#), [PositionSendBackwards](#), [PositionBringToFront](#), [PositionBringForwards](#),  
[PositionRotate](#)

## Example

```
' Places an object behind other objects on a graphics page
GraphicsBuilder.LibraryObjectPlace "include", "agitator", "agit_1_Pos1_g", 2, True
GraphicsBuilder.PositionAt 200, 200
GraphicsBuilder.PositionSendToBack
```

## Composite Genie Functions

Use Composite Genie functions to place or locate Composite Genies.

<a href="#">CompositeGeniePlace</a>	Places a Composite Genie at the specified co-ordinates on the active page.
<a href="#">PageSelectFirstCompositeGenie</a>	Selects the first Composite Genie on the active page.
<a href="#">PageSelectNextCompositeGenie</a>	Selects the next Composite Genie on the active page.

For details and a VB example on handling return and error values, see [Error Handling](#).

### See Also

[Automation Events](#)

[Function Categories](#)

### CompositeGeniePlace

Places a Composite Genie at the specified co-ordinates on the active page.

---

**Note:** This function does not support implicit value type casting.

---

### Syntax

**CompositeGeniePlace**(*templateID*, *KeyValuePair*, *nXPos*, *nYPos*)

*templateID*

Visual template ID of the Composite Genie template. This can be found in the root node attribute "Guid" of the Composite Genie's XML template.

*KeyValuePair*

This is a special key value pair that comes from the Graphics Builder. Note that this is different from the .NET key value pair.

*nXPos*

The horizontal offset position on the page where you want to place the Composite Genie.

*nYPos*

The vertical offset position on the page where you want to place the Composite Genie.

### HResult

The HResult is retrieved from the calling function's exception-handling mechanism. For example, if the calling function is .NET-based, a COM exception such as System.Runtime.InteropServices.COMException will be thrown if the HRESULT is not S\_OK.

The possible values for HResult include:

- S\_OK indicates success

- E\_FAIL indicates an error condition in the system. For example, the Composite Genie template may have a syntactical error, or if one of the Content Item Genies is missing.
- E\_INVALIDARG indicates that one of the arguments is incorrect. For example, one of the key names in the KeyValuePair does not exist or one of the value types does not match the parameter definition.
- E\_ABORT is returned if you attempt to place the Composite Genie at an invalid location on the page.

## Troubleshooting

If you encounter E\_FAIL, it is recommended that you manually paste the Composite Genie on the page.

If you encounter E\_INVALIDARG, check that the parameter names exist and parameter values match the type described in the visual template.

## Related Functions

[PageSelectFirstCompositeGenie](#), [PageSelectNextCompositeGenie](#)

### PageSelectFirstCompositeGenie

Selects the first Composite Genie on the active page.

## Syntax

**PageSelectFirstCompositeGenie()**

## HResult

The HResult is retrieved from the calling function's exception-handling mechanism. For example, if the calling function is .NET-based, a COM exception such as System.Runtime.InteropServices.COMException will be thrown if the HRESULT is not S\_OK.

The possible values for HResult include:

- E\_ABORT is returned when no more Composite Genies are found.

## Related Functions

[PageSelectNextCompositeGenie](#), [CompositeGeniePlace](#)

### PageSelectNextCompositeGenie

Selects the next Composite Genie on the active page.

## Syntax

**PageSelectNextCompositeGenie()**

## HResult

The HResult is retrieved from the calling function's exception-handling mechanism. For example, if the calling function is .NET-based, a COM exception such as System.Runtime.InteropServices.COMException will be thrown if the HRESULT is not S\_OK.

The possible values for HResult include:

- E\_ABORT is returned when no more Composite Genies are found.

## Related Functions

[PageSelectFirstCompositeGenie](#), [CompositeGeniePlace](#)

## Dynamic Properties Functions

With these functions, you can modify the dynamic properties of the graphics objects in your project (movement, scaling, rotation, sliders, dynamic color fill).

The error E\_HANDLE is returned if there is no selected or active object, or if an object does not support this type of property. E\_INVALIDARG is returned if an argument is out of range.

<a href="#">PropertiesAccessDisableGet</a>	Reads the current values set on the Access   Disable tab of the Object Properties dialog.
<a href="#">PropertiesAccessDisablePut</a>	Sets the values on the Access   Disable tab of the Object Properties dialog.
<a href="#">PropertiesAccessGeneralGet</a>	Reads the values on the Access   General tab of the Object Properties dialog.
<a href="#">PropertiesAccessGeneralPut</a>	Sets the values on the Access   General tab of the Object Properties dialog.
<a href="#">PropertiesAttributeNameGetEx</a>	Retrieves the Name of a graphics object on the current page.
<a href="#">PropertiesAttributeNamePutEx</a>	Sets the Name of a graphics object on the current page.
<a href="#">PropertiesButtonGet</a>	Reads the values for a button object from the Appearance   General tab of the Object Properties dialog.
<a href="#">PropertiesButtonGetEx</a>	Reads the values, including custom fill color for a button object from the Appearance   General tab of the Object Properties dialog.
<a href="#">PropertiesButtonPut</a>	Sets the values on the Appearance   General tab of the Object Properties dialog for a button object.

<a href="#">PropertiesCicodeObjectGet</a>	Reads the values set for a Cicode object on the Cicode   General tab of the Object Properties dialog.
<a href="#">PropertiesCicodeObjectPut</a>	Sets the values for a Cicode object on the Cicode   General tab of the Object Properties dialog.
<a href="#">PropertiesDisplayValueGet</a>	Reads the type and expressions configured on the Appearance   Display Value tab of the Object Properties dialog.
<a href="#">PropertiesDisplayValuePut</a>	Sets the values and expressions on the Appearance   Display Value tab of the Object Properties dialog.
<a href="#">PropertiesDisplayValueTextGet</a>	Reads the text for a specific index from the Appearance   Display Value tab of the Object Properties dialog.
<a href="#">PropertiesDisplayValueTextPut</a>	Sets the text for a specific index on the Appearance   Display Value tab of the Object Properties dialog.
<a href="#">PropertiesFillColourColourGet</a>	Reads the current color value set for the specified index point on the Fill   Color tab of the Object Properties dialog. This function has been superseded by the function <a href="#">PropertiesFillColourColourGetEx</a> .
<a href="#">PropertiesFillColourColourGetEx</a>	Reads the current color value set for the specified index point on the Fill   Color tab of the Object Properties dialog.
<a href="#">PropertiesFillColourColourPut</a>	Sets the color at the specific index on the Fill   Color tab of the Object Properties dialog. This function has been superseded by the function <a href="#">PropertiesFillColourColourPutEx</a> .
<a href="#">PropertiesFillColourColourPutEx</a>	Sets the color at the specific index on the Fill   Color tab of the Object Properties dialog.
<a href="#">PropertiesFillColourGet</a>	Reads the values set on the Fill   Color tab of the Object Properties dialog for the current object.
<a href="#">PropertiesFillColourPut</a>	Sets the values on the Fill   Color tab of the Object Properties dialog.
<a href="#">PropertiesFillLevelGet</a>	Reads the values set on the Fill   Level tab of the Object Properties dialog. This function has been superseded by the function <a href="#">PropertiesFillLevelGetEx</a> .
<a href="#">PropertiesFillLevelGetEx</a>	Reads the values set on the Fill   Level tab of the Object Properties dialog.
<a href="#">PropertiesFillLevelPut</a>	Sets the values on the Fill   Level tab of the Object

	Properties dialog. This function has been superseded by the function PropertiesFillLevelPutEx.
PropertiesFillLevelPutEx	Sets the values on the Fill   Level tab of the Object Properties dialog.
PropertiesInputKeyboardGet	Reads the values set on the Input   Keyboard Command tab of the Object Properties dialog
PropertiesInputKeyboardPut	Sets the values on the Input   Keyboard Commands tab of the Object Properties dialog
PropertiesInputTouchGet	Reads the values set on the Input   Touch tab of the Object Properties dialog
PropertiesInputTouchPut	Sets the values on the Input   Touch tab of the Object Properties dialog.
PropertiesShowDialog	Shows the property dialog for an object or a form for Genies.
PropertiesSymbolSetGet	Reads the type and expressions configured on the Appearance   General tab of the Object Properties dialog.
PropertiesSymbolSetPut	Sets the type defined for a symbol set on the Appearance   General tab of the Object Properties dialog.
PropertiesSymbolSetSymbolGet	Retrieves the Element name and Library name of the "Index" element of the currently selected object.
PropertiesSymbolSetSymbolPut	Sets the Element name and Library name of the "Index" element of the currently selected object.
PropertiesTransCentreOffsetExpressGet	Retrieve the express properties.
PropertiesTransCentreOffsetExpressPut	Set the express properties.
PropertiesTransformationGet	Reads the property values set on the Movement, Scaling and Slider tabs of the Object Properties dialog.
PropertiesTransformationPut	Sets values for the properties on the Movement, Scaling and Slider tabs of the Object Properties dialog.
PropertiesTrendGet	Reads the values for a trend object as set on the Appearance   General tab of the Object Properties dialog. This function has been superseded by the function PropertiesTrendGetEx.
PropertiesTrendGetEx	Reads the values for a trend object as set on the

	Appearance   General tab of the Object Properties dialog.
PropertiesTrendPut	Sets the values for a trend object that appear on the Appearance   General tab of the Object Properties dialog. This function has been superseded by the function PropertiesTrendPutEx.
PropertiesTrendPutEx	Sets the values for a trend object that appear on the Appearance   General tab of the Object Properties dialog
PropertyVisibility	Sets the Hidden when argument on the Appearance   Visibility tab of the Object Properties dialog.

For details and a VB example on handling return and error values, see [Error Handling](#).

## See Also

[Automation Events](#)

[Function Categories](#)

## PropertiesAccessDisableGet

Reads the current values set on the **Access | Disable** tab of the Object Properties dialog for the current object.

## Syntax

**PropertiesAccessDisableGet(Expression, DisableFlag, DisableStyle)**

*Expression:*

The string for the Disable when command.

*DisableFlag:*

TRUE if the object is configured to disable when an insufficient area or privilege setting is encountered.

*DisableStyle:*

The disable style setting:

0 = Embossed

1 = Grayed

2 = Hidden

## Return Value

The requested values, as a string.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PropertiesAccessDisablePut](#)

### PropertiesAccessDisablePut

Sets the values on the **Access | Disable** tab of the Object Properties dialog for the current object.

## Syntax

**PropertiesAccessDisablePut**(*Expression*, *DisableFlag*, *DisableStyle*)

*Expression*:

The string for the Disable when command.

*DisableFlag*:

TRUE if the object is configured to disable when an insufficient area or privilege setting is encountered.

*DisableStyle*:

The disable style setting:

0 = Embossed

1 = Grayed

2 = Hidden

## Return Value

0 (zero) if successful, otherwise an error is returned

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesAccessDisableGet](#)

### PropertiesAccessGeneralGet

Reads the values on the **Access | General** tab of the Object Properties dialog for the current object.

## Syntax

**PropertiesAccessGeneralGet**(STRING *pbstrName*, STRING *Description*, STRING *Tooltip*, INT *Area*, INT *Privilege*, STRING *LogDevice*)

*pbstrName*:

Name of the graphics object.

*Description*:

Description string for the object.

*Tooltip:*

Tooltip string for the object.

*Area:*

1 to 255 representing the current area setting, or 0 if the Same area as page check box is ticked.

*Privilege:*

1 to 255 representing the current privilege setting, or 0 if the No privilege restrictions check box is ticked.

*LogDevice:*

The name of the log device as a string.

## Return Value

The requested values, as a string.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesAccessGeneralPut](#)

### PropertiesAccessGeneralPut

Sets the values on the **Access | General** tab of the Object Properties dialog for the current object.

## Syntax

**PropertiesAccessGeneralPut**(STRING *pbstrName*, STRING *Description*, STRING *Tooltip*, INT *Area*, INT *Privilege*,  
STRING *LogDevice*)

*pbstrName:*

Name of the graphics object.

*Description:*

Description string for the object.

*Tooltip:*

Tooltip string for the object.

*Area:*

1 to 255 representing the current area setting, or 0 if the Same area as page check box is ticked.

*Privilege:*

1 to 255 representing the current privilege setting, or 0 if the No privilege restrictions check box is ticked.

*LogDevice:*

The name of the log device as a string.

## Return Value

0 (zero) if successful, otherwise an error is returned

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesAccessGeneralGet](#)

### PropertiesAttributeNameGetEx

Retrieves the name of a graphics object on the current page.

## Syntax

**PropertiesAttributeNameGetEx(INT nAN, STRING\* pbstrName)**

*nAN*:

Animation number of graphics object

*pbstrName*:

Name of graphics object

---

**Note:** Names need to be unique within the group. Names should not start with a number or contain the following chars <>:"^|?\*.

---

## Return Value

The requested values, as a string

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageOpen](#), [PageSave](#), [PageSaveAs](#), [PageClose](#)

### PropertiesAttributeNamePutEx

Sets the Name of a graphics object on the current page.

## Syntax

**PropertiesAttributeNamePutEx(INT nAN, STRING\* pbstrName)**

*nAN*:

Animation number of graphics object

*pbstrName*:

Name of graphics object

**Note:** Names need to be unique within the group. Names should not start with a number or contain the following chars <>:"^|?\*.

## Return Value

An error if the name cannot be set.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PageOpen](#), [PageSave](#), [PageSaveAs](#), [PageClose](#)

### PropertiesButtonGet

Reads the values for a button object from the **Appearance | General** tab of the Object Properties dialog.

## Syntax

**PropertiesButtonGet(ButtonType, Text, TextFont, Library, SymbolName)**

*ButtonType*:

Defines the button type:

0 = Text

1 = Border 3D Target

2 = Border Target

3 = Target

4 = Symbol

5 = XP Style button with text

6 = XP Style Button with Symbol

*Text*:

Button text. This argument is only valid for ButtonType = 0 and 5 (text).

*TextFont*:

The font use for the button text. This argument is only valid for ButtonType = 0 and 5 (text).

*Library*:

Library where the button symbol can be found. This argument is only valid for ButtonType = 4 and 6 (symbol).

*SymbolName*:

Name of the symbol to be displayed for a button. This argument is only valid for ButtonType = 4 and 6 (symbol).

## Return Value

The requested values, as a string.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesButtonPut](#)

### PropertiesButtonGetEx

Reads the values for a button object from the **Appearance | General** tab of the Object Properties dialog.

## Syntax

**PropertiesButtonGetEx**(*Type*, *Text*, *Font*, *Library*, *Element*, *CustomFillColor*, *FillColorUp*, *FillColorDown*)

*Type:*

Defines the button type:

0 = Text

1 = Border 3D Target

2 = Border Target

3 = Target

4 = Symbol

5 = XP Style button with text

6 = XP Style Button with Symbol

*Text:*

Button text. This argument is only valid for button Type = 0 and 5 (text).

*Font:*

The font use for the button text. This argument is only valid for button Type = 0 and 5 (text).

*Library:*

Library where the button symbol can be found. This argument is only valid for button Type = 4 and 6 (symbol).

*Element:*

Name of the symbol to be displayed for a button. This argument is only valid for button Type = 4 and 6 (symbol).

*CustomFillColor:*

TRUE if a custom fill color is defined for the button, otherwise FALSE.

*FillColorUp:*

Fill color (RBB value converted to decimal) defined for the button in the Up state.

*FillColorDown:*

Fill color (RBB value converted to decimal) defined for the button in the Down state.

## Return Value

The requested values, as a string.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesButtonPut](#)

### PropertiesButtonPut

Sets the values on the **Appearance | General** tab of the Object Properties dialog for a button object.

## Syntax

**PropertiesButtonPut**(*Type*, *Text*, *TextFont*, *Library*, *SymbolName*)

*ButtonType*:

Defines the button type:

0 = Text

1 = Border 3D Target

2 = Border Target

3 = Target

4 = Symbol

5 = XP Style button with text

6 = XP Style Button with Symbol

*Text*:

Button text. This argument is only valid for *ButtonType* = 0 and 5 (text).

*TextFont*:

The font use for the button text. This argument is only valid for *ButtonType* = 0 and 5 (text).

*Library*:

Library where the button symbol can be found. This argument is only valid for *ButtonType* = 4 and 6 (symbol).

*SymbolName*:

Name of the symbol to be displayed for a button. This argument is only valid for *ButtonType* = 4 and 6 (symbol).

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesButtonGet](#)

### PropertiesCicodeObjectGet

Reads the values set for a Cicode object on the **Cicode | General** tab of the Object Properties dialog.

## Syntax

**PropertiesCicodeObjectGet**(*Expression*, *Library*, *SymbolName*)

*Expression*:

The command expression.

*Library*:

Name of the library where the symbol used can be found.

*SymbolName*:

Name of the symbol used.

## Return Value

The requested values, as a string.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesCicodeObjectPut](#)

### PropertiesCicodeObjectPut

Sets the values for a Cicode object on the **Cicode | General** tab of the Object Properties dialog.

## Syntax

**PropertiesCicodeObjectPut**(*Expression*, *Library*, *SymbolName*)

*Expression*:

The command expression.

*Library*:

Name of the library where the symbol used can be found.

*SymbolName*:

Name of the symbol used.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesCicodeObjectGet](#)

## PropertiesDisplayValueGet

Reads the type and expressions configured on the **Appearance | Display Value** tab of the Object Properties dialog for a number or text object.

## Syntax

**PropertiesDisplayValueGet**(*SymbolSetType*, *ExpressionA*, *ExpressionB*, *ExpressionC*, *ExpressionD*, *ExpressionE*)

*SymbolSetType*:

Defines the symbol set type:

0 = On / Off

1 = Multi-state

2 = Array

3 = Numeric

4 = String

*ExpressionA*:

This is the main expression:

- ON text when for type On / Off.
- Conditions A for type Multi-state.
- Array expression for type Array.
- Numeric Expression for type Numeric.
- String Expression for type String.

*ExpressionB*:

Conditions B, only used for multistate type.

*ExpressionC*:

Conditions C, only used for multistate type.

*ExpressionD*:

Conditions D, only used for multistate type.

*ExpressionE*:

Conditions E, only used for multistate type.

## Return Value

The requested values, as a string

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesDisplayValuePut](#)

## Example

```
` Gets the properties on the Appearance/DisplayValue sheet for a number or text object  
  
GraphicsBuilder.PropertiesDisplayValueGet nType, Expression1, Expression2, Expression3,  
Expression4, Expression5
```

## PropertiesDisplayValuePut

Sets the fields that appear on the **Appearance | Display Value** tab of the Object Properties dialog for a number or text object. This includes the type setting and related expressions.

## Syntax

**PropertiesDisplayValueGet**(*SymbolSetType*, *ExpressionA*, *ExpressionB*, *ExpressionC*, *ExpressionD*, *ExpressionE*)

*SymbolSetType*:

Defines the symbol set type:

0 = On / Off

1 = Multi-state

2 = Array

3 = Numeric

4 = String

*ExpressionA*:

This is the main expression:

- ON text when for type On / Off.
- Conditions A for type Multi-state.
- Array expression for type Array.
- Numeric Expression for type Numeric.
- String Expression for type String.

*ExpressionB*:

Conditions B, only used for multistate type.

*ExpressionC*:

Conditions C, only used for multistate type.

*ExpressionD*:

Conditions D, only used for multistate type.

*ExpressionE*:

Conditions E, only used for multistate type.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesDisplayValueGet](#)

### PropertiesDisplayValueTextGet

Reads the text for a specific index from the **Appearance | Display Value** tab of the Object Properties dialog for a number or text object of type Multistate, Array or Numeric.

## Syntax

**PropertiesDisplayValueTextGet(Index, Text)**

*Index:*

The position of the text:

0..31 for type Multistate.

0..255 for type Array.

0 for type Numeric.

*Text:*

The text written to the field:

- State text for type Multi-state.

- Array text for type Array.

- Format for type Numeric.

## Return Value

The requested values, as a string.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesDisplayValueTextPut](#), [PropertiesDisplayValuePut](#), [PropertiesCicodeObjectPut](#)

### PropertiesDisplayValueTextPut

Sets the text for a specific index on the **Appearance | Display Value** tab of the Object Properties dialog for a number or text object of type Multistate, Array, or Numeric.

## Syntax

**PropertiesDisplayValueTextGet(Index, Text)**

*Index:*

The position of the text:

0..31 for type Multistate.

0..255 for type Array.

0 for type Numeric.

*Text:*

The text written to the field:

- State text for type Multi-state.
- Array text for type Array.
- Format for type Numeric.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesDisplayValueTextGet](#), [PropertiesSymbolSetPut](#), [PropertiesSymbolSetGet](#)

### PropertiesFillColourColourGet

Reads the current color value set for the specified index point on the **Fill | Color** tab of the Object Properties dialog for Array, Threshold and Gradient types.

---

**Note:** As this function does not support True Color functionality, it has been superseded by the function [PropertiesFillColourColourGetEx](#).

---

## Syntax

**PropertiesFillColourColourGet(Index, ColourNo, Limit, Operator)**

*Index:*

Specify the index you would like to read the current color for. This values depends on the type of color fill selected:

0 - 31 for type Multi-state

0 - 255 for type Array

0 - 255 for type Threshold

0- 1 for Gradient

*ColourNo:*

A value between 0 and 255 representing the color applied to the Index setting.

*Limit:*

A value between 0 and 100 representing the threshold limit. Used for type Threshold only.

*Operator:*

The value representing the current operator used for the threshold limit setting:

- 0 : < (less than)
- 1 : > (greater than)
- 2 : <= (less than or equal to)
- 3 : >= (greater than or equal to)

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesFillColourColourPut](#), [PropertiesFillColourGet](#), [PropertiesFillColourPutEx](#)

### PropertiesFillColourColourGetEx

Reads the current color value set for the specified index point on the **Fill | Color** tab of the Object Properties dialog for Array, Threshold and Gradient types.

## Syntax

**PropertiesFillColourColourGet(Index, OnColourNo, OffColourNo, Limit, Operator)**

*Index:*

Specify the index you would like to read the current color for. This value depends on the type of color fill selected:

0 - 31 for type Multi-state

0 - 255 for type Array

0 - 255 for type Threshold

0 - 1 for Gradient

*OnColourNo:*

An RGB value representing the "on" color applied to the Index setting.

*OffColourNo:*

An RGB value representing the "off" color applied to the Index setting.

*Limit:*

A value between 0 and 100 representing the threshold limit. Used for type Threshold only.

*Operator:*

The value representing the current operator used for the threshold limit setting:

- 0 : < (less than)
- 1 : > (greater than)
- 2 : <= (less than or equal to)

3 : >= (greater than or equal to)

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PropertiesFillColourColourPutEx](#), [PropertiesFillColourGet](#), [PropertiesFillColourPut](#)

### PropertiesFillColourColourPut

Sets the color at the specific index on the **Fill | Color** tab of the Object Properties dialog for type Array, Threshold and Gradient.

**Note:** As this function does not support True Color functionality, it has been superseded by the function.

## Syntax

**PropertiesFillColourColourPut**(*Index*, *ColourNo*, *Limit*, *Operator*)

*Index*:

Specify the index you would like to set the current color for. This value depends on the type of color fill selected:

0 - 31 for type Multi-state

0 - 255 for type Array

0 - 255 for type Threshold

0- 1 for Gradient

To set the fill color for on/off mode use index values of OFF = 0 or ON = 1.

*ColourNo*:

A value between 0 and 255 representing the color applied to the Index setting.

*Limit*:

A value between 0 and 100 representing the threshold limit. Used for type Threshold only.

*Operator*:

The value representing the current operator used for the threshold limit setting:

0 : < (less than)

1 : > (greater than)

2 : <= (less than or equal to)

3 : >= (greater than or equal to)

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesFillColourColourGet](#), [PropertiesFillColourGet](#), [PropertiesFillColourPut](#)

### PropertiesFillColourColourPutEx

Sets the color at the specific index on the **Fill | Color** tab of the Object Properties dialog for type Array, Threshold and Gradient.

## Syntax

**PropertiesFillColourColourPutEx**(*Index*, *OnColourNo*, *OffColourNo*, *Limit*, *Operator*)

*Index*:

Specify the index you want to read the current color for. This values depends on the type of color fill selected:

0 - 31 for type Multi-state

0 - 255 for type Array

0 - 255 for type Threshold

0- 1 for Gradient

*OnColourNo*:

An RGB value representing the "on" color applied to the Index setting.

*OffColourNo*:

An RGB value representing the "off" color applied to the Index setting.

*Limit*:

A value between 0 and 100 representing the threshold limit. Used for type Threshold only.

*Operator*:

The value representing the current operator used for the threshold limit setting:

0 : < (less than)

1 : > (greater than)

2 : <= (less than or equal to)

3 : >= (greater than or equal to)

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesFillColourColourGetEx](#), [PropertiesFillColourPut](#)

## PropertiesFillColourGet

Reads the values set on the **Fill | Color** tab of the Object Properties dialog for the current object.

## Syntax

**PropertiesFillColourGet**(*FillColourType*, *ExpressionA*, *ExpressionB*, *ExpressionC*, *ExpressionD*, *ExpressionE*,  
*RangeFlag*, *RangeMin*, *RangeMax*)

*FillColourType*:

The fill color type:

0 = On / Off

1 = Multi-state

2 = Array

3 = Threshold

4 = Gradient

*ExpressionA*:

This is the main expression:

- ON color when for type On / Off
- Conditions A for type Multi-state
- Array expression for type Array
- Color expression for type Animated

*ExpressionB*:

Conditions B, only used for multistate symbol sets.

*ExpressionC*:

Conditions C, only used for multistate symbol sets.

*ExpressionD*:

Conditions D, only used for multistate symbol sets.

*ExpressionE*:

Conditions E, only used for multistate symbol sets.

*RangeFlag*:

If set to TRUE, checks the Specify range check box. Flag is only valid for Threshold and Gradient types.

*RangeMin*:

This floating point value sets the minimum range of the tag value. Only necessary if the argument RangeFlag is set to TRUE.

*RangeMax*:

This floating point value sets the maximum range of the tag value. Only necessary, if the argument RangeFlag is set to TRUE.

## Return Value

The requested values, as a string.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PropertiesFillColourPut](#), [PropertiesFillColourColourPut](#)

### PropertiesFillColourPut

sets the values on the **Fill | Color** tab of the Object Properties dialog for the current object.

## Syntax

**PropertiesFillColourPut**(*FillColourType*, *ExpressionA*, *ExpressionB*, *ExpressionC*, *ExpressionD*, *ExpressionE*,  
*RangeFlag*, *RangeMin*, *RangeMax*)

*FillColourType*:

The fill color type:

0 = On / Off

1 = Multi-state

2 = Array

3 = Threshold

4 = Gradient

*ExpressionA*:

This is the main expression:

- ON color when for type On / Off
- Conditions A for type Multi-state
- Array expression for type Array
- Color expression for type Animated

*ExpressionB*:

Conditions B, only used for multistate symbol sets.

*ExpressionC*:

Conditions C, only used for multistate symbol sets.

*ExpressionD*:

Conditions D, only used for multistate symbol sets.

*ExpressionE*:

Conditions E, only used for multistate symbol sets.

*RangeFlag*:

If set to TRUE, checks the Specify range check box. Flag is only valid for Threshold and Gradient types.

*RangeMin:*

This floating point value sets the minimum range of the tag value. Only necessary if the argument RangeFlag is set to TRUE.

*RangeMax:*

This floating point value sets the maximum range of the tag value. Only necessary, if the argument RangeFlag is set to TRUE.

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PropertiesFillColourGet](#), [PropertiesFillColourColourGet](#), [PropertiesFillColourColourPut](#)

### PropertiesFillLevelGet

Reads the values set on the **Fill | Level** tab of the Object Properties dialog for the current object.

**Note:** As this function does not support True Color functionality, it has been superseded by the function [PropertiesFillLevelGetEx](#).

## Syntax

**PropertiesFillLevelGet(Expression, RangeFlag, RangeMin, RangeMax, OffsetMin, OffsetMax, FillDirection, BackgroundColour)**

*Expression:*

The level expression.

*RangeFlag:*

TRUE if the Specify range check box is selected.

*RangeMin:*

The minimum floating point value in the range of the tag. This argument is only valid if RangeFlag is set to TRUE.

*RangeMax:*

The maximum floating point value in the range of the tag. This argument is only valid if RangeFlag is set to TRUE.

*OffsetMin:*

The value between 0 and 100 representing the percentage of the area displayed as filled when the tag value is at its minimum.

*OffsetMax:*

The value between 0 and 100 representing the percentage of the area displayed as filled when the tag value is at its maximum.

*FillDirection:*

The current fill direction setting:

0 = up

1 = down

2 = left

3 = right

*BackgroundColour:*

A value between 0 and 255 representing the background color setting.

## Return Value

The requested values, as a string.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesFillColourPut](#)

### PropertiesFillLevelGetEx

Reads the values set on the **Fill | Level** tab of the Object Properties dialog for the current object.

## Syntax

**PropertiesFillLevelGetEx**(*Expression*, *RangeFlag*, *RangeMin*, *RangeMax*, *OffsetMin*, *OffsetMax*, *FillDirection*, *OnColour*, *OffColour*)

*Expression:*

The level expression.

*RangeFlag:*

TRUE if the Specify range check box is selected.

*RangeMin:*

The minimum floating point value in the range of the tag. This argument is only valid if RangeFlag is set to TRUE.

*RangeMax:*

The maximum floating point value in the range of the tag. This argument is only valid if RangeFlag is set to TRUE.

*OffsetMin:*

The value between 0 and 100 representing the percentage of the area displayed as filled when the tag value is at its minimum.

*OffsetMax:*

The value between 0 and 100 representing the percentage of the area displayed as filled when the tag value is at its maximum.

*FillDirection:*

The current fill direction setting:

0 = up

1 = down

2 = left

3 = right

*OnColour:*

An RGB value representing the background "on" color setting.

*OffColour:*

An RGB value representing the background "off" color setting.

## Return Value

The requested values, as a string.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesFillColourPut](#)

### PropertiesFillLevelPut

Sets the values on the Fill | Level tab of the Object Properties dialog for the current object.

---

**Note:** As this function does not support True Color functionality, it is superseded by the function [PropertiesFillLevelPutEx](#).

---

## Syntax

**PropertiesFillLevelPut**(*Expression*, *RangeFlag*, *RangeMin*, *RangeMax*, *OffsetMin*, *OffsetMax*, *FillDirection*, *BackgroundColour*)

*Expression:*

The level expression.

*RangeFlag:*

TRUE if the Specify range check box is selected.

*RangeMin:*

The minimum floating point value in the range of the tag. This argument is only valid if RangeFlag is set to TRUE.

*RangeMax:*

The maximum floating point value in the range of the tag. This argument is only valid if RangeFlag is set to TRUE.

*OffsetMin:*

The value between 0 and 100 representing the percentage of the area displayed as filled when the tag value is at its minimum.

*OffsetMax:*

The value between 0 and 100 representing the percentage of the area displayed as filled when the tag value is at its maximum.

*FillDirection:*

The current fill direction setting:

0 = up

1 = down

2 = left

3 = right

*BackgroundColour:*

A value between 0 and 255 representing the background color setting.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesFillLevelPutEx](#), [PropertiesFillLevelGet](#)

### PropertiesFillLevelPutEx

Sets the values on the **Fill | Level** tab of the Object Properties dialog for the current object.

## Syntax

**PropertiesFillLevelPutEx(Expression, RangeFlag, RangeMin, RangeMax, OffsetMin, OffsetMax, FillDirection, OnColour, OffColour)**

*Expression:*

The level expression.

*RangeFlag:*

TRUE if the Specify range check box is selected.

*RangeMin:*

The minimum floating point value in the range of the tag. This argument is only valid if RangeFlag is set to TRUE.

*RangeMax:*

The maximum floating point value in the range of the tag. This argument is only valid if RangeFlag is set to TRUE.

*OffsetMin:*

The value between 0 and 100 representing the percentage of the area displayed as filled when the tag value is at its minimum.

*OffsetMax:*

The value between 0 and 100 representing the percentage of the area displayed as filled when the tag value is at its maximum.

*FillDirection:*

The current fill direction setting:

0 = up

1 = down

2 = left

3 = right

*OnColour:*

An RGB value representing the background "on" color setting.

*OffColour:*

An RGB value representing the background "off" color setting.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesFillLevelGetEx](#)

## PropertiesInputKeyboardGet

Reads the values set on the **Input | Keyboard Command** tab of the Object Properties dialog for the current object.

## Syntax

**PropertiesInputKeyboardGet(Index, KeySequence, Command, Area, Privilege, LogMessage)**

*Index:*

0 to 255 for the key sequence.

*KeySequence:*

String of the keys to be pressed.

*Command:*

Expression for the key sequence command.

*Area:*

0 to 255 for the area, where 0 ticks the check box Same area as object.

*Privilege:*

0 to 255 for the privilege, where 0 ticks the check box Same privilege as object.

*LogMessage:*

The message text to be logged.

## Return Value

The requested values, as a string.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesInputKeyboardPut](#)

### PropertiesInputKeyboardPut

Sets the values on the Input | Keyboard Commands tab of the Object Properties dialog for the current object.

## Syntax

**PropertiesInputKeyboardPut**(*Index*, *KeySequence*, *Command*, *Area*, *Privilege*, *LogMessage*)

*Index*:

0 to 255 for the key sequence.

*KeySequence*:

String of the keys to be pressed.

*Command*:

Expression for the key sequence command.

*Area*:

0 to 255 for the area, where 0 ticks the check box Same area as object.

*Privilege*:

0 to 255 for the privilege, where 0 ticks the check box Same privilege as object.

*LogMessage*:

The message text to be logged.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesInputKeyboardGet](#)

### PropertiesInputTouchGet

Reads the values set on the **Input | Touch** tab of the Object Properties dialog for the current object.

## Syntax

**PropertiesInputTouchGet**(*Action*, *Expression*, *LogMessage*, *RepeatRate*)

*Action*:

The type of keyboard action:

0 = Up

1 = Down

2 = Repeat

*Expression*:

The expression configured for the selected keyboard action (either up, down or repeat).

*LogMessage*:

The message text to be logged.

*RepeatRate*:

A value between 1 and 32000 representing the repeat rate in milliseconds.

## Return Value

The requested values, as a string.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesInputTouchPut](#)

## PropertiesInputTouchPut

Sets the values on the **Input | Touch** tab of the Object Properties dialog for the current object.

## Syntax

**PropertiesInputTouchPut**(*Action*, *Expression*, *LogMessage*, *RepeatRate*)

*Action*:

The type of keyboard action:

0 = Up

1 = Down

2 = Repeat

*Expression*:

The expression configured for the selected keyboard action (either up, down or repeat).

*LogMessage*:

The message text to be logged.

*RepeatRate*:

A value between 1 and 32000 representing the repeat rate in milliseconds.

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PropertiesInputTouchGet](#)

### PropertiesShowDialog

Shows the properties dialog for an object or a form for Genies.

## Syntax

**PropertiesShowDialog**

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

### PropertiesSymbolSetGet

Reads the type and expressions configured on the **Appearance | General** tab of the Object Properties dialog for a symbol set.

## Syntax

**PropertiesSymbolSetGet**(*SymbolSetType*, *ExpressionA*, *ExpressionB*, *ExpressionC*, *ExpressionD*, *ExpressionE*)

*SymbolSetType*:

Defines the symbol set type:

0 = On / Off

1 = Multi-state

2 = Array

3 = Animated

*ExpressionA*:

This is the main expression:

- ON symbol when for type On / Off
- Conditions A for type Multi-state

- Array expression for type Array
- Animate when for type Animated

*ExpressionB:*

Conditions B, only used for multistate symbol sets.

*ExpressionC:*

Conditions C, only used for multistate symbol sets.

*ExpressionD:*

Conditions D, only used for multistate symbol sets.

*ExpressionE:*

Conditions E, only used for multistate symbol sets.

## Return Value

The requested values, as a string.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesSymbolSetPut](#)

## Example

```
` Gets the properties on the Appearance/General sheet for a symbol set
GraphicsBuilder.PropertiesSymbolSetGet nType, Expression1, Expression2, Expression3,
Expression4, Expression5
```

## PropertiesSymbolSetPut

Sets the type defined for a symbol set on the **Appearance | General** tab of the Object Properties dialog, as well any expressions used

## Syntax

**PropertiesSymbolSetPut**(*SymbolSetType*, *ExpressionA*, *ExpressionB*, *ExpressionC*, *ExpressionD*, *ExpressionE*)

*SymbolSetType:*

Defines the symbol set type:

0 = On / Off

1 = Multi-state

2 = Array

3 = Animated

*ExpressionA:*

This is the main expression:

- ON symbol when for type On / Off
- Conditions A for type Multi-state
- Array expression for type Array
- Animate when for type Animated

*ExpressionB:*

Conditions B, only used for multistate symbol sets.

*ExpressionC:*

Conditions C, only used for multistate symbol sets.

*ExpressionD:*

Conditions D, only used for multistate symbol sets.

*ExpressionE:*

Conditions E, only used for multistate symbol sets.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesSymbolSetGet](#)

## Example

```
` Sets the properties on the Appearance General sheet for a symbol set
GraphicsBuilder.PropertiesSymbolSetPut 0, "ON / OFF", "", "", "", ""
```

## PropertiesSymbolSetSymbolGet

Retrieves the Element name and Library name of the "Index" element of the currently selected object.

"Index" refers to the element within the currently selected object. For example:

- If the currently selected object is an On/Off symbol set, index can be a value in the range 0..1
- If the currently selected object is a multistate symbol set, index can be a value in the range 0..31
- If the currently selected object is an array symbol set, index can be a value in the range 0..255
- If the currently selected object is an animated symbol set, index can be a value in the range 0..255
- On return, "Element" will contain the name of the symbol set element name for the "Index" element
- On return, "Library" will contain the name of the symbol set library name for the "Index" element, for example:

```
Index=0, Element="detail_entrycoil1_grey_01", Library="steelmill"
Index=1, Element="detail_entrycoil1_green_01", Library="steelmill"
```

## Syntax

**PropertiesSymbolSetSymbolGet(Index, Library, Element)**

## Return Value

N/A

**Note:** For details on handling return and error values, see [Error Handling](#).

## Example

```
Public Sub Test()
Dim gb As GraphicsBuilder.GraphicsBuilder
.
.
.
gb.PropertiesSymbolSetSymbolPut(1, "Library", "Element")
Dim sLibrary As String
Dim sElement As String
gb.PropertiesSymbolSetSymbolGet(1, sLibrary, sElement)
End Sub
```

## Related Functions

[PropertiesSymbolSetSymbolPut](#)

### PropertiesSymbolSetSymbolPut

Sets the Element name and Library name of the "Index" element of the currently selected object.

"Index" refers to the element within the currently selected object. For example:

- If the currently selected object is an On/Off symbol set, index can be a value in the range 0..1
- If the currently selected object is a multistate symbol set, index can be a value in the range 0..31
- If the currently selected object is an array symbol set, index can be a value in the range 0..255
- If the currently selected object is an animated symbol set, index can be a value in the range 0..255
- On return, "Element" will contain the name of the symbol set element name for the "Index" element
- On return, "Library" will contain the name of the symbol set library name for the "Index" element, for example:

```
Index=0, Element="detail_entrycoil1_grey_01", Library="steelmill"
Index=1, Element="detail_entrycoil1_green_01", Library="steelmill"
```

## Syntax

**PropertiesSymbolSetSymbolPut(Index, Library, Element)**

## Return Value

N/A

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Example

```
Public Sub Test()
Dim gb As GraphicsBuilder.GraphicsBuilder
.
.
.
gb.PropertiesSymbolSetSymbolPut(1, "Library", "Element")
Dim sLibrary As String
Dim sElement As String
gb.PropertiesSymbolSetSymbolGet(1, sLibrary, sElement)
End Sub
```

## Related Functions

[PropertiesSymbolSetSymbolGet](#)

### PropertiesTransCentreOffsetExpressGet

Retrieve the express properties.

## Syntax

**PropertiesTransCentreOffsetExpressGet**(*movementRotationalExpress*, *scalingHorizontalExpress*,  
*scalingVerticalExpress*, *sliderRotationalExpress*)

*movementRotationalExpress*  
Movement Rotational Express

*scalingHorizontalExpress*  
Scaling Horizontal Express

*scalingVerticalExpress*  
Scaling Vertical Express

*sliderRotationalExpress*  
Slider Rotational Express

## Return Value

N/A

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Example

```
Public Sub Test()
Dim gb As GraphicsBuilder.GraphicsBuilder
gb = New GraphicsBuilder.GraphicsBuilder
.
.
.
gb.PropertiesTransCentreOffsetExpressPut(0, 0, 0, 0)
Dim nMovRot As Short
Dim nScaleHorz As Short
Dim nScaleVert As Short
Dim nSliderRot As Short
gb.PropertiesTransCentreOffsetExpressGet(nMovRot, nScaleHorz, nScaleVert, nSliderRot)
End Sub
```

## Related Functions

[PropertiesTransCentreOffsetExpressPut](#)

### PropertiesTransCentreOffsetExpressPut

Sets the express properties.

## Syntax

**PropertiesTransCentreOffsetExpressPut**(*movementRotationalExpress*, *scalingHorizontalExpress*,  
*scalingVerticalExpress*, *sliderRotationalExpress*)

*movementRotationalExpress*  
Movement Rotational Express

*scalingHorizontalExpress*  
Scaling Horizontal Express

*scalingVerticalExpress*  
Scaling Vertical Express

*sliderRotationalExpress*  
Slider Rotational Express

## Return Value

N/A

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Example

```
Public Sub Test()
Dim gb As GraphicsBuilder.GraphicsBuilder
```

```
gb = New GraphicsBuilder.GraphicsBuilder
.
.
.
gb.PropertiesTransCentreOffsetExpressPut(0, 0, 0, 0)
Dim nMovRot As Short
Dim nScaleHorz As Short
Dim nScaleVert As Short
Dim nSliderRot As Short
gb.PropertiesTransCentreOffsetExpressGet(nMovRot, nScaleHorz, nScaleVert, nSliderRot)
End Sub
```

## Related Functions

[PropertiesTransCentreOffsetExpressGet](#)

### PropertiesTransformationGet

Reads the property values set on the **Movement**, **Scaling** and **Slider** tabs of the Object Properties dialog for the current object.

## Syntax

**PropertiesTransformationGet**(*Action*, *Expression*, *RangeFlag*, *RangeMin*, *RangeMax*, *OffsetMin*, *OffsetMax*,  
*CustomFlag*, *CentreOffsetRight*, *CentreOffsetDown*)

*Action*:

Selects the tab on the Object Properties dialog that data will be read from:

0 = MovementHorizontal

1 = MovementVertical

2 = MovementRotational

3 = ScalingHorizontal

4 = ScalingVertical

5 = SliderHorizontal

6 = SliderVertical

7 = SliderRotational

*Expression*:

The main expression in Field:

- **Movement expression** for the actions MovementHorizontal or MovementVertical
- **Angle expression** for action MovementRotational
- **Scaling expression** for actions ScalingHorizontal or ScalingVertical
- **Tag** for actions SliderHorizontal, SliderVertical or SliderRotational

*RangeFlag*:

TRUE if Specify range is checked

*RangeMin*:

The minimum floating point value. 0 (zero) if RangeFlag is not set.

*RangeMax:*

This maximum floating point value. 0 (zero) if RangeFlag is set to TRUE.

*OffsetMin:*

The value of Angle at minimum for the actions MovementRotational and SliderRotational, or Offset at minimum for other actions.

*OffsetMax:*

The value of Angle at maximum for the actions MovementRotational and SliderRotational, or Offset at maximum for other actions.

*CustomFlag:*

TRUE if custom is selected for the center axis offset setting for the actions MovementRotational, SliderRotational, Scaling Horizontal or ScalingVertical.

*CentreOffsetRight:*

A value between 0 and 32767 representing the customized setting for center offset right. 0 (zero) if CustomFlag is not set.

*CentreOffsetDown:*

A value between 0 and 32767 representing the customized setting for center offset down. 0 (zero) if CustomFlag is not set.

## Return Value

The requested values, as a string.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesTransformationPut](#)

### PropertiesTransformationPut

Sets values for the properties on the **Movement**, **Scaling** and **Slider** tabs of the Object Properties dialog.

## Syntax

**PropertiesTransformationGet**(*Action*, *Expression*, *RangeFlag*, *RangeMin*, *RangeMax*, *OffsetMin*, *OffsetMax*, *CustomFlag*, *CentreOffsetRight*, *CentreOffsetDown*)

*Action:*

Selects the tab on the Object Properties dialog that data will be read from:

0 = MovementHorizontal

1 = MovementVertical

2 = MovementRotational

3 = ScalingHorizontal

4 = ScalingVertical

5 = SliderHorizontal

6 = SliderVertical

7 = SliderRotational

*Expression:*

The main expression in Field:

- **Movement expression** for the actions MovementHorizontal or MovementVertical
- **Angle expression** for action MovementRotational
- **Scaling expression** for actions ScalingHorizontal or ScalingVertical
- **Tag** for actions SliderHorizontal, SliderVertical or SliderRotational

*RangeFlag:*

TRUE if Specify range is checked

*RangeMin:*

The minimum floating point value. 0 (zero) if RangeFlag is not set.

*RangeMax:*

This maximum floating point value. 0 (zero) if RangeFlag is set to TRUE.

*OffsetMin:*

The value of Angle at minimum for the actions MovementRotational and SliderRotational, or Offset at minimum for other actions.

*OffsetMax:*

The value of Angle at maximum for the actions MovementRotational and SliderRotational, or Offset at maximum for other actions.

*CustomFlag:*

TRUE if custom is selected for the center axis offset setting for the actions MovementRotational, SliderRotational, Scaling Horizontal or ScalingVertical.

*CentreOffsetRight:*

A value between 0 and 32767 representing the customized setting for center offset right. 0 (zero) if CustomFlag is not set.

*CentreOffsetDown:*

A value between 0 and 32767 representing the customized setting for center offset down. 0 (zero) if CustomFlag is not set.

## Return Value

The requested values, as a string.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesTransformationGet](#)

## PropertiesTrendGet

Reads the values for a trend object as set on the **Appearance | General** tab of the Object Properties dialog.

---

**Note:** As this function does not support True Color functionality, it has been superseded by the [function](#).

---

## Syntax

**PropertiesTrendGet**(*NumberOfSamples*, *PixelPerSample*, *Expression1*, *Colour1*, *Expression2*, *Colour2*,  
*Expression3*, *Colour3*, *Expression4*, *Colour4*, *Expression5*, *Colour5*, *Expression6*, *Colour6*, *Expression7*, *Colour7*,  
*Expression8*, *Colour8*)

*NumberOfSamples*:

A value between 0 and 32767 representing the number of samples in a trend display.

*PixelPerSample*:

A value between 1 and 32, representing the width of each sample in pixels.

*Expression1*:

String argument for the field Pen1.

*Colour1*:

A value between 0 and 255 representing the color of trend Pen1.

*Expression2*:

String argument for the field Pen2.

*Colour2*:

A value between 0 and 255 representing the color of trend Pen2.

*Expression3*:

String argument for the field Pen3.

*Colour3*:

A value between 0 and 255 representing the color of trend Pen3.

*Expression4*:

String argument for the field Pen4.

*Colour4*:

A value between 0 and 255 representing the color of trend Pen4.

*Expression5*:

String argument for the field Pen5.

*Colour5*:

A value between 0 and 255 representing the color of trend Pen5.

*Expression6*:

String argument for the field Pen6.

*Colour6*:

A value between 0 and 255 representing the color of trend Pen6.

*Expression7*:

String argument for the field Pen7.

*Colour7:*

A value between 0 and 255 representing the color of trend Pen7.

*Expression8:*

String argument for the field Pen8.

*Colour8:*

A value between 0 and 255 representing the color of trend Pen8.

## Return Value

The requested values, as a string.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesTrendPut](#)

### PropertiesTrendGetEx

Reads the values for a trend object as set on the **Appearance | General** tab of the Object Properties dialog.

## Syntax

**PropertiesTrendGetEx**(*NumberOfSamples*, *PixelPerSample*, *Expression1*, *OnColour1*, *OffColour1*, *Expression2*, *OnColour2*, *OffColour2*, *Expression3*, *OnColour3*, *OffColour3*, *Expression4*, *OnColour4*, *OffColour4*, *Expression5*, *OnColour5*, *OffColour5*, *Expression6*, *OnColour6*, *OffColour6*, *Expression7*, *OnColour7*, *OffColour7*, *Expression8*, *OnColour8*, *OffColour8*)

*NumberOfSamples:*

A value between 0 and 32767 representing the number of samples in a trend display.

*PixelPerSample:*

A value between 1 and 32, representing the width of each sample in pixels.

*Expression1:*

String argument for the field Pen1.

*OnColour1:*

An RGB value representing the "on" color of trend Pen1.

*OffColour1:*

An RGB value representing the "off" color of trend Pen1.

*Expression2:*

String argument for the field Pen2.

*OnColour2:*

An RGB value representing the "on" color of trend Pen2.

*OffColour2:*

An RGB value representing the "off" color of trend Pen2.

*Expression3:*

String argument for the field Pen3.

*OnColour3:*

An RGB value representing the "on" color of trend Pen3.

*OffColour3:*

An RGB value representing the "off" color of trend Pen3.

*Expression4:*

String argument for the field Pen4.

*OnColour4:*

An RGB value representing the "on" color of trend Pen4.

*OffColour4:*

An RGB value representing the "off" color of trend Pen4.

*Expression5:*

*String argument for the field Pen5.*

*OnColour5:*

An RGB value representing the "on" color of trend Pen5.

*OffColour5:*

An RGB value representing the "off" color of trend Pen5.

*Expression6:*

String argument for the field Pen6.

*OnColour6:*

An RGB value representing the "on" color of trend Pen6.

*OffColour6:*

An RGB value representing the "off" color of trend Pen6.

*Expression7:*

String argument for the field Pen7.

*OnColour7:*

An RGB value representing the "on" color of trend Pen7.

*OffColour7:*

An RGB value representing the "off" color of trend Pen7.

*Expression8:*

String argument for the field Pen8.

*OnColour8:*

An RGB value representing the "on" color of trend Pen8.

*OffColour8:*

An RGB value representing the "off" color of trend Pen8.

## Return Value

The requested values, as a string.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PropertiesTrendPutEx](#)

### PropertiesTrendPut

Sets the values for a trend object that appear on the **Appearance | General** tab of the Object Properties dialog.

**Note:** As this function does not support True Color functionality, it has been superseded by the functions [PropertiesTrendPutEx](#).

## Syntax

**PropertiesTrendGet**(*NumberOfSamples*, *PixelPerSample*, *Expression1*, *Colour1*, *Expression2*, *Colour2*,  
*Expression3*, *Colour3*, *Expression4*, *Colour4*, *Expression5*, *Colour5*, *Expression6*, *Colour6*, *Expression7*, *Colour7*,  
*Expression8*, *Colour8*)

*NumberOfSamples*:

A value between 0 and 32767 representing the number of samples in a trend display.

*PixelPerSample*:

A value between 1 and 32, representing the width of each sample in pixels.

*Expression1*:

String argument for the field Pen1.

*Colour1*:

A value between 0 and 255 representing the color of trend Pen1

*Expression2*:

String argument for the field Pen2.

*Colour2*:

A value between 0 and 255 representing the color of trend Pen2.

*Expression3*:

String argument for the field Pen3.

*Colour3*:

A value between 0 and 255 representing the color of trend Pen3.

*Expression4*:

String argument for the field Pen4.

*Colour4*:

A value between 0 and 255 representing the color of trend Pen4.

*Expression5*:

String argument for the field Pen5.

*Colour5:*

A value between 0 and 255 representing the color of trend Pen5.

*Expression6:*

String argument for the field Pen6.

*Colour6:*

A value between 0 and 255 representing the color of trend Pen6.

*Expression7:*

String argument for the field Pen7.

*Colour7:*

A value between 0 and 255 representing the color of trend Pen7.

*Expression8:*

String argument for the field Pen8.

*Colour8:*

A value between 0 and 255 representing the color of trend Pen8.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PropertiesTrendGet](#)

### PropertiesTrendPutEx

Sets the values for a trend object that appear on the **Appearance | General** tab of the Object Properties dialog.

## Syntax

**PropertiesTrendPutEx**(*NumberOfSamples*, *PixelPerSample*, *Expression1*, *OnColour1*, *OffColour1*, *Expression2*, *OnColour2*, *OffColour2* , *Expression3*, *OnColour3*, *OffColour3* , *Expression4*, *OnColour4*, *OffColour4* , *Expression5*, *OnColour5*, *OffColour5*, *Expression6*, *OnColour6*, *OffColour6* , *Expression7*, *OnColour7*, *OffColour7*, *Expression8*, *OnColour8*, *OffColour8* )

*NumberOfSamples:*

A value between 0 and 32767 representing the number of samples in a trend display.

*PixelPerSample:*

A value between 1 and 32, representing the width of each sample in pixels.

*Expression1:*

String argument for the field Pen1.

*OnColour1:*

An RGB value representing the "on" color of trend Pen1.

*OffColour1:*

An RGB value representing the "off" color of trend Pen1.

*Expression2:*

String argument for the field Pen2.

*OnColour2:*

An RGB value representing the "on" color of trend Pen2.

*OffColour2:*

An RGB value representing the "off" color of trend Pen2.

*Expression3:*

String argument for the field Pen3.

*OnColour3:*

An RGB value representing the "on" color of trend Pen3.

*OffColour3:*

An RGB value representing the "off" color of trend Pen3.

*Expression4:*

String argument for the field Pen4.

*OnColour4:*

An RGB value representing the "on" color of trend Pen4.

*OffColour4:*

An RGB value representing the "off" color of trend Pen4.

*Expression5:*

String argument for the field Pen5.

*OnColour5:*

An RGB value representing the "on" color of trend Pen5.

*OffColour5:*

An RGB value representing the "off" color of trend Pen5.

*Expression6:*

String argument for the field Pen6.

*OnColour6:*

An RGB value representing the "on" color of trend Pen6.

*OffColour6:*

An RGB value representing the "off" color of trend Pen6.

*Expression7:*

String argument for the field Pen7.

*OnColour7:*

An RGB value representing the "on" color of trend Pen7.

*OffColour7:*

An RGB value representing the "off" color of trend Pen7.

*Expression8:*

String argument for the field Pen8.

*OnColour8:*

An RGB value representing the "on" color of trend Pen8.

*OffColour8:*

An RGB value representing the "off" color of trend Pen8.

**Return Value**

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

**Related Functions**

[PropertiesTrendGetEx](#)

**PropertyVisibility**

Sets the Hidden when argument on the **Appearance | Visibility** tab of the Object Properties dialog.

**Syntax**

**PropertyVisibility**(*Text*)

*Text:*

The argument string.

**Return Value**

If retrieving the current setting, the argument string. If enabling or disabling the option, 0 (zero) if successful. In both cases, an error is returned if unsuccessful.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

**Note:** This function is implemented in the C++ environment as two separate functions:

`put_PropertyVisibility` enables or disables this option, and `get_PropertyVisibility` retrieves the current option setting.

---

**Event Functions**

The following events use the automation `Idispatch` mechanism to fire events in specific situations.

<b>BrokenLink</b>	This event is fired if a missing link is encountered
-------------------	--

	while executing the functions <a href="#">ProjectUpdatePages()</a> or <a href="#">PageOpen()</a> . Details of the missing object are provided through the parameters <i>Project</i> , <i>Library</i> , <i>Object</i> , <i>GenieOrSymbol</i> .
<a href="#">PasteGenie</a>	When the <a href="#">LibrarySelectionHooksEnabled()</a> attribute is set to TRUE, this event is fired when: the paste Genie menu item is selected; the paste genie toolbar button is pressed; or F11 is pressed.
<a href="#">PasteSymbol</a>	When the <a href="#">LibrarySelectionHooksEnabled()</a> attribute is set to TRUE, this event is fired when the paste symbol menu item is selected, the paste symbol toolbar button is pressed, or F6 is pressed.
<a href="#">ProjectChange</a>	This event is fired whenever a new project is selected in Plant SCADA Studio.
<a href="#">Selection</a>	When <a href="#">SelectionEventEnabled()</a> is set to TRUE, this event is fired every time a selection is made within a graphics page. The dimension of the selection rectangle is passed as parameters.
<a href="#">SwapObject</a>	When the <a href="#">LibrarySelectionHooksEnabled()</a> attribute is set to TRUE, this event is fired when pressing the CTRL+SHIFT keys and double-clicking on the object in the graphics page.

For details and a VB example on handling return and error values, see [Error Handling](#).

## See Also

[Automation Events](#)

[Function Categories](#)

## BrokenLink

This event is fired if a missing link is encountered while executing the functions [ProjectUpdatePages](#) or [PageOpen](#). Details of the missing object are provided through the parameters *Project*, *Library*, *Object*, *GenieOrSymbol*.

## Syntax

**BrokenLink(*Project*, *Library*, *Object*, *GenieOrSymbol*)**

*Project*:

The name of the project.

*Library*:

The name of the library.

*Object:*

The name of the symbol or Genie.

*GenieOrSymbol:*

Identifies if the object is a symbol or Genie: 1 = Genie; 2 = symbol.

## See Also

[Automation Events](#)

### PasteGenie

When the `LibSelectionHooksEnabled` attribute is set to TRUE, this event is fired when the Paste Genie menu item is selected, the Paste Genie toolbar button is pressed, or when F11 is pressed.

## Syntax

**PasteGenie**

## See Also

[Automation Events](#)

### PasteSymbol

When the `LibSelectionHooksEnabled` attribute is set to TRUE, this event is fired when the paste symbol menu item is selected, the paste symbol toolbar button is pressed, or F6 is pressed.

## Syntax

**PasteSymbol**

## See Also

[Automation Events](#)

### ProjectChange

This event is fired whenever a new project is selected in Plant SCADA Studio.

## Syntax

**ProjectChange**

## See Also

[Automation Events](#)

## Selection

When [SelectionEventEnabled](#) is set to TRUE, this event is fired every time a selection is made within a graphics page. The dimension of the selection rectangle is passed as parameters.

## Syntax

**Selection** (*FromXPosition*, *FromYPosition*, *ToXPosition*, *ToYPosition*)

*FromXPosition*:

Distance from the left-hand side of the page to top-left hand corner of the selection rectangle (in pixels).

*FromYPosition*:

Distance from the top of the page to the top-left hand corner of the selection rectangle (in pixels).

*ToXPosition*:

Distance from the left-hand side of the page to the bottom-right hand corner of the selection rectangle (in pixels).

*ToYPosition*:

Distance from the top of the page to the bottom-right hand corner of the selection rectangle (in pixels).

## See Also

[Automation Events](#)

## SwapObject

When the [LibSelectionHooksEnabled](#) attribute is set to TRUE, this event is fired when pressing the CTRL+SHIFT keys and double-clicking the object in the graphics page.

## Syntax

**SwapObject**

## See Also

[Automation Events](#)

## Library Object Functions

With Library Object functions you can use and manipulate the objects stored in libraries in your project. This includes such objects as Genies, Super Genies, Symbols, and so on.

<a href="#">LibraryFirstObject</a>	Returns the name of the first library object.
<a href="#">LibraryNextObject</a>	Returns the name of the next library object when following a call of the function LibraryFirstObject.
<a href="#">LibraryObjectFirstProperty</a>	Returns the name and value of the active Genie's first property.
<a href="#">LibraryObjectFirstPropertyEx</a>	Returns the name and value of a specified Genie's first property.
<a href="#">LibraryObjectHotspotGet</a>	Retrieves the hotspot marker in a Genie or symbol page.
<a href="#">LibraryObjectHotspotPut</a>	Positions the hotspot marker in a Genie or symbol page.
<a href="#">LibraryObjectName</a>	Returns the name of the selected object.
<a href="#">LibraryObjectNextProperty</a>	Returns the name and value of the active Genie's "next" property when implemented following a call of the function LibraryObjectFirstProperty.
<a href="#">LibraryObjectNextPropertyEx</a>	Returns the name and value of the "next" property of the Genie specified by the implementation of LibraryObjectFirstPropertyEx.
<a href="#">LibraryObjectPlace</a>	Places a library object (a symbol or genie) on the active Plant SCADA graphics page at the default location (top left corner).
<a href="#">LibraryObjectPlaceEx</a>	Places a library object (a symbol or genie) on the active Plant SCADA graphics page at the specified location.
<a href="#">LibraryObjectPutProperty</a>	Sets the value of a specified property for the active genie.
<a href="#">LibSelectionHooksEnabled</a>	Writing a TRUE value with this function enables library selection hooks. When enabled, selecting Paste Genie or Paste Symbol (or their equivalent function key of toolbar button) will not show the standard selection dialog, but will fire the automation event PasteSymbol or PasteGenie instead.
<a href="#">LibraryShowPasteDialog</a>	Shows either the paste Genie or Paste Symbol dialog.

For details and a VB example on handling return and error values, see [Error Handling](#).

## See Also

[Automation Events](#)

[Function Categories](#)

## LibraryFirstObject

Returns the name of the first library object. Can be used with [LibraryNextObject](#) to iterate through library objects.

## Syntax

**LibraryFirstObject**(*Project*,*Library*,*Type*,*Object*)

*Project*:

The name of the project that contains the object library you would like to browse.

*Library*:

Specifies the library that contains the objects you would like to browse.

*Type*:

Specifies the type of the object you want to browse.

1 = Genie

2 = Super Genie

3 = Template

*Object*:

Returns the name of the first object as a string.

## Return Value

The name of the first object as string values.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[LibraryNextObject](#), [LibraryObjectPlace](#), [LibraryObjectFirstProperty](#), [LibraryObjectNextProperty](#),  
[LibraryObjectFirstPropertyEx](#), [LibraryObjectPutProperty](#),

## Example

```
On Error Resume Next
Err.Clear
GraphicsBuilder.LibraryFirstObject ProjectName,LibraryName,1,ObjectName
While Err.Number = 0
Debug.Print ObjectName
GraphicsBuilder.LibraryNextObject ObjectName
Wend
```

## LibraryNextObject

Returns the name of the next library object when following a call of the function [LibraryFirstObject](#). By using multiple calls of this function, you can iterate through a library objects.

## Syntax

**LibraryNextObject**(*ObjectName*)

*ObjectName*:

Returns the name of the next object as a string.

## Return Value

The name of the next object as string values.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[LibraryFirstObject](#), [LibraryObjectPlace](#), [LibraryObjectFirstProperty](#), [LibraryObjectNextProperty](#),  
[LibraryObjectFirstPropertyEx](#), [LibraryObjectPutProperty](#), [LibraryObjectName](#)

## Example

```
On Error Resume Next
Err.Clear
GraphicsBuilder.LibraryFirstObject ProjectName,LibraryName,1,ObjectName
While Err.Number = 0
    Debug.Print ObjectName
    GraphicsBuilder.LibraryNextObject ObjectName
Wend
```

## LibraryObjectFirstProperty

Returns the name and value of the active Genie's first property. Can be used with [LibraryObjectNextProperty](#) to step through a genie's properties.

## Syntax

**LibraryObjectFirstProperty**(*PropertyName*, *PropertyValue*)

*PropertyName*:

Returns the name of the active genie's first property as a string.

*PropertyValue*:

Returns the value of the active genie's first property as a string.

## Return Value

The name and value of the Genie's first property as string values

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[LibraryObjectPlace](#), [LibraryObjectNextProperty](#), [LibraryObjectPutProperty](#), [LibraryObjectName](#)

## Example

```
On Error Resume Next
Err.Clear
GraphicsBuilder.LibraryObjectFirstProperty PropName, PropValue
While Err.Number = 0
    Debug.Print PropName, PropValue
    GraphicsBuilder.LibraryObjectNextProperty PropName, PropValue
Wend
```

## LibraryObjectFirstPropertyEx

Returns the name and value of a specified Genie's first property. Can be used in conjunction with [LibraryObjectNextPropertyEx](#) to step through the specified Genie's properties.

## Syntax

**LibraryObjectFirstPropertyEx**(*Project*, *Library*, *Object*, *PropertyName*, *PropertyValue*)

*Project*:

The name of the project where the Genie is located.

*Library*:

The name of the library where the Genie is located.

*Object*:

The name of the genie.

*PropertyName*:

Returns the name of the active genie's first property as a string.

*PropertyValue*:

Returns the value of the active genie's first property as a string.

## Return Value

The name and value of the specified Genie's first property as string values.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[LibraryObjectPlace](#), [LibraryObjectNextProperty](#), [LibraryObjectPutProperty](#), [LibraryObjectName](#)

## Example

```
' Retrieves the first property of the specified Genie
GraphicsBuilder.LibraryObjectFirstPropertyEx "include", "motors", "Motor_2_east", PropName,
PropValue
```

## LibraryObjectHotspotGet

Retrieves the hotspot marker in a Genie or symbol page. Fails if not a Genie or symbol page.

## Syntax

**LibraryObjectHotspotGet(*Xposition*, *YPosition*)**

*Xposition*:

Absolute X position in pixels from the left side of the page.

*YPosition*:

Absolute Y position in pixels from the top of the page.

## Return Value

X and Y values for the hotspot, where X represents the number of pixels from the left hand side of the page, and Y represents the number of pixels from the top of the page.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[LibraryObjectHotspotPut](#)

## LibraryObjectHotspotPut

Positions the hotspot marker in a Genie or symbol page. Fails if not a Genie or symbol page.

## Syntax

**LibraryObjectHotspotPut(*Xposition*, *YPosition*)**

*Xposition*:

Absolute X position in pixels from the left side of the page.

*YPosition*:

Absolute Y position in pixels from the top of the page.

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[LibraryObjectHotspotGet](#)

### LibraryObjectName

Returns the name of the selected object, if it is a library object.

## Syntax

**LibraryObjectName**(*Project*, *Library*, *Object*, *GenieOrSymbol*)

*Project*:

The name of the project that contains the object library you would like to source.

*Library*:

Specifies the library that contains the symbol or genie you would like to retrieve the name of.

*Object*:

The name of the symbol or genie as a string.

*GenieOrSymbol*:

Indicates whether the object you want to retrieve the name for is a symbol or a genie.

1 = Genie

2 = Symbol

## Return Value

The name of the specified object as a string.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[LibraryObjectFirstProperty](#), [LibraryObjectNextProperty](#), [LibraryObjectPutProperty](#)

## Example

```
On Error Resume Next
Err.Clear
GraphicsBuilder.LibraryObjectName Project, File, Page, LibType
If Err.Number = 0 Then
    Debug.Print Project; "."; File; "."; Page
```

```
Else
    Debug.Print "not a library object"
End If
```

## LibraryObjectNextProperty

Returns the name and value of the active Genie's "next" property when implemented following a call of the function [LibraryObjectFirstProperty](#). By using multiple calls of this function, you can iterate through an object's properties.

## Syntax

**LibraryObjectNextPropertyEx**(*PropertyName*, *PropertyValue*)

*PropertyName*:

Returns the name of the active genie's next property as a string.

*PropertyValue*:

Returns the value of the active genie's next property as a string.

## Return Value

The name and value of the Genie's next property as string values

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[LibraryObjectPlace](#), [LibraryObjectFirstProperty](#), [LibraryObjectPutProperty](#), [LibraryObjectName](#)

## Example

```
On Error Resume Next
Err.Clear
GraphicsBuilder.LibraryObjectFirstProperty PropName, PropValue
While Err.Number = 0
    Debug.Print PropName, PropValue
    GraphicsBuilder.LibraryObjectNextProperty PropName, PropValue
Wend
```

## LibraryObjectNextPropertyEx

Returns the name and value of the "next" property of the Genie specified by the implementation of [LibraryObjectFirstPropertyEx](#). By using multiple calls of this function, you can iterate through the specified genie's properties.

## Syntax

**LibraryObjectNextPropertyEx**(*PropertyName*, *PropertyValue*)

*PropertyName:*

Returns the name of the active genie's next property as a string.

*PropertyValue:*

Returns the value of the active genie's next property as a string.

## Return Value

The name and value of the Genie's next property as string values.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

[LibraryObjectFirstPropertyEx](#)

## Example

```
On Error Resume Next
Err.Clear
GraphicsBuilder.LibraryObjectFirstPropertyEx "include", "motors", "Motor_2_east", PropName,
PropValue
While Err.Number = 0
    Debug.Print PropName, PropValue
    GraphicsBuilder.LibraryObjectNextProperty PropName, PropValue
Wend
```

## LibraryObjectPlace

Places a library object (a symbol or genie) on the active Plant SCADA graphics page at the default location (top left corner). This function will not succeed if the specified object is not found.

## Syntax

**LibraryObjectPlace**(*Project*, *Library*, *Object*, *GenieOrSymbol*, *Linked*)

*Project:*

The name of the project that contains the object library you would like to source.

*Library:*

Specifies the library that contains the symbol or genie you would like to place on the active Plant SCADA graphics page.

*Object:*

The name of the symbol or genie you would like to place on the active Plant SCADA graphics page.

*GenieOrSymbol:*

Indicates whether the object you want to use is a symbol or a genie.

0 = Library type unknown (will automatically select genie or symbol)

1 = Genie

2 = Symbol

*Linked:*

If set to TRUE, the object will remain linked to the library it came from. (select TRUE for Genies). Can only be set to FALSE if GenieOrSymbol is set to 2 (Symbol).

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[LibraryObjectFirstProperty](#), [LibraryObjectNextProperty](#), [LibraryObjectPutProperty](#), [LibraryObjectName](#)

## Example

```
' Adds an object to the current Plant SCADA graphics page
GraphicsBuilder.LibraryObjectPlace "include", "agitator", "agit_1_Pos1_g", 2, True
GraphicsBuilder.PositionAt 200, 200
```

## LibraryObjectPlaceEx

Places a library object (a symbol or genie) on the active Plant SCADA graphics page at the specified location. This function will not succeed if the specified object is not found.

## Syntax

**LibraryObjectPlaceEx**(*Project*, *Library*, *Object*, *GenieOrSymbol*, *Linked*, *Xposition*, *YPosition*)

*Project*:

The name of the project that contains the object library you would like to source.

*Library*:

Specifies the library that contains the symbol or genie you would like to place on the active Plant SCADA graphics page.

*Object*:

The name of the symbol or genie you would like to place on the active Plant SCADA graphics page.

*GenieOrSymbol*:

Indicates whether the object you want to use is a symbol or a genie.

0 = Library type unknown (will automatically select genie or symbol)

1 = Genie

2 = Symbol

*Linked*:

If set to TRUE, the object will remain linked to the library it came from. (Select TRUE for Genies).

*Xposition*:

Absolute X position in pixels from the left hand side of the page.

***YPosition:***

Absolute Y position in pixels from the top of the page.

**Return Value**

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

**Related Functions**

[LibraryObjectFirstProperty](#), [LibraryObjectNextProperty](#), [LibraryObjectPutProperty](#), [LibraryObjectName](#)

**Example**

```
' Adds an object to the current graphics page at 200 pixels from the left and top
GraphicsBuilder.LibraryObjectPlaceEx "include", "agitator", "agit_1_Pos1_g", 2, True, 200,
200
```

**LibraryObjectPutProperty**

Sets the value of a specified property for the active genie. The field name is case-sensitive.

**Syntax**

**LibraryObjectPutProperty(*PropertyName*, *PropertyValue*)**

***PropertyName:***

The name of the property to be modified, as returned by the function [LibraryObjectFirstProperty](#) or [LibraryObjectNextProperty](#).

***PropertyValue:***

The value to be written to the property as a string.

**Return Value**

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

**Related Functions**

[LibraryObjectPlace](#), [LibraryObjectFirstProperty](#), [LibraryObjectNextProperty](#),

**Example**

```
GraphicsBuilder.LibraryObjectPlace "include", "motors", "Motor_1_east", 1, True
```

```
GraphicsBuilder.LibraryObjectPutProperty "Tag", "My test genie"
```

## LibraryShowPasteDialog

Shows either the Paste Genie or Paste Symbol dialog.

## Syntax

**LibraryShowPasteDialog(*GenieOrSymbol*)**

*GenieOrSymbol*:

Indicates whether the object you want to use is a symbol or a genie.

1 = Genie

2 = Symbol

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PasteSymbol](#), [PasteGenie](#)

## LibSelectionHooksEnabled

Writing a TRUE value with this function enables library selection hooks. When enabled, selecting **Paste Genie** or **Paste Symbol** (or their equivalent function key or toolbar button) will not show the standard selection dialog, but will fire the automation event [PasteSymbol](#) or [PasteGenie](#) instead.

Additionally, when hooks are enabled, pressing CTRL + SHIFT and double-clicking a Plant SCADA page will fire the event [SwapObject](#).

## Syntax

**LibSelectionHooksEnabled(*HooksEnabled*)**

*HooksEnabled*:

A setting of TRUE enables library selection hooks.

## Return Value

Enables library selection hooks, or retrieves the current library selection hooks setting.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PasteSymbol](#), [PasteGenie](#)

---

**Note:** This function is implemented in the C++ environment as two separate functions:

[put\\_LibSelectionHooksEnabled](#) enables selection hooks, and [get\\_LibSelectionHooksEnabled](#) retrieves

---

---

the current selection hooks setting.

---

## Metadata Functions

Use these Metadata functions to configure the metadata of the current object on a page.

<a href="#">PropertiesAddMetadata</a>	Adds a new Metadata entry to the properties of the current object.
<a href="#">PropertiesSelectFirstMetadata</a>	Selects the first metadata entry from the properties of the current object.
<a href="#">PropertiesSelectNextMetadata</a>	Selects the next metadata entry from the properties of the current object.
<a href="#">PropertiesSelectMetadataByName</a>	Selects the specified metadata entry in the current page.
<a href="#">PropertiesDeleteMetadata</a>	Deletes the selected metadata from the properties of the current object.
<a href="#">PropertiesMetadataName</a>	Sets or retrieves the name of the currently selected object metadata.
<a href="#">PropertiesMetadataValue</a>	Sets or retrieves the value of the currently selected object metadata.

For details and a VB example on handling return and error values, see [Error Handling](#).

## See Also

[Automation Events](#)

[Function Categories](#)

## PropertiesAddMetadata

Adds a new metadata entry to the current object properties. This function will return an error if metadata with the specified name already exists.

## Syntax

**PropertiesAddMetadata** (*Name*, *Value*)

*Name*:

The name of the new metadata entry to be added to the properties of the current object

*Value*:

The value of the new metadata to be added to the current object properties.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Example

Adding a new element and setting its properties:

```
GraphicsBuilder.PropertiesAddMetadata("MyName", "MyValue")
```

## Related Functions

[PropertiesMetadataValue](#), [PropertiesMetadataName](#), [PropertiesSelectNextMetadata](#),  
[PropertiesSelectFirstMetadata](#), [PropertiesDeleteMetadata](#), [PropertiesSelectMetadataByName](#)

## PropertiesDeleteMetadata

Deletes the selected metadata from the properties of the current object. After an item has been deleted, a call to [PropertiesSelectNextMetadata](#) will select the item immediately following the deleted item.

## Syntax

```
PropertiesDeleteMetadata()
```

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Example

Delete any metadata starting with "a":

```
Dim name As String
On Error Resume Next
Err.Clear()
GraphicsBuilder.PropertiesSelectFirstMetadata()
While (Err.Number = 0)
    name = GraphicsBuilder.PropertiesMetadataName
    If (name.ToLower().StartsWith("a")) Then
        GraphicsBuilder.PropertiesDeleteMetadata()
    End If
    GraphicsBuilder.PropertiesSelectNextMetadata()
End While
```

## Related Functions

[PropertiesMetadataValue](#), [PropertiesMetadataName](#), [PropertiesSelectNextMetadata](#),  
[PropertiesSelectFirstMetadata](#), [PropertiesAddMetadata](#), [PropertiesSelectMetadataByName](#)

## PropertiesMetadataName

Sets or retrieves the name of the currently selected object metadata.

### Syntax

*Name* = PropertiesMetadataName  
PropertiesMetadataName (*Name*)

### Return Value

The name of the currently selected metadata item (as a string). An error is returned if unsuccessful.

## Related Functions

[PropertiesMetadataValue](#), [PropertiesDeleteMetadata](#), [PropertiesSelectNextMetadata](#),  
[PropertiesSelectFirstMetadata](#), [PropertiesAddMetadata](#), [PropertiesSelectMetadataByName](#)

## PropertiesMetadataValue

Sets or retrieves the value of the currently selected object metadata.

### Syntax

*Val* = PropertiesMetadataValue  
PropertiesMetadataValue(*Def*)

### Return Value

The value of the currently selected metadata (as a string), or 0 (zero) if successfully used to set the default. An error is returned if unsuccessful.

## Related Functions

[PropertiesMetadataName](#), [PropertiesDeleteMetadata](#), [PropertiesSelectNextMetadata](#),  
[PropertiesSelectFirstMetadata](#), [PropertiesAddMetadata](#), [PropertiesSelectMetadataByName](#)

## PropertiesSelectFirstMetadata

Selects the first metadata entry from the properties of the current object.

### Syntax

**PropertiesSelectFirstMetadata()**

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Example

Determines whether the page properties of the current object has defined metadata:

```
On Error Resume Next
Err.Clear()
GraphicsBuilder.PropertiesSelectFirstMetadata()
If (Err.Number <> 0)
    ' The object has no metadata
End If
```

## Related Functions

[PropertiesMetadataValue](#), [PropertiesMetadataName](#), [PropertiesSelectNextMetadata](#), [PropertiesDeleteMetadata](#),  
[PropertiesAddMetadata](#), [PropertiesSelectMetadataByName](#)

## PropertiesSelectMetadataByName

Selects the specified metadata in the current page.

## Syntax

**PropertiesSelectMetadataByName(BSTR Name)**

*Name:*

The name of the metadata to be selected.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Example

Determining whether an metadata with a particular name exists:

```
On Error Resume Next
Err.Clear()
GraphicsBuilder.PropertiesSelectMetadataByName( "MyName" )
If (Err.Number <> 0)
    ' The metadata does not exist
End If
```

## Related Functions

[PropertiesDeleteMetadata](#)

## PropertiesSelectNextMetadata

Selects the next metadata entry from the properties of the current object.

## Syntax

```
PropertiesSelectNextMetadata()
```

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Example

Print metadata entries in the current object properties:

```
On Error Resume Next
Err.Clear()
GraphicsBuilder.PropertiesSelectFirstMetadata()
While (Err.Number = 0)
    Console.Out.WriteLine(GraphicsBuilder.PropertiesMetadataName)
    GraphicsBuilder.PropertiesSelectNextMetadata()
End While
```

## Related Functions

[PropertiesMetadataValue](#), [PropertiesSelectNextMetadata](#), [PropertiesSelectFirstMetadata](#),  
[PropertiesAddMetadata](#), [PropertiesSelectMetadataByName](#)

## Miscellaneous Functions

These functions are used for special interactions with the Graphics Builder; for example, an external drag-and-drop action could be performed by requesting the active window handle.

<a href="#">BrokenLinkCancelEnabled</a>	Writing a TRUE value enables the functions ProjectUpdatePages or PageOpen to exit and report the error E-POINTER when encountering the first broken link (missing reference) during execution.
<a href="#">ClipboardCopy</a>	Copies the selected object(s) to the Windows Clipboard.
<a href="#">ClipboardCut</a>	Cuts the selected object(s) to the Windows Clipboard.
<a href="#">ClipboardPaste</a>	Paste the elements of the Windows Clipboard on to the active page.
<a href="#">ConvertToBitmap</a>	Converts the active object to a bitmap. Unable to convert if no active object.

<a href="#">GetOffBitmap</a>	Returns Bitmap data for an object.
<a href="#">GetOnBitmap</a>	Returns Bitmap data for an object.
<a href="#">Quit</a>	Exits the Plant SCADA development environment.
<a href="#">SelectionEventEnabled</a>	Writing a true value with this function enables an event to be fired for every selection performed on a graphics page. You can also use this function to retrieve the current setting for this option.
<a href="#">UnLockObject</a>	Make an object selectable.

For details and a VB example on handling return and error values, see [Error Handling](#).

## See Also

[Automation Events](#)

[Function Categories](#)

## BrokenLinkCancelEnabled

Writing a TRUE value enables the functions ProjectUpdatePages or PageOpen to exit and report the error E-POINTER when encountering the first broken link (missing reference) during execution. If set to FALSE, these functions will succeed, but will issue a BrokenLink event for every unresolved reference on a page.

## Syntax

**BrokenLinkCancelEnabled(*CancelEnabled*)**

*CancelEnabled*:

TRUE if enabled.

## Return Value

If retrieving the current setting, TRUE or FALSE. If setting this option, 0 (zero) if successful. In both cases, an error is returned if unsuccessful.

For details and a VB example on handling return and error values, see [Error Handling](#).

## Related Functions

[BrokenLink](#), [ProjectUpdatePages](#), [PageOpen](#)

---

**Note:** This function is implemented in the C++ environment as two separate functions: `put_BrokenLinkCancelEnabled` enables or disables this option, and `get_BrokenLinkCancelEnabled` retrieves the current setting.

## ClipboardCopy

Copies the selected object(s) to the Windows clipboard.

### Syntax

**ClipboardCopy**

### Related Functions

[ClipboardCut](#), [ClipboardPaste](#)

## ClipboardCut

Cuts the selected object(s) to the Windows clipboard.

### Syntax

**ClipboardCut**

### Related Functions

[ClipboardCopy](#), [ClipboardPaste](#)

## ClipboardPaste

Paste the elements of the Windows Clipboard on to the active page.

### Syntax

**ClipboardPaste**

### Related Functions

[ClipboardCut](#), [ClipboardCopy](#)

## ConvertToBitmap

Converts the active object to a bitmap. Fails if no active object.

### Syntax

**ConvertToBitmap**

## GetOffBitmap

Returns bitmap data for an object. The data can be saved directly as the contents of a bitmap file, and includes both a BITMAPFILEHEADER and a BITMAPV4HEADER with transparency information. Object needs to be a bitmap or an error message will be raised.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject\(\)](#) and [PageSelectNextObject\(\)](#) functions, you can access your objects and change or read their properties.

The On and Off versions are provided to support flashing colors (similar to other automation interface functions). The 'Off' version will return an error if there are no flashing colors for the selected bitmap.

## Syntax

**GetOffBitmap(Bitmap)**

*Bitmap:*

Returns the bitmap as SAFEARRAY.

```
Result = GraphicsBuilder.GetOffBitmap(bitmap)
```

## Return Value

Success or failure.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## GetOnBitmap

Returns bitmap data for an object. The data can be saved directly as the contents of a bitmap file, and includes both a BITMAPFILEHEADER and a BITMAPV4HEADER with transparency information. Object needs to be a bitmap or an error message will be raised.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

The On and Off versions are provided to support flashing colors (similar to other automation interface functions)

## Syntax

**GetOnBitmap(Bitmap)**

*Bitmap:*

Returns the bitmap as SAFEARRAY.

```
Result = GraphicsBuilder.GetOnBitmap(bitmap)
```

## Return Value

Success or failure.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

## Quit

Exits the Plant SCADA development environment.

## SelectionEventEnabled

Writing a true value with this function enables an event to be fired for every selection performed on a graphics page. You can also use this function to retrieve the current setting for this option.

## Syntax

**SelectionEventEnabled**(*EventEnabled*)

*EventEnabled*:

Set to TRUE to enable selection events.

## Return Value

If retrieving the current setting, TRUE or FALSE. If setting this option, 0 (zero) if successful. In both cases, an error is returned if unsuccessful.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[Selection](#)

---

**Note:** This function is implemented in the C++ environment as two separate functions:  
`put_SelectionEventEnabled` enables or disables this option, and `get_SelectionEventEnabled` retrieves the current option setting.

## UnLockObject

Make an object selectable.

## Syntax

**UnLockObject**

## Return Value

N/A

---

**Note:** For details on handling return and error values, see [Error Handling](#).

## Example

```
Public Sub Example()
Dim gb As GraphicsBuilder.GraphicsBuilder
.
.
.
gb.PageSelectFirstObjectInGenie()
gb.PageSelectNextObjectInGenie()
gb.PageTemplateSelectFirstObject()
gb.PageTemplateSelectNextObject()
gb.UnLockObject()
End Sub
```

## Related Functions

N/A

## Object Drawing and Property Functions

With these functions, you can draw objects and manipulate the properties of objects.

---

**Note:** Freehand line drawing is not supported, as the same output can be achieved using the DrawPolygon functions.

---

Only General and 3D properties are supported. Movement, Scaling, Fill, and so on are not accessible.

The settings are applied to or read from the selected object. Typically, the last placed object is the selected object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

<a href="#">Attribute3dEffects</a>	Applies a 3D effect to an object, or retrieves the current 3D effect setting.
<a href="#">Attribute3dEffectDepth</a>	Applies a level of depth to a 3D effect, or retrieves the current depth setting.
<a href="#">AttributeAN</a>	Retrieves the animation number (AN) of the active object.
<a href="#">AttributeBaseCoordinates</a>	Returns the base coordinates of an object.
<a href="#">AttributeClass</a>	Retrieves the class of the active object as a string.
<a href="#">AttributeCornerRadius</a>	Sets or retrieves the corner radius value for the current object.
<a href="#">AttributeEllipseStyle</a>	Applies a style to an ellipse, or retrieves the current ellipse style setting.
<a href="#">AttributeEndAngle</a>	Sets the end angle of an arc or pie-slice, or retrieves the end angle.

<a href="#">AttributeExtentX</a>	Retrieves the X coordinate that represents the extent of the active object.
<a href="#">AttributeExtentY</a>	Retrieves the Y coordinate that represents the extent of the active object.
<a href="#">AttributeFillColour</a>	Sets the fill color for an object, or retrieves a value representing the current fill color.
<a href="#">AttributeFillOffColourEx</a>	Sets the fill color for an object, or retrieves a value representing the current fill color.
<a href="#">AttributeFillOnColourEx</a>	Sets the fill color for an object, or retrieves a value representing the current fill color.
<a href="#">AttributeGradientMode</a>	Sets or retrieves the direction of the gradient for the current object.
<a href="#">AttributeGradientOffColour</a>	Sets or retrieves the "Off" portion of the gradient colour for the current object.
<a href="#">AttributeGradientOnColour</a>	Sets or retrieves the "On" portion of the gradient colour for the current object.
<a href="#">AttributeHiLightColour</a>	Sets the highlight color applied to the 3D effects raised, lowered or embossed, or retrieves the current highlight color setting.
<a href="#">AttributeLineColour</a>	Applies a color to a line, or retrieves the current color setting. This function has been replaced by the functions AttributeLineOnColourEx and AttributeLineOffColourEx.
<a href="#">AttributeLineOnColourEx</a>	This function supports True Color functionality and replaces AttributeLineColour.
<a href="#">AttributeLineOffColourEx</a>	This function supports True Color functionality and replaces AttributeLineColour.
<a href="#">AttributeLineStyle</a>	Applies a style to a line, or retrieves the current style setting.
<a href="#">AttributeLineWidth</a>	Sets the width of a line, or retrieves its current width.
<a href="#">AttributeLoLightColour</a>	Sets the lowlight color applied to the 3D effects raised, lowered or embossed, or retrieves the current lowlight color setting. As this function does not support True Colour functionality, it has been superseded by the functions AttributeLoLightOffColourEx and AttributeLoLightOnColourEx.

<a href="#">AttributeLoLightOffColourEx</a>	Sets the lowlight "off" color applied to the 3D effects raised, lowered or embossed, or retrieves the current lowlight color setting.
<a href="#">AttributeLoLightOnColourEx</a>	Sets the lowlight "on" color applied to the 3D effects raised, lowered or embossed, or retrieves the current lowlight color setting.
<a href="#">AttributeNodeCoordinatesFirst</a>	Returns the coordinates of the first node of a free hand line, polygon or pipe.
<a href="#">AttributeNodeCoordinatesNext</a>	Returns the coordinates of any following nodes of a free hand line, polygon or pipe when implemented after AttributeNodeCoordinatesFirst.
<a href="#">AttributePolygonOpen</a>	Defines whether a polygon (polyline) is set to open mode (i.e. its two end points are not joined) or closed (its two ends are joined).
<a href="#">AttributeRectangleStyle</a>	Sets the rectangle style, or retrieves the rectangle style setting.
<a href="#">AttributeSetFill</a>	Displays the object as filled, or retrieves the current fill value.
<a href="#">AttributeShadowColour</a>	Sets the shadow color when a shadowed 3D effect is used, or retrieves the current shadow color setting. As this function does not support True Color functionality, it has been superseded by the functions AttributeShadowOffColourEx and AttributeShadowOnColourEx.
<a href="#">AttributeShadowOffColourEx</a>	Sets the "off" shadow color when a shadowed 3D effect is used, or retrieves the current shadow color setting.
<a href="#">AttributeShadowOnColourEx</a>	Sets the "on" shadow color when a shadowed 3D effect is used, or retrieves the current shadow color setting.
<a href="#">AttributeStartAngle</a>	Sets the start angle of an arc or pie-slice, or retrieves the start angle.
<a href="#">AttributeTransformationMatrixGet</a>	Reads the elements of the transformation matrix.
<a href="#">AttributeTransformationMatrixPut</a>	Sets the elements of the transformation matrix.
<a href="#">AttributeX</a>	Retrieves the X coordinate of the active object.
<a href="#">AttributeY</a>	Retrieves the Y coordinate of the active object.

<a href="#">DrawButton</a>	Draws a button on the active page.
<a href="#">DrawCicodeObject</a>	Places a Cicode object on the page at the specified location.
<a href="#">DrawEllipse</a>	Draws an ellipse on the active page.
<a href="#">DrawLine</a>	Draws a line on the active page.
<a href="#">DrawNumber</a>	Places a number object on the page at the specified location.
<a href="#">DrawPipeEnd</a>	Terminates the drawing of a pipe on the active page.
<a href="#">DrawPipeSection</a>	Draws a section of pipe on the active page.
<a href="#">DrawPipeStart</a>	Initiates the process of drawing a pipe on the active page by defining a starting point that DrawPipeSection() can be applied to.
<a href="#">DrawPolygonEnd</a>	Terminates the drawing of a polygon on the active page.
<a href="#">DrawPolygonLine</a>	Draws a line on the active page that forms part of a polygon.
<a href="#">DrawPolygonStart</a>	Initiates the process of drawing a polygon on the active page by defining a starting point that DrawPolygonLine() can be applied to.
<a href="#">DrawRectangle</a>	Draws a rectangle on the active page.
<a href="#">DrawSymbolSet</a>	Places a Symbol Set object on the page at the specified location.
<a href="#">DrawText</a>	Draws an alphanumeric string at the specified location.
<a href="#">DrawTrend</a>	Draws a trend object on the active page.

For details and a VB example on handling return and error values, see [Error Handling](#).

## See Also

[Automation Events](#)  
[Function Categories](#)

## Attribute3dEffects

Applies a 3D effect to an object, or retrieves the current 3D effect setting.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**Attribute3dEffects(*Effects*)**

*Effects*:

A value between 0 and 4 representing the 3D effect type.

0 = none

1 = raised

2 = lowered

3 = shadowed

4 = embossed

## Return Value

If retrieving the current 3D effect setting, a value between 0 and 4 representing the effect type. If applying a 3D effect, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report with the error E\_INVALIDARG. If there is no active object, they will exit with a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[Attribute3dEffectDepth](#), [AttributeShadowColour](#), [AttributeHiLightColour](#), [AttributeLoLightColour](#)

## Example

```
' Applies a 3D effect (embossed) to an object
GraphicsBuilder.Attribute3dEffects = 4
' Retrieves the current 3D effect applied to an object
MyVariable = GraphicsBuilder.Attribute3dEffects
```

**Note:** This function is implemented in the C++ environment as two separate functions: put\_Attribute3dEffect applies a 3D effect, and get\_Attribute3dEffect retrieves the current 3D effect setting.

## Attribute3dEffectDepth

Applies a level of depth to a 3D effect, or retrieves the current depth setting.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**Attribute3dEffectDepth(*EffectDepth*)**

*EffectDepth*:

A value between 0 and 32 representing the depth of the 3D effect used.

## Return Value

If retrieving the current depth setting for a 3D effect, a value between 0 and 32. If applying depth to a 3D effect, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit with a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[Attribute3dEffects](#), [AttributeShadowColour](#), [AttributeHiLightColour](#), [AttributeLoLightColour](#)

## Example

```
' Applies depth to a 3D effect for the current object
GraphicsBuilder.Attribute3dEffectDepth = 28

' Retrieves the 3D depth for the current object
MyVariable = GraphicsBuilder.Attribute3dEffectDepth
```

**Note:** This function is implemented in the C++ environment as two separate functions:

`put_Attribute3dEffectDepth` applies depth to 3D effect, and `get_Attribute3dEffectDepth` retrieves the current 3D depth setting.

## AttributeAN

Retrieves the animation number (AN) of the active object. This is a read only function.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeAN(*AN*)**

*AN*:

A value between 0 and 65536.

## Return Value

A value between 0 and 65536. If values are out of range on writing to the attribute, the function will exit and

report the error E\_INVALIDARG. If there is no active object, they will exit with a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AttributeX](#), [AttributeY](#)

## Example

```
' Retrieves the AN for the current object
MyVariable = GraphicsBuilder.AttributeAN
```

## AttributeBaseCoordinates

Returns the base coordinates of an object. If you use these coordinates, also apply the transformation matrix.

Refer to functions [AttributeTransformationMatrixPut](#) and [AttributeTransformationMatrixGet](#).

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeBaseCoordinates**(*FromXPosition*, *FromYPosition*, *ToXPosition*, *ToYPosition*)

*FromXPosition*:

Distance from the left hand side of the page to top left hand corner of the object, measured in pixels.

*FromYPosition*:

Distance from the top of the page to the top left hand corner of the object, measured in pixels.

*ToXPosition*:

Distance from the left hand side of the page to the bottom right hand corner of the object, measured in pixels.

*ToYPosition*:

Distance from the top of the page to the bottom right hand corner of the object, measured in pixels.

## Return Value

The base coordinates of the current object. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit with a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AttributeTransformationMatrixPut](#), [AttributeTransformationMatrixGet](#)

## AttributeClass

Retrieves the class of the active object as a string. This is a read only function.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeClass(*Class*)**

*Class*:

A string depicting the class of the object. The class options include: "Draw", "Line", "Square", "Circle", "Polyline", "Pipe", "Text", "Button", "Set", "Trend", "Advanced Animation", "Bitmap", "Group", "ActiveX", "Symbol" and "Genie".

## Return Value

A string depicting the class of the object. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit with a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Example

```
' Retrieves the Class for the current object
MyVariable = GraphicsBuilder.AttributeClass
```

## AttributeCornerRadius

Sets or retrieves the corner radius value on the **General | Appearance** tab of the Object Properties dialog for the current object. This is only supported on rectangle objects.

## Syntax

**AttributeCornerRadius(*nRadius*)**

*nRadius*:

Defines the radius of the corner. Values from 0- 32 pixels are permitted.

## Return Value

If retrieving the current corner radius, a value between 0 and 32. If applying a corner radius, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E\_INVALIDARG. If there is no active object, they exit with a return value of E\_HANDLE.

## AttributeEllipseStyle

Applies a style to an ellipse, or retrieves the current ellipse style setting.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeEllipseStyle(*EllipseStyle*)**

*EllipseStyle*:

A value representing the current ellipse style.

0 = normal ellipse

1 = pie slice

2 = arc

## Return Value

If retrieving the current ellipse style setting, a value between 0 and 2 representing one of three style options. If applying a style setting, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit with a return value of E\_HANDLE.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Example

```
' Applies a style (arc) to an ellipse
GraphicsBuilder.AttributeEllipseStyle = 2

' Retrieves a value representing the style applied to an ellipse
MyVariable = GraphicsBuilder.AttributeEllipseStyle
```

---

**Note:** This function is implemented in the C++ environment as two separate functions:

`put_AttributeEllipseStyle` applies a style to an ellipse, and `get_AttributeEllipseStyle` retrieves the current ellipse style setting.

---

## AttributeEndAngle

Sets the end angle of an arc or pie-slice, or retrieves the end angle.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeEndAngle(*Angle*)**

**Angle:**

A value between 0 and 360 representing the end angle (in degrees).

## Return Value

If retrieving the end angle, a value between 0 and 360. If applying an end angle, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit and report a return value of E\_HANDLE.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AttributeStartAngle](#)

## Example

```
' Sets the end angle of an arc
GraphicsBuilder.AttributeEndAngle = 45

' Retrieves the start angle for an arc
MyVariable = GraphicsBuilder.AttributeEndAngle
```

---

**Note:** This function is implemented in the C++ environment as two separate functions: put\_AttributeEndAngle applies an end angle setting, and get\_AttributeEndAngle retrieves the current end angle setting.

---

## AttributeExtentX

Retrieves the X coordinate that represents the extent of the active object. For example, if the active object were a line, it would be the end coordinate. This is a read only function.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeExtentX(*XPosition*)**

***XPosition:***

A value between 0 and 65536.

## Return Value

A value between 0 and 65536. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit and report a return value of E\_HANDLE.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AttributeExtentY](#), [AttributeAN](#)

## Example

```
' Retrieves the X coordinate for the extent of the current object
MyVariable = GraphicsBuilder.AttributeExtentX
```

### AttributeExtentY

Retrieves the Y coordinate that represents the extent of the active object. For example, if the active object were a line, it would be the end coordinate. This is a read only function.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeExtentY(YPosition)**

*YPosition:*

A value between 0 and 65536

## Return Value

A value between 0 and 65536. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit and report a return value of E\_HANDLE.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AttributeExtentX](#), [AttributeAN](#)

## Example

```
' Retrieves the Y coordinate for the extent of the current object
MyVariable = GraphicsBuilder.AttributeExtentY
```

### AttributeFillColour

Sets the fill color for an object, or retrieves a value representing the current fill color.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

---

**Note:** As this function does not support True Color functionality, it has been superseded by the functions [AttributeFillOffColourEx](#) and [AttributeFillOnColourEx](#).

---

## Syntax

**AttributeFillColour(*FillColour*)**

*FillColour:*

A value between 0 and 255 representing the fill color.

## Return Value

If retrieving the current fill color, a value between 0 and 255. If applying a fill color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit and report a return value of E\_HANDLE.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AttributeSetFill](#)

## Example

```
' Sets the fill color for an object
GraphicsBuilder.AttributeFillColour = 125

' Retrieves the value of the fill color
MyVariable = GraphicsBuilder.AttributeFillColour
```

---

**Note:** This function is implemented in the C++ environment as two separate functions:  
put\_AttributeFillColour applies a fill color, and get\_AttributeFillColour retrieves the current fill color setting.

---

## AttributeFillOffColourEx

Sets the fill color for an object, or retrieves a value representing the current fill color.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects, and change or read their properties.

## Syntax

**AttributeFillOffColourEx(*FillColour*)**

*FillColour:*

An RGB value.

## Return Value

If retrieving the current fill color, an RGB value. If applying a fill color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit and report a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AttributeSetFill](#)

## Example

```
' Sets the fill color for an object
GraphicsBuilder.AttributeFillOffColourEx = &hFF0000

' Retrieves the value of the fill color
MyVariable = GraphicsBuilder.AttributeFillOffColourEx
```

**Note:** This function is implemented in the C++ environment as two separate functions:

`put_AttributeFillOffColourEx` applies a fill color, and `get_AttributeFillOffColourEx` retrieves the current fill color setting.

## AttributeFillOnColourEx

Sets the fill color for an object, or retrieves a value representing the current fill color.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeFillOnColourEx(*FillColour*)**

*FillColour:*

An RGB value.

## Return Value

If retrieving the current fill color, an RGB value. If applying a fill color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E\_INVALIDARG. If there is no active object, they exit and report a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AttributeSetFill](#)

## Example

```
' Sets the fill color for an object
GraphicsBuilder.AttributeFillOnColourEx = &hFF0000

' Retrieves the value of the fill color
MyVariable = GraphicsBuilder.AttributeFillOnColourEx
```

**Note:** This function is implemented in the C++ environment as two separate functions:

`put_AttributeFillOnColourEx` applies a fill color, and `get_AttributeFillOnColourEx` retrieves the current fill color setting.

## AttributeGradientMode

Sets or retrieves the direction of the gradient on the **General | Appearance** tab of the Object Properties dialog for the current object.

This function is only supported on rectangle objects.

## Syntax

**AttributeGradientMode(*Mode*)**

*Mode:*

Direction of the gradient:

- 0 - Off
- 1 - Left To Right
- 2 - Right To Left
- 3 - Top To Bottom
- 4 - Bottom To Top
- 5 - Horizontal Edge To Middle
- 6 - Middle To Horizontal Edge
- 7 - Vertical Edge To Middle
- 8 - Middle To Vertical Edge

## Return Value

If retrieving the gradient mode, the direction of the gradient as specified above. If applying a gradient mode, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error `E_INVALIDARG`. If there is no active object, they will exit and report a return value of `E_HANDLE`.

## AttributeGradientOffColour

Sets or retrieves the "Off" portion of the gradient color on the **General | Appearance** tab of the Object Properties dialog for the current object.

This is only supported on rectangle objects.

## Syntax

**AttributeGradientOffColour(***Color***)**

*Color*:

Off portion of the gradient color.

## Return Value

If retrieving the gradient color, an RGB encoded color. If applying a gradient color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E\_INVALIDARG. If there is no active object, they exit and report a return value of E\_HANDLE.

## AttributeGradientOnColour

Sets or retrieves the "On" portion of the gradient color on the **General | Appearance** tab of the Object Properties dialog for the current object.

This function is only supported on rectangle objects.

## Syntax

**AttributeGradientOnColour(***Color***)**

*Color*:

On portion of the gradient color.

## Return Value

If retrieving the gradient color, an RGB encoded color. If applying a gradient color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit and report a return value of E\_HANDLE.

## AttributeHiLightColour

Sets the highlight color applied to the 3D effects raised, lowered or embossed, or retrieves the current highlight color setting.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access objects and change or read their properties.

---

**Note:** As this function does not support True Color functionality, it has been superseded by the functions [AttributeHiLightOnColourEx](#) and [AttributeHiLightOffColourEx](#).

---

## Syntax

**AttributeHiLightColour(*HiLightColour*)**

*HiLightColour*:

A value between 0 and 255 representing the highlight color.

## Return Value

If retrieving the current highlight color setting, a value between 0 and 255. If applying a highlight color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit and report a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[Attribute3dEffects](#), [Attribute3dEffectDepth](#), [AttributeShadowColour](#), [AttributeLoLightColour](#)

## Example

```
' Applies a highlight color to a 3D effect
GraphicsBuilder.AttributeHiLightColour = 125

' Retrieves a value representing a 3D effect's highlight color
MyVariable = GraphicsBuilder.AttributeHiLightColour
```

**Note:** This function is implemented in the C++ environment as two separate functions:  
`put_AttributeHiLightColour` applies a highlight color setting, and `get_AttributeHiLightColour` retrieves the current highlight color setting.

## AttributeHiLightOffColourEx

Sets the highlight color applied to the 3D effects raised, lowered or embossed, or retrieves the current highlight color setting.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject\(\)](#) and [PageSelectNextObject\(\)](#) functions, you can access your objects, and change or read their properties.

## Syntax

**AttributeHiLightOffColourEx(*HiLightColour*)**

*HiLightColour*:

An RGB value.

## Return Value

If retrieving the current highlight color setting, an RGB value. If applying a highlight color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E\_INVALIDARG. If there is no active object, they exit and report a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[Attribute3dEffects](#), [Attribute3dEffectDepth](#), [AttributeShadowOffColourEx](#), [AttributeShadowOnColourEx](#),  
[AttributeLoLightOffColourEx](#), [AttributeLoLightOnColourEx](#)

## Example

```
' Applies a highlight color to a 3D effect
GraphicsBuilder.AttributeHiLightOffColourEx = &hFF0000

' Retrieves a value representing a 3D effect's highlight color
MyVariable = GraphicsBuilder.AttributeHiLightOffColourEx
```

**Note:** This function is implemented in the C++ environment as two separate functions:

`put_AttributeHiLightOffColourEx` applies a highlight color setting, and

`get_AttributeHiLightOffColourEx` retrieves the current highlight color setting.

## AttributeHiLightOnColourEx

Sets the highlight color applied to the 3D effects raised, lowered or embossed, or retrieves the current highlight color setting.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects, and change or read their properties.

## Syntax

**AttributeHiLightOnColourEx(HiLightColour)**

*HiLightColour:*

An RGB value.

## Return Value

If retrieving the current highlight color setting, an RGB value. If applying a highlight color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E\_INVALIDARG. If there is no active object, they exit and report a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[Attribute3dEffects](#), [Attribute3dEffectDepth](#), [AttributeShadowOffColourEx](#), [AttributeShadowOnColourEx](#),  
[AttributeLoLightOffColourEx](#), [AttributeLoLightOnColourEx](#)

## Example

```
' Applies a highlight color to a 3D effect
GraphicsBuilder.AttributeHiLightOnColourEx = &hFF0000

' Retrieves a value representing a 3D effect's highlight color
MyVariable = GraphicsBuilder.AttributeHiLightOnColourEx
```

**Note:** This function is implemented in the C++ environment as two separate functions:

`put_AttributeHiLightOnColourEx` applies a highlight color setting, and `get_AttributeHiLightOnColourEx` retrieves the current highlight color setting.

## AttributeLineColour

Applies a color to a line, or retrieves the current color setting.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

**Note:** This function, as it does not support True Color functionality, has been superseded by the functions [AttributeLineOnColourEx](#) and [AttributeLineOffColourEx](#).

## Syntax

**AttributeLineColour**(*LineColour*)

*LineColour*:

A value between 0 and 255 representing a particular color.

## Return Value

If retrieving the current line color, a value between 0 and 255. If setting the line color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit and report a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AttributeLineWidth](#), [AttributeLineStyle](#)

## Example

```
' Applies a color to the current line
GraphicsBuilder.AttributeLineColour = 125

' Retrieves the value of the color applied to the current line
MyVariable = GraphicsBuilder.AttributeLineColour
```

**Note:** This function is implemented in the C++ environment as two separate functions:

`put_AttributeLineColour` applies a particular color to a line, and `get_AttributeLineColour` retrieves the current color setting.

## AttributeLineOffColourEx

Applies a color to a line, or retrieves the current "off" color setting. The function uses RGB colors for each state of a color instead of a palette index.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects, and change or read their properties.

## Syntax

**AttributeLineOffColourEx**(*LineColour*)

*LineColour:*

An RGB value.

## Return Value

If retrieving the current line color, an RGB value. If setting the line color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error `E_INVALIDARG`. If there is no active object, they exit and report a return value of `E_HANDLE`.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AttributeLineWidth](#), [AttributeLineStyle](#)

## Example

```
' Applies a color to the current line
GraphicsBuilder.AttributeLineOffColourEx = &hFF0000

' Retrieves the value of the color applied to the current line
MyVariable = GraphicsBuilder.AttributeLineOffColourEx
```

**Note:** This function is implemented in the C++ environment as two separate functions:

`put_AttributeLineOffColourEx` applies a particular color to a line, and `get_AttributeLineOffColourEx` retrieves the current color setting.

## AttributeLineOnColourEx

Applies a color to a line, or retrieves the current "on" color setting. The function uses RGB colors for each state of a color instead of a palette index.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects, and change or read their properties.

## Syntax

**AttributeLineOnColourEx**(*LineColour*)

*LineColour*:

An RGB value.

## Return Value

If retrieving the current line color, an RGB value. If setting the line color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E\_INVALIDARG. If there is no active object, they exit and report a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AttributeLineWidth](#), [AttributeLineStyle](#)

## Example

```
' Applies a color to the current line
GraphicsBuilder.AttributeLineOnColourEx = &hFF0000

' Retrieves the value of the color applied to the current line
MyVariable = GraphicsBuilder.AttributeLineOnColourEx
```

**Note:** This function is implemented in the C++ environment as two separate functions:

`put_AttributeLineOnColourEx` applies a particular color to a line, and `get_AttributeLineOnColourEx` retrieves the current color setting.

## AttributeLineStyle

Applies a style to a line, or retrieves the current style setting. You can only apply a line style if the line width is set to 1.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeLineStyle(*LineStyle*)**

*LineStyle*:

A value between 0 and 4 representing the style applied to a line. Line style only works if line width is set to 1.

0 = solid

1 = dashed

2 = dot

3 = dash dot

4 = dash dot dot

## Return Value

If retrieving the current line style, a value between 0 and 4 that represents a particular style. If setting the line style, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit and report a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AttributeLineWidth](#), [AttributeLineColour](#)

## Example

```
' Applies a style (dash dot) to the current line
GraphicsBuilder.AttributeLineStyle = 3

' Retrieves the style applied to the current line
MyVariable = GraphicsBuilder.AttributeLineStyle
```

**Note:** This function is implemented in the C++ environment as two separate functions:

`put_AttributeLineStyle` applies a particular line style, and `get_AttributeLineStyle` retrieves the current style setting.

## AttributeLineWidth

Sets the width of a line, or retrieves its current width.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeLineWidth(*LineWidth*)**

**LineWidth:**

A value between 0 and 32 representing the line width in pixels.

## Return Value

If retrieving the current width of a line, a value between 1 and 32 (representing pixels) is returned. If setting the line width, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit and report a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AttributeLineStyle](#), [AttributeLoLightColour](#)

## Example

```
' Sets the width for the current line
GraphicsBuilder.AttributeLineWidth = 1

' Retrieves the width of the current line
MyVariable = GraphicsBuilder.AttributeLineWidth
```

**Note:** This function is implemented in the C++ environment as two separate functions:  
`put_AttributeLineWidth` sets the value for the width of a line, and `get_AttributeLineWidth` retrieves the current line width setting.

## AttributeLoLightColour

Sets the lowlight color applied to the 3D effects raised, lowered or embossed, or retrieves the current lowlight color setting.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeLoLightColour(LoLightColour)**

*LoLightColour:*

A value between 0 and 255 representing the lowlight color.

## Return Value

If retrieving the current lowlight color setting, a value between 0 and 255. If applying a lowlight color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit and

report a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[Attribute3dEffects](#), [Attribute3dEffectDepth](#), [AttributeShadowColour](#), [AttributeHiLightColour](#)

## Example

```
' Applies a lowlight color to a 3D effect
GraphicsBuilder.AttributeLoLightColour = 45

' Retrieves a value representing a 3D effect's lowlight color
MyVariable = GraphicsBuilder.AttributeLoLightColour
```

**Note:** This function is implemented in the C++ environment as two separate functions:

`put_AttributeLoLightColour` applies a lowlight color setting, and `get_AttributeLoLightColour` retrieves the current lowlight color setting.

## AttributeLoLightOffColourEx

Sets the lowlight color applied to the 3D effects raised, lowered or embossed, or retrieves the current lowlight color setting.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeLoLightOffColourEx(LoLightColour)**

*LoLightColour:*

An RGB value.

## Return Value

If retrieving the current lowlight color setting, an RGB value. If applying a lowlight color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E\_INVALIDARG. If there is no active object, they exit and report a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[Attribute3dEffects](#), [Attribute3dEffectDepth](#), [AttributeShadowOffColourEx](#), [AttributeShadowOnColourEx](#),  
[AttributeHiLightOffColourEx](#), [AttributeHiLightOnColourEx](#)

## Example

```
' Applies a lowlight color to a 3D effect
GraphicsBuilder.AttributeLoLightOffColourEx = &hFF0000

' Retrieves a value representing a 3D effect's lowlight color
MyVariable = GraphicsBuilder.AttributeLoLightOffColourEx
```

**Note:** This function is implemented in the C++ environment as two separate functions:

`put_AttributeLoLightOffColourEx` applies a lowlight color setting, and `get_AttributeLoLightOffColourEx` retrieves the current lowlight color setting.

## AttributeLoLightOnColourEx

Sets the lowlight color applied to the 3D effects raised, lowered or embossed, or retrieves the current lowlight color setting.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeLoLightOnColourEx**(*LoLightColour*)

*LoLightColour*:

An RGB value.

## Return Value

If retrieving the current lowlight color setting, an RGB value. If applying a lowlight color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E\_INVALIDARG. If there is no active object, they exit and report a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[Attribute3dEffects](#), [Attribute3dEffectDepth](#), [AttributeShadowOffColourEx](#), [AttributeShadowOnColourEx](#),  
[AttributeHiLightOffColourEx](#), [AttributeHiLightOnColourEx](#)

## Example

```
' Applies a lowlight color to a 3D effect
GraphicsBuilder.AttributeLoLightOnColourEx = &hFF0000

' Retrieves a value representing a 3D effect's lowlight color
MyVariable = GraphicsBuilder.AttributeLoLightOnColourEx
```

**Note:** This function is implemented in the C++ environment as two separate functions:

---

`put_AttributeLoLightOnColourEx` applies a lowlight color setting, and `get_AttributeLoLightOnColourEx` retrieves the current lowlight color setting.

---

## AttributeNodeCoordinatesFirst

Returns the coordinates of the first node of a free hand line, polygon or pipe.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeNodeCoordinatesFirst(*XPosition*, *YPosition*)**

*XPosition*:

Distance from the left-hand side of the page to the first node of an object, measured in pixels.

*YPosition*:

Distance from the top of the page to the first node of an object, measured in pixels.

## Return Value

The coordinates of the current object's first node. If values are out of range on writing to the attribute, the function will exit and report the error `E_INVALIDARG`. If there is no active object, they will exit and report a return value of `E_HANDLE`.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AttributeNodeCoordinatesNext](#)

## AttributeNodeCoordinatesNext

Returns the coordinates of any following nodes of a free hand line, polygon or pipe when implemented after `AttributeNodeCoordinatesFirst`.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeNodeCoordinatesNext(*XPosition*, *YPosition*)**

*XPosition*:

Distance from the left-hand side of the page to the first node of an object, measured in pixels.

*YPosition*:

Distance from the top of the page to the first node of an object, measured in pixels.

## Return Value

The coordinates of the object's following nodes, or E\_ABORT if no more nodes are left. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit and report a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AttributeNodeCoordinatesFirst](#)

### AttributePolygonOpen

Defines whether a polygon (polyline) is set to open mode (that is, its two end points are not joined) or closed (its two ends are joined). It can also be used to retrieve the current open mode setting.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributePolygonOpen(OpenClose)**

*OpenClose:*

TRUE = Polygon is drawn in open mode; FALSE = Polygon is drawn in closed mode.

## Return Value

If retrieving the current open mode setting for a polygon, TRUE or FALSE is returned. If setting the open mode, 0 (zero) is returned if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit and report a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Example

```
' Sets a polygon to Open mode
GraphicsBuilder.AttributePolygonOpen = TRUE

' Determines if the current polygon is defined as Open
MyVariable = GraphicsBuilder.AttributePolygonOpen
```

**Note:** This function is implemented in the C++ environment as two separate functions:

`put_AttributePolygonOpen` sets the open mode for a polygon, and `get_AttributePolygonOpen` retrieves the current open mode setting.

## AttributeRectangleStyle

Sets the rectangle style, or retrieves the rectangle style setting.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeRectangleStyle(Style)**

*Style:*

0 = none

1 = border

2 = extra line

3 = border and an extra line

## Return Value

If retrieving the current rectangle style setting, a value between 0 and 3. If applying a rectangle style, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit and report a return value of E\_HANDLE.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[Attribute3dEffects](#), [Attribute3dEffectDepth](#), [AttributeShadowColour](#), [AttributeHiLightColour](#)

## Example

```
' Applies a style to a rectangle
GraphicsBuilder.AttributeRectangleStyle = 1

' Retrieves a value representing a rectangle style
MyVariable = GraphicsBuilder.AttributeRectangleStyle
```

---

**Note:** This function is implemented in the C++ environment as two separate functions:

`put_AttributeRectangleStyle` applies a rectangle style, and `get_AttributeRectangleStyle` retrieves the current rectangle style setting.

## AttributeSetFill

Displays the object as filled, or retrieves the current fill value.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeSetFill(*SetFill*)**

*SetFill*:

TRUE if the object drawn filled.

## Return Value

If retrieving the current fill setting, TRUE if the object is displayed as filled. If applying a fill setting, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit and report a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AttributeFillColour](#)

## Example

```
' Displays an object as filled
GraphicsBuilder.AttributeSetFill = TRUE

' Retrieves the current fill setting
MyVariable = GraphicsBuilder.AttributeSetFill
```

**Note:** This function is implemented in the C++ environment as two separate functions: put\_AttributeSetFill applies a fill setting, and get\_AttributeSetFill retrieves the current fill setting.

## AttributeShadowColour

Sets the shadow color when a shadowed 3D effect is used, or retrieves the current shadow color setting.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

**Note:** As this function does not support True Color functionality, it has been superseded by the functions [AttributeShadowOffColourEx](#) and [AttributeShadowOnColourEx](#).

## Syntax

**AttributeShadowColour(*ShadowColour*)**

*ShadowColour*:

A value between 0 and 255 representing the shadow color.

## Return Value

If retrieving the current shadow color setting, a value between 0 and 255. If applying a shadow color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit and report a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[Attribute3dEffects](#), [Attribute3dEffectDepth](#), [AttributeHiLightColour](#), [AttributeLoLightColour](#)

## Example

```
' Applies a shadow color to a shadowed 3D effect
GraphicsBuilder.AttributeShadowColour = 125

' Retrieves the current shadow color
MyVariable = GraphicsBuilder.AttributeShadowColour
```

**Note:** This function is implemented in the C++ environment as two separate functions:

`put_AttributeShadowColour` applies a shadow color setting, and `get_AttributeShadowColour` retrieves the current shadow color setting.

## AttributeShadowOffColourEx

Sets the shadow color when a shadowed 3D effect is used, or retrieves the current shadow color setting.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeShadowOffColourEx(ShadowColour)**

*ShadowColour:*

An RGB value.

## Return Value

If retrieving the current shadow color setting, an RGB value. If applying a shadow color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E\_INVALIDARG. If there is no active object, they exit and report a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[Attribute3dEffects](#), [Attribute3dEffectDepth](#), [AttributeHiLightOffColourEx](#), [AttributeHiLightOnColourEx](#),  
[AttributeLoLightOffColourEx](#), [AttributeLoLightOnColourEx](#)

## Example

```
' Applies a shadow color to a shadowed 3D effect
GraphicsBuilder.AttributeShadowOffColourEx = &hFF0000

' Retrieves the current shadow color
MyVariable = GraphicsBuilder.AttributeShadowOffColourEx
```

This function is implemented in the C++ environment as two separate functions:

`put_AttributeShadowOffColourEx` applies a shadow color setting, and `get_AttributeShadowOffColourEx` retrieves the current shadow color setting.

## AttributeShadowOnColourEx

Sets the shadow color when a shadowed 3D effect is used, or retrieves the current shadow color setting.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects, and change or read their properties.

## Syntax

**AttributeShadowOnColourEx(*ShadowColour*)**

*ShadowColour*:

An RGB value.

## Return Value

If retrieving the current shadow color setting, an RGB value. If applying a shadow color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error `E_INVALIDARG`. If there is no active object, they will exit and report a return value of `E_HANDLE`.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[Attribute3dEffects](#), [Attribute3dEffectDepth](#), [AttributeHiLightOffColourEx](#), [AttributeHiLightOnColourEx](#),  
[AttributeLoLightOffColourEx](#), [AttributeLoLightOnColourEx](#)

## Example

```
' Applies a shadow color to a shadowed 3D effect
GraphicsBuilder.AttributeShadowOnColourEx = &hFF0000
```

```
' Retrieves the current shadow color
MyVariable = GraphicsBuilder.AttributeShadowOnColourEx
```

**Note:** This function is implemented in the C++ environment as two separate functions:  
`put_AttributeShadowOnColourEx` applies a shadow color setting, and `get_AttributeShadowOnColourEx` retrieves the current shadow color setting.

## AttributeStartAngle

Sets the start angle of an arc or pie-slice, or retrieves the start angle.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeStartAngle(*Angle*)**

*Angle*:

A value between 0 and 360 representing the start angle (in degrees).

If retrieving the start angle, a value between 0 and 360. If applying a start angle, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error `E_INVALIDARG`. If there is no active object, they will exit and report a return value of `E_HANDLE`.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AttributeEndAngle](#)

## Example

```
' Sets the start angle of an arc
GraphicsBuilder.AttributeStartAngle = 45

' Retrieves the start angle for an arc
MyVariable = GraphicsBuilder.AttributeStartAngle
```

**Note:** This function is implemented in the C++ environment as two separate functions:  
`put_AttributeStartAngle` applies a start angle setting, and `get_AttributeStartAngle` retrieves the current start angle setting.

## AttributeTransformationMatrixGet

Reads the elements of the transformation matrix. If A and D are both 1, and others are 0, the object is not transformed (identity matrix).

This function applies to the selected object, which is typically the last placed object. By using the

[PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeTransformationMatrixGet(*A, B, C, D, H, K*)**

*A*:

Element A of the transformation matrix

*B*:

Element A of the transformation matrix

*C*:

Element A of the transformation matrix

*D*:

Element A of the transformation matrix

*H*:

Element A of the transformation matrix

*K*:

Element A of the transformation matrix

## Return Value

The elements of the transformation matrix. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit and report a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AttributeTransformationMatrixPut](#)

### AttributeTransformationMatrixPut

Sets the elements of the transformation matrix.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeTransformationMatrixPut(*A, B, C, D, H, K*)**

*A*:

Element A of the transformation matrix

*B:*

Element A of the transformation matrix

*C:*

Element A of the transformation matrix

*D:*

Element A of the transformation matrix

*H:*

Element A of the transformation matrix

*K:*

Element A of the transformation matrix

## Return Value

0 (zero) if successful, otherwise an error is returned. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit and report a return value of E\_HANDLE.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AttributeTransformationMatrixGet](#)

### AttributeX

Retrieves the X coordinate of the active object. This is a read only function.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeX(*XPosition*)**

*XPosition:*

A value between 0 and 65536.

## Return Value

A value between 0 and 65536. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit and report a return value of E\_HANDLE.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AttributeY](#), [AttributeAN](#)

## Example

```
' Retrieves the X coordinate for the current object
MyVariable = GraphicsBuilder.AttributeX
```

## AttributeY

Retrieves the Y coordinate of the active object. This is a read-only function.

This function applies to the selected object, which is typically the last placed object. By using the [PageSelectFirstObject](#) and [PageSelectNextObject](#) functions, you can access your objects and change or read their properties.

## Syntax

**AttributeY(YPosition)**

*YPosition:*

A value between 0 and 65536.

## Return Value

A value between 0 and 65536. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active object, they will exit and report a return value of E\_HANDLE.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AttributeX](#), [AttributeAN](#)

## Example

```
' Retrieves the Y coordinate for the current object
MyVariable = GraphicsBuilder.AttributeY
```

## DrawButton

Draws a button on the active page.

## Syntax

**DrawButton(FromXPosition, FromYPosition, ToXPosition, ToYPosition)**

**FromXPosition:**

Distance from the left hand side of the page to top left hand corner of the button to be drawn, measured in pixels.

**FromYPosition:**

Distance from the top of the page to the top left hand corner of the button to be drawn, measured in pixels.

**ToXPosition:**

Distance from the left hand side of the page to the bottom right hand corner of the button to be drawn, measured in pixels.

**ToYPosition:**

Distance from the top of the page to the bottom right hand corner of the button to be drawn, measured in pixels.

## Return Value

0 (zero) if successful; otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[DrawLine](#), [DrawEllipse](#), [DrawRectangle](#), [DrawPolygonStart](#), [DrawPolygonLine](#), [DrawPolygonEnd](#), [DrawPipeStart](#), [DrawPipeSection](#), [DrawPipeEnd](#), [DrawText](#), [DrawNumber](#), [DrawSymbolSet](#), [DrawTrend](#), [DrawCicodeObject](#)

## Example

```
GraphicsBuilder.DrawButton 50, 70, 400, 200
```

## DrawCicodeObject

Places a Cicode object on the page at the specified location.

## Syntax

**DrawCicodeObject(XPosition, YPosition)**

**XPosition:**

Distance in pixels from the left of the page to the point where you would like the Cicode object to be placed.

**YPosition:**

Distance in pixels from the top of the page to the point where you would like the Cicode object to be placed.

## Return Value

0 (zero) if successful; otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[DrawLine](#), [DrawEllipse](#), [DrawRectangle](#), [DrawPolygonStart](#), [DrawPolygonLine](#), [DrawPolygonEnd](#), [DrawPipeStart](#), [DrawPipeSection](#), [DrawPipeEnd](#), [DrawText](#), [DrawNumber](#), [DrawSymbolSet](#), [DrawTrend](#)

## Example

```
GraphicsBuilder.DrawCicodeObject 500, 100
```

### DrawEllipse

Draws an ellipse on the active page.

## Syntax

**DrawEllipse(FromXPosition, FromYPosition, ToXPosition, ToYPosition)**

*FromXPosition*:

Distance in pixels from the left hand side of the page to top left hand corner of the rectangle that will enclose the ellipse.

*FromYPosition*:

Distance in pixels from the top of the page to the top left hand corner of the rectangle that will enclose the ellipse.

*ToXPosition*:

Distance in pixels from the left hand side of the page to the bottom right hand corner of the rectangle that will enclose the ellipse.

*ToYPosition*:

Distance in pixels from the top of the page to the bottom-right hand corner of the rectangle that will enclose the ellipse.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[DrawButton](#), [DrawLine](#), [DrawRectangle](#), [DrawPolygonStart](#), [DrawPolygonLine](#), [DrawPolygonEnd](#), [DrawPipeStart](#), [DrawPipeSection](#), [DrawPipeEnd](#), [DrawText](#), [DrawNumber](#), [DrawSymbolSet](#), [DrawCicodeObject](#)

## Example

```
GraphicsBuilder.DrawEllipse 50, 70, 400, 200
```

## DrawLine

Draws a line on the active page.

## Syntax

**DrawLine**(*FromXPosition*, *FromYPosition*, *ToXPosition*, *ToYPosition*)

*FromXPosition*:

Distance in pixels from the left-hand side of the page to top left-hand corner of the rectangle that will enclose the line.

*FromYPosition*:

Distance in pixels from the top of the page to the top-left hand corner of the rectangle that will enclose the line.

*ToXPosition*:

Distance in pixels from the left-hand side of the page to the bottom right-hand corner of the rectangle that will enclose the line.

*ToYPosition*:

Distance in pixels from the top of the page to the bottom-right hand corner of the rectangle that will enclose the line.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[DrawButton](#), [DrawRectangle](#), [DrawPolygonStart](#), [DrawPolygonLine](#), [DrawPolygonEnd](#), [DrawPipeStart](#), [DrawPipeSection](#), [DrawPipeEnd](#), [DrawText](#), [DrawNumber](#), [DrawSymbolSet](#), [DrawCicodeObject](#)

## Example

```
GraphicsBuilder.DrawLine 50, 70, 400, 70
```

## DrawNumber

Places a number object on the page at the specified location.

## Syntax

**DrawNumber**(*XPosition*, *YPosition*)

*XPosition*:

Distance in pixels from the left of the page to the point where you would like the number object to be placed.

*YPosition*:

Distance in pixels from the top of the page to the point where you would like the number object to be placed.

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[DrawLine](#), [DrawEllipse](#), [DrawRectangle](#), [DrawPolygonStart](#), [DrawPolygonLine](#), [DrawPolygonEnd](#), [DrawPipeStart](#), [DrawPipeSection](#), [DrawPipeEnd](#), [DrawText](#), [DrawButton](#), [DrawTrend](#), [DrawCicodeObject](#)

## Example

```
GraphicsBuilder.DrawNumber 500, 100
```

## DrawPipeEnd

Terminates the drawing of a pipe on the active page. To work successfully, this function needs to follow an instance of [DrawPipeSection](#).

## Syntax

**DrawPipeEnd**

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[DrawPipeSection](#), [DrawPipeStart](#)

## Example

```
GraphicsBuilder.DrawPipeStart 50, 290
GraphicsBuilder.DrawPipeSection 200, 350
GraphicsBuilder.DrawPipeSection 350, 250GraphicsBuilder.DrawPipeSection 400, 350
GraphicsBuilder.DrawPipeEnd
```

## DrawPipeSection

Draws a section of pipe on the active page. To work successfully, this function needs to have a starting point defined by the function [DrawPipeStart](#), or it needs to follow a previous incidence of itself.

## Syntax

**DrawPipeSection(*XPosition*, *YPosition*)**

*XPosition*:

Distance in pixels from the left of the page to the point where you would like the section of pipe to end.

*YPosition*:

Distance in pixels from the top of the page to the point where you would like the section of pipe to end.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[DrawPipeStart](#), [DrawPipeEnd](#)

## Example

```
GraphicsBuilder.DrawPipeStart 50, 290
GraphicsBuilder.DrawPipeSection 200, 350
GraphicsBuilder.DrawPipeSection 350, 250
GraphicsBuilder.DrawPipeSection 400, 350
GraphicsBuilder.DrawPipeEnd
```

## DrawPipeStart

Initiates the process of drawing a pipe on the active page by defining a starting point that [DrawPipeSection](#) can be applied to.

## Syntax

**DrawPipeStart(*XPosition*, *YPosition*)**

*XPosition*:

Distance in pixels from the left of the page to the point where you would like the section of pipe to start.

*YPosition*:

Distance in pixels from the top of the page to the point where you would like the section of pipe to start.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[DrawPipeSection](#),

## Example

```
GraphicsBuilder.DrawPipeStart 50, 290
GraphicsBuilder.DrawPipeSection 200, 350
GraphicsBuilder.DrawPipeSection 350, 250
GraphicsBuilder.DrawPipeSection 400, 350
GraphicsBuilder.DrawPipeEnd
```

## DrawPolygonEnd

Terminates the drawing of a polygon on the active page. To work successfully, this function needs to follow an instance of [DrawPolygonLine](#).

## Syntax

**DrawPolygonEnd**

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[DrawPolygonLine](#), [DrawPolygonStart](#)

## Example

```
GraphicsBuilder.DrawPolygonStart 50, 290
GraphicsBuilder.DrawPolygonLine 200, 350
GraphicsBuilder.DrawPolygonLine 350, 250
GraphicsBuilder.DrawPolygonLine 400, 350
GraphicsBuilder.DrawPolygonEnd
```

## DrawPolygonLine

Draws a line on the active page that forms part of a polygon. To work successfully, this function needs to have a starting point defined by the function [DrawPolygonStart](#) or a previous incidence of itself.

## Syntax

**DrawPolygonLine(*XPosition*, *YPosition*)**

*XPosition:*

Distance in pixels from the left of the page to the point where you would like the line to end.

*YPosition:*

Distance in pixels from the top of the page to the point where you would like the line to end.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[DrawPolygonStart](#), [DrawPolygonEnd](#)

## Example

```
GraphicsBuilder.DrawPolygonStart 50, 290
GraphicsBuilder.DrawPolygonLine 200, 350
GraphicsBuilder.DrawPolygonLine 350, 250
GraphicsBuilder.DrawPolygonLine 400, 350
GraphicsBuilder.DrawPolygonEnd
```

## DrawPolygonStart

Initiates the process of drawing a polygon on the active page by defining a starting point that [DrawPolygonLine](#) can be applied to.

## Syntax

**DrawPolygonStart(*XPosition*, *YPosition*)**

*XPosition:*

Distance in pixels from the left of the page to the point where you would like the start the polygon.

*YPosition:*

Distance in pixels from the top of the page to the point where you would like the start the polygon.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[DrawPolygonLine](#), [DrawPolygonEnd](#)

## Example

```
GraphicsBuilder.DrawPolygonStart 50, 290
GraphicsBuilder.DrawPolygonLine 200, 350
GraphicsBuilder.DrawPolygonLine 350, 250
GraphicsBuilder.DrawPolygonLine 400, 350
GraphicsBuilder.DrawPolygonEnd
```

## DrawRectangle

Draws a rectangle on the active page.

## Syntax

**DrawRectangle**(*FromXPosition*, *FromYPosition*, *ToXPosition*, *ToYPosition*)

*FromXPosition*:

Distance from the left hand side of the page to top left hand corner of the rectangle to be drawn, measured in pixels.

*FromYPosition*:

Distance from the top of the page to the top left hand corner of the rectangle to be drawn, measured in pixels.

*ToXPosition*:

Distance from the left hand side of the page to the bottom right hand corner of the rectangle to be drawn, measured in pixels.

*ToYPosition*:

Distance from the top of the page to the bottom right hand corner of the rectangle to be drawn, measured in pixels.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[DrawButton](#), [DrawLine](#), [DrawEllipse](#), [DrawPolygonStart](#), [DrawPolygonLine](#), [DrawPolygonEnd](#), [DrawPipeStart](#), [DrawPipeSection](#), [DrawPipeEnd](#), [DrawText](#), [DrawNumber](#), [DrawSymbolSet](#), [DrawCicodeObject](#)

## Example

```
GraphicsBuilder.DrawRectangle 50, 70, 400, 200
```

## DrawSymbolSet

Places a symbol set object on the page at the specified location.

## Syntax

**DrawSymbolSet(*XPosition*, *YPosition*)**

*XPosition*:

Distance in pixels from the left of the page to the point where you would like the object to be placed.

*YPosition*:

Distance in pixels from the top of the page to the point where you would like the object to be placed.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[DrawLine](#), [DrawEllipse](#), [DrawPolygonStart](#), [DrawPolygonLine](#), [DrawPolygonEnd](#), [DrawPipeStart](#), [DrawPipeSection](#),  
[DrawPipeEnd](#), [DrawText](#), [DrawNumber](#), [DrawButton](#), [DrawTrend](#), [DrawCicodeObject](#), [DrawRectangle](#)

## Example

```
GraphicsBuilder.DrawSymbolSet 500, 100
```

## DrawText

Draws an alphanumeric string at the specified location.

## Syntax

**DrawText(*Text*, *XPosition*, *YPosition*)**

*Text*:

The text to be pasted on to the active graphics page.

*XPosition*:

Distance in pixels from the left of the page to the point where you would like the text to be placed.

*YPosition*:

Distance in pixels from the top of the page to the point where you would like the text to be placed.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[DrawLine](#), [DrawEllipse](#), [DrawPolygonStart](#), [DrawPolygonLine](#), [DrawPolygonEnd](#), [DrawPipeStart](#), [DrawPipeSection](#),  
[DrawPipeEnd](#), [DrawNumber](#), [DrawButton](#), [DrawTrend](#), [DrawCicodeObject](#), [DrawRectangle](#)

## Example

```
GraphicsBuilder.DrawText "My Text", 500, 100
```

## DrawTrend

Draws a trend object on the active page.

## Syntax

**DrawTrend(FromXPosition, FromYPosition, ToXPosition, ToYPosition)**

*FromXPosition*:

Distance from the left hand side of the page to top left hand corner of the trend object, measured in pixels.

*FromYPosition*:

Distance from the top of the page to the top left hand corner of the trend object, measured in pixels.

*ToXPosition*:

Distance from the left hand side of the page to the bottom right hand corner of the trend object, measured in pixels.

*ToYPosition*:

Distance from the top of the page to the bottom right hand corner of the trend object, measured in pixels.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[DrawLine](#), [DrawEllipse](#), [DrawPolygonStart](#), [DrawPolygonLine](#), [DrawPolygonEnd](#), [DrawPipeStart](#), [DrawPipeSection](#),  
[DrawPipeEnd](#), [DrawNumber](#), [DrawButton](#), [DrawCicodeObject](#), [DrawRectangle](#), [DrawSymbolSet](#), [DrawText](#)

## Example

```
GraphicsBuilder.DrawTrend 50, 70, 400, 200
```

## Options Functions

These relate to the options found under the Graphics Builder's Tools menu. They do not throw an exception in

Visual Basic.

<a href="#">OptionDisplayPropertiesOnNew</a>	Simulates the option available on the Graphics Builder Tools menu that automatically shows the properties for a new object when inserted. You can use this function to set this option, or you can retrieve its current setting.
<a href="#">OptionSnapToGrid</a>	Simulates the option available on the Graphics Builder Tools menu that automatically snaps objects to a grid. You can use this function to set this option, or you can retrieve its current setting.
<a href="#">OptionSnapToGuidelines</a>	Simulates the option available on the Graphics Builder Tools menu that automatically snaps objects to guidelines. You can use this function to set this option, or you can retrieve its current setting.

For details and a VB example on handling return and error values, see [Error Handling](#).

## See Also

[Automation Events](#)

[Function Categories](#)

### OptionDisplayPropertiesOnNew

Simulates the option available on the Graphics Builder Tools menu that automatically shows the properties for a new object when inserted. You can use this function to set this option, or you can retrieve its current setting.

## Syntax

**OptionDisplayPropertiesOnNew(OptionValue)**

*OptionValue:*

TRUE activates the option, FALSE deactivates the option.

## Return Value

If retrieving the current setting, TRUE or FALSE. If enabling or disabling the option, 0 (zero) if successful. In both cases, an error is returned if unsuccessful.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[OptionSnapToGrid](#), [OptionSnapToGuidelines](#)

**Note:** This function is implemented in the C++ environment as two separate functions:

---

`put_OptionDisplayPropertiesOnNew` enables or disables this option, and `get_OptionDisplayPropertiesOnNew` retrieves the current option setting.

---

## OptionSnapToGrid

Simulates the option available on the Graphics Builder Tools menu that automatically snaps objects to a grid. You can use this function to set this option, or you can retrieve its current setting.

### Syntax

**OptionSnapToGrid(*OptionValue*)**

*OptionValue*:

TRUE activates the option, FALSE deactivates the option.

### Return Value

If retrieving the current setting, TRUE or FALSE. If enabling or disabling the option, 0 (zero) if successful. In both cases, an error is returned if unsuccessful.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[OptionDisplayPropertiesOnNew](#), [OptionSnapToGuidelines](#)

---

**Note:** This function is implemented in the C++ environment as two separate functions: `put_OptionSnapToGrid` enables or disables this option, and `get_OptionSnapToGrid` retrieves the current option setting.

---

## OptionSnapToGuidelines

Simulates the option available on the Graphics Builder Tools menu that automatically snaps objects to guidelines. You can use this function to set this option, or you can retrieve its current setting.

### Syntax

**OptionSnapToGuidelines(*OptionValue*)**

*OptionValue*:

TRUE activates the option, FALSE deactivates the option.

### Return Value

If retrieving the current setting, TRUE or FALSE. If enabling or disabling the option, 0 (zero) if successful. In both cases, an error is returned if unsuccessful.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

### [OptionDisplayPropertiesOnNew](#), [OptionSnapToGrid](#)

**Note:** This function is implemented in the C++ environment as two separate functions:

`put_OptionSnapToGuidelines` enables or disables this option, and `get_OptionSnapToGuidelines` retrieves the current option setting.

## Page Functions

Using the page functions, you can manipulate the pages in your project (for example, open, close, save, delete), and select objects on those pages. This includes templates, symbols, Genies, Super Genies.

<a href="#">PageActiveWindowHandle</a>	Retrieves the window handle of the active page.
<a href="#">PageClose</a>	Closes the current page. This function will also close an unsaved page.
<a href="#">PageConvertWindowCoordinates</a>	Converts the raw window coordinates to page coordinates which account for the scroll bar position.
<a href="#">PageDelete</a>	Deletes the specified page.
<a href="#">PageDeleteEx</a>	Deletes a specified symbol, Genie or Super Genie from a library.
<a href="#">PageDeleteObject</a>	Deletes the currently selected object.
<a href="#">PageDeleteTemplate</a>	Deletes a specified graphics page template.
<a href="#">PageGroupSelectedObjects</a>	Groups the currently selected objects.
<a href="#">PageImport</a>	Imports a graphics file on to the page as a bitmap.
<a href="#">PageKeyboardCommandsPut</a>	Displays keyboard command options that are available in the Graphics Builder via the automation interface.
<a href="#">PageNew</a>	Creates a new Plant SCADA graphics page.
<a href="#">PageNewEx</a>	Creates a new symbol, Genie or Super Genie page.
<a href="#">PageNewLibrary</a>	Creates a new library. Does not succeed, if project is read-only or not valid.
<a href="#">PageNewTemplate</a>	Creates a new Plant SCADA graphics page template.
<a href="#">PageOpen</a>	Opens an existing Plant SCADA graphics page.
<a href="#">PageOpenEx</a>	Opens the specified symbol, Genie or Super Genie page, if it is found.
<a href="#">PageOpenTemplate</a>	Opens a specified graphics page template.

<a href="#">PagePrint</a>	Prints the current page.
<a href="#">PageUpdated</a>	The event fired when a Graphics Builder page is updated.
<a href="#">PageSave</a>	Saves the page using its current name.
<a href="#">PageSaveAs</a>	Saves the page with a new name within a specified project.
<a href="#">PageSaveAsEx</a>	Saves a symbol, Genie, Super Genie or template page to the specified location.
<a href="#">PageSelect</a>	Select an opened page.
<a href="#">PageSelectFirst</a>	Selects the first page currently open in the Graphics Builder. Does not succeed if there are no pages open.
<a href="#">PageSelectFirstObject</a>	Selects the first object on the active page, based on its z-order number. This will not succeed if no object exists. Note that an object can also be a group object, in which case this function will iterate through the items of a group.
<a href="#">PageSelectFirstObjectEx</a>	Selects the first object on the active page, based on its z-order number. This will not succeed if no object exists. This function will not iterate through the items of a group.
<a href="#">PageSelectFirstObjectInGenie</a>	Select the first sub-object in the currently selected genie
<a href="#">PageSelectFirstObjectInGroup</a>	Selects the first object in a group.
<a href="#">PageSelectNext</a>	Selects the next page currently open in the Graphics Builder. Does not succeed if there are no pages open.
<a href="#">PageSelectNextObject</a>	Selects the next object on the active page, based on its z-order number. This function will not succeed if no object exists. Note that an object can also be a group object, in which case this function will iterate through the items of a group.
<a href="#">PageSelectNextObjectEx</a>	Selects the next object on the active page, based on its z-order number. This function will not succeed if no object exists. This function will not iterate through the items of a group.
<a href="#">PageSelectNextObjectInGenie</a>	Select the next sub-object in the currently selected genie.

<a href="#">PageSelectNextObjectInGroup</a>	Selects the next object in a group.
<a href="#">PageSelectObject</a>	Selects an object using a specified AN number.
<a href="#">PageSelectObjectAdd</a>	Selects an additional object using a specified AN number. This can be used to select multiple objects for a succeeding PageGroupSelectedObjects operation.
<a href="#">PageTemplateSelectFirstObject</a>	Restart the template enumeration sequence.
<a href="#">PageTemplateSelectNextObject</a>	Select the next object in the template.
<a href="#">PageThumbnailToClipboard</a>	Creates a thumbnail image of the current page and copies it to the clipboard.
<a href="#">PageUngroupSelectedObject</a>	Ungroups the currently selected grouped object.

For details and a VB example on handling return and error values, see [Error Handling](#).

## See Also

[Automation Events](#)

[Function Categories](#)

## PageActiveWindowHandle

Retrieves the window handle of the active page.

## Syntax

**PageActiveWindowHandle(*WindowHandle*)**

*WindowHandle*:

The active window handle. The handle is NULL if there is no active window. In VB, this is the return value.

## Return Value

The active window handle. The handle is NULL if there is no active window.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Example

```
' Retrieves the window handle of the active pageMyVariable =
GraphicsBuilder.PageActiveWindowHandle
```

## PageClose

Closes the current page. This function will also close an unsaved page.

## Syntax

**PageClose**

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PageNew](#), [PageOpen](#), [PageSaveAs](#), [PageSave](#)

## Example

```
' Closes the current Plant SCADA graphics page
GraphicsBuilder.PageClose
```

## PageConvertWindowCoordinates

Converts the raw window coordinates to page coordinates which account for the scroll bar position.

## Syntax

**PageConvertWindowCoordinates(*XPosition*, *YPosition*)**

*XPosition*:

The raw window X coordinate as input. The page X coordinates as output.

*YPosition*:

The raw window Y coordinate as input. The page Y coordinate as output.

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## PageDelete

Deletes the specified page.

## Syntax

**PageDelete(*Project*, *Page*, *Flag*)**

*Project*:

The name of the project where the page can be found.

*Page:*

The name of the page to be deleted.

*Flag:*

If the flag is set, associated records are deleted.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageNew](#), [PageOpen](#), [PageSaveAs](#), [PageSave](#)

## Example

```
' Deletes the current Plant SCADA graphics page
GraphicsBuilder.PageDelete "Example", "TestPage", True
```

## PageDeleteEx

Deletes the specified page.

## Syntax

**PageDelete(*Project*, *Page*, *Flag*)**

*Project:*

The name of the project where the page can be found.

*Page:*

The name of the page to be deleted.

*Flag:*

If the flag is set, associated records are deleted.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageNew](#), [PageOpen](#), [PageSaveAs](#), [PageSave](#)

## Example

```
' Deletes the current Plant SCADA graphics page
GraphicsBuilder.PageDelete "Example", "TestPage", True
```

## PageDeleteObject

Deletes a specified symbol, Genie or Super Genie from a library.

## Syntax

**PageDeleteEx**(*Project*, *Library*, *Element*, *PageType*)

*Project*:

The name of the project where the element can be found.

*Library*:

The name of the library where the element can be found.

*Element*:

Name of the symbol, Genie or Super genie.

*PageType*:

0 = Symbol

1 = Genie

2 = Super Genie

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageNew](#), [PageOpen](#), [PageSaveAs](#), [PageSave](#)

## Example

```
' Deletes the specified symbol
GraphicsBuilder.PageDeleteEx "Example", "TestLibrary", "TestObject", 0
' Deletes the specified Genie
GraphicsBuilder.PageDeleteEx "Example", "TestLibrary", "TestObject", 1
' Deletes the specified Super Genie
GraphicsBuilder.PageDeleteEx "Example", "TestLibrary", "TestObject", 2
```

## PageDeleteTemplate

Deletes a specified graphics page template.

### Syntax

**PageDeleteTemplate**(*Project*, *Style*, *Template*, *Resolution*, *Titlebar*)

*Project*:

The name of the project that contains the template.

*Style*:

The style of the template you would like to delete.

*Template*:

The name of the template you would like to delete.

*Resolution*:

The resolution of the template.

0 = Default

1 = VGA

2 = SVGA

3 = XGA

4 = SXGA

5 = User

*Titlebar*:

Set to TRUE to select a titlebar.

### Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

### Related Functions

[PageOpen](#), [PageSave](#), [PageSaveAs](#), [PageClose](#)

### Example

```
' Deletes a graphics page template
GraphicsBuilder.PageDeleteTemplate "include", "standard", "blank", 2, True
```

## PageGroupSelectedObjects

Groups the currently selected objects.

## Syntax

**PageGroupSelectedObjects**

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageSelectObject](#), [PageSelectObjectAdd](#), [PageSelectFirstObject](#), [PageSelectNextObject](#),  
[PageUngroupSelectedObject](#)

## PageImport

Imports a graphics file on to the page as a bitmap.

## Syntax

**PageImport(FileName)**

*FileName*:

The name of the graphic file, including the complete path.

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## PageKeyboardCommandsPut

Allows you to assign keyboard commands to an object or group. A keyboard command is a particular key sequence which executes a command when it is typed in by the operator at runtime.

## Syntax

**PageKeyboardCommandsPut(*nIndex*, *KeySequence*, *Command*, *nArea*, *nPrivilege*, *Logging*)**

*nIndex*:

Position of the keyboard command (as you can define more than one).

*KeySequence*

Key sequences that the operator can enter to execute a command.

*Command*

The commands (set of instructions) to be executed immediately when the selected key sequence is entered. The

commands can be a maximum of 253 characters long.

*nArea*

Use this to assign the keyboard command to the same area as the object/group.

*nPrivilege*

Use this to assign the keyboard command the same privilege as the object/group.

*Logging*

Enables sending a log message to the Log Device.

**Return Value**

Indicates success or failure. 0 indicates success.

## PageNew

Creates a new Plant SCADA graphics page.

## Syntax

**PageNew(*Project, Style, Template, Resolution, Titlebar, Linked*)**

*Project:*

The name of the project that contains the template you would like to apply to the page.

*Style:*

The style you would like to apply to your new Plant SCADA graphics page. Plant SCADA templates are grouped into styles.

*Template:*

Specifies the template you would like to apply to your new Plant SCADA graphics page.

*Resolution:*

Sets the appropriate resolution for the page being created.

0 = Default

1 = VGA

2 = SVGA

3 = XGA

4 = SXGA

5 = User

6 = WUXGA

7 = HD1080

*Titlebar:*

Set to TRUE to include a titlebar on your new Plant SCADA graphics page.

*Linked:*

Set to TRUE to link the page to the library.

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PageOpen](#), [PageSave](#), [PageSaveAs](#), [PageClose](#)

## Example

```
' Creates a new Plant SCADA graphics page
GraphicsBuilder.PageNew "include", "standard", "blank", 2, True, True
```

## PageNewEx

Creates a new symbol, Genie or Super Genie page.

## Syntax

**PageNewEx(*PageType*)**

*PageType*:

Specifies the type of page you would like to create:

0 = Symbol

1 = Genie

2 = Super Genie

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PageOpen](#), [PageSave](#), [PageSaveAs](#), [PageClose](#)

## Example

```
' Creates a symbol as a new graphics page
GraphicsBuilder.PageNewEx "Example", "boiler", "tubes1", 0

' Creates a Genie as a new graphics page
GraphicsBuilder.PageNewEx "Example", "example", "dial", 1

' Creates a Super Genie as a new graphics page
```

```
GraphicsBuilder.PageNewEx "Example", "utility", "!sysinfo", 2
```

## PageNewLibrary

Creates a new library. Fails, if project is read-only or not valid.

## Syntax

**PageNewLibrary**(*Project*, *Library*, *LibraryType*)

*Project*:

The name of the project where the library is created.

*Library*:

The new library name (or style for templates).

*LibraryType*:

Type:

0 = Symbol

1 = Genie

2 = Super Genie

3 = Template

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageOpen](#), [PageSave](#), [PageSaveAs](#), [PageClose](#)

## Example

```
' Creates a new symbol library
GraphicsBuilder.PageNewLibrary "Example", "newlibrary", 0
:
' Creates a new Genie library
GraphicsBuilder.PageNewLibrary "demo", "newlibrary", 1
:
' Creates a new Super Genie library
GraphicsBuilder.PageNewLibrary "Example", "newlibrary", 2
:
' Creates a new template style
GraphicsBuilder.PageNewLibrary "Example", "newstyle", 3
```

## PageNewTemplate

Creates a new Plant SCADA graphics page template.

**PageNewTemplate**(*Project*, *Style*, *Template*, *Resolution*, *Titlebar*, *Linked*)

*Project*:

The name of the project that will contain the template.

*Style*:

The style you would like to apply to your new template.

*Template*:

The name you would like to give to your new template.

*Resolution*:

Sets the appropriate resolution for the template being created.

0 = Default

1 = VGA

2 = SVGA

3 = XGA

4 = SXGA

5 = User

*Titlebar*:

Set to TRUE to include a titlebar on the template.

*Linked*:

Set to TRUE to link the page to the library.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageOpen](#), [PageSave](#), [PageSaveAs](#), [PageClose](#)

## Example

```
' Creates a new Plant SCADAGraphics page template
GraphicsBuilder.PageNewTemplate "include", "standard", "blank", 2, True, True
```

## PageOpen

Opens an existing Plant SCADA graphics page.

## Syntax

**PageOpen(*Project*, *Page*)**

*Project*:

The name of the project that contains the page you would like to open.

*Page*:

The name of the page you would like to open.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageNew](#), [PageSave](#), [PageSaveAs](#), [PageClose](#)

## Example

```
' Opens an existing Plant SCADA graphics page
GraphicsBuilder.PageOpen "Example", "Genies"
```

## PageOpenEx

Opens the specified symbol, Genie or Super Genie page, if it is found. See *BrokenLinkCancelEnabled* for more information if a missing reference is encountered.

## Syntax

**PageOpenEx(*Project*, *Library*, *Element*, *PageType*)**

*Project*:

The name of the project where the element can be found.

*Library*:

The name of the library where the element can be found.

*Element*:

The name of the symbol, Genie or Super Genie.

*PageType*:

Type:

0 = Symbol

1 = Genie

2 = Super Genie

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PageNew](#), [PageSave](#), [PageSaveAs](#), [PageClose](#)

## Example

```
' Opens a symbol saved as a graphics page
GraphicsBuilder.PageOpenEx "Example", "boiler", "tubes1", 0

' Opens a Genie saved as a new graphics page
GraphicsBuilder.PageOpenEx "Example", "example", "dial", 1

' Opens a Super Genie saved as a graphics page
GraphicsBuilder.PageOpenEx "Example", "utility", "!sysinfo", 2
```

## PageOpenTemplate

Opens a specified graphics page template.

## Syntax

**PageOpenTemplate**(*Project*, *Style*, *Template*, *Resolution*, *Titlebar*)

*Project*:

The name of the project that contains the template.

*Style*:

The style of the template you would like to open.

*Template*:

The name of the template you would like to open.

*Resolution*:

The resolution of the template.

0 = Default

1 = VGA

2 = SVGA

3 = XGA

4 = SXGA

5 = User

*Titlebar*:

Set to TRUE to select a titlebar.

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PageOpen](#), [PageSave](#), [PageSaveAs](#), [PageClose](#)

## Example

```
' Opens a graphics page template
GraphicsBuilder.PageOpenTemplate "include", "standard", "blank", 2, True
```

## PagePrint

Prints the current page.

## Syntax

[PagePrint](#)

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PageSelectFirstObject](#), [PageSelectNextObject](#)

## PageSave

Saves the page using its current name.

## Syntax

[PageSave](#)

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PageOpen](#), [PageNew](#), [PageSaveAs](#), [PageClose](#)

## Example

```
' Saves an existing Plant SCADA graphics page
GraphicsBuilder.PageSave
```

## PageSaveAs

Saves the page with a new name within a specified project.

## Syntax

**PageSaveAs**(*Project*, *Page*)

*Project*:

The name of the project you would like to save the page to.

*Page*:

The name you would like to apply to the page.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageNew](#), [PageOpen](#), [PageSave](#), [PageClose](#)

## Example

```
' Saves a Plant SCADA graphics page
GraphicsBuilder.PageSaveAs "Example", "MyPage"
```

## See Also

[Page Properties - General](#)

## PageSaveAsEx

Saves a symbol, Genie, Super Genie or template page to the specified location.

## Syntax

**PageSaveAsEx(*Project*, *Library*, *Element*)**

*Project*:

The name of the project where the element is to be saved.

*Library*:

The name of the library (or style for templates) where the element is to be saved.

*Element*:

The new name for the symbol, Genie, Super Genie, or template to be saved.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageNew](#), [PageSave](#), [PageSaveAs](#), [PageClose](#)

## Example

```
' Renames and saves the currently selected element to the specified location
GraphicsBuilder.PageSaveAsEx "Example", "TestLibrary", "TestObject"
```

## PageSelect

Selects an opened page.

## Syntax

**PageSelect**

sBase - Base name

sFile - File name

## Return Value

N/A

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Example

```
Public Sub Example()
Dim gb As GraphicsBuilder.GraphicsBuilder
.'
```

```
.  
. .  
gb.PageSelect("BaseName", "Filename")  
End Sub
```

## Related Functions

[PageSelectNext](#)

### PageSelectFirst

Selects the first page currently open in the Graphics Builder. Fails if there are no pages open.

## Syntax

[PageSelectFirst](#)

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageSelectNext](#)

## Example

```
' Selects the first page in Graphics Builder  
GraphicsBuilder.PageSelectFirst  
If Err.Number <> 0 Then  
Output.Print "PageSelectFirst Error" + "; Err.Number = " + Hex(Err.Number)  
Else  
Output.Print "PageSelectFirst OK" End If
```

### PageSelectFirstObject

Selects the first object on the active page, based on its z-order number. This will exit and return an error if no object exists.

## Syntax

[PageSelectFirstObject](#)

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PageSelectObject](#), [PageSelectNextObject](#), [PageDeleteObject](#), [PageSelectFirstObjectEx](#)

### PageSelectFirstObjectEx

Selects the first object on the active page, based on its z-order number. This will exit and return an error if no object exists. This function will not iterate through the items of a group.

## Syntax

**PageSelectFirstObjectEx**

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PageSelectObject](#), [PageSelectNextObject](#), [PageDeleteObject](#), [PageSelectFirstObject](#)

### PageSelectFirstObjectInGenie

Select the first sub-object in the currently selected genie

## Syntax

**PageSelectFirstObjectInGenie**

## Return Value

N/A

**Note:** For details on handling return and error values, see [Error Handling](#).

## Example

```
Public Sub Example()
Dim gb As GraphicsBuilder.GraphicsBuilder
```

```
.  
. .  
gb.PageSelectFirstObjectInGenie()  
gb.PageSelectNextObjectInGenie()  
gb.PageTemplateSelectFirstObject()  
gb.PageTemplateSelectNextObject()  
gb.UnLockObject()  
End Sub
```

## Related Functions

[PageSelectNextObjectInGenie](#)

### PageSelectFirstObjectInGroup

Selects the first object in a group.

## Syntax

**PageSelectFirstObjectInGroup**

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageSelectObject](#), [PageSelectNextObject](#), [PageDeleteObject](#), [PageSelectFirstObject](#)

### PageSelectNext

Selects the next page currently open in the Graphics Builder. Fails if there are no pages open.

## Syntax

**PageSelectNext**

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageSelectFirst](#)

## Example

```
' Selects the next page in Graphics Builder
GraphicsBuilder.PageSelectNext
If Err.Number <> 0 Then
Output.Print "PageSelectNext Error" + "; Err.Number = " + Hex(Err.Number)
Else
Output.Print "PageSelectNext OK" End If
```

## PageSelectNextObject

Selects the next object on the active page, based on its z-order number. This function will exit and return an error if no object exists. An object can also be a group object, in which case this function will iterate through the items of a group.

## Syntax

[PageSelectNextObject](#)

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageSelectObject](#), [PageSelectFirstObject](#), [PageDeleteObject](#), [PageSelectNextObjectEx](#)

## PageSelectNextObjectEx

Selects the next object on the active page, based on its z-order number. This function will exit and return an error if no object exists. This function will not iterate through the items of a group.

## Syntax

[PageSelectNextObjectEx](#)

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageSelectObject](#), [PageSelectFirstObject](#), [PageDeleteObject](#), [PageSelectNextObjectEx](#)

### PageSelectNextObjectInGenie

Select the next sub-object in the currently selected genie.

## Syntax

**PageSelectNextObjectInGenie**

## Return Value

N/A

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Example

```
Public Sub Example()
Dim gb As GraphicsBuilder.GraphicsBuilder
.
.
.
gb.PageSelectFirstObjectInGenie()
gb.PageSelectNextObjectInGenie()
gb.PageTemplateSelectFirstObject()
gb.PageTemplateSelectNextObject()
gb.UnLockObject()
End Sub
```

## Related Functions

[PageSelectFirstObjectInGenie](#)

### PageSelectNextObjectInGroup

Selects the next object in a group.

## Syntax

**PageSelectNextObjectInGroup**

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageSelectFirstObjectInGroup](#)

### PageSelectObject

Selects an object using a specified AN number.

## Syntax

**PageSelectObject(AN)**

*AN:*

The AN number of the object you would like to select.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageSelectObjectAdd](#), [PageSelectFirstObject](#), [PageSelectNextObject](#), [PageDeleteObject](#)

### PageSelectObjectAdd

Selects an additional object using a specified AN number. This can be used to select multiple objects for a succeeding PageGroupSelectedObjects operation.

## Syntax

**PageSelectObjectAdd(AN)**

*AN:*

The AN number of the object you would like to select.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageSelectObject](#), [PageSelectFirstObject](#), [PageSelectNextObject](#), [PageGroupSelectedObjects](#)

### PageTemplateSelectFirstObject

Restart the template enumeration sequence.

## Syntax

**PageTemplateSelectFirstObject**

## Return Value

N/A

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Example

```
Public Sub Example()
Dim gb As GraphicsBuilder.GraphicsBuilder
.
.
.
gb.PageSelectFirstObjectInGenie()
gb.PageSelectNextObjectInGenie()
gb.PageTemplateSelectFirstObject()
gb.PageTemplateSelectNextObject()
gb.UnLockObject()
End Sub
```

## Related Functions

[PageTemplateSelectNextObject](#)

### PageTemplateSelectNextObject

Select the next object in the template.

## Syntax

**PageTemplateSelectNextObject**

## Return Value

N/A

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Example

```
Public Sub Example()
Dim gb As GraphicsBuilder.GraphicsBuilder
.
.
.
gb.PageSelectFirstObjectInGenie()
gb.PageSelectNextObjectInGenie()
gb.PageTemplateSelectFirstObject()
gb.PageTemplateSelectNextObject()
gb.UnLockObject()
End Sub
```

## Related Functions

[PageTemplateSelectFirstObject](#)

### PageThumbnailToClipboard

Creates a thumbnail image of the current page and copies it to the clipboard.

## Syntax

**PageThumbnailToClipboard(*Size*)**

*Size*:

The size of the thumbnail image in pixels.

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

### PageUngroupSelectedObject

Ungroups the currently selected grouped object.

## Syntax

**PageUngroupSelectedObject**

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageSelectObject](#), [PageSelectObjectAdd](#), [PageSelectFirstObject](#), [PageSelectNextObject](#),  
[PageGroupSelectedObjects](#)

## PageUpdated

The event fired when a Graphics Builder page is updated.

## Syntax

**PageUpdated**(*sProject*, *sPage*)

*sProject*

The name of the project.

*sPage*

The name of the page.

## Return Value

N/A

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageSelectNext](#)

## Example

```
Sub PageUpdated(ByVal bstrProject As String, ByVal bstrPage As String, ByVal bLastPage As Boolean)
    ' Add your code here
End sub
```

## Page Properties Functions

Using the page properties functions, you can manipulate the properties of the pages in your project.

<a href="#">PageAddAssociation</a>	Adds a new association to the current page. This function will return an error if an association with the specified name already exists.
<a href="#">PageAssociationDefault</a>	Sets or retrieves the default for the currently selected page association.

<a href="#">PageAssociationDescription</a>	Sets or retrieves the description for the currently selected page association.
<a href="#">PageAssociationName</a>	Retrieves the name of the currently selected page association.
<a href="#">PageAssociationValueOnError</a>	Sets or retrieves the value-on-error for the currently selected page association.
<a href="#">PageAppearanceGet</a>	Retrieves the appearance properties of a page. This function, since it does not support True Color functionality, has been superseded by PageAppearanceGetEx.
<a href="#">PageAppearanceGetEx</a>	Retrieves the appearance properties of a page. This function supports True Color functionality and replaces PageAppearanceGet.
<a href="#">PageArea</a>	Retrieves or sets the PageArea property for the current graphics page.
<a href="#">PageClusterInherit</a>	Retrieves or sets the cluster context inherit flag setting for current graphics page.
<a href="#">PageClusterName</a>	Retrieves or sets the cluster context name property for the current graphics page.
<a href="#">PageDeleteAssociation</a>	Deletes the selected association from the current page. After an item has been deleted, a call to PageSelectNextAssociation will select the item immediately following the deleted item.
<a href="#">PageDescription</a>	Sets or retrieves the description attached to the active Plant SCADA graphics page.
<a href="#">PageEnvironmentAdd</a>	Adds a new environment variable to the current page. This function will return an error if an environment variable with the specified name already exists.
<a href="#">PageEnvironmentFirst</a>	Retrieves the first environment variable in the current page. This function will return an error if there are no environment variables in the page.
<a href="#">PageEnvironmentNext</a>	Retrieves the next environment variable in the current page. This function will return an error if there are no more environment variables in the page.
<a href="#">PageEnvironmentRemove</a>	Removes an environment variable from the current page. This function will return an error if an environment variable with the specified name does not exist.

<a href="#">PageEventGet</a>	Gets the event (command or value) that will be triggered at runtime depending on the page behavior.
<a href="#">PageEventGetEx</a>	Gets the event (command or value) that will be triggered at runtime when the window is activated or deactivated.
<a href="#">PageEventPut</a>	Sets the event (command or value) that will be triggered at runtime.
<a href="#">PageEventPutEx</a>	Sets the event (command or value) that will be triggered at runtime, including when the window is activated or deactivated.
<a href="#">PageLogDevice</a>	Retrieves or sets the LogDevice property setting for the current graphics page.
<a href="#">PageName</a>	Returns the name of the active page. This is a read only attribute.
<a href="#">PageNext</a>	Retrieves the name of the page currently defined as "next" for the active graphics page, or sets the page you would like defined as next.
<a href="#">PagePrevious</a>	Retrieves the name of the page currently defined as "previous" to the active graphics page, or sets the page you would like defined as previous to the current page.
<a href="#">PageScanTime</a>	Retrieves or sets the PageScanTime property for the current graphics page.
<a href="#">PageSelectAssociationByName</a>	Selects the specified association in the current page.
<a href="#">PageSelectFirstAssociation</a>	Selects the first association in the current page.
<a href="#">PageSelectNextAssociation</a>	Selects the next association in the current page.
<a href="#">PageTitle</a>	Sets or retrieves the title of the active Plant SCADA graphics page.

For details and a VB example on handling return and error values, see [Error Handling](#).

## See Also

[Automation Events](#)

[Function Categories](#)

## PageAddAssociation

Adds a new association to the current page. This function will return an error if an association with the specified name already exists.

## Syntax

**PageAddAssociation(*Name*)**

*Name*:

The name of the new association to be added to the current page.

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Example

Adding a new element and setting its properties:

```
GraphicsBuilder.PageAddAssociation("MyAssociation")
GraphicsBuilder.SelectAssociationByName("MyAssociation")
GraphicsBuilder.PageAssociationsDefault = "TAG0"
GraphicsBuilder.PageAssociationValueOnError = "Oops"
GraphicsBuilder.PageAssociationDescription = "My Association"
```

## Related Functions

[PageAssociationDefault](#), [PageAssociationDescription](#)

## PageAppearanceGet

Retrieves the appearance properties of a page.

**Note:** As this function does not support True Color functionality, this function has been superseded by the function [PageAppearanceGetEx](#).

## Syntax

**PageAppearanceGet(*Project*, *Style*, *Template*, *Resolution*, *Titlebar*, *Width*, *Height*, *Color*)**

*Project*:

The name of the project that contains the template.

*Style*:

The style of the template.

*Template*:

The name of the template.

*Resolution:*

The resolution of the template.

0 = Default

1 = VGA

2 = SVGA

3 = XGA

4 = SXGA

5 = User

6 = WUXGA

7 = HD1080

*Titlebar:*

TRUE if titlebar is selected.

*Width:*

The width of the page in pixels.

*Height:*

The height of the page in pixels.

*Color:*

The color of the page background.

## Return Value

The requested values, as a string

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageOpen](#), [PageSave](#), [PageSaveAs](#), [PageClose](#)

### PageAppearanceGetEx

Retrieves the appearance properties of a page.

## Syntax

**PageAppearanceGetEx**(*Project*, *Style*, *Template*, *Resolution*, *Titlebar*, *Width*, *Height*, *OnColour*, *OffColour*)

*Project:*

Name of project that contains the template.

*Style:*

Style of template.

*Template:*

Name of template.

*Resolution:*

Resolution of template.

0 = Default

1 = VGA

2 = SVGA

3 = XGA

4 = SXGA

5 = User

6 = WUXGA

7 = HD1080

*Titlebar:*

TRUE if titlebar is selected.

*Width:*

Width of the page in pixels.

*Height:*

Height of the page in pixels.

*OnColour:*

"On" color of the page background as an RGB value.

*OffColour:*

"Off" color of the page background as an RGB value.

## Return Value

The requested values, as a string

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageOpen](#), [PageSave](#), [PageSaveAs](#), [PageClose](#)

### PageArea

Retrieves or sets the PageArea property for the current graphics page.

## Syntax

**PageArea(*Area*)**

*Area:*

1... 255 as a string, or blank to assign the page to every area.

## Return Value

If retrieving the current PageArea setting, 1... 255 is returned as a string if a numeric value is used, or a group name is returned as a string if security has been set up using a preconfigured group. A blank string is returned if the active page is assigned to every area.

If you are using the function to apply an area setting, 0 (zero) is returned if successful, or an E\_INVALIDARG error if the value you want to apply is out of range.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PageName](#), [PageTitle](#), [PageDescription](#), [PagePrevious](#), [ProjectNext](#), [PageScanTime](#), [PageLogDevice](#)

## Example

```
' Assigns a page to one of the areas defined in a Plant SCADA project
GraphicsBuilder.PageArea = "1"

' Retrieves the name of the area the current page is assigned to
MyVariable = GraphicsBuilder.PageArea
```

**Note:** This function is implemented in the C++ environment as two separate functions: put\_PageArea applies an Area setting for active graphics page, and get\_PageArea retrieves the current Area setting.

## See Also

[Page Properties - General](#)

### PageAssociationDefault

Sets or retrieves the default for the currently selected page association.

## Syntax

```
Def=PageAssociationDefault
PageAssociationDefault(Def)
Def:
```

Default substitution string to be used if the page association has not been performed using the Ass(..) Cicode function at runtime. The default needs to be either a literal string enclosed in single quotes (e.g. 'a literal value') or a valid tag name.

## Return Value

The default for the currently selected association (as a string), or 0 (zero) if successfully used to set the default. An error is returned if unsuccessful.

## Related Functions

[PageAddAssociation](#), [PageAssociationDescription](#)

### PageAssociationDescription

Sets or retrieves the description for the currently selected page association.

## Syntax

*Description* = PageAssociationDescription

**PageAssociationDescription**(*Description*)

*Description*:

Free text description of the association.

## Return Value

The description of the currently selected association (as a string), or 0 (zero) if successfully used to set the description. An error is returned if unsuccessful.

## Related Functions

[PageAddAssociation](#)

### PageAssociationName

Retrieves the name of the currently selected page association.

## Syntax

Name = PageAssociationName

## Return Value

The name of the currently selected association (as a string). An error is returned if unsuccessful.

## Related Functions

[PageAddAssociation](#), [PageAssociationDescription](#)

### PageAssociationValueOnError

Sets or retrieves the value-on-error for the currently selected page association.

## Syntax

*ValOnErr* = PageAssociationValueOnError

**PageAssociationValueOnError(*ValOnErr*)**

*ValOnErr*:

Value to be used if the substitution was not performed and a default value was not defined, or a tag name was specified that did not resolve.

## Return Value

The value-on-error for the currently selected association (as a string), or 0 (zero) if successfully used to set the value-on-error. An error is returned if unsuccessful.

## Related Functions

[PageAddAssociation](#), [PageAssociationDescription](#)

## PageClusterInherit

Retrieves or sets the Cluster context inherit flag property setting for the current graphics page.

## Syntax

**PageClusterInherit(*bInherit*)**

*bInherit*:

The setting of the cluster context inherit flag as a boolean value.

## Return Value

The cluster context inherit flag for the active graphics page (as a boolean value), or 0 (zero) if successfully used to set the inherit flag. In both cases, an error is returned if unsuccessful.

## Related Functions

[PageClusterName](#)

## Example

```
' Sets the cluster context inherit flag for current page
GraphicsBuilder.PageClusterInherit = "1"

' Retrieves the cluster context inherit flag for current page
MyVariable = GraphicsBuilder.PageClusterInherit
```

## See Also

[Page Properties - General](#)

### PageClusterName

Retrieves or sets the Cluster context name property setting for the current graphics page.

## Syntax

**PageClusterName(ClusterName)**

*ClusterName*:

The name of the cluster as a string.

## Return Value

The ClusterName setting for the active graphics page (as a string), or 0 (zero) if successfully used to set the ClusterName. In both cases, an error is returned if unsuccessful.

## Related Functions

[PageClusterInherit](#)

## Example

```
' Sets the cluster context name property setting for current page
GraphicsBuilder.PageClusterName = "Cluster1"
' Retrieves the cluster context name property setting for current page
MyVariable = GraphicsBuilder.PageClusterName
```

## See Also

[Page Properties - General](#)

### PageDeleteAssociation

Deletes the selected association from the current page.

After an item has been deleted, a call to [PageSelectNextAssociation](#) will select the item immediately following the deleted item.

## Syntax

**PageDeleteAssociation()**

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Example

Deleting any associations starting with "a":

```
Dim name As String
On Error Resume Next
Err.Clear()
GraphicsBuilder.SelectFirstAssociation()
While (Err.Number = 0)
name = GraphicsBuilder.PageAssociationName
If (name.ToLower().StartsWith("a")) Then
GraphicsBuilder.PageDeleteAssociation()
End If
GraphicsBuilder.PageSelectNextAssociation()
End While
```

## Related Functions

### PageDescription

Sets or retrieves the description attached to the active Plant SCADA graphics page.

## Syntax

**PageDescription**(*Description*)

*Description:*

The description applied to the active graphics page, as a string.

## Return Value

The description of the active graphics page as a string, or 0 (zero) if successfully used to apply a description to the active graphics page. In both cases, an error is returned if unsuccessful.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageName](#), [PageTitle](#), [PagePrevious](#), [ProjectNext](#), [PageScanTime](#), [PageLogDevice](#), [PageNext](#), [PageArea](#)

## Example

```
' Attaches a description to the active graphics page
GraphicsBuilder.PageDescription = "MyDescription"
```

```
' Retrieves the description for the active graphics page MyVariable =  
GraphicsBuilder.PageDescription
```

**Note:** This function is implemented in the C++ environment as two separate functions: put\_PageDescription sets the title of the active graphics page, and get\_PageDescription retrieves the title of the active graphics page.

## See Also

[Page Properties - General](#)

### PageEnvironmentAdd

Adds a new environment variable to the current page. This function will return an error if an environment variable with the specified name already exists

## Syntax

**PageEnvironmentAdd(*Name*, *Value*)**

*Name*:

Specifies the name of the new environment variable.

*Value*:

Specifies the value to associate with the new environment variable

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Example

Adding a new environment variable:

```
GraphicsBuilder.PageEnvironmentAdd("Foo", "Bar")
```

## Related Functions

[PageEnvironmentFirst](#)

### PageEnvironmentFirst

Retrieves the first environment variable in the current page. This function will return an error if there are no environment variables in the page.

## Syntax

**PageEnvironmentFirst(*Name*, *Value*)**

*Name*:

Receives the name of the first environment variable in the current page.

*Value*:

Receives the value associated with the first environment variable.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Example

Printing out environment variables:

```
Dim name As String
Dim value As String
Dim prevName As String
On Error Resume Next
Err.Clear()
GraphicsBuilder.PageEnvironmentFirst(name, value)
While (Err.Number = 0)
Console.Out.WriteLine(name + "=" + value)
prevName = name
GraphicsBuilder.PageEnvironmentNext(prevName, name, value)
End While
```

## Related Functions

[PageEnvironmentAdd](#)

## PageEnvironmentNext

Retrieves the next environment variable in the current page. This function will return an error if there are no more environment variables in the page.

## Syntax

**PageEnvironmentNext(*currentName*, *nextName*, *nextValue*)**

*currentName*:

Specifies the name of the current environment variable.

*nextName*:

Receives the name of the next environment variable.

*nextValue*:

Receives the value associated with the next environment variable.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[PageEnvironmentFirst](#)

### PageEnvironmentRemove

Removes an environment variable from the current page. This function will return an error if an environment variable with the specified name does not exist.

## Syntax

**PageEnvironmentRemove(*Name*)**

*Name*:

Specifies the name of the environment variable to be removed.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Example

Deleting an existing environment variable

```
GraphicsBuilder.PageEnvironmentRemove("Foo")
```

Updating an existing environment variable

```
GraphicsBuilder.PageEnvironmentRemove("Foo")
GraphicsBuilder.PageEnvironmentAdd("Foo", "Bar2")
```

## Related Functions

[PageEnvironmentFirst](#)

### PageEventGet

Gets the event (command or value) that will be triggered at runtime depending on the page behavior.

See Page Properties - Events for more information.

## Syntax

**PageEventGet( )**

## Return Value

Command (Cicode function) or value that was set for the page event.

## Example

```
{  
    string strEventOnEntryCommand,  
    strEventOnExitCommand,  
    strEventWhileShownCommand,  
    strEventOnShownCommand,  
    strEventOnActivateCommand,  
    strEventOnDeactivateCommand,  
    strEventOnEntryCommandExp,  
    strEventOnExitCommandExp,  
    strEventWhileShownCommandExp,  
    strEventOnShownCommandExp,  
    strEventOnActivateCommandExp,  
    strEventOnDeactivateCommandExp  
    .  
  
    gb.ProjectSelect("Example_AlarmIndicator");  
    gb.PageOpen("Example_AlarmIndicator", "Test");  
    .  
    strEventOnEntryCommand = "PageEventsPutEntryCommandEx";  
    strEventOnExitCommand = "PageEventsPutExitCommandEx";  
    strEventWhileShownCommand = "PageEventsPutShownCommandEx";  
    strEventOnShownCommand = "PageEventsPutShownCommandEx";  
    strEventOnActivateCommand = "PageEventsPutActivateCommandEx";  
    strEventOnDeactivateCommand = "PageEventsPutDeactivateCommandEx";  
    gb2.PageEventsPutEx(strEventOnEntryCommand, strEventOnExitCommand,  
    strEventWhileShownCommand,  
    strEventOnShownCommand,  
    strEventOnActivateCommand, strEventOnDeactivateCommand);  
    gb2.PageEventsGetEx(out strEventOnEntryCommandExp, out strEventOnExitCommandExp, out  
    strEventWhileShownCommandExp,  
    out strEventOnShownCommandExp, out strEventOnActivateCommandExp, out  
    strEventOnDeactivateCommandExp);  
    .  
    .  
    strEventOnEntryCommand = "PageEventsPutEntryCommand";  
    strEventOnExitCommand = "PageEventsPutExitCommand";  
    strEventWhileShownCommand = "PageEventsPutShownCommand";  
    strEventOnShownCommand = "PageEventsPutShownCommand";  
  
    gb2.PageEventsPut(strEventOnEntryCommand, strEventOnExitCommand, strEventWhileShownCommand,  
    strEventOnShownCommand);  
    gb2.PageEventsGet(out strEventOnEntryCommandExp, out strEventOnExitCommandExp, out  
    strEventWhileShownCommandExp,  
    out strEventOnShownCommandExp);  
}
```

## Related Functions

[PageEventPut](#)

### PageEventGetEx

Gets the event (command or value) that will be triggered at runtime when the window is activated or deactivated.

See Page Properties - Events for more information

## Syntax

**PageEventGetEx ()**

## Return Value

Command (Cicode function) or value that was set for the window.

## Example

```
{  
    string strEventOnEntryCommand,  
    strEventOnExitCommand,  
    strEventWhileShownCommand,  
    strEventOnShownCommand,  
    strEventOnActivateCommand,  
    strEventOnDeactivateCommand,  
    strEventOnEntryCommandExp,  
    strEventOnExitCommandExp,  
    strEventWhileShownCommandExp,  
    strEventOnShownCommandExp,  
    strEventOnActivateCommandExp,  
    strEventOnDeactivateCommandExp  
  
    gb.ProjectSelect("Example_AlarmIndicator");  
    gb.PageOpen("Example_AlarmIndicator", "Test");  
  
    strEventOnEntryCommand = "PageEventsPutEntryCommandEx";  
    strEventOnExitCommand = "PageEventsPutExitCommandEx";  
    strEventWhileShownCommand = "PageEventsPutShownCommandEx";  
    strEventOnShownCommand = "PageEventsPutShownCommandEx";  
    strEventOnActivateCommand = "PageEventsPutActivateCommandEx";  
    strEventOnDeactivateCommand = "PageEventsPutDeactivateCommandEx";  
  
    gb2.PageEventsPutEx(strEventOnEntryCommand, strEventOnExitCommand,  
    strEventWhileShownCommand,  
    strEventOnShownCommand,  
    strEventOnActivateCommand, strEventOnDeactivateCommand);  
    gb2.PageEventsGetEx(out strEventOnEntryCommandExp, out strEventOnExitCommandExp, out  
    strEventWhileShownCommandExp,  
    out strEventOnShownCommandExp, out strEventOnActivateCommandExp, out
```

```
strEventOnDeactivateCommandExp);  
  
strEventOnEntryCommand = "PageEventsPutEntryCommand";  
strEventOnExitCommand = "PageEventsPutExitCommand";  
strEventWhileShownCommand = "PageEventsPutShownCommand";  
strEventOnShownCommand = "PageEventsPutShownCommand";  
  
gb2.PageEventsPut(strEventOnEntryCommand, strEventOnExitCommand, strEventWhileShownCommand,  
strEventOnShownCommand);  
gb2.PageEventsGet(out strEventOnEntryCommandExp, out strEventOnExitCommandExp, out  
strEventWhileShownCommandExp,  
out strEventOnShownCommandExp);  
}
```

## Related Functions

[PageEventPutEx](#)

### PageEventPut

Sets the event (command or value) that will be triggered at runtime either:

- on page entry.
- exiting the page.
- when the page is first displayed and objects on it have been initialized.
- or executed continually for the entire time the page is open.

## Syntax

```
PageEventPut (sEventOnEntryCommand, sEventOnExitCommand, sEventWhileShownCommand,  
sEventOnShownCommand,sOnActivateCommand, sOnDeactivateCommand )
```

## Example

```
{  
string strEventOnEntryCommand,  
strEventOnExitCommand,  
strEventWhileShownCommand,  
strEventOnShownCommand,  
strEventOnActivateCommand,  
strEventOnDeactivateCommand,  
strEventOnEntryCommandExp,  
strEventOnExitCommandExp,  
strEventWhileShownCommandExp,  
strEventOnShownCommandExp,  
strEventOnActivateCommandExp,  
strEventOnDeactivateCommandExp  
  
gb.ProjectSelect("Example_AlarmIndicator");
```

```
gb.PageOpen("Example_AlarmIndicator", "Test");

strEventOnEntryCommand = "PageEventsPutEntryCommandEx";
strEventOnExitCommand = "PageEventsPutExitCommandEx";
strEventWhileShownCommand = "PageEventsPutShownCommandEx";
strEventOnShownCommand = "PageEventsPutShownCommandEx";
strEventOnActivateCommand = "PageEventsPutActivateCommandEx";
strEventOnDeactivateCommand = "PageEventsPutDeactivateCommandEx";

gb2.PageEventsPutEx(strEventOnEntryCommand, strEventOnExitCommand,
strEventWhileShownCommand,
strEventOnShownCommand,
strEventOnActivateCommand, strEventOnDeactivateCommand);
gb2.PageEventsGetEx(out strEventOnEntryCommandExp, out strEventOnExitCommandExp, out
strEventWhileShownCommandExp,
out strEventOnShownCommandExp, out strEventOnActivateCommandExp, out
strEventOnDeactivateCommandExp);

strEventOnEntryCommand = "PageEventsPutEntryCommand";
strEventOnExitCommand = "PageEventsPutExitCommand";
strEventWhileShownCommand = "PageEventsPutShownCommand";
strEventOnShownCommand = "PageEventsPutShownCommand";

gb2.PageEventsPut(strEventOnEntryCommand, strEventOnExitCommand, strEventWhileShownCommand,
strEventOnShownCommand);
gb2.PageEventsGet(out strEventOnEntryCommandExp, out strEventOnExitCommandExp, out
strEventWhileShownCommandExp,
out strEventOnShownCommandExp);
}
```

## Related Functions

[PageEventGet](#)

### PageEventPutEx

Sets the event (command or value) that will be triggered at runtime either:

- on page entry.
- exiting the page.
- when the page is first displayed and objects on it have been initialized.
- or executed continually for the entire time the page is open.
- when the window becomes active (in context).
- when the window is no longer active.

See Page Properties - Events for more information

## Syntax

**PageEventPutEx** (*sEventOnEntryCommand*, *sEventOnExitCommand*, *sEventWhileShownCommand*,  
*sEventOnShownCommand*, *sOnActivateCommand*, *sOnDeactivateCommand*)

## Example

```
{  
    string strEventOnEntryCommand,  
    strEventOnExitCommand,  
    strEventWhileShownCommand,  
    strEventOnShownCommand,  
    strEventOnActivateCommand,  
    strEventOnDeactivateCommand,  
    strEventOnEntryCommandExp,  
    strEventOnExitCommandExp,  
    strEventWhileShownCommandExp,  
    strEventOnShownCommandExp,  
    strEventOnActivateCommandExp,  
    strEventOnDeactivateCommandExp  
  
    gb.ProjectSelect("Example_AlarmIndicator");  
    gb.PageOpen("Example_AlarmIndicator", "Test");  
  
    strEventOnEntryCommand = "PageEventsPutEntryCommandEx";  
    strEventOnExitCommand = "PageEventsPutExitCommandEx";  
    strEventWhileShownCommand = "PageEventsPutShownCommandEx";  
    strEventOnShownCommand = "PageEventsPutShownCommandEx";  
    strEventOnActivateCommand = "PageEventsPutActivateCommandEx";  
    strEventOnDeactivateCommand = "PageEventsPutDeactivateCommandEx";  
  
    gb2.PageEventsPutEx(strEventOnEntryCommand, strEventOnExitCommand,  
    strEventWhileShownCommand,  
    strEventOnShownCommand,  
    strEventOnActivateCommand, strEventOnDeactivateCommand);  
    gb2.PageEventsGetEx(out strEventOnEntryCommandExp, out strEventOnExitCommandExp, out  
    strEventWhileShownCommandExp,  
    out strEventOnShownCommandExp, out strEventOnActivateCommandExp, out  
    strEventOnDeactivateCommandExp);  
  
    strEventOnEntryCommand = "PageEventsPutEntryCommand";  
    strEventOnExitCommand = "PageEventsPutExitCommand";  
    strEventWhileShownCommand = "PageEventsPutShownCommand";  
    strEventOnShownCommand = "PageEventsPutShownCommand";  
  
    gb2.PageEventsPut(strEventOnEntryCommand, strEventOnExitCommand, strEventWhileShownCommand,  
    strEventOnShownCommand);  
    gb2.PageEventsGet(out strEventOnEntryCommandExp, out strEventOnExitCommandExp, out  
    strEventWhileShownCommandExp,  
    out strEventOnShownCommandExp);  
}
```

## Related Functions

[PageEventGetEx](#)

### PageLogDevice

Retrieves or sets the LogDevice property setting for the current graphics page.

## Syntax

**PageLogDevice(*LogDevice*)**

*LogDevice*:

The name of the log device as a string.

## Return Value

The LogDevice setting for the active graphics page (as a string), or 0 (zero) if successfully used to set the LogDevice. In both cases, an error is returned if unsuccessful.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PageName](#), [PageTitle](#), [PageDescription](#), [PagePrevious](#), [PageNext](#), [PageScanTime](#)

## Example

```
' Sets the LogDevice for the current graphics page
GraphicsBuilder.PageLogDevice = "MyDevice"
' Retrieves the name of the LogDevice for the current page
MyVariable = GraphicsBuilder.PageLogDevice
```

**Note:** This function is implemented in the C++ environment as two separate functions: put\_PageLogDevice applies a LogDevice setting for active graphics page, and get\_PageLogDevice retrieves the current LogDevice setting.

## See Also

[Page Properties - General](#)

### PageName

Returns the name of the active page. This is a read only attribute.

## Syntax

**PageName(*PageName*)**

*PageName:*

Returns the name of the active page as a string.

## Return Value

The name of the active Plant SCADA graphics page as a string.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageTitle](#), [PageDescription](#), [PagePrevious](#), [PageNext](#), [PageArea](#), [PageScanTime](#), [PageLogDevice](#)

## Example

```
Debug.Print "PageName"; GraphicsBuilder.PageName
```

## See Also

[Page Properties - General](#)

## PageNext

Retrieves the name of the page currently defined as "next" for the active graphics page, or sets the page you would like defined as next.

## Syntax

**PageNext(*PageName*)**

*PageName:*

The name of the page defined as next for the active graphics page, as a string.

## Return Value

The name of the page defined as next for the active graphics page (as a string), or 0 (zero) if successfully used to set the page that is defined as next. In both cases, an error is returned if unsuccessful.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[PageName](#), [PageTitle](#), [PageDescription](#), [PagePrevious](#), [PageArea](#), [PageScanTime](#), [PageLogDevice](#)

## Example

```
' Defines a page as the one that follows the current page in a browse sequence
```

```
GraphicsBuilder.PageNext = "MyPage3"
' Retrieves the name of the page that follows the current page in a browse sequence
MyVariable = GraphicsBuilder.PageNext
```

**Note:** This function is implemented in the C++ environment as two separate functions: put\_PageNext sets the page defined as next for the active graphics page, and get\_PageNext retrieves the name of the next graphics page.

## See Also

[Page Properties - General](#)

### PagePrevious

Retrieves the name of the page currently defined as "previous" to the active graphics page, or sets the page you would like defined as previous to the current page.

## Syntax

**PagePrevious(***PageName***)**

*PageName*:

The name of the page defined as previous for the active graphics page, as a string.

## Return Value

The name of the page defined as previous to the active graphics page (as a string), or 0 (zero) if successfully used to set the page that is previous to the active graphics page. In both cases, an error is returned if unsuccessful.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PageTitle](#), [PageDescription](#), [PageNext](#), [PageArea](#), [PageScanTime](#), [PageLogDevice](#)

## Example

```
' Defines a page as previous to the current page in a browse sequence
GraphicsBuilder.PagePrevious = "MyPage1"
' Retrieves the name for the page defined as previous to the current page
MyVariable = GraphicsBuilder.PagePrevious
```

**Note:** This function is implemented in the C++ environment as two separate functions: put\_PagePrevious sets the page defined as previous to the active graphics page, and get\_PagePrevious retrieves the name of the previous graphics page.

## See Also

[Page Properties - General](#)

## PageScanTime

Retrieves or sets the PageScanTime property for the current graphics page.

## Syntax

**PageScanTime(ScanTime)**

*ScanTime:*

A value between 1 and 60000 as a string, or blank to set to default.

## Return Value

If retrieving the current PageScanTime setting, the value returned is between 1 and 60000 as a string, or a blank string if set to default.

If you are using the function to apply a ScanTime setting, 0 (zero) is returned if successful, or an E\_INVALIDARG error if the value you want to apply is out of range.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PageName](#), [PageTitle](#), [PageDescription](#), [PagePrevious](#), [PageNext](#), [PageLogDevice](#)

## Example

```
' Assigns a ScanTime value to the current graphics page
GraphicsBuilder.PageScanTime = "2000"
' Retrieves the ScanTime setting for the current page
MyVariable = GraphicsBuilder.PageScanTime
```

**Note:** This function is implemented in the C++ environment as two separate functions: `put_PageScanTime` applies an ScanTime setting to active graphics page, and `get_PageScanTime` retrieves the current ScanTime setting.

## See Also

[Page Properties - General](#)

## PageSelectAssociationByName

Selects the specified association in the current page.

## Syntax

**PageSelectAssociationByName(Name)**

*Name*

The name of the association to be selected.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Example

Determining whether an association with a particular name exists:

```
On Error Resume Next
Err.Clear()
GraphicsBuilder.SelectAssociationByName("MyAssociation")
If (Err.Number <> 0)
    ' The association does not exist
End If
```

## Related Functions

[PageAddAssociation](#)

## PageSelectFirstAssociation

Selects the first association in the current page.

## Syntax

`PageSelectFirstAssociation()`

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Example

Determine whether the current page has associations in the page properties:

```
On Error Resume Next
Err.Clear()
GraphicsBuilder.SelectFirstAssociation()
If (Err.Number <> 0)
    ' The page has no associations
End If
```

## Related Functions

[PageAddAssociation](#), [PageAssociationDescription](#)

## PageSelectNextAssociation

Selects the next association in the current page.

### Syntax

```
PageSelectNextAssociation()
```

### Return Value

0 (zero) if successful, otherwise an error is returned.

### Example

Print associations in the current page's properties:

```
On Error Resume Next
Err.Clear()
GraphicsBuilder.SelectFirstAssociation()
While (Err.Number = 0)
Console.Out.WriteLine(GraphicsBuilder.PageAssociationName)
GraphicsBuilder.SelectNextAssociation()
End While
```

## Related Functions

[PageAddAssociation](#), [PageAssociationDescription](#)

### PageTitle

Sets or retrieves the title of the active Plant SCADA graphics page.

### Syntax

```
PageTitle(Title)
```

*Title*:

The title of the active page as a string.

### Return Value

The title of the active graphics page as a string, or 0 (zero) if successfully used to set the title of the active graphics page. In both cases, an error is returned if unsuccessful.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[PageName](#), [PageDescription](#), [PagePrevious](#), [PageNext](#), [PageArea](#), [PageScanTime](#), [PageLogDevice](#)

## Example

```
' Sets the title of the active graphics page
GraphicsBuilder.PageTitle = "MyTitle"

' Retrieves the title of the active graphics page
MyVariable = GraphicsBuilder.PageTitle
```

**Note:** This function is implemented in the C++ environment as two separate functions: put\_PageTitle sets the title of the active graphics page, and get\_PageTitle retrieves the title of the active graphics page.

## See Also

[Page Properties - General](#)

## Project Functions

These functions operate on the project level. Some are initiated within Plant SCADA Studio. If they experience an error in Visual Basic, they throw an exception with a return value E\_FAIL.

<a href="#">ProjectCompile</a>	This function starts the Plant SCADA compiler with the current project.
<a href="#">ProjectFirst</a>	Retrieves the name of the first project defined in Plant SCADA.
<a href="#">ProjectFirstInclude</a>	Retrieves the name of the first included project defined for the current Plant SCADA project.
<a href="#">ProjectNext</a>	Retrieves the name of the next project defined in Plant SCADA.
<a href="#">ProjectNextInclude</a>	Retrieves the name of the next included project defined for the current Plant SCADA project.
<a href="#">ProjectPackDatabase</a>	Packs the current project's database files.
<a href="#">ProjectPackLibraries</a>	Packs the library files for the current Plant SCADA project.
<a href="#">ProjectSelect</a>	Selects the passed project as the current project within Plant SCADA.
<a href="#">ProjectSelected</a>	Retrieves the name of the project that is currently selected in Plant SCADA.

ProjectUpdatePages	Updates the pages for the current Plant SCADA project.
ProjectUpgrade	Performs a project upgrade on the current Plant SCADA project.
ProjectUpgradeAll	Performs a project upgrade on the Plant SCADA projects.

For details and a VB example on handling return and error values, see [Error Handling](#).

## See Also

[Automation Events](#)

[Function Categories](#)

## ProjectCompile

This function starts the Plant SCADA compiler with the current project. There are currently no functions to check errors or trigger the compiler's cancel function.

## Syntax

**ProjectCompile**

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[ProjectSelect](#), [ProjectSelected](#), [ProjectFirst](#), [ProjectNext](#), [ProjectFirstInclude](#), [ProjectNextInclude](#), [ProjectUpgrade](#), [ProjectUpgradeAll](#), [ProjectPackLibraries](#), [ProjectUpdatePages](#), [ProjectPackDatabase](#)

## Example

```
GraphicsBuilder.ProjectCompile
If Err.Number <> 0 Then
    Debug.Print "Error in ProjectCompile"
    Err.Clear
Else
    Debug.Print "ProjectCompile OK"
End If
```

## ProjectFirst

Retrieves the name of the first project defined in Plant SCADA. Can be used in conjunction with [ProjectNext](#) to call the projects currently defined in Plant SCADA.

## Syntax

**ProjectFirst(*Project*)**

*Project:*

The name of the project.

## Return Value

The name of the first Plant SCADA project. If no project exists, an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[ProjectSelect](#), [ProjectSelected](#), [ProjectNext](#), [ProjectFirstInclude](#), [ProjectNextInclude](#), [ProjectUpgrade](#), [ProjectUpgradeAll](#), [ProjectPackLibraries](#), [ProjectUpdatePages](#), [ProjectPackDatabase](#), [ProjectCompile](#)

## Example

```
On Error Resume Next
sProject = GraphicsBuilder.ProjectSelected
If Err.Number <> 0 Then
    Debug.Print "Error in ProjectSelected"
    Err.Clear
Else
    Debug.Print "Selected project:", sProject
End If
Debug.Print "list of projects:"
sProject = GraphicsBuilder.ProjectFirstWhile Err.Number = 0
Debug.Print sProject
sProject = GraphicsBuilder.ProjectNext
Wend
```

## ProjectFirstInclude

Retrieves the name of the first included project defined for the current Plant SCADA project. Can be used in conjunction with [ProjectNextInclude](#) to call the projects defined as included for current Plant SCADA project.

## Syntax

**ProjectFirstInclude(*Project*)**

*Project:*

The name of the project.

## Return Value

The name of the first include project for the current Plant SCADA project. If no project exists, an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[ProjectSelect](#), [ProjectSelected](#), [ProjectFirst](#), [ProjectNext](#), [ProjectNextInclude](#), [ProjectUpgrade](#), [ProjectUpgradeAll](#), [ProjectPackLibraries](#), [ProjectUpdatePages](#), [ProjectPackDatabase](#), [ProjectCompile](#)

## ProjectNext

Retrieves the name of the next project defined in Plant SCADA. Can be used with [ProjectFirst](#) to call projects currently defined in Plant SCADA.

## Syntax

**ProjectNext(*Project*)**

*Project*:

The name of the project.

## Return Value

The name of the next Plant SCADA project. If no project exists, an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[ProjectSelected](#), [ProjectFirst](#), [ProjectFirstInclude](#), [ProjectNextInclude](#), [ProjectUpgrade](#), [ProjectUpgradeAll](#), [ProjectPackLibraries](#), [ProjectUpdatePages](#), [ProjectPackDatabase](#), [ProjectCompile](#)

## Example

```
On Error Resume Next
sProject = GraphicsBuilder.ProjectSelected
If Err.Number <> 0 Then
    Debug.Print "Error in ProjectSelected"
    Err.Clear
Else
    Debug.Print "Selected project:", sProject
End If
Debug.Print "list of projects:"
sProject = GraphicsBuilder.ProjectFirst
```

```
While Err.Number = 0
Debug.Print sProject
sProject = GraphicsBuilder.ProjectNext
Wend
```

## ProjectNextInclude

Retrieves the name of the next included project defined for the current Plant SCADA project. Can be used with [ProjectFirstInclude](#) to call the projects defined as included for current Plant SCADA project.

## Syntax

**ProjectNextInclude(*Project*)**

*Project*:

The name of the project.

## Return Value

The name of the next include project for the current Plant SCADA project. If no project exists, an error is returned.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[ProjectSelect](#), [ProjectSelected](#), [ProjectFirst](#), [ProjectNext](#), [ProjectFirstInclude](#), [ProjectUpgrade](#), [ProjectUpgradeAll](#),  
[ProjectUpdatePages](#), [ProjectPackDatabase](#), [ProjectCompile](#)

## ProjectPackDatabase

Packs the current project's database files.

## Syntax

**ProjectPackDatabase**

## Return Value

0 (zero) if successful, otherwise an error is returned.

---

**Note:** This function displays a cancel dialog. It will exit and report error code E\_ABORT, if the cancel button is pressed. For details on handling return and error values, see [Error Handling](#).

## Related Functions

[ProjectSelect](#), [ProjectSelected](#), [ProjectFirst](#), [ProjectNext](#), [ProjectFirstInclude](#), [ProjectNextInclude](#), [ProjectUpgrade](#),  
[ProjectUpgradeAll](#), [ProjectPackLibraries](#), [ProjectUpdatePages](#), [ProjectCompile](#)

## Example

```
GraphicsBuilder.ProjectPackDatabase
If Err.Number <> 0 Then
    Debug.Print "Error in ProjectPackDatabase"
    Err.Clear
Else
    Debug.Print "ProjectPackDatabase OK"
End If
```

## ProjectPackLibraries

Packs the library files for the current Plant SCADA project.

## Syntax

**ProjectPackLibraries**

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** This function displays a cancel dialog. It will exit and report error code E\_ABORT, if the cancel button is pressed. For details on handling return and error values, see [Error Handling](#).

## Related Functions

[ProjectSelect](#), [ProjectSelected](#), [ProjectFirst](#), [ProjectNext](#), [ProjectFirstInclude](#), [ProjectNextInclude](#), [ProjectUpgrade](#), [ProjectUpgradeAll](#), [ProjectUpdatePages](#), [ProjectPackDatabase](#), [ProjectCompile](#)

## Example

```
GraphicsBuilder.ProjectPackLibraries
If Err.Number <> 0 Then
    Debug.Print "Error in ProjectPackLibraries"
    Err.Clear
Else
    Debug.Print "ProjectPackLibraries OK"
End If
```

## ProjectSelect

Selects the passed project as the current project within Plant SCADA Studio.

## Syntax

**ProjectSelect(*Project*)**

*Project*:

The name of the project.

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[ProjectSelected](#), [ProjectFirst](#), [ProjectNext](#), [ProjectFirstInclude](#), [ProjectNextInclude](#), [ProjectUpgrade](#), [ProjectUpgradeAll](#), [ProjectPackLibraries](#), [ProjectUpdatePages](#), [ProjectPackDatabase](#), [ProjectCompile](#)

## Example

```
GraphicsBuilder.ProjectSelect "Example"
```

### ProjectSelected

Retrieves the name of the project that is currently selected in Plant SCADA.

## Syntax

**ProjectSelected**(*Project*)

*Project*:

The name of the project.

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[ProjectSelect](#), [ProjectFirst](#), [ProjectNext](#), [ProjectFirstInclude](#), [ProjectNextInclude](#), [ProjectUpgrade](#), [ProjectUpgradeAll](#), [ProjectPackLibraries](#), [ProjectUpdatePages](#), [ProjectPackDatabase](#), [ProjectCompile](#)

## Example

```
On Error Resume Next
sProject = GraphicsBuilder.ProjectSelected
If Err.Number <> 0 Then
    Debug.Print "Error in ProjectSelected"
    Err.Clear
Else
    Debug.Print "Selected project:", sProject
End If
```

```
Debug.Print "list of projects:"  
sProject = GraphicsBuilder.ProjectFirst  
While Err.Number = 0  
Debug.Print sProject  
sProject = GraphicsBuilder.ProjectNext  
Wend
```

## ProjectUpdatePages

Updates the pages for the current Plant SCADA project. If you encounter missing references during the update, see [BrokenLinkCancelEnabled](#).

## Syntax

**ProjectUpdatePages(*FastUpdate*)**

*FastUpdate*:

Set to TRUE to enable a fast update.

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** This function displays a cancel dialog. It will exit and report error code E\_ABORT, if the cancel button is pressed. For details on handling return and error values, see [Error Handling](#).

## Related Functions

[ProjectSelect](#), [ProjectSelected](#), [ProjectFirst](#), [ProjectNext](#), [ProjectFirstInclude](#), [ProjectNextInclude](#), [ProjectUpgrade](#), [ProjectUpgradeAll](#), [ProjectPackLibraries](#), [ProjectPackDatabase](#), [ProjectCompile](#)

## Example

```
GraphicsBuilder.ProjectUpdatePages True  
If Err.Number <> 0 Then  
    Debug.Print "Error in ProjectUpdatePages"  
    Err.Clear  
Else  
    Debug.Print "ProjectUpdatePages OK"  
End If
```

## ProjectUpgrade

Performs a project upgrade on the current Plant SCADA project.

## Syntax

**ProjectUpgrade**

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** This function displays a cancel dialog. It will exit and report error code E\_ABORT, if the cancel button is pressed. For details on handling return and error values, see [Error Handling](#).

## Related Functions

[ProjectSelect](#), [ProjectSelected](#), [ProjectFirst](#), [ProjectNext](#), [ProjectFirstInclude](#), [ProjectNextInclude](#),  
[ProjectUpgradeAll](#), [ProjectPackLibraries](#), [ProjectUpdatePages](#), [ProjectPackDatabase](#), [ProjectCompile](#)

## Example

```
On Error Resume Next
Err.Clear
GraphicsBuilder.ProjectUpgrade
If Err.Number <> 0 Then
    Debug.Print "Error in ProjectUpgrade"
    Err.Clear
Else
    Debug.Print "ProjectUpgrade OK"
End If
```

## ProjectUpgradeAll

Performs a project upgrade on Plant SCADA projects. This function produces the same result as setting Upgrade=1 in the Citect.ini file.

## Syntax

**ProjectUpgradeAll**

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[ProjectSelect](#), [ProjectSelected](#), [ProjectFirst](#), [ProjectFirstInclude](#), [ProjectNextInclude](#), [ProjectUpgrade](#),  
[ProjectPackLibraries](#), [ProjectUpdatePages](#), [ProjectPackDatabase](#), [ProjectCompile](#)

## Example

```
On Error Resume Next
Err.Clear
GraphicsBuilder.ProjectUpgrade
```

```
If Err.Number <> 0 Then
    Debug.Print "Error in ProjectUpgrade"
    Err.Clear
Else
    Debug.Print "ProjectUpgrade OK"
End If
```

## Specific Functions

The specific functions category currently includes only the [Visible](#) function.

<a href="#">Visible</a>	Controls visibility of the Plant SCADA Graphics Builder, or retrieves its current visible state.
-------------------------	--

For details and a VB example on handling return and error values, see [Error Handling](#).

## See Also

[Automation Events](#)

[Function Categories](#)

### Visible

Controls visibility of the Plant SCADA Graphics Builder or retrieves its current visible state.

## Syntax

**Visible**

### Return Value

If determining the current visible state of the Graphics Builder, TRUE or FALSE is returned. If applying a setting to this function, 0 (zero) is returned if successful, otherwise an error is returned.

**Note:** For details on handling return and error values, see [Error Handling](#).

### Example

```
' Make Plant SCADA Graphics Builder appear
GraphicsBuilder.Visible = TRUE
' Retrieve the current visible state of the Graphics Builder
MyVariable = GraphicsBuilder.Visible
```

**Note:** This function is implemented in the C++ environment as two separate functions: `put_Visible` sets the visible state of the Graphics Builder, and `get_Visible` retrieves the current state of the Graphics Builder (TRUE = visible).

## Text Property Functions

These functions allow you to read and modify the properties of the text objects in your project.

<a href="#">AttributeText</a>	Sets the text for a text object, or retrieves the current text.
<a href="#">AttributeTextColour</a>	Applies a color to the selected text, or retrieves the current font color setting.
<a href="#">AttributeTextOffColourEx</a>	Applies the "off" color to the selected text, or retrieves the current font color setting.
<a href="#">AttributeTextOnColourEx</a>	Applies the "on" color to the selected text, or retrieves the current font color setting.
<a href="#">AttributeTextFont</a>	Applies a specific font to the selected text, or retrieves the font setting.
<a href="#">AttributeTextFontSize</a>	Applies a font size to the selected text, or retrieves the current font size.
<a href="#">AttributeTextJustification</a>	Applies a specific justification setting to selected text, or retrieves the current text justification value.
<a href="#">AttributeTextStyle</a>	Sets a specific text style, or retrieves the current text style setting.

The following object functions are also valid for text objects:

<a href="#">Attribute3dEffects</a>	Applies a 3D effect to an object, or retrieves the current 3D effect setting.
<a href="#">Attribute3dEffectDepth</a>	Applies a level of depth to a 3D effect, or retrieves the current depth setting.
<a href="#">AttributeHiLightColour</a>	Sets the highlight color applied to the 3D effects raised, lowered or embossed, or retrieves the current highlight color setting.
<a href="#">AttributeLoLightColour</a>	Sets the lowlight color applied to the 3D effects raised, lowered or embossed, or retrieves the current lowlight color setting.
<a href="#">AttributeShadowColour</a>	Sets the shadow color when a shadowed 3D effect is used, or retrieves the current shadow color setting.

For details and a VB example on handling return and error values, see [Error Handling](#).

## See Also

[Automation Events](#)

[Function Categories](#)

## AttributeText

Sets the text for a text object, or retrieves the current text.

## Syntax

**AttributeText(*Text*)**

*Text*:

The text object's text as a string.

## Return Value

If retrieving the current text for the object, the text is returned as a string. If setting the text, a 0 (zero) is returned if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active text object, these functions throw an exception with a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AttributeTextStyle](#), [AttributeTextJustification](#), [AttributeTextFont](#), [AttributeTextFontSize](#), [AttributeTextColour](#)

## Example

```
' Sets the text for the currently text object
GraphicsBuilder.AttributeText = "TestText"

' Retrieves text for the current text object
MyVariable = GraphicsBuilder.AttributeText
```

This function is implemented in the C++ environment as two separate functions: put\_AttributeText sets the text for the currently selected text object, and get\_AttributeText retrieves the text for the current text object.

## AttributeTextColour

Applies a color to the selected text, or retrieves the current font color setting.

**Note:** As this function does not support True Color functionality, it has been superseded by the functions [AttributeTextOnColourEx](#) and [AttributeTextOffColourEx](#).

## Syntax

**AttributeTextColour(*TextColour*)**

*TextColour*:

A value between 0 and 255 representing the font color.

## Return Value

If retrieving the current font color, a value between 0 and 255. If applying a particular font color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E\_INVALIDARG. If there is no active text object, these functions throw an exception with a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AttributeText](#), [AttributeTextStyle](#), [AttributeTextJustification](#), [AttributeTextFont](#), [AttributeTextFontSize](#)

## Example

```
' Applies a color to the selected text
GraphicsBuilder.AttributeTextColour = 255

`Retrieves the current font color setting
MyVariable = GraphicsBuilder.AttributeTextColour
```

**Note:** This function is implemented in the C++ environment as two separate functions:  
put\_AttributeTextColour applies a color to the currently selected text, and get\_AttributeTextColour retrieves the current text color.

## AttributeTextOffColourEx

Applies the "off" color to the selected text, or retrieves the current font color setting.

## Syntax

**AttributeTextOffColourEx(*TextColour*)**

*TextColour*:

An RGB value.

## Return Value

If retrieving the current font color, an RGB value. If applying a particular font color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E\_INVALIDARG. If there is no active text object, these functions throw an exception with a return value of E\_HANDLE.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AttributeText](#), [AttributeTextStyle](#), [AttributeTextJustification](#), [AttributeTextFont](#), [AttributeTextFontSize](#)

## Example

```
' Applies a color to the selected text
GraphicsBuilder.AttributeTextOffColourEx = &hFF0000

`Retrieves the current font color setting
MyVariable = GraphicsBuilder.AttributeTextOffColourEx
```

---

**Note:** This function is implemented in the C++ environment as two separate functions:  
`put_AttributeTextOffColourEx` applies a color to the currently selected text, and  
`get_AttributeTextOffColourEx` retrieves the current text color.

---

## AttributeTextOnColourEx

Applies the "on" color to the selected text, or retrieves the current font color setting.

## Syntax

**AttributeTextOnColourEx(*TextColour*)**

*TextColour:*

An RGB value.

## Return Value

If retrieving the current font color, an RGB value. If applying a particular font color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error `E_INVALIDARG`. If there is no active text object, these functions throw an exception with a return value of `E_HANDLE`.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AttributeText](#), [AttributeTextStyle](#), [AttributeTextJustification](#), [AttributeTextFont](#), [AttributeTextFontSize](#)

## Example

```
' Applies a color to the selected text
GraphicsBuilder.AttributeTextOnColourEx = &hFF0000

`Retrieves the current font color setting
MyVariable = GraphicsBuilder.AttributeTextOnColourEx
```

---

**Note:** This function is implemented in the C++ environment as two separate functions: `put_AttributeTextOnColourEx` applies a color to the currently selected text, and `get_AttributeTextOnColourEx` retrieves the current text color.

---

## AttributeTextFont

Applies a specific font to the selected text, or retrieves the font setting.

## Syntax

**AttributeTextFont(*TextFont*)**

*TextFont*:

The font name as a string.

## Return Value

If retrieving the current font, the name of the font as a string, for example "courier". If applying a particular font, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error `E_INVALIDARG`. If there is no active text object, these functions throw an exception with a return value of `E_HANDLE`.

---

**Note:** For details on handling return and error values, see [Error Handling](#).

---

## Related Functions

[AttributeText](#), [AttributeTextStyle](#), [AttributeTextJustification](#), [AttributeTextFontSize](#), [AttributeTextColour](#)

## Example

```
' Applies the font Courier to the selected text
GraphicsBuilder.AttributeTextFont = "Courier"

' Retrieves the font setting
MyVariable = GraphicsBuilder.AttributeTextFont
```

---

**Note:** This function is implemented in the C++ environment as two separate functions: `put_AttributeTextFont` applies a font to the currently selected text, and `get_AttributeTextFont` retrieves the current font setting.

---

## AttributeTextFontSize

Applies a font size to the selected text, or retrieves the current font size.

## Syntax

**AttributeTextFontSize(*TextFontSize*)**

*TextFontSize*:

A value between 0 and 65535 representing the font size.

## Return Value

If retrieving the current font size, a value between 0 and 65535. If applying a particular font size, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active text object, these functions throw an exception with a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AttributeText](#), [AttributeTextStyle](#), [AttributeTextJustification](#), [AttributeTextFont](#), [AttributeTextColour](#)

## Example

```
' Applies the font size to the selected text
GraphicsBuilder.AttributeTextFontSize = 12

' Retrieves the font size
MyVariable = GraphicsBuilder.AttributeTextFontSize
```

**Note:** This function is implemented in the C++ environment as two separate functions:

`put_AttributeTextFontSize` sets the font size, and `get_AttributeTextFontSize` retrieves the current font size.

## AttributeTextJustification

Applies a specific justification setting to selected text, or retrieves the current text justification value.

## Syntax

**AttributeTextJustification**(*TextJustification*)

*TextJustification*:

A value depicting the type of justification used:

0 = left justified

1 = right justified

2 = centered

## Return Value

If retrieving the current text justification, a value between 0 and 2 depicting the type of justification used. If applying justification, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active text object, these functions throw an exception with a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AttributeText](#), [AttributeTextStyle](#), [AttributeTextFont](#), [AttributeTextFontSize](#), [AttributeTextColour](#)

## Example

```
' Applies right justification to the selected text
GraphicsBuilder.AttributeTextJustification = 1

' Retrieves the current text justification value
MyVariable = GraphicsBuilder.AttributeTextJustification
```

**Note:** This function is implemented in the C++ environment as two separate functions:  
`put_AttributeTextJustification` applies justification to the currently selected text, and  
`get_AttributeTextJustification` retrieves the current justification setting.

## AttributeTextStyle

Sets a specific text style, or retrieves the current text style setting.

## Syntax

**AttributeTextStyle(*TextStyle*)**

*TextStyle*:

A value depicting text style:

0 = normal

1 = bold

2 = italic

4 = underline

8 = strikeout

You can superimpose styles by adding the above values.

## Return Value

If retrieving the current text style, a value between 0 and 8 depicting the applied style. If applying a text style, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E\_INVALIDARG. If there is no active text object, these functions throw an exception with a return value of E\_HANDLE.

**Note:** For details on handling return and error values, see [Error Handling](#).

## Related Functions

[AttributeText](#), [AttributeTextJustification](#), [AttributeTextFont](#), [AttributeTextFontSize](#), [AttributeTextColour](#)

## Example

```
' Sets the normal text style
GraphicsBuilder.AttributeTextStyle = 0

' Retrieves the current text style setting
MyVariable = GraphicsBuilder.AttributeTextStyle
```

**Note:** This function is implemented in the C++ environment as two separate functions:

`put_AttributeTextStyle` applies a style to the currently selected text, and `get_AttributeTextStyle` retrieves the current text style setting.

## Using External Databases

You can store and update runtime data from your plant floor in a database external to Plant SCADA. You can also use Plant SCADA to send information from the database (such as a recipe) to I/O Devices in your plant.

Plant SCADA supports two types of databases:

- [dBASE Databases](#)
- [SQL Databases](#)

By using an external application to read and write the same database records, you can effectively interface to an external application through, for example, a relational database.

Plant SCADA provides two methods of communication with a database:

- using Devices
- using the [Using Structured Query Language](#).

Devices are most useful when the automatic reporting and logging features of Plant SCADA are used.

## dBASE Databases

The dBASE file format has become an industry standard for data storage, and Plant SCADA supports the standard dBASE format. You can use any database editor (that supports dBASE III files) to create a database, and to read and write database records.

To connect to a database, you need to define a Plant SCADA Device. Devices specify the format and location of the database, for example:

Name	Recipe
Format	{Name,16}{Water,8}{Sugar,8}{Flour,8} {Salt,8}{Yeast,8}{Milk,8}
Header	Name
File Name	[DATA]:RECIPE.DBF

Type	DBASE_DEV
Comment	Recipe Device (dBASE file)

## SQL Databases

SQL (Structured Query Language) also has become an industry standard. SQL is a command language that allows you to define, manipulate, and control data in SQL databases - and in other relational databases. Most database servers support SQL. SQL gives you direct access to database servers on other platforms, such as computers, mini-computers, and mainframe computers.

You can use the Plant SCADA Device functions to set up the format and locations of each of your SQL databases. Alternatively, use the SQL functions for direct control over SQL transactions.

You need to define a Plant SCADA Device to specify the format and location of the database, for example:

Name	Recipe

In the Format field, specify the field names in the SQL database, for example:

Format	{Name,16}{Water,8}{Sugar,8}{Flour,8} {Salt,8}{Yeast,8}{Milk,8}

The Header is the database connection string for ODBC connection, for example:

Header	DSN = ORACLEDATABASE

Enter the database table name in the File Name field:

File Name	RECIPE
Type	SQL_DEV
Comment	Recipe Device (SQL)

## See Also

[Using Structured Query Language](#)

## Using Structured Query Language

You can use Structured Query Language (SQL) functions for direct access to an SQL database, instead of accessing the database as a device. Using direct database access can provide greater flexibility.

The SQL functions provide access to SQL databases through any ODBC-compatible database driver, for example, MS Access, FoxPro or Paradox.

**NOTICE****PERFORMANCE DEGRADATION**

To avoid system and database performance degradation, please consider the following:

- Build queries that require an acceptable use of system resources (memory, CPU, threads).
- Limit scope of queries with parameter-driven WHERE clauses.
- Avoid using SELECT \* to return all columns.
- Do not perform queries with null values for date time parameters.
- Try to balance load between relational database management systems (RDBM) and SCADA, for example, consider sorting data on the client side rather than the server side.
- Consider using stored procedures to minimize the number of round trips to the server.

**Failure to follow these instructions can result in equipment damage.**

SQL functions provide absolute control and flexibility over connections and queries. Querying a database via the Device system is slower than constructing a query via a SQL function due to the generic nature of a device.

## ActiveX Data Objects

Plant SCADA 7.30 introduced the use of the ADO.NET library of SQL functions, which can be programmed using any of .NET managed languages, and is supported and further developed by its manufacturer. Further, the ADO.NET approach enables you to have multiple recordsets per connection, and multiple queries per recordset. Previous versions restricted you to 1 recordset per connection and 1 query per connection. You can also delegates connections, disconnections and query executions to separate application threads, meaning that SQL queries need no longer be blocking actions.

The technology facilitates connections to databases from various vendors and obtains data sets by executing either ANSI SQL commands or any database specific dialects of SQL (if a dedicated provider is used) such as T-SQL or PL/SQL.

Connections can be established to databases which:

- Have an ODBC driver
- Have an OleDB provider
- Use MS SQL Server.

Specifically, the Cicode SQL DB interface can be used to connect to either Historian or Historical Alarm databases as long as they use databases fulfilling one of the above conditions and store their data in types that can be read by the Cicode SQL interface.

## See Also

[Limitations When Using Legacy SQL Functions](#)

[Database Connection Objects](#)

[Connecting to an SQL Database](#)

[Basic SQL Scenarios](#)

## Alarm Server Database Specification

### Limitations When Using Legacy SQL Functions

The SQL library used in Plant SCADA prior to release 7.30 had a number of limitations that are documented in Tech Note TN6222, available from the AVEVA Knowledge and Support Center. This library was not designed to handle large scale SQL data transactions. For large volumes of data it is recommended that CitectHistorian be used. Note that new functions using ADO.NET do not share these limitations.

If you intend to use the embedded SQL library in Plant SCADA, consider the following:

- Use "Simple" model for a SQL database to limit transaction information logging to transaction log file.
- Use the fixed size of transaction log to restrict the log file growing.
- Back up the database regularly to keep the transaction log size under control. When SQL Server finishes backing up a database or its transaction log, it automatically truncates the inactive portion of the transaction log. If the transaction log file is full, your database transactions will cease.
- Keep working tables as small as possible. When Plant SCADA SQL adds or appends a new row to data table, it uses SELECT \* first to get column information. If there are hundreds and thousands of records in the table, this action will certainly hinder the performance.
- Use SQL Server trigger to remove records from the working tables to the permanent tables. In this way, sizes of the working tables that are directly interfaced with Plant SCADA are not growing unrestricted.

### Database Connection Objects

The new implementation introduces database connection objects as containers for parameters describing SQL connections to the database.

Users can execute SQL queries via DB connection objects and return either disconnected SCADA recordsets or the connected SCADA recordset holding results of the queries.

Disconnected SCADA recordsets have no direct association to their DB connection objects and their lifetime is not limited by the DB connection objects' lifetime. They should be closed in Cicode when no longer necessary.

The connected SCADA recordset is directly associated to the DB connection object. The relation is 1-1 and there cannot be more than one connected recordset per one connection. The associated connected recordset is called further a default recordset.

SQL functions from the previous SCADA are kept and their functionality, from a user's point of view, is the same as before (see Cicode functions for minor exceptions).

A set of new Cicode SQL functions is introduced together with the new ADO interface. These new functions can be divided into four categories:

- Creation/connection – functions servicing separate creation of DB connection objects and separate connection initialization
- Multiple recordsets per connection – functions enabling users to obtain and use handles to disconnected recordsets (the old database access methods limited the number of recordsets per connection to one)
- Parameterization – functions helping to provide more secure way of building SQL queries
- Multiple queries per connection – functions enabling users to obtain and use handles to queries (the old database access methods limited the number of queries per connection to 1)

## See Also

[Basic SQL Scenarios](#)

## Basic SQL Scenarios

Two basic scenarios of using the Cicode SQL database interface are:

- **Accessing and analyzing data in a CCicode procedure**

The CCicode SQL database interface enables users to create a connection object, connect it to a database, execute SQL queries and step through records in either the default connected recordset, using legacy Cicode functions, or a disconnected recordset, using the new Cicode functions. See examples in SQLCreate and SQLConnect Cicode function descriptions.

In previous versions, this scenario could be problematic because connecting to databases and accessing data might hold SCADA Kernel's Tasks and temporarily block the entire application. The new implementation delegates connections, disconnections and query executions to separate application threads. The SCADA kernel thread, which initializes one of those actions, is suspended and waits for synchronization while other threads can continue processing their tasks.

- **Accessing and displaying data on a graphics page**

Typically, this scenario is used when the result of a query is displayed on a graphic page in the form of a table with the possibility of scrolling forward and backward through the data.

In this case the data from a disconnected SCADA recordset is displayed on a graphic page using the SQLGetField and SQLIsNullField SQL functions.

Data pertaining to the SQL query is copied to the disconnected SCADA recordset just after executing the query. This behavior means that there is no additional data fetching when you iterate through disconnected recordsets later.

---

**Note:** When executing queries, there is a direct relation between data size, execution time and memory consumption: the more data fetched the longer execution time and the larger amount of physical memory taken by the SCADA recordset. Take care to design your queries so that they return only amount of data compatible with available resources and performance expectations.

---

Since recordsets reside in memory while active, it is your responsibility to close unused SCADA recordsets when they are no longer required.

Disconnected SCADA recordsets will not depend on the existence of either the database connection object or the SQL connection to a database. The recordsets can therefore be seen as disconnected immediately after executing a query. They do not synchronize with the source database after that point. Subsequently, the SQL connection to a database can either stay open or be closed.

## See Also

[Connecting to an SQL Database](#)

[Executing Legacy SQL Commands](#)

[Using a Transaction](#)

[Expressing Dates and Times in SQL](#)

## Connecting to an SQL Database

### Using Legacy Functions

Before you can use SQL commands, you need to connect to the SQL database. In previous versions of Plant SCADA you would use the SQLConnect function to provide the access. It has the format:

```
SQLConnect(sConnect);
```

where sConnect is the connection string, for example:

```
INT hSQL;  
hSQL = SQLConnect("DSN=DBASE_FILES;DB=C:\ODBC\EMP;LCK=NONE;CS=ANSI");  
! Connect to a dBASE Compatible Database File.
```

```
INT hSQL;  
hSQL = SQLConnect("DSN=EXCEL_FILE;DB=C:\ODBC\EMP;FS=10");  
! Connect to an Excel File.
```

```
INT hSQL;  
hSQL = SQLConnect("DSN=ORACLE_TABLES;SRVR=X:ACCTS;UID=SCOTT;PWD=TIGER");  
! Connect to an Oracle Database.
```

Refer to the documentation that accompanied your SQL server for details about connecting to an SQL database.

To close a legacy database connection, use the SQLDisconnect function with the handle to the connection object.

### Using ADO.NET Functions

The new interface requires two calls in order to open a connection to a database. The first call uses SQLCreate which prepares the database connection object. A second call SQLOpen establishes the SQL connection to the database.

SQLOpen can be used to reopen previously closed connections.

**Note:** SQL connections to databases can be closed sometimes by either databases or ADO.NET when certain conditions are fulfilled (for example through idle time expiration, active pooling mechanism logic).

To close an ADO.NET database connection, use the SQLClose function to close the connection and then the SQLQueryDispose with the handle to the connection object.

New functions can be used with the pre-existing ones. However, as a good programming practice, it is advised not to mix the old and the new interface. A hardware alarm will be generated if a SQLDisconnect is detected when there are connections opened with SQLCreate/SQLOpen, or if SQLClose/SQLDispose are detected and there are connections opened with SQLConnect.

### See Also

[Executing Legacy SQL Commands](#)

[Working with Recordsets](#)

### Working with Recordsets

ADO.NET uses disconnected recordsets to view data from a database. These are created by the function

SQLGetRecordset. A handle to the created recordset is returned and can be used in subsequent functions.

To access data from a specified column and row, use the SQLGetField function with a recordset handle. Null detection can be performed by using the SQLIsNullField function. See the code example in the SQLCreate function for a simple example.

ADO.NET allows you to create many queries per single database connection object. Queries are created using the function and disposed by the SQLQueryDispose function.

## See Also

[Parameterized Queries](#)

## Parameterized Queries

Parameterized queries are now possible with the new ADO.NET SQL interface. This provides the possibility to define queries with parameters which can be substituted later by concrete values (integers, reals or strings). This is especially useful when a query is built using end user data, such as user names and passwords, inserted via graphical pages (forms). When correctly applied, parameterized queries can prevent SQL injection attacks.

Parameters are defined for each DB connection object and concrete values are substituted into queries just before executing the function from the SQLExec family.

The parameters exist in DB connection objects as long as the objects exist. The function SQLParamsClearAll can be used to remove all parameters from a specified DB connection object. There is currently no way to iterate through all existing parameters in a DB connection object nor to remove a single selected parameter from the object's collection.

Each database provider (ODBC, OleDb, SQL Server) implements parameterized queries in different ways. As SQL queries can be very complicated, it is difficult to provide a generic converter for unifying the parameterization syntax. So to use parameterized queries, you should have knowledge of how they work on a particular provider and structure your Cicode accordingly.

See the examples in SQLParamsSetAsInt and SQLParamsSetAsString.

## Executing Legacy SQL Commands

SQL allows you to manipulate data in a non-procedural manner; you specify an operation in terms of what is to be done, not how to do it. SQL commands allow you to:

- Create tables in the database.
- Store information in tables.
- Select exactly the information you need from your database.
- Make changes to your data and to the structure of a table.
- Combine and calculate data.

The SQLExec() function executes any SQL command that your SQL server supports. For example, to create a database table, you would execute the SQL "CREATE TABLE" command:

```
SQLExec ( hSQL, "CREATE TABLE recipe ('Name' CHAR(16), 'Water' CHAR(8),
'Sugar' CHAR(8), 'Flour' CHAR(8), 'Salt' CHAR(8), 'Yeast' CHAR(8), 'Milk' CHAR(8))");
```

To add records into the database table, use the "INSERT INTO" command. The command has the following syntax:

```
INSERT INTO <filename> [(<col_name>, . . .)] VALUES (<expr>, . . .)
```

This command adds the values for each field in the table, for example:

```
SQLExec(hSQL, "INSERT INTO recipe VALUES ('Bread', '10', '5', '7', '1', '1', '2')");
```

Column names are optional; however, if you omit column (field) names, the values are inserted into the fields in the same order as the values.

To read data from an SQL database, use the SQL "SELECT" command. You can use the "SELECT" command to read an entire set of records, or a row, from the table. You can then use the SQLGetField() function to read the data in each field, for example:

```
SQLExec(hSQL, "SELECT * FROM recipe WHERE NAME = 'Bread'");  
If SQLNext(hSQL) = 0 Then  
PLC_Water = SQLGetField(hSQL, "WATER");  
PLC_Sugar = SQLGetField(hSQL, "SUGAR");  
PLC_Flour = SQLGetField(hSQL, "FLOUR");  
PLC_Salt = SQLGetField(hSQL, "SALT");  
PLC_Yeast = SQLGetField(hSQL, "YEAST");  
PLC_Milk = SQLGetField(hSQL, "MILK");  
END
```

To delete database records, use the SQL "DELETE" command. The command has the following syntax:

```
DELETE FROM <filename> [WHERE <conditions>]
```

This command deletes values from the table, for example:

```
SQLExec(hSQL, "DELETE FROM recipe WHERE NAME = 'Bread'");
```

## See Also

[Using a Transaction](#)

## Using a Transaction

You can use a database transaction for more sophisticated database operations. A database transaction allows you to execute a series of SQL commands and then either commit the changes to the database, or 'roll back' (cancel) the changes, for example:

```
SQLBeginTran(hSQL); ! Begin the transaction  
SQLExec(hSQL, "UPDATE recipe SET water = '12' WHERE NAME = 'Bread'");  
SQLExec(hSQL, "UPDATE recipe SET milk = '1' WHERE NAME = 'Bread'");  
IF . . . THEN  
    SQLCommit(hSQL); ! Commit the transaction  
ELSE  
    SQLRollBack(hSQL);! Cancel the transaction  
END  
Check the ODBC-compatibility level of your database driver if you cannot use transactions.
```

## See Also

[Expressing Dates and Times in SQL](#)

## Expressing Dates and Times in SQL

The way in which SQL dates are expressed depends upon the particular database system. With dBASE, you normally specify a date in braces, for example {02/18/95}. For Oracle, use the format: to\_date('02/18/95', 'MM/DD/YY'). Other ODBC drivers might require another format - a common ODBC format is: 'YYYY-MM-DD'.

---

**Note:** Date references in an external database have to be based on the Gregorian Calendar, or the database tables needs to be exported to text files before use in Plant SCADA. Dates in Microsoft Access database tables exported as text files are stored as Gregorian values.

---

### See Also

[Conversions between Database and Cicode Types](#)

### Database Independent Date-Time Syntax

For database independence, you can use the following syntax for dates and times:

[<format>'YYYY-MM-DD HH:MM:SS.FFFFFFF' ]

Where:

- <format> (the first character after the opening square bracket) needs to be one of the following:
  - d - date
  - t - time
  - dt - date and time.

Whether you are specifying a date, time, or date and time, you need to provide the complete 26 character string, for example:

[d'1995-02-18 00::00.000000' ]

Refer to the documentation that accompanied your SQL server for further information about SQL commands.

### Conversions between Database and Cicode Types

Any type of database field can be requested in statements. SCADA converts values of the fields to MBCS 8-bit strings which may not always be possible. For example:

- Single byte database strings or numbers can be converted to MBCS 8-bit strings, processed and displayed by SCADA
- Multi-byte strings can be converted to MBCS. Processing and displaying can be done as far as: SCADA supports MBCS and is properly parameterized, OS has Language packs installed and is properly parameterized
- Blobs (binary large objects) cannot be converted.

An error is returned when conversion to 8-bit strings is not possible.

**NOTICE****LACK OF DATA CONVERSION**

Ensure that all your database fields can be converted. Otherwise, when conversion is possible, some data may not be converted, but there is no indication that has occurred. For example when SCADA isn't properly parameterized.

**Failure to follow these instructions can result in equipment damage.**

There are a large number of custom DB types that need macros/functions to be called on the database side of the SQL connection for their proper conversion to strings.

**MS SQL Server DB types and only their subset**

MS SQL Server DB type	Format of the CiCode string returned by either SQLRecordsetGetField or SQLGetField*
CHAR(), NCHAR(), VARCHAR(), NTEXT, NVARCHAR(), TEXT, XML	The text is converted to MBCS 8-bit string.
BIT, BIGINT, INT, SMALLINT, TINYINT	Integer as a text string
DECIMAL(,), FLOAT, NUMERIC(,), REAL, MONEY, SMALLMONEY	Real as a text string.
UNIQUEIDENTIFIER	A unique id stored as a 16-byte binary value and usually converted to a GUID-like string
SQLVARIANT	The sqlvariant is a type which can encapsulate other types. The format of the resultant string depends on the encapsulated type and ability to convert it to a string.
DATE, DATETIME, DATETIME2(), DATETIMEOFFSET, SMALLDATETIME, TIME()	Date and time as a string. Only one format and UTC. If users wish to have date/time in a different format, have to use DB side conversion macros/functions in their SELECT queries.
BINARY(), VARBINARY(), IMAGE, GEOGRAPHY, GEOMETRY, HIERARCHYID, TIMESTAMP	No conversion possible. The returned string is empty. An error is set during the query execution by either SQLExec() / SQLNext() or SQLGetRecordset(). Moreover, Geography, Geometry and HierarchyID need additional CLR types not distributed with .NET platform. If need to use objects of those types in queries directly (without explicit conversions by macros), the CLR types need to be installed either separately or with SQL Server. Otherwise such queries return error 307. A separate setup with the CLR types can be downloaded from <a href="http://www.microsoft.com/">http://www.microsoft.com/</a>

<b>MS SQL Server DB type</b>	<b>Format of the CiCode string returned by either SQLRecordsetGetField or SQLGetField*</b>
	en-us/download/details.aspx?id=30440.

## MS Access types and only their subset

<b>MS Access DB type</b>	<b>Format of the CiCode string returned by either SQLRecordsetGetField or SQLGetField*</b>
Text, Memo, Hyperlink, Lookup Wizard	The text is converted to MBCS 8-bit string.
Yes/No, AutoNumber	Integer as a text string
Number, Currency	Integer as a text string or (if necessary) Real as a text string.
Date/Time	Date and time as a string. Only one format and UTC. If users wish to have date/time in a different format, have to use DB side conversion macros/functions in their SELECT queries.
OLE Object	No conversion possible. The returned string is empty. An error is set during the query execution by either SQLExec()/SQLNext() or SQLGetRecordset().

\* The string has to be shorter than 255 characters, otherwise an error is thrown.

The Cicode runtime engine tries to automatically convert strings to target data types when a cast operation is necessary. The following conversions are potentially allowed: STRING to INT and STRING to REAL.

## Exchanging Data with Other Applications

You can transfer data between Plant SCADA and other software for storage, analysis, and post processing, or to control and tune your Plant SCADA system.

Plant SCADA uses the following methods to exchange data:

- Using DDE (Dynamic Data Exchange) (DDE), where Plant SCADA can act as a:
  - **DDE server** providing tag values to requesting clients
  - **DDE client** to request data from other applications.
- Using ODBC Drivers (ODBC), where Plant SCADA functions as an ODBC server, allowing other applications to read Plant SCADA variables directly.
- By using a common [external database](#), where Plant SCADA and other applications use the same database to store and share information.

Plant SCADA also supports importing and linking variable tag data from external databases (see [Link Tags to an External Data Source](#)).

## Using DDE (Dynamic Data Exchange)

**Note:** Plant SCADA's DDE Server is disabled by default. To enable the DDE Server, set the parameter [\[DDE\]Enable](#) to 1 in the Citect.ini file. In addition, the Cicode function [DDEPost](#) is disabled on the DDE Server.

Microsoft Windows DDE allows the continuous and automatic exchange of data between different Windows applications on the same machine without the need for any user intervention. For example, your company's Production group might use a spreadsheet application to graphically represent plant-floor data (product output). This could be dynamically updated with the latest live data using DDE to read values directly from Plant SCADA.

Windows DDE uses the DDE protocol to send messages between applications that share data. Dynamic Data Exchange occurs between a DDE client application (which requests the data or service) and a DDE server application (which provides the data or service).

The DDE Client starts the exchange by establishing a conversation with the DDE server, and requesting data or services. The DDE server responds to these requests by providing the data or services to the DDE Client. The DDE Client terminates the conversation when it no longer needs the DDE server's data or services.

**Note:** As the DDE protocol is not designed for high-speed data transfer, the use of DDE is only appropriate when data communication speed is not critical.

- For information about DDE conversations, see [DDE Conversations and Client Syntax](#).
- To establish a DDE conversation between applications on the same computer, see [Setting up DDE Conversations](#).
- To establish DDE conversations between applications running on different computers over the same network, see [Network DDE](#).
- To establish DDE Conversations with the Plant SCADA tag database directly, see [Connecting to the Tag Database using DDE](#).

**Note:** When reading or writing to Plant SCADA tags using DDE, you might unknowingly add to your Plant SCADA license point count. Once the dynamic point count is greater than the license point count, the software protection mechanism will terminate Plant SCADA runtime. Therefore, when accessing tags via DDE, it's important to remain aware of how many points you have used. For details, see license point count in the Installation and Configuration Guide.

## DDE Conversations and Client Syntax

Two applications participating in Dynamic Data Exchange are said to be engaged in a **DDEconversation**. The application that initiates the conversation is the **DDE Client**, and the application that responds to the DDE Client is the **DDE server**.

An application can have several DDE conversations running at the same time. The application can be the DDE Client in some conversations (requesting data or services), and the DDE server (the data/service provider) in others. Each request or response in a DDE conversation specifies the data or service to be sent or received.

**Note:** A DDE conversation is sometimes referred to as a **channel** or a **link**.

The syntax sent by the DDE Client when it tries to establish a DDE conversation with the DDE server, consists of three parts:

- The name of the application to retrieve the data from.
- The file or topic name which contains the data to be retrieved.

- The cell range, value, field, or data item that's being requested.

These are combined in the format:

<DDE server application name> | <DDE Topic name>!<DDE Data item name>

Where:

- <**DDE server applicationname**> identifies the DDE server application.
- | (pipe character) separates the DDE server application name from the DDE Topic Name with no spaces between them.
- <**DDE Topic name**> identifies the context of the data. For DDE Servers that operate on file-based documents, DDE topic names are typically file names. For other DDE Servers, they are other DDE application-specific strings.
- ! (exclamation character) separates the DDE Topic Name from the DDE Data item name with no spaces.
- <**DDE Data item name**> is a string that identifies the data item that a DDE server can pass to a DDE Client during a DDE transaction. In some instances, the DDE Data item name is optional. Refer to the DDE application documentation for particulars.

---

**Note:** In the DDE Client syntax structure example above, every placeholder shown inside arrow brackets (<placeholder>) has to be replaced with the actual name of the item that it describes. Do not include the arrow brackets and the placeholder words they contain in the statement, and are shown here only for your information.

As the DDE protocol was designed in an era before long file names, DDE only supports the use of short (8 character) file names. To overcome this limitation, enclose the three parts of the DDE syntax within single quotes respectively. For example:

Citect|Variable! 'Process Variable 1'

This instructs DDE to treat the characters within the quotes as strings, thus permitting them to contain long file names, the space character ( ), the pipe character (|), the exclamation or bang character (!), or any other non alphanumeric character.

## See Also

[Setting up DDE Conversations](#)

### Setting up DDE Conversations

The DDE protocol itself does not support the launch of applications, so both the DDE Client application and the DDE server application needs to already be running before any DDE conversations can occur (unless the calling application is coded to detect and launch the DDE server application when necessary).

At the beginning of a DDE conversation, a DDE Client requests the services of a DDE server using DDE Client syntax (which contains the DDE server application name, topic or file name, and the data item name in the request). For DDE Client syntax details, see [DDE Conversations and Client Syntax](#).

To set up an application as a DDE Client, that is, to request data from a DDE server application, you need to use appropriate values in the DDE Client syntax as follows:

## DDE server application name

The DDE server name is usually the DDE server application name, for example the DDE server name for Plant SCADA is "Citect", the DDE server name for Microsoft Excel is "Excel", the DDE server name for Microsoft Word is "WinWord", and the DDE server name for Microsoft Access is "MSAccess". Most DDE Servers respond to only one name.

## DDE topic name

The DDE Topic name for Plant SCADA is either "**Data**" (if you use the Cicode DDE functions) or "**Variable**" (if you use Plant SCADA as the DDE server and want to access the variable tag database directly). The DDE Topic name for Microsoft Excel is the name of the worksheet (which may also include the workbook name enclosed in square brackets). The DDE Topic name for Microsoft Word is the document name. The DDE Topic name for Microsoft Access is the Database name and Table name, Query name or an SQL string as detailed in the following note:

---

**Note:** The proper DDE Client syntax of the DDE Topic name section for accessing a Microsoft Access database is constructed like this: "<DataBaseName>; TABLE <TableName>".

The < DataBaseName > placeholder is for the name of the Access database file followed by a semicolon ( ; ). You might have to include the file path; however this might not be the case (i.e. if it is known that Access will be running with the target file open). The .MDB suffix is optional (as .MDB is the default suffix for Access databases), unless the full path was included.

The **TABLE <TableName>** is the command string to instruct Access which table data you intend to converse with. DDE also supports the use of **QUERY <QueryName>** or **SQL <SQLString>** in place of **TABLE <TableName>**.

The use of the semi-colon ( ; ) after the '< DataBaseName >' placeholder, and the use of UPPERCASE for the 'TABLE', 'QUERY', and 'SQL' commands in the DDE string syntax are necessary. The whole section needs to be enclosed in quotes ( " ).

## DDE Data item name

The DDE Data item name for Plant SCADA depends upon the DDE Topic name being used. When using 'Variable' as the DDE Topic name to access the variable tag database directly, the DDE Data item name is the Plant SCADA variable tag name. When using 'Data' as the DDE Topic name to access a value posted using the Cicode DDEPost() function, the DDE Data item name is the posted name.

The DDE Data item name for Microsoft Excel is the cell range in Row number Column number format (for example R1C1). The DDE Data item name for Microsoft Word is a bookmark name. The DDE Data item name for Microsoft Access is dependant upon which topic name was used. Refer to the Microsoft Access Help for details.

---

**Note:** Microsoft introduced security measures with Office which, by default, block Office applications from being DDE Clients. For security details, see [Using DDE with Microsoft Office Applications](#).

Plant SCADA can perform as both a DDE server and as a DDE client as necessary. To set up Plant SCADA to use DDE, see [Exchanging Plant SCADA Data via DDE](#).

## DDE Function Types

There are two classes of DDE functions in Cicode, the original **DDE** functions and the later **DDEh** functions.

- **DDE functions**

The original Cicode DDE functions do not return a DDE Channel Number and were designed to insulate the user from the need to manage DDE Channels. The DDERead(), DDEPost(), DDEWrite(), and DDEExec() functions each perform a single exchange of data. Each of these functions starts a DDE conversation with the external application, sends or receives the data (or command), and ends the conversation - in one operation.

- **DDEh functions**

The Cicode **DDEh** functions were introduced to afford more control over DDE communications, especially for Network DDE and for circumstances where it is necessary to explicitly terminate and re-initiate a DDE Channel (after deleting rows from a table for example).

The DDE handle (*DDEh...*) functions return a handle to the conversation - a DDE channel number.

Use the DDEh handle functions for Network DDE and for Access DDE.

## See Also

[Exchanging Plant SCADA Data via DDE](#)

## Exchanging Plant SCADA Data via DDE

Plant SCADA runtime can exchange data as a DDE server or a DDE Client.

Plant SCADA behaves as a DDE server when providing other applications with access to its data. When acting as a DDE server, Plant SCADA runtime can:

- Provide DDE access to the complete variable tag database **automatically** with no further setup necessary
- Provide access to selected variable values by posting select Plant SCADA data using DDE

Plant SCADA behaves as a DDE client when requesting other applications to provide access to their data. When acting as a DDE Client, Plant SCADA runtime can:

- Read data directly from another application
- Write data directly to another application

---

**Note:** You can also execute commands in another application from Plant SCADA with the DDEExec() function.

Similarly, you can run Cicode functions from another application by passing the functions through that application's DDE Execute command.

---

## See Also

[Connecting to the Tag Database using DDE](#)

## Connecting to the Tag Database using DDE

Plant SCADA runtime behaves as a DDE server and automatically provides DDE access to the complete variable tag database with no further setup necessary.

To create DDE links to the Plant SCADA variable tags, use the DDE Client syntax. For syntax details, see [DDE Conversations and Client Syntax](#).

In the DDE Client call, the DDE Application name needs to be "Citect", the DDE Topic name needs to be

"Variable", and the DDE Data item name needs to be the Plant SCADA tag name.

For instance, the PV1 tag value can be accessed from a cell in Excel containing the following formula:

```
=Citect|Variable!PV1
```

If the Plant SCADA variable tag name contains spaces or non-alphanumeric characters, the DDE data item section of the DDE Client call syntax needs to be enclosed within single quotes. For example:

```
=Citect|Variable! 'Process Variable 1'
```

Plant SCADA runtime and the DDE Client application (for example Excel) need to both be running on the same computer. For information about DDE conversations, see [DDE Conversations and Client Syntax](#).

To establish a DDE conversation between applications on the same computer, see [Setting up DDE Conversations](#).

To establish DDE conversations between applications running on different computers over the same network, see [Network DDE](#).

## Posting Select Data using DDE

You might have a tag naming convention which is not DDE compatible, or inappropriate for use in a DDE call. Plant SCADA provides the ability to publish specific tags under different names in DDE. This involves using the DDEPost function.

To make selected Plant SCADA variable values or the results of calculations available to external DDE Client applications currently running on the same computer, use the Cicode DDEPost() function to have Plant SCADA runtime behave as a DDE server.

This conversation is one-way, which allows an external DDE Client application (like Excel, Word, etc.) to read the value from Plant SCADA using DDE. The Client application cannot change this value in Plant SCADA.

You can use this function to present data under a different name than its source. For instance, the following example presents the value of variable tag PV1 as "Process1".

The following Cicode example posts the value of variable PV1 using DDE:

```
DDEPost("Process1", PV1)
```

Once posted, this value can be accessed, for example, from a cell in Excel containing the following formula:

```
=Citect|Data!Process1
```

or from a field in Microsoft Word containing the following function:

```
{DDEAuto Citect Data Process1 }
```

### Notes:

- The name of the posted value (for example Process1) not to exceed 8 characters or contain spaces if you want to link to it via a Microsoft Word DDE field. Microsoft Excel, however, accepts long data item names if enclosed within single quotes (for example 'Process Variable 1').
- You need to use Data as the DDE Topic name when accessing posted values. See [Setting up DDE Conversations](#).

This method of DDE connection requires that both Plant SCADA runtime and the DDE Client application (for example Excel or Word) are running on the same computer at the same time. Once the DDE connection has been made, the DDE Client would normally display the updated value of the Plant SCADA DDE posting as soon as it becomes available, usually within milliseconds of the post.

Unfortunately, this posting method of DDE linking is subject to breakage whenever Plant SCADA runtime is closed, even if Plant SCADA runtime is subsequently restarted. The DDE Client application might not detect the

lack of connection, as updates to the data in the DDE Client (for example Excel) only occur after each post by the DDE server (Plant SCADA runtime). If no further posts occur, and in this scenario none will (as the DDE link is disconnected), the DDE Client application receives no update, and subsequently might display data which could be out of date. This disconnected state will remain until the DDE link in the DDE Client application is refreshed or the DDE Client application is restarted.

## See Also

[Writing Values to a DDE Application](#)

## Writing Values to a DDE Application

To write a Plant SCADA variable value directly to an external DDE server application currently running on the same computer as Plant SCADA, use the Cicode DDEWrite() function.

For example, writing data from Plant SCADA to a Microsoft Office Application (using Plant SCADA as the DDE Client and the Office application as the DDE server), could be done using the following Cicode DDEWrite() function examples:

```
! Write PV1 to Excel
! Assumes the existence of Sheet1
DDEWrite("Excel", "Sheet1", "R1C1", PV1);

! Write PV1 to Word
! Assumes the existence of TestDDE.doc already loaded in Word
! containing the bookmark named TagPV1
DDEWrite("Word", "TestDDE", "TagPV1", PV1);
! MS Access does not support direct DDE writes.
```

This DDE function is one-way, so to update the tag value in the remote DDE server application, this function will need to be called every time the value of this Plant SCADA variable changes.

Writing directly to a DDE server assumes a prior knowledge of the DDE server. In the above example, the spreadsheet or document needs to exist and Excel or Word (as appropriate) needs to be running for the DDE communication to be successful.

**Note:**

- Instead of using the once only DDEWrite(), you could use the multiple use DDE handle function DDEhPoke().
- Verify that remote requests are enabled in the other application. For example, in Excel, you need to verify the **Ignore other applications** check box is cleared (the default setting).
- The High security setting (if selected) on Microsoft Office does not appear to affect the use of remote DDE Client requests with those Office applications. See [Using DDE with Microsoft Office Applications](#).

## See Also

[Reading Values from a DDE Application](#)

## Reading Values from a DDE Application

To read a value into a Plant SCADA variable directly from an external DDE server application currently running on

the same computer as Plant SCADA, use the Cicode DDERead() function. For example:

```
PV1 = DDERead("Excel", "[Book1]Sheet1", R1C1);
```

The data from Row 1, Column 1 on Sheet 1 of Book 1 in Excel is read and stored in the Plant SCADA variable "PV1". This DDE function is one-way, and will need to be called whenever the value needs to be updated in Plant SCADA.

Reading from a DDE server assumes a prior knowledge of the DDE server. In the above example, Excel needs to be running and the appropriately named spreadsheet needs to exist. The Plant SCADA variable PV1 needs to also have been previously declared.

---

**Note:** Instead of using the once only DDERead(), use the multiple use DDE handle function DDEhRequest().

---

Verify that remote requests are enabled in the other application. For example, in Excel, you need to confirm that the **Ignore other applications** check box is cleared (the default setting).

The High security setting (if selected) on Microsoft Office does not appear to affect the use of remote DDE Client requests with those Office applications. See [Using DDE with Microsoft Office Applications](#).

## Using DDE with Microsoft Office Applications

Microsoft has introduced security measures with Office which, by default, block Office applications from being DDE Clients. See [Microsoft Office Security](#).

Microsoft Office applications appear to support varying degrees of long file names with DDE. See [Long File Names in DDE](#).

To enable DDE remote requests in Microsoft Excel, you need to verify the **Ignore other applications** check box is cleared (the default setting).

## Long File Names in DDE

According to MSDN Knowledge Base article Q109397, DDE does not support long file names, so DOS alias names needs to be used for directory and file names longer than eight characters (i.e. C:\mydocu~1\file.mdb).

Different Microsoft Office applications differ in their support for the use of long file names when used as DDE Clients. For instance, Microsoft Word does not appear to support the use of DDE data item names which exceed 8 characters, while Microsoft Excel however, accepts long data item names if enclosed within single quotes. See [Posting Select Data using DDE](#) for an example.

## Microsoft Office Security

In the interests of data security, Microsoft Office has its security settings set to high by default. To view or change your security level in Excel or Word, choose **Tools | Macro**, click **Security**, and click the **Security Level** tab.

- If you have your security level set to **High** (the default setting), then communication with external DDE Servers will not be available unless they are digitally signed and trusted. What you see in Excel cells that use the DDE function is #N/A , and with no additional explanation as to why the DDE functions aren't working. The High security setting (if selected) does not appear to affect the use of remote DDE Client requests with those Office applications as DDE Servers.
- If you set your security level to **Medium**, you are asked if you want to run any DDE Servers that are not digitally signed and trusted and that are referenced by DDE functions.

- If you set your security level to **Low**, external DDE Servers are run regardless of whether they are digitally signed and trusted, or not.

---

**Note:** If you need to manipulate another application's objects from Microsoft Office, consider using OLE Automation.

---

## Network DDE

Network DDE is a version of the DDE protocol for use across a network. For information about DDE, see [Using DDE \(Dynamic Data Exchange\)](#). For details about setting up Plant SCADA to use DDE, see [Exchanging Plant SCADA Data via DDE](#).

Just like the establishment of a DDE conversation between applications running on the same computer, a network DDE conversation uses the same DDE protocol to share data between applications running on separate computers connected to a common network.

Network DDE is a Windows Service used to initiate and maintain the network connections, security, and shared file space needed for using DDE over a network, and needs to be running on both computers at the same time. For startup details, see .

A network DDE trusted share needs to be manually created for the Network DDE server application on the Network DDE server application machine. This instructs the Network DDE Service (on the DDE server application machine), as to which application and topic to connect with. It is this share name which the Network DDE Client application can subsequently communicate with. For details, see [Setting up Network DDE Shares](#).

The Network DDE Client starts the Network DDE conversation by connecting to the Network DDE Share on the Network DDE server computer. For details, see [Using Network DDE](#).

## Starting Network DDE Services

For Network DDE to function, NetDDE.EXE needs to be installed and running on both machines before attempting to conduct a Network DDE conversation. NetDDE.exe is a Windows Service system file that is used to communicate the shared dynamic data exchange used by Network DDE. It has no graphical user interface (it runs as a background Windows service).

It is necessary to initiate the automatic activation of Network DDE Services, or manually run NetDDE.EXE on both machines before attempting connection.

### To manually start Network DDE services:

- On the Windows Start menu, click **Start | Run**, type in "netdde" (without the quotes) and press the **Enter** key. Do so on both machines.

### To automatically start the Network DDE Services on machine startup:

- Use the Windows Services Manager (select **Start | Control Panel | Administrative Tools | Services**) to set the **Network DDE** service from **Manual** to **Automatic**. To do so, right-click the service and select **Properties** from the pop-up menu. On the **General** tab select **Automatic** from the drop-down list of the **Startup type** field. Click **OK**. Close evry window and restart the machine.

**To verify that the NetDDE Services are running:**

- The Windows Task Manager lists NetDDE.exe on the **Processes** tab when running. To view the Windows Task Manager, press **CTRL+ALT+DEL**.
- The Service Administrative Tools also lists the status of Network DDE and Network DDE DSDM. To view the Windows Services Manager, select **Start | Settings | Control Panel | Administrative Tools | Services**.

---

**Note:** If you are using only the Microsoft Client Service for NetWare Networks, the NW IPX/SPX/NetBIOS compatible protocol needs to be enabled for NetDDE.exe to load.

---

**To test that Network DDE is operational between two machines on the same network**

Microsoft Windows ships with a network DDE application called Chat. It is installed in the system32 folder.

1. On the Windows Start menu, click **Start | Run**, type in "winchat" (without the quotes) and press the **Enter** key. Do so on both machines.
2. On one machine, select the Chat menu **Conversation | Dial** or click the dial button. The **Select Computer** dialog will display.
3. Select the other computer from the list, and click **OK**. Chat will attempt to establish a network DDE conversation between the computers.

---

**Note:** If Chat is not already running on the other computer, it times-out and states that the other computer didn't answer. If however, the other computer is already running Chat, it will keep dialling until answered.

---

4. On the other machine, (while it is being dialed), select the Chat menu **Conversation | Answer** or click the answer button. Type in a message and it will display in the Chat window on the other machine. The conversation will continue until either machine hangs up.

Once a Chat conversation is established, it proves that both machines are properly set-up and capable of handling network DDE conversations. You can view the share properties for Chat\$ by using the DDEShares.exe application as described in [Setting up Network DDE Shares](#).

You don't have to run Chat to use Network DDE with Plant SCADA. This Network DDE test only uses Chat as an example to confirm Network DDE functionality between two machines. Once you have established that Network DDE is functional, close the Chat windows, create the **Network DDE Trusted Share** for your Network DDE server application, and connect to the share using Network DDE. See [Connecting to a Network DDE Shared Application](#).

## Setting up Network DDE Shares

To be able to create a DDE link over a network, the computer serving as the Network DDE server needs to be setup to provide a **NetworkDDE Share** to establish a network DDE Channel.

---

**Note:** You only have to create a DDE Share if you intend to use DDE between two separate applications running on different machines, and you only have to create the DDE Share on the machine that contains the application which will be the DDE server.

---

The Windows DDESHARE.EXE utility enables users to manage DDE shares. The 32-bit version is shipped with each Microsoft operating system and is located in the Windows\System32 sub-directory.

**To manually launch the DDE Share utility:**

- On the Windows Start menu, click **Start | Run**, type in "ddeshare" (without the quotes) and press the **Enter** key.

When DDEShare.EXE is running, it displays the DDE Share utility window containing two icons which launch the DDE Shares dialog, and the DDE Trusted Shares dialog:

In the DDE Share utility, double-click the left icon (without the check mark) to display the DDE Shares dialog box:

The DDE Shares dialog is used to create, manage, and delete global DDE shares on your computer, and to view the DDE shares of any computer on the network. You can use this dialog to confirm the names of shares available on any machine on the same network. From the DDE Shares menu, select **Shares | Select Computer** and choose the computer name you're interested in from the list.

## DDE Shares

There are three types of DDE shares: old style, new style, and static. Plant SCADA only supports the static type. The names of static shares follow the convention

<ShareName>\$

so to set up a Plant SCADA server computer as a Network DDE share, use the name "Citect\$" as the sharename on that computer. To expose the Plant SCADA runtime variable tag database for suitable DDE linking, use the word "Variable" as the DDE Share Topic name.

**Note:** The trailing dollar sign (\$) is necessary as part of the DDE share name syntax.

### To create a DDE share:

1. In the DDE Shares dialog, click **Add a Share**. The **DDE Shares Properties** dialog appears. Verify that the fields are blank, or unchecked, except for the **Grant access to every item** radio button which needs to be checked.
2. Click **Permissions**. The DDE Share Name Permissions dialog appears.
3. **Read and Link** is the default permission setting. If you want to write data to the DDE Share application, change the permission to **Full Control**.
4. Click **OK**.
5. Click **OK** to save the Share, and return to the **DDE Shares** dialog.

## See Also

[Using DDE Trusted Shares](#)

## Using DDE Trusted Shares

When a network DDE Client user connects to a network DDE Share from a remote computer, Network DDE accepts the request only if both:

- The user who created the share has granted trusted status to the share.
- The user who created the share is currently logged on to the server computer.

To link to the Plant SCADA tag database, and permit write actions from an external application using Network DDE, the DDE Client computer needs to be granted appropriate Trusted status.

### To create a trusted share:

1. On the DDE Shares dialog, highlight the new 'Citect\$' share entry, and click **Trust Share to display the Trusted**

Share Properties dialog box.

2. Check **Initiate to Application Enable** to allow new connections to the DDE share.
3. Click **OK**.

#### To view the trusted shares:

1. In the DDESHARE utility double-click the right icon (with the check mark) to display the DDE Trusted Shares dialog.
2. The DDE Trusted Shares dialog lists the DDE shares that are trusted in the current user's context. You can view and modify trusted share properties and remove DDE shares from the list of trusted shares.
3. Once setup is completed, close the DDE Share utility dialog box.

## Using Network DDE

Microsoft Network DDE Service needs to be running on both computers to communicate using Network DDE. For startup details, see [Starting Network DDE Services](#).

Before a Network DDE Client can establish a DDE conversation with a Network DDE server application, the Network DDE server application computer needs to already have setup a **Network DDE Share**. For details, see [Setting up Network DDE Shares](#).

---

**Note:** You cannot connect using Network DDE to a shared application on the same machine. You can only connect using Network DDE to a shared application on another machine (which needs to also be on the same network).

---

To connect to a Network DDE shared application, you use an altered version of the DDE syntax, which replaces the "<ApplicationName>" with "<ComputerName>\NDDE\$" and replaces the "<TopicName>" with the Network DDE server Share "<ShareName>", and continues to use the "<DataItemName>" as normal.

At first glance, there appears to be no way to specify the DDE Application or Topic names in the Network DDE syntax call, and indeed, that is the case. However, the DDE Application and Topic names are defined in the DDE server Share settings. So, when the Network DDE server machine receives the call (from the Network DDE Client) containing the Share name, it knows which application and topic to connect with. See [Connecting to a Network DDE Shared Application](#).

## Connecting to a Network DDE Shared Application

The network DDE Client specifies the remote DDE server share in the normal DDE Client syntax by replacing the DDE Application name and DDE Topic name with the DDE server computer name and DDE server share name in the call. For DDE client syntax details, see [DDE Conversations and Client Syntax](#).

With Network DDE Client syntax, the DDE Application name is replaced with the following string enclosed in single quotes:

```
'\\<ComputerName>\NDDE$'
```

where "<ComputerName>" is the name of the computer running the DDE server application, and "NDDE\$" notifies Windows on the remote computer that the calling DDE Client wishes to establish a Network DDE channel. You cannot omit the NDDE\$ string, or it won't work.

The DDE Topic name is replaced with the following string also enclosed in single quotes:

```
'<ShareName>'
```

where "<ShareName>" is the name of the DDE Trusted Share previously set-up on the DDE server computer. The DDE Share on the DDE server machine contains the details of which application and topic to create the Network DDE link with. Most often, DDE server share names end with a \$ character.

---

**Note:** You need to use a separate DDE share name on the remote computer for each combination of DDE application name and DDE topic name you want to share. You can not declare the topic as a wild card (\*).

---

For example, to create a Network DDE link with the following criteria:

- Plant SCADA variable tag name: "PV1"
- Plant SCADA server computer name: "PlantSvr"
- Remote DDE Share name: "Citect\$"

you would construct a Network DDE Client call containing:

```
'\\PlantSvr\NDDE$' | 'Citect$' !PV1
```

In Excel, the following formula could be placed directly into a worksheet cell:

```
='\\PlantSvr\NDDE$' | 'Citect$' !PV1
```

If prompted for a username and password, use one that has appropriate permissions on the DDE server computer.

---

**Note:** You cannot omit the DDE syntax pipe character (|) or exclamation character (!), nor can you enclose those characters within quotes ('').

---

## Using ODBC Drivers

Plant SCADA supports the Open Database Connectivity (ODBC) standard. Many manufacturers of database packages also supply an ODBC database driver for their software. As well as these, there are independent parties manufacturing ODBC database drivers for various databases, such as Intersolv Q+E with their DataDirect ODBC Pack. Usually, any ODBC driver for your database will work.

### Installing the ODBC driver

You need to install and setup up your ODBC driver from the Windows Control Panel. To do this:

1. Open the Windows Control Panel.
2. Click the **ODBC** icon to start the ODBC setup utility (if you do not already have an ODBC icon in your control panel, you might need to install the icon. See the documentation provided with your ODBC driver).
3. Click **Add** to set up a DataSource.

---

**Note:** If your ODBC driver is not included in the list of Installed ODBC Drivers, return to the ODBC setup utility and install your driver (select the **Drivers...** button).

---

4. From the **Add Data Source** dialog box, select your ODBC driver. The Setup dialog is displayed.
5. Enter the Data Source Name of the driver that you used previously. For example, if you had used SQL with a dBASEIII database, then your connection string would have been "DRV=QEDBF". To avoid changing the connection strings throughout your project, use a Data Source Name of **QEDBF**.
6. Run your Plant SCADA project.

Some Cicode SQL functions will not work if your ODBC driver has limited functionality. This situation is not

frequently encountered, and in most cases affects only the ability to use transactions with the SQLBeginTran(), SQLCommit(), and SQLRollBack() functions. If you are using the Intersolv Q+E ODBC drivers, it is unlikely you will experience any loss of functionality.

You might need to change the data source in the Control Panel each time you switch from using one ODBC-compatible driver to another, for example from a dBASE file to an Access database. Click the ODBC icon and select from the list of available data sources. (Refer to the documentation supplied with your driver for more information.)

**Note:**

1. For maximum compatibility with the Cicode SQL functions, the ODBC driver has to provide a minimum of functions. For example, if your driver does not support the ODBC function SQLTransact, you cannot use the Cicode functions SQLBeginTran(), SQLCommit(), and SQLRollback().
2. Any functions you might have created in early versions are backward-compatible. Q+E drivers are now ODBC-compliant, so you need to upgrade your old Q+E database driver to a Q+E ODBC database driver.

## About the ODBC driver

Plant SCADA connects directly to the Microsoft Access ODBC driver, which allows applications to access information stored in MDB (Microsoft Access Database) files without actually running Microsoft Access. (Microsoft Access uses the "Jet Engine" DLL to access information stored in MDB files.)

ODBC normally implies heavy use of SQL statements to manipulate data. SQL statements can become quite complex and verbose. To implement them in Cicode they often have to be separated into sub-strings so that the maximum string length for Cicode variables is not exceeded.

With Access, it is possible to call queries that have been defined in Access so that the SQL statements become quite simple and straightforward. The Access tables & queries can be used to implement **RELATIONSHIPS** and **JOINS**, to **SORT** & **SELECT** only those rows (records) and return only those columns (fields) of particular interest at the time.

Developing queries in Access also has an advantage that the resulting Recordsets can be viewed in Access to test that they contain the data that is expected. The queries can incorporate SQL Functions (such as BETWEEN & AND).

The Jet Engine can also call upon the VBA Expression Service. This means that many non ANSI functions can also be used (both in SQL statements and Access Query Definitions) provided there is no need to migrate to a non Access system at a later date. Refer to VBA Functions Reference in the Access or Excel help system (only those functions with (VBA) after them and are appropriate to an SQL environment, are likely to work in an SQL statement).

## Setting up ODBC

To use ODBC, the Access ODBC Driver needs to be installed. This can be obtained from Microsoft and is included with Microsoft Office. The installation programme (for example, for Microsoft Office) will copy the necessary drivers and the Jet Engine DLL into the appropriate Windows directories when the appropriate Data Access/ODBC options are selected.

With the Driver installed on the PC the ODBC Icon can be selected from the Control Panel and a Data Service Name set up for the desired MDB. This is used in the DSN= part of the connect string.

The Jet Engine DLL is quite large (1 MB) and the execution speed of the runtime (and your application) can be impacted if the Windows Virtual Memory Manager (VMM) swaps it out of memory. The next time an SQL is

executed there will be short delay while the DLL is loaded back into memory. To force the VMM to keep the DLL in memory, design a simple dummy table with only one record and one field and set up a Cicode task that frequently (say every 10-15 seconds) calls a SELECT query based only on the dummy table. This has no significant effect on CPU load and keeps the DLL in memory.

## Getting the correct syntax with ODBC

The ODBC syntax for SQLs varies from the Access syntax in some ways. A good way to get the syntax correct and view the resulting Recordset is to use the query designer in **Microsoft Query** then copy the SQL text from it into Cicode. Because MS Query uses ODBC, any syntax that works in it will work when called via ODBC from Cicode. MS Query can also be used to confirm that the DSN is correct.

MS Query tends to create SQL text that is more complex than necessary. In particular it includes the path with the file name which is not necessary because the path is already defined in the DSN entry. It is considered bad practice to hard code file paths. MS Query also tends to prefix column (field) names with the table names to avoid any chance of ambiguity. Again this is not always necessary and it is desirable to keep the SQL text as brief as possible in your code.

The SQL statement text generated by the query designer can be pasted into Execute SQL window (under the File menu of MS Query), any surplus text removed and the SQL statement tested until the simplest syntax that works can be found. There is provision to save the SQL text if necessary. The final version of the SQL statement can be used with confidence in Cicode.

## Programming style with ODBC

Most of the sample code in the following topics do not include error checking and reporting:

1. Reading data from an access table with ODBC
2. Writing data to an access table with ODBC
3. Deleting rows from an Access table with ODBC
4. Calling action queries with ODBC
5. Parameter queries using ODBC

---

**Note:** These examples exclude error checking to keep them as simple as possible. However, both common sense and accepted programming practices mandate that you include error checking in your ODBC code.

Give consideration to implementing most of the complexity of queries in Access Query Definitions where they are easier to design and the results are easily viewed. A WHERE clause can be used when calling the query to select only the desired rows at run time. Where tables have many columns (fields), the Access Query Definitions can be used to restrict any particular call to view only the fields of interest.

It is helpful to build the SQL test up into strings. Firstly the ODBC function calls become simpler. Secondly the strings can be passed to TraceMsg() to make debugging simpler.

Remember that the Jet Engine runs on the same PC as Plant SCADA and that complex queries returning large Recordsets can have an adverse impact on CPU and memory resources. However, excessive impact on PC resources can be avoided by careful table, query and relationship design.

If there is a need to execute the queries on a **Remote Computer**, the code can set up on a Reports Server or an Event Server. This is especially relevant if the code is to be triggered by an event in a PLC. If the code is to be triggered by a User at a Display Station, and the query is considered too CPU intensive, the Display Station can be used to set the PLC bit that calls to code or call the Report using the Cicode Report() function. Another possibility

is to use the Cicode MsgRPC() function to call a Cicode function (with parameters, if necessary) on a remote computer. Each of these alternatives require Plant SCADA to be running on the remote computer.

## See Also

[Using Plant SCADA as an ODBC Server](#)

## ODBC Compatibility

This section describes the necessary and optional ODBC functions that your database driver needs to support:

### Essential functions

The Plant SCADA SQL devices and Cicode functions only work if the database driver supports the following ODBC functions:

Level 0	Level 1
SQLAllocConnect	SQLColumns
SQLAllocEnv	SQLDriverConnect
SQLAllocStmt	SQLGetData
SQLBindCol	SQLGetFunctions
SQLColAttributes	SQLGetInfo
SQLDescribeCol	SQLGetTypeInfo
SQLDisconnect	SQLParamData
SQLError	SQLPutData
SQLExecDirect	SQLSetConnectOption
SQLExecute	SQLSetStmtOption
SQLFetch	
SQLFreeStmt	
SQLGetCursorName	
SQLNumResultCols	
SQLPrepare	
SQLRowCount	
SQLSetParam	

## Optional functions

Plant SCADA SQL devices and Cicode functions also use the ODBC functions listed in the table below. While these functions are not necessary for operation, if they are absent in a driver, the error code 307 (SQL database error) will be returned if the ODBC driver does not support the necessary ODBC functions. To get the message associated with the error, call the Cicode function SQLErrMsg.

### Level 0

SQLTransact	If the driver does not support this function, the Cicode functions SQLBeginTran(), SQLCommit(), and SQLRollBack() are not supported.
-------------	--

### Level 1

SQLSpecial Columns	Plant SCADA uses this function if it is available. There is no loss of functionality otherwise.
SQLTables	(no effect on Plant SCADA)

### Level 2

SQLData Sources	The driver does not need to support this function. It is provided by ODBC.
SQLExtended Fetch	Without this function, Plant SCADA cannot take advantage of the native database's ability to fetch records at random.
SQLSetScroll Options	Without this function, Plant SCADA cannot take advantage of the native database's ability to fetch records at random.
SQLMore Results	(no effect on Plant SCADA)
SQLNativeSql	(no effect on Plant SCADA)
SQLProcedure Columns	(no effect on Plant SCADA)

## See Also

[Using Plant SCADA as an ODBC Server](#)

## Using Plant SCADA as an ODBC Server

ODBC server support allows Plant SCADA to function as an SQL database server. This will allow third-party applications that support ODBC to access data directly from Plant SCADA. This means that users can have direct access to data in Plant SCADA without having to develop Cicode or reports to export the data.

**Note:** A Plant SCADA ODBC Server is disabled by default. To enable an ODBC Server, you need to set the parameter [ODBC]Server to 1 in the Citect.ini file on the server computer.

Currently, the Plant SCADA ODBC server allows variable tags to be accessed. The table for the variable tags is named '**TAGS**' and the format is as follows.

NAME	Variable tag name	read only
VALUE	The current runtime value	read/write

Plant SCADA can only function as a database server at runtime. Using tags through ODBC at runtime can still add to your Plant SCADA License point count. Once the dynamic point count is greater than the license point count, the software protection mechanism will shutdown Plant SCADA runtime. Therefore, when accessing tags via the ODBC server, it's important to keep aware of how many points you have used. For details see License point count in the Installation and Configuration Guide.

### Setting up the Plant SCADA ODBC server:

You need to have TCP/IP installed on your computer first.

1. Choose **Start | Settings | Control Panel**.
2. Double-click the ODBC icon.
3. Click **Add** on the **User DSN** tab.

---

**Note:** If you select other tabs you will see that the Plant SCADA ODBC driver has been automatically installed.

---

4. Select the **Plant SCADA Driver** from the list and click **Finish**.
5. Enter "**Plant**" in the **Data Source** field. If you do not want to use this name, make sure the name you use is one word.
6. Enter the Computer Name in the **Host** field. The Computer Name is specified in the Network section of the Control Panel.
7. Click **OK**.

### Accessing the Plant SCADA ODBC server using MS Query (V2.00):

All ODBC capable applications use different ways to construct queries for accessing Plant SCADA tags. The example instructions for using MS Query, given here, show a simple implementation. MS Excel and MS Access use the same method.

1. Verify that you have MS Query installed on your computer.
2. Set up the Plant SCADA ODBC server.
3. Run Plant SCADA.
4. Run MS Query.
5. From the **File** menu (in MS Query) select **New Query**.
6. Select the Plant SCADA Data Source Name (DSN) from the Available Data Sources list. Click the **Use** button.
7. Select the **Tags** table. Click the **Add** button and then the **Close** button.
8. You can now run a query to extract the Tag data from Plant SCADA. The simplest way to see this is by double-clicking **Names** and **Tags**.

### Accessing the Plant SCADA ODBC server using MS Query (V8.00):

Unlike Version 2.00, User DSNS are not used by Version 8.00. Instead it uses File DSNS which by default are stored in **Program Files\Common Files\ODBC\Data** Source folder. File DSNS are not stored in the Windows registry,

they are text files given the .DSN extension. When you connect to an existing data source, only the available File DSNs that are stored on that PC are displayed. MS Query V8.00 does not display User or System DSNs. The simplest solution is to create a File DSN that points to a User DSN.

#### To create a file DSN that points to a user DSN:

1. Use a text editor (Notepad for example) and create a file containing the following two lines:

```
[ODBC]  
DSN=<MyUsrDSN>
```

where <MyUserDSN> is the name of an existing user DSN that you have created via the ODBC icon in the Control Panel.

2. Click **Save As** on the **File** menu and type a name that includes a .DSN file extension. For example, "Plant\_File.dsn" is a valid name. Include the quotation marks so that the .DSN file name extension is added correctly. Save it to the default File DSN directory listed above, then it will appear in the DSN list box.
3. Open the ODBC Manager from the Control Panel and verify that you can see your newly created File.DSN.
4. Open the ODBC Manager from the Control Panel and verify that you have created a User DSN called <MyUsrDSN>. For example:
  - Select Plant SCADA Driver and click **Finish**.
  - Type "Plant" in the **Data Source** field (i.e., <MyUsrDSN>).
  - Enter *Computer Name* in the **Host** field.

When you run MS Query, you can now select your File DSN from the list.

#### See Also

[Reading Data from an Access Table with ODBC](#)

### Reading Data from an Access Table with ODBC

You can use a SELECT query to read data from an Access table or to call an Access query.

A query is preferred over a table if there are many more columns in the table than are needed at the time, if the data needs to be sorted, or if you need to relate or join several tables. The Cicode necessary is as follows:

```
Function SQLTest  
    INT hSQL, iResult;  
    hSQL = SQLConnect("DSN=ODBCTest;UID=YourUID_C;PWD=YourPWD");  
    IF hSQL <> -1 Then  
        iResult = SQLExec(hSQL, "SELECT * FROM qryRecipes  
        WHERE Recipe Between '3000' And '6000'");  
        IF iResult = 0 Then  
            WHILE SQLNext(hSQL) = 0 DO  
                TraceMsg(">" + SQLGetField(hSQL, "Recipe")  
                + "<>" + SQLGetField(hSQL, "Flour")  
                + "<>" + SQLGetField(hSQL, "Water")  
                + "<>" + SQLGetField(hSQL, "Cocoa") + "<");  
            END  
            SQLDisconnect(hSQL);  
        ELSE  
            Message("SQL Error", SQLErrMsg, 48);  
        END
```

```
    END
ELSE
    Message("SQL Error", SQLErrMsg, 48);
END
END
```

## See Also

[Appending Data with ODBC](#)

## Appending Data with ODBC

To append data to an Access table using ODBC, you can use an SQL INSERT statement.

```
Function SQLInsert
    INT hSQL, iResult;
    hSQL = SQLConnect("DSN=ODBCTest;UID=YourUID;PWD=YourPWD");
    IF hSQL <> -1 Then
        iResult = SQLExec(hSQL, "INSERT INTO tblRecipes
        (Recipe, Flour, Water, Cocoa) VALUES ('X1234', 2, 3, 4)");
        SQLDisconnect(hSQL);
    END
END
```

To avoid having to deal with SQL statements, you can use the standard Cicode device functions to append records to an Access table. First configure an SQL device. If the table has many fields that do not need to be written to, define only those fields that are necessary in the device definition (this keeps the device definition simple and reduces the number of DevWrite instructions). DevOpen, DevWrite and DevClose can then be used to add records to the table.

Plant SCADA accepts successive DevWrites until they equal the number of fields in the device definition at which time it constructs an SQL INSERT statement. The DevWrites needs to contain data for fields in the same order as the device definition. Do a DevOpen followed immediately by successive DevWrites for as many records as are necessary then a DevClose to avoid data being out of context.

## See Also

[Editing Data with ODBC](#)

## Editing Data with ODBC

To edit data in an Access table there needs to be a unique (usually primary) key to identify the row (record) to be changed. This is to be able to provide a WHERE clause that will apply only to that row in an SQL UPDATE.

To edit data, read the data in the normal way, keeping track of the unique key. Any changed values can later be written to the same row using an UPDATE query with a WHERE clause.

```
Function SQLUpdate
    INT hSQL, iResult;
    hSQL = SQLConnect("DSN=ODBCTest;UID=YourUID;PWD=YourPWD");
    IF hSQL <> -1 Then
        iResult = SQLExec(hSQL, "UPDATE tblRecipes SET Flour = 20,
        Water = 30,Cocoa = 40 WHERE Recipe = 'X1234'");
```

```
SQLDisconnect(hSQL);
END
END
```

**Note:** The ODBC/SQL environment does not let you edit the "Current Record" (there is no current record). Consequently you cannot use DevAppend or DevSetField to add or modify records.

## See Also

[Deleting Rows from an Access Table](#)

## Deleting Rows from an Access Table

The DELETE keyword is used with a WHERE clause to delete the necessary row or rows. If the WHERE clause is not based on a primary key, more than one record may be deleted.

```
Function SQLDelete
    INT hSQL, iResult;
    hSQL = SQLConnect("DSN=ODBCTest;UID=YourUID_C;PWD=YourPWD");
    IF hSQL <> -1 Then
        iResult = SQLExec(hSQL, "DELETE FROM tblRecipes WHERE
        Recipe = 'X1234'");
        SQLDisconnect(hSQL);
    END
END
```

## See Also

[Calling Action Queries with ODBC](#)

## Calling Action Queries with ODBC

Access ACTION queries cannot be called in a SELECT query such as:

```
"SELECT * FROM qdeDeleteRecipe"
```

To call an Access ACTION via ODBC, use the Call statement in SQLExec:

```
{"{Call qdeDeleteRecipe}"}
```

The statement needs to be enclosed in {curly} braces.

The Call statement can be used to Call SELECT queries, the resulting Recordset being accessible in the normal way.

## See Also

[Parameter Queries](#)

## Parameter Queries

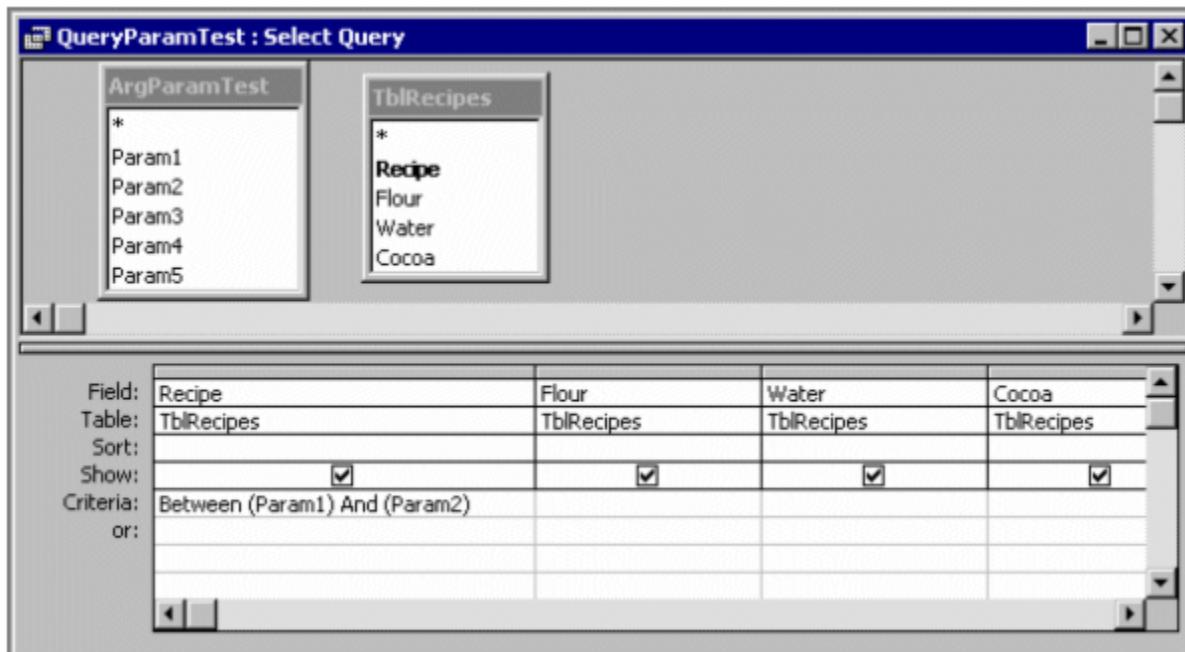
Many ODBC Servers will accept PARAMETERS in a Call statement so that PARAMETERS can be defined in a Query

Definition on the server and their values supplied by ODBC Clients at run time. Unfortunately, the Access Jet Engine uses Parameter Markers which are not supported in the standard Call statement. The method outlined here can be used as a work-around.

For each query that requires PARAMETERS, design a arguments table with the same name as the query but with a different prefix. For example, if the query is qryParamTest, the TABLE could be called argParamTest.

The TABLE could have, say, five fields called Param1, Param2, Param3, Param4, Param5.

Add this table to the Access Query Definition for qryParamTest. Then the field names can be used as PARAMETERS anywhere in the Query Definition.



A simple Cicode function can be written to which the query name (without the prefix) and PARAMETERS are passed. The function inserts "arg" in front of the query name and executes a DELETE from the TABLE (to verify that it is empty) and then performs an INSERT to leave the table containing ONE RECORD with the desired PARAMETERS in the appropriate fields. The function then prefixes the query Name with "qry" and calls the query.

```
Function SQLCall(INT hSQL, STRING sQueryName, STRING sArg1 = " ", STRING sArg2 = " ",
    STRING sArg3 = " ", STRING sArg4 = " ", STRING sArg5 = " ")
    STRING sTable, sQuery;
    sTable = "arg" + sQueryName;
    sQuery = "qry" + sQueryName;
    SQLExec(hSQL, "DELETE FROM " + sTable);
    SQLExec(hSQL, "INSERT INTO " + sTable + " (Param1, Param2, Param3, Param4,
    Param5) VALUES ('" + sArg1 + "', '" + sArg2 + "', '" + sArg3 + ',
    '" + sArg4 + "', '" + sArg5 + "')");
    SQLExec(hSQL, "{Call " + sQuery + "}");
END
```

Calling the Function from Cicode is then as simple as:

```
SQLCall(hSQL, "ParamTest", "2000", "4000");
```

The default parameters for SQLCall needs to be SPACES if "Allow Zero Length" is "No" in the Access table Definitions for fields Param1, Param2 etc.

The function can be used to call many different PARAMETER queries.

An advantage of this work-around is that, even after Plant SCADA has been shut down, the query can be called from Access and, because the PARAMETERS are still stored in the arguments table, the resulting Recordset can be viewed in Access.

Another method is to design queries that perform any necessary joins, sorting and field selection the call them using a WHERE clause to select the desired rows (records).

## See Also

[Access and Cicode Date/Time Conversions](#)

## Access and Cicode Date/Time Conversions

Access and Cicode have different Date/Time variables data types. Date references in an external database has to be based on the Gregorian Calendar.

You can convert between date and time in three ways:

- **Convert to real numbers** - In both Cicode and Access you can equate real numbers to data/time variables, like this:

```
AccessTime = 25568.66667 + (CicodeTime/86400);  
CitectTime = 86400*(AccessTime - 25568.66667);
```

- **Convert to strings** - The Data and/or Time are converted to and from text strings using the standard conversion functions available in each environment.
- **Use the #Date/Time# SQL syntax** - The Jet Engine will convert and Dates and Time strings enclosed in # markers. This date could be useful in a WHERE clause:

```
SELECT * FROM qryMyQuery WHERE 'Date' BETWEEN #3/20/96# AND #3/27/96#
```

The American Date format is used in this case, the Jet Engine DLL ignores the local Date and Time settings as set in Windows Control Panel.

## Integration with Historian

Plant SCADA supports the ability to automatically historize and publish variables within CitectHistorian. This capability offers a tightly integrated solution for collecting historical data, as variables can be passed to the Historian environment with minimal configuration. It is a particularly useful feature if your SCADA project has a large number of variables that need to be chronicled.

When a Plant SCADA project is added to Historian as a control system data source, the **Use SCADA Configuration To Historize and Publish** feature allows tags, trends and alarms to be imported and automatically configured within the Historian environment.

The automated configuration process will implement any equipment hierarchies defined in the source SCADA system, and apply the data acquisition method appropriate to the source variable type.

To prepare variables for automated configuration within Historian, you need to initially identify them within your Plant SCADA project. For more information, see [Preparing Variables for Historian Integration](#).

---

**Note:** CitectHistorian requires a valid license. Contact your local authorized AVEVA distributor for more information on the licensing options that are available.

## Preparing Variables for Historian Integration

To allow variables to be automatically configured in Historian, you initially need to identify the variables you would like included in this process within Plant SCADA. This involves adjusting the **Historize** property for each required variable to "true".

The following variable types are supported:

- Variable tags
- Trend tags
- Digital alarms
- Time-stamped alarms
- Analog alarms
- Advanced alarms
- Multi-digital alarms
- Time-stamped digital alarms
- Time-stamped analog alarms

For each variable you would like to be automatically configured in Historian, you need to set the **Historize** field to "true".

---

**Note:** If a variable has the Historian field set to "false", it will still be added to the Historian data sources directory when an import schema is run. You will still be able to manually historize and publish the variable within the Historian environment.

---

For more information, see the topic *Automating the configuration of SCADA attributes* in the CitectHistorian user documentation.

## Using Microsoft Excel to Edit DBF Tables

Plant SCADA allows you to edit and save .dbf files (tables) used in Plant SCADA by opening them in Microsoft® Office Excel®.

Microsoft Office Excel 2016 does not allow you to save files in .dbf format though you may open and edit them using the File > Open command. In order to overcome this limitation Plant SCADA includes an Add-In for Microsoft Excel called the **Plant SCADA Project DBF AddIn**. When this add-In is loaded into Excel, it allows you to browse, open, edit and save Plant SCADA .dbf files in the correct format.

The Project DBF AddIn is an optional component included in the installation of Plant SCADA and is accessed from the installer. If you did not select installation of the add-In during installation, you can install it at a later time by navigating to the ProjectDBFAddIn folder of your Plant SCADA installation and running the setup.exe in that folder.

## Compatibility

This add-In is specifically designed for use with Microsoft Excel 2016 or later. Earlier versions of Excel are not supported.

## Integration with Microsoft Excel

When you open Microsoft Excel, an additional toolbar, or an additional tab, called Project DBF Addin is available depending on the version of Excel that you are using.

The fields and buttons on this toolbar allow you to choose:

1. The path to the master.dbf.
2. A Plant SCADA project from the master.dbf .
3. A .dbf file (table) belonging to the selected project.
4. Save changes made to a .dbf file (table).
5. Save changes without re-indexing the table, or save and re-index.
6. Save a dbf file with a different name.
7. Open a dbf file in any location

However, you can also open .dbf files using the File > Open command.

## Protected Tables

The installation of the add-In incorporates a list of tables that by default are excluded from being opened using the add-In interface. This list is in a file called ExcludedProjectTables.xml and can be found in the add-In installation directory.

These tables are system tables that are maintained by Plant SCADA, and it is recommended that these files are not edited externally in Microsoft Excel. To minimize the likelihood of additional tables being edited with Microsoft Excel you can add their names to the ExcludedProjectTables file using a text editor. You can also view the default list of tables using a text editor, or an XML compliant browser.

## Read Only Tables

Projects which are read only, or contain read only tables, are displayed in the list of tables when selected from the Add-In buttons. If you try to open a read-only table (.dbf file) an alert (sometimes referred to as a "warning") message is displayed advising that the table is read only. You can still open and edit a read only table but you will not be able to save it.

A table could be read-only for the following reasons:

- The project is read-only.
- The table is open in the Plant SCADA Studio.
- The user trying to edit the file does not have read and/or write privileges.

If you attempt to save a read-only table an alert message is displayed advising that the table is read only.

## See Also

[Project DBF AddIn Functionality](#)

## Project DBF AddIn Functionality

The Project DBF Addin is automatically available to Excel when the add-in is installed by the Plant SCADA installation. When you open Microsoft Excel you will see a new toolbar called Project DBF Add-in, containing the following fields and buttons:

- Master.dbf location
- SaveDBF table
- SCADA project
- SCADA table
- Save DBF
- Save As
- Open DBF
- Save Only/Save and re-index

### To access the tables using the Add-In

1. If the location of the Master.dbf file is not displayed in the master.dbf location field, it will display Enter path to master.dbf. Click on this field to display a dialog box where you can enter the location. This populates the SCADA Project field.
2. Click the drop down arrow in the SCADA Project Field to select the Project in which the .dbf file (table) resides that you want to open. This populates the SCADA Table field.

---

**Note:** Any tables listed in the ExcludedProjectTables.xml are excluded from the list of tables.

3. Click the drop down arrow in the SCADA Table field to select the table. The table will open in Microsoft Excel as a new worksheet named with the table name. If the table is a read only table an alert (sometimes referred to as a "warning") message is displayed advising that it is read only.

Alternatively, if the dbf file that you wish to open is not the Master.dbf file, or is not located in the root dbf file location, click the Open DBF button and browse to the location of the dbf file that you want to open.

### To edit the contents of the table in Microsoft Excel

A table in .dbf format is displayed in Microsoft Excel in the following equivalent format:

- A field is displayed as a column.
- A record is displayed as a row.
- A field within a record (data) is displayed as a cell.

You may edit any of the data in the table, shown as cells within a row. You may delete a record shown as a row, and add a record. However you cannot change the structure of the table by modifying the name of any Field, shown as a column, add a column or delete a column. If you do change the structure an alert message will be displayed when you attempt to save the table.

If you edit a cell in the table take care not to exceed the maximum field length set for that field in the table.

### To save a table in Microsoft Excel

1. Select to Save and re-index the table, or Save Only, from the drop down control on the toolbar.

If you do not re-index an indexed table the table will be saved more quickly, but you will need to manually re-index the table when you have completed your edits. To re-index manually, in the **Project** activity, select **Home**, and then click **Pack** on the Command Bar. To pack included projects, click **Pack with Included Projects**.

2. Click the SaveDBF table button.

When you save the table the Add-In performs the following checks prior to saving the table:

- Verifies that the file type is .dbf.
- Checks for any changes in table structure that may have been made. For restrictions on changes to the table structure, refer to "To edit the contents of the table in Microsoft Excel".
- Updates the existing table according to the data in the Microsoft Excel worksheet.
- Re-indexes the updated table if it is was an indexed table (if Save and re-index was selected).

If the edited table in the Microsoft Excel worksheets does not pass every check, an alert (sometimes referred to as a "warning") message is displayed and the existing table on disk will not be affected until you correct the condition that caused any check to not pass.

You can also save a dbf file with a different name by clicking the Save As button on the toolbar and entering a name for the file. This method of saving also includes the option of saving the dbf file with or without re-indexing the table.

---

**Note:** If you attempt to save a table in .dbf format in Microsoft Excel using the Microsoft Excel File > Save command an alert message is displayed advising you that the file cannot be saved in the current format and that use the SaveDBF button on the Microsoft Excel toolbar.

---

## Using an OPC AE Server

OPC (OLE for Process Control) is a set of communication standards maintained by the OPC Foundation for the industrial automation industry. OPC provides a common platform for applications to share data from typically disparate sources, such as PLCs and databases, without the need to comprehend native protocols.

The OPC Alarms and Events (OPC AE) specification is similar to the original OPC Data Access specification, except that it addresses alarm and event data instead of real-time process control data. Plant SCADA supports a runtime OPC AE Server that supports the OPC AE interface specifications.

---

**Note:** To view Plant SCADA alarm information such as areas and privileges via OPC AE, your OPC AE client needs to support the capability to log on as specific user. This will implement the "IOpcSecurityPrivate" interface.

---

To get started, you need to register an OPC AE Server on a computer running a Plant SCADA alarm server. This is achieved via the General Options page of the Plant SCADA Setup Wizard (see [Setup Wizard - General Options Setup](#)).

You can then use a third-party OPC client application, such as Matrikon, to access your system and view alarm and event data. See the topic [Using a Third-party OPC Client to Access Alarm Server Data](#) for configuration details.

---

**Note:** The OPC AE Server does not support DCOM for remote communications. It is recommended that you run any connecting OPC client applications on the computer that is registered as the OPC AE Server.

---

## Using a Third-party OPC Client to Access Alarm Server Data

You can use a third-party OPC client application, such as Matrikon, to access your Plant SCADA system and view alarm server data. To do this, you need to initially register a Plant SCADA alarm server as a runtime OPC AE Server. This requires administrative privileges.

**Note:** The OPC AE Server does not support DCOM for remote communications. It is recommended that you run any connecting OPC client applications on the computer that is registered as the OPC AE Server.

### To connect a third-party client application to an OPC AE Server:

1. Run the Setup Wizard.
2. On the **General Options Setup** page, dialog select **Register**.  
The will register the alarm server as an OPC AE server and make it accessible to the third-party OPC applications.
3. Run the third party application.
4. From the list of available servers select the following:

Serck.ScxV6OPCAE.<AlarmServerName>.<IP address>

Where:

<Server Name> is the name of the registered alarm server.

<IP Address> is the address defined in the Network Address settings.

5. Connect to the OPC AE Server and create an event subscription within your third party application.

The events retrieved from the OPC AE Server will be listed. The information returned to the OPC client may differ slightly to what is presented on Plant SCADA's alarm pages.

**Note:** Do not attempt to connect an OPC DA application to a Plant SCADA alarm server. This is not supported and will not be enabled in any future releases. This is only intended for use with OPC AE clients.

### See Also

[Using an OPC AE Server](#)

[General Options Setup](#)

## Using an OPC DA Server

OPC (OLE for Process Control) is a set of communication standards maintained by the OPC Foundation for the industrial automation industry. OPC provides a common platform for applications to share data from typically disparate sources, such as PLCs and databases, without the need to comprehend native protocols.

The OPC Data Access solution (OPC DA) provides specifications for client and server applications that are focused on the continuous communication of real-time data. To this end, Plant SCADA supports a runtime OPC DA server that implements the mandatory OPC DA v2.05 and OPC DA v3 interface specifications.

This allows Plant SCADA to provide real-time data to any compliant OPC DA clients, including applications such as AMPLA, OSI-PI and Historian.

In order to allow OPC DA Server to be accessed by the OPC DA Clients, the OPC DA Server process must be configured in the SCADA project. This behavior is different to releases 7.20 and earlier where no configuration

step was required.

**Note:**

- The OPC DA Server does not fully support WriteVQT functionality, as a Plant SCADA I/O server does not allow quality and timestamp changes that are generated from an external source.
- Connecting a Plant SCADA OPC DA client to an Plant SCADA OPC DA server is not recommended, as the client was not designed to be used in this way. This mode of operation has not been validated.
- The OPC DA server uses regular expressions. For example, if retrieving tags with 'bit' in the name, you need to define "bit.\*\*" instead of "bit\*".

---

To determine which OPC DA interfaces are supported by the Plant SCADA OPC DA Server, see the topic [Supported OPC DA Server Interfaces](#) in the reference section of this chapter.

## See Also

[Configure an OPC DA Server](#)

[OPC DA Server DCOM Settings](#)

[Operating the OPC DA Server at Runtime](#)

[OPC DA Server Diagnostics](#)

[OPC DA Server Reference Information](#)

## Configure an OPC DA Server

A Plant SCADA OPC DA Server is configured in the Topology activity.

**To add an OPC DA Server process:**

1. In the **Topology** activity, select **Edit**.
2. From the drop down menu below the Command Bar, select **OPC DA Server**.
3. Add a new row to the grid.
4. Type the required information in each column, or in the Property Grid (see [OPC DA Server Properties](#) for a description of each property).
5. Click **Save**.

You can configure as many OPC DA servers as required in a project, however you can only have one per computer.

---

**Note:** To successfully operate an OPC DA Server, you will need to correctly configure its DCOM settings based on your client requirements. See the topic [OPC DA Server DCOM Settings](#).

## See Also

[OPC DA Server Properties](#)

## OPC DA Server Properties

An [Using an OPC DA Server](#) has the following properties:

## General Properties

Option	Description
Server Name	A unique name for the OPC DA server.
Comment	Any useful comment (optional).

## Communication Properties

Option	Description
Comment	Any useful comment (optional).
Network addresses	The IP address of the computer that will host the OPC DA server. You can only configure one server per computer.

## Other Properties

Option	Description
Browsing Hierarchy	<p>The field allows you to select the type of hierarchy to use for OPC item browsing. The options are:</p> <ul style="list-style-type: none"> <li>• Flat</li> <li>• Hierarchy</li> </ul> <p>Selecting <b>Flat</b> will present tags at a single level represented by their tag names. Selecting <b>Hierarchy</b> presents the tags in a directory structure that represents the clusters and equipment definitions that are used.</p> <p>No selection is the default, which is the equivalent of selecting <b>Flat</b>.</p> <p>For more information, see <a href="#">OPC Item Browsing</a>.</p>
Field	Selecting <b>True</b> will expose the Field tag extension as an OPC DA vendor-specific property of the OPC item. No selection is the default, which is the equivalent of selecting <b>False</b> .
Valid	Selecting <b>True</b> will expose the last valid value for the Valid tag extension as an OPC DA vendor-specific property of the OPC item. No selection is the default, which is the equivalent of selecting <b>False</b> .

Option	Description
OverrideMode	Selecting <b>True</b> will expose the OverrideMode tag extension as an OPC DA vendor-specific property of the OPC item. No selection is the default, which is the equivalent of selecting <b>False</b> .
Override	Selecting <b>True</b> will expose the Override tag extension as an OPC DA vendor-specific property of the OPC item. No selection is the default, which is the equivalent of selecting <b>False</b> .
ControlMode	Selecting <b>True</b> will expose the ControlMode tag extension as an OPC DA vendor-specific property of the OPC item. No selection is the default, which is the equivalent of selecting <b>False</b> .
Status	Selecting <b>True</b> will expose the Status tag extension as an OPC DA vendor-specific property of the OPC item. No selection is the default, which is the equivalent of selecting <b>False</b> .

## Project Properties

Option	Description
Project	The project in which the OPC DA server is configured.

## See Also

[Configure an OPC DA Server](#)

## OPC DA Server DCOM Settings

To manage user access to an OPC DA Server, it is recommended that you create a Windows user group with appropriate DCOM settings on the host server rather than dealing with Windows user identities individually. You can then just add users to this group as required.

See the Windows documentation for information on how to create a user group. Once the required user group has been created, you need to configure the required DCOM settings.

**Note:** These settings do not apply to an OPC AE Server. Plant SCADA's OPC AE Server does not support DCOM.

## To configure the machine-wide user group DCOM settings

1. Launch the Windows Component Services manager. To do this, go to Control Panel, open **Administrative Tools** and then **Component Services**.

2. Expand the **Component Services** folder, and the **Computers** folder.
3. Right click on the **My Computer** folder, and select **Properties**.
4. Go to the **COM Security** tab.
5. In the **Access Permissions** section , click on the **Edit Limits** button. Make the following adjustments:
  - add the OPC DA Server users group you have created
  - allow both **Local Access** and **Remote Access** for the users group
  - click **OK**In the **Access Permissions** section, now click on the **Edit Default...** button. Make the following adjustments:
  - add the OPC DA Server users group you have created
  - allow both **Local Access** and **Remote Access** for the users group
  - click **OK**
6. In the **Launch and Activation Permissions** section, click on the **Edit Limits** button. Make the following adjustments:
  - add the OPC DA Server users group you have created
  - allow **Local Launch**, **Remote Launch**, **Local Activation** and **Remote Activation** for the users group
  - click **OK**
  - You can now exit the Properties dialog.

## To configure the OPC DA Server specific settings

1. Launch the Windows Component Services manager. To do this, go to Control Panel, then **System and Security, Administrative Tools** and then **Component Services**.
2. Expand the **Component Services** folder, the **Computers** folder, the **My Computer** folder, and the **DCOM Config** folder.
3. Locate the "AVEVA SCADA OPC DA Server" component and select **Properties**.
4. Go to the **Security** tab.
5. In the **Launch and Activation Permissions** section, select **Customize** and click on the **Edit** button. Make the following adjustments:
  - add the OPC DA Server users group you have created
  - allow **Local Launch**, **Remote Launch**, **Local Activation** and **Remote Activation** for the users group
  - click **OK**
6. Go to the **Identity** tab. This is where you define which user accounts can run the OPC DA Server. The setting you choose will have the following implications:
  - **The interactive user** is the default option. This means the OPC DA Server will run using the security context of the Windows user currently logged in to the local computer. If there is no active Windows user logged in, or if the current user identity doesn't have the launching and activation permissions for the OPC DA Server, a connection will be unsuccessful.
  - **The launching user** - Each login session invisibly spawns multiple instances of the Runtime Manager, the OPC DA Server and the client if multiple users connect at the same time. This setting is considered a resource consuming option.

- **This user** allows you to identify a specific user. A connection will not be successful if there is already an instance of the Runtime Manager running under the active Windows session. Similarly, launching the Runtime Manager using a local Windows login will be unsuccessful if an instance of the Runtime Manager has already been launched by a DCOM connection. However, this option does avoid the situation where multiple instances of the Runtime Manager and the OPC DA Server are launched.
7. Once you have selected an option, you can exit the Properties dialog.

## To configure the connectivity environment settings

The way you configure a server's connectivity settings depends on whether it is on a domain or part of a workgroup. The following points describe how you should set up different client/server combinations.

- **If the server is on a domain and the client is on a domain:**

On the server computer, add the domain login identity that the client uses to the OPC DCOM users group you have created.

- **If the server is on a domain and the client is part of a workgroup:**

Create a matching Windows login identity on the server with the same password as the Windows login identity on the client machine. Add this Windows login identity to the OPC DCOM users group you have created.

- **If the server is part of a workgroup and the client is on a domain:**

Create a matching Windows login identity on the server with the same password as the domain login identity on the client machine. Add this Windows logon identity to the OPC DCOM users group you have created.

- **If the server is part of the same workgroup as the client:**

Create a matching Windows login identity on the server with the same password as the Windows login identity on the client machine. Add this Windows login identity to the OPC DCOM users group you have created.

---

**Note:** The registry entry for OPC Client application needs to be configured to accept callbacks. An indication that this is not being done as required, is that all synchronous OPC DA APIs work as expected but data updates and other asynchronous operation never complete.

---

**Note:** If you intend to run your OPC DA server as a service, you will also need to specify the user account under which the service will operate. For more information , see [Running an OPC DA Server as a Service](#).

---

## See Also

[OPC DA Server Properties](#)

## Operating the OPC DA Server at Runtime

An OPC DA Server operates as part of the Plant SCADA server management infrastructure, which means it can be controlled and monitored by the Runtime Manager.

The Runtime Manager will display the operational status and messages for the OPC DA Server as it does for other Plant SCADA processes. Be aware, however, that you will need to run the Runtime Manager locally on the OPC DA Server.

You can also configure an OPC DA Server to start in response to a client connection. See [Starting the OPC DA Server on a Client Connection](#).

## See Also

[OPC Item Browsing](#)

## Starting the OPC DA Server on a Client Connection

The OPC DA server can be configured to start when a request is received from a client.

When the parameter [OpcDaServer]AutoStartByClient is enabled, the Runtime Manager will start the OPC DA server process when a client request is received.

## See Also

[OPC DA Server Properties](#)

## OPC Item Browsing

When you create an OPC DA server in Plant SCADA, you can determine if it will present items to a client application as a hierarchy (representing the clusters and equipment definitions used), or as a flat structure.

The ability to use a flat structure allows the server to comply with the item browsing definitions supported by version 2.x of the OPC DA standard.

You can set the type of browsing supported by an OPC DA server via the **Browsing Hierarchy** field on the [OPC DA Server Properties](#) dialog. By default, the browsing hierarchy will be set to flat.

The browse position will be initially set as the root of the address space. After the initial call to browse items, the client will be able to continue browsing from the last browse point.

In the case where a SCADA project is configured with a large number of tags it is recommended to test browsing of OPC DA items to establish if excessive memory consumption occurs on IOservers or OpcDaServer processes running in the system. In general you should attempt to browse tags using third party OPC DA Client installed on the machine where the OPC DA Process is running.

---

**Note:**

1. Data Type filtering is not supported in the browsing function. This is due to the fact that Plant SCADA data types are not known until tags are subscribed to, if the user interface and other clients are subscribed at different points in time, clients could get a different list of tags for the same filter being applied. This is an accepted relaxation of the OPC Foundation CTT rule.

2. The actual "fully qualified" subscription name for an OPC DA Item might differ to the name returned by the OPC DA Browsing APIs, which is in accordance with OPC DA v2.05 and 3.0 specification. For example, in a multiclustered system "Tag1" might be defined in several clusters so its "fully qualified" item id might be "Cluster1.Tag1". Subscription to "Tag1" might still work where there is no ambiguity. In general it is recommended that OPC DA Clients obtain "fully qualified" ITEM ID via GetItemID if OPC DA v2.05 is used, or from OPCITEMPROPERTY structure if OPC DA v3.0 is used.

---

## See Also

[Configure an OPC DA Server](#)

## OPC DA Server Diagnostics

There are two ways you can monitor the performance of your OPC DA server:

### OPC DA Server Logging

Logging for the OPC DA Server is enabled through the Citect.ini parameter [\[Debug\]CategoryFilter](#), which allows you to specify the messages that are sent to the tracelog.dat file.

In the case of an OPC DA server, the tracelog file will use a name based on the following:

tracelog.OpcServer.<OPC server name>.dat

[Debug]CategoryFilter will support the following category definitions:

- OpcDaServerComApi - enables traces on a COM API level. Only return status of each OPC DA COM API if logged
- OpcDaBrowse205 - enables tracing related to OPC DA v2.05 item browsing
- OpcDaBrowse300 - enables tracing related to OPC DA v3.00 item browsing
- OpcDaServer - enables extensive tracing of OPC DA server object APIs
- OpcDaGroups - enables extensive tracing of OPC DA group object APIs

### OPC DA Server Kernel Statistics

A Kernel page is available that displays low-level diagnostic information about the runtime status of the OPC DA server. It can be accessed from the main kernel window by typing "Page Table OPCServerConnections".

The information available from this table includes the following:

- OPC Handle - the handle which the OPC server uses to reference a particular OPC connection. This handle can be cross referenced with log file entries.
- Index - the server object index, which is incremented each time a new client connects.
- #Groups - the number of groups created within the OPC server instance (including both active and inactive groups).
- #Items - the number of items created within OPC server instances (including all active/inactive items in all groups).
- #Reads - the number of reads that have been requested for items.
- #Writes - the number of writes that have been request for items.
- #Updates - the number of updates that have been received for subscribed items.

For more information on using the Kernel's Table window, see the topic [Page Table](#) in the *Using The Kernel* section of the Plant SCADA help.

## OPC DA Server Reference Information

The available reference information for an OPC DA Server includes the following topics:

- [Supported OPC DA Server Interfaces](#)
- [Client Request Data Type Conversions](#)
- [OPC DA Server Mandatory Properties.](#)

### Supported OPC DA Server Interfaces

The following table indicates which OPC DA interfaces are supported by the Plant SCADA OPC DA Server. A highlighted field indicates that the particular OPC API is supported by the server.

<b>OPC DA Server required interfaces</b>	<b>Version 1.0</b>	<b>Version 2.0</b>	<b>Version 3.0</b>
<b>OPC Server</b>			
IUnknown	Required	Required	Required
IOPCServer	Required	Required	Required
IOPCCommon	N/A	Required	Required
IConnectionPointContainer	N/A	Required	Required
IOPCItemProperties	N/A	Required	N/A
IOPCBrowse	N/A	N/A	Required
IOPCServerPublicGroups	Optional	Optional	N/A
IOPCBrowseServerAddressSpace	Optional	Optional	N/A
IOPCItemIO	N/A	N/A	Required
<b>OPC Group</b>			
IUnknown	Required	Required	Required
IOPCItemMgt	Required	Required	Required
IOPCGroupStateMgt	Required]	Required	Required
IOPCGroupStateMgt2	N/A	N/A	Required
IOPCPublicGroupStateMgt	Optional	Optional	N/A

<b>OPC DA Server required interfaces</b>	<b>Version 1.0</b>	<b>Version 2.0</b>	<b>Version 3.0</b>
IOPCSyncIO	Required	Required	Required
IOPCSyncIO2	N/A	N/A	Required
IOPCAsyncIO2	N/A	Required	Required
IOPCAsyncIO3	N/A	N/A	Required
IOPCItemDeadbandMgt	N/A	N/A	Required
IOPCItemSamplingMgt	N/A	N/A	Optional
IConnectionPointContainer	N/A	Required	Required
IOPCAsyncIO	Required	Optional	N/A
IDataObject	Required	Optional	N/A

## See Also

[OPC DA Server Reference Information](#)

## Client Request Data Type Conversions

The following topics describe the outcome of converting Plant SCADA and OPC data types during read and write requests by an OPC client.

- OPC Client Read Requests
- OPC Client Write Requests

## See Also

[OPC DA Server Reference Information](#)

## OPC Client Read Request Conversions

The following table describes the outcomes that can be expected when converting Plant SCADA data types to OPC data types during client read requests.

<b>OPC Data Types (To)</b>	<b>Plant SCADA Data Types (From)</b>							
BYTE	INT	UINT,	LONG	ULONG,	REAL	STRING	DIGITAL	

<b>OPC Data Types (To)</b>	<b>Plant SCADA Data Types (From)</b>							
		<b>BCD</b>		<b>LONG-BCD</b>		<b>(256 bytes)</b>		
<b>I1</b>	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range	OK
<b>UI1</b>	OK	Overflow if out of range	OK					
<b>I2</b>	OK	OK	Overflow if out of range	OK				
<b>UI2</b>	OK	Overflow if out of range	OK	Overflow if out of range	OK			
<b>I4</b>	OK	OK	OK	OK	Overflow if out of range	Overflow if out of range	Overflow if out of range	OK
<b>UI4</b>	OK	Overflow if out of range	OK	Overflow if out of range	OK	Overflow if out of range	Overflow if out of range	OK
<b>R4</b>	OK	OK	OK	OK	OK	OK	Overflow if out of range	OK
<b>R8</b>	OK	OK	OK	OK	OK	OK	Overflow if out of range	OK
<b>CY</b>	OK	OK	OK	OK	OK	OK	Overflow if out of range	OK
<b>DATE</b>	OK	OK	OK	OK	OK	OK	Overflow if out of range	OK
<b>BSTR</b>	OK	OK	OK	OK	OK	OK	OK	OK
<b>BOOL</b>	OK	OK	OK	OK	OK	OK	Overflow if out of	OK

OPC Data Types (To)	Plant SCADA Data Types (From)						
							range

**Note:** For read operations that result in an overflow, OPC E\_FAIL will be returned and read quality will be set to BAD.

## See Also

[OPC Client Write Request Conversions](#)

[OPC DA Server Reference Information](#)

## OPC Client Write Request Conversions

The following table describes the outcomes that can be expected when converting Plant SCADA data types to OPC data types during client write requests.

OPC Data Types (From)	Plant SCADA Data Types (To)							
BYTE	INT	UINT, BCD	LONG	ULONG, LONG- BCD	REAL	STRING (256 bytes)	DIGITAL	
I1	Overflow if out of range	OK	Overflow if out of range	OK	Overflow if out of range	OK	OK	Overflow if out of range
UI1	OK	Overflow if out of range	OK	OK	OK	OK	OK	Overflow if out of range
I2	Overflow if out of range	OK	Overflow if out of range	OK	Overflow if out of range	OK	OK	Overflow if out of range
UI2	Overflow if out of range	Overflow if out of range	OK	OK	OK	OK	OK	Overflow if out of range
I4	Overflow if out of range	Overflow if out of range	Overflow if out of range	OK	Overflow if out of range	OK	OK	Overflow if out of range
UI4	Overflow if out of	Overflow if out of	Overflow if out of	Overflow if out of	OK	OK	OK	Overflow if out of

OPC Data Types (From)	Plant SCADA Data Types (To)							
	range	range	range	range				range
<b>R4</b>	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range	OK	Overflow if out of range
<b>R8</b>	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range	OK	Overflow if out of range
<b>CY</b>	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range	OK	Overflow if out of range
<b>DATE</b>	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range
<b>BSTR</b>	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range	Overflow if out of range	>Overflow if out of range
<b>BOOL</b>	OK	OK	OK	OK	OK	OK	OK	OK

**Note:** For write operations that result in an overflow, OPC E\_FAIL will be returned and the target SCADA value will not be changed.

## See Also

[OPC Client Read Request Conversions](#)

[OPC DA Server Reference Information](#)

## OPC DA Server Mandatory Properties

These properties are mandatory and need to be implemented by all OPC DA Servers.

**Note:** The property IDs 2, 3 and 4 may not provide actual values in some cases. Refer to the Notes column below for more information.

Property ID	Description	Variant data type	Notes
1	"Item Canonical DataType" (VARTYPE stored in an I2)	VT_I2	Variant data type of the item.
2	"Item Value" (VARIANT)	Depends on DATATYPE	This value might not be

Property ID	Description	Variant data type	Notes
	<p><b>Note:</b> the type of value returned is as indicated by the "Item Canonical DataType" above and depends on the item.</p> <p>This will behave like a read from DEVICE.</p>		valid at the time the properties were obtained and it can only be considered if Property ID 4 (Quality) is set to GOOD.
3	<p>"Item Quality" (OPCQUALITY stored in an I2).</p> <p>This will behave like a read from DEVICE.</p>	VT_I2	<p>Quality of the value.</p> <p>In the majority of cases it will be set to BAD_WAITING_FOR_INITI AL_DATA and in that case Property ID 2 (Value) should not be considered as useful.</p>
4	<p>"Item Timestamp" (will be converted from FILETIME).</p> <p>This will behave like a read from DEVICE.</p>	VT_DATE	<p>Timestamp of the value.</p> <p>If Property ID 2 is set to BAD_WAITING_FOR_INITI AL_DATA, this value might represent the timestamp obtained by the OPC DA Server process and that might be on fixed offset to the actual timestamp provided by the I/O server.</p> <p>Actual data provided by the Read or OnDataChange will bring correct timestamp as stamped by the device.</p>
5	<p>"Item Access Rights" (OPCACCESSRIGHTS stored in an I4)</p>	VT_I4	Access rights.
6	<p>"Server Scan Rate" in milliseconds.</p> <p>This represents the fastest rate at which the server could obtain data from the underlying data source. This value generally represents the</p>	VT_24	Scan rate

Property ID	Description	Variant data type	Notes
	<p>optimum fastest RequestedUpdateRate which could be used if this item were added to an OPCGroup.</p> <p>The accuracy of this value can be greatly affected by system load and other factors.</p>		

## See Also

[OPC DA Server Reference Information](#)

# Plant SCADA Access Anywhere

Plant SCADA Access Anywhere enables you to remotely access a running Plant SCADA client from a mobile device (such as a tablet or smartphone) or a laptop computer.

You can view and control the client through a web browser without the need to install anything. This allows you to interact with the Plant SCADA client from anywhere in real-time.

The Access Anywhere documentation includes the following guides:

- [Access Anywhere Quick Start Guide](#) — an introductory guide to Access Anywhere that will get most users up and running with a basic configuration.
- [Access Anywhere Installation and Configuration Guide](#) — a more comprehensive description of the installation process that includes advanced configuration information.
- [Access Anywhere Secure Gateway Installation Guide](#) — provides instruction for installing Secure Gateway, a complementary component that provides encrypted remote access to Plant SCADA clients.
- [Access Anywhere Web Client User Guide](#) — describes how to use an HTML5-compatible web browser to remotely connect to a Plant SCADA client.

## Access Anywhere Quick Start Guide

Welcome to the AVEVA Plant SCADA Access Anywhere Quick Start Guide. This guide provides an overview of the steps that need to be carried out so that you can use your Plant SCADA Access Anywhere.

## About Plant SCADA Access Anywhere

Plant SCADA Access Anywhere enables you to remotely access a running Plant SCADA client with a mobile device including tablets, smart phones and laptop computers.

You can view and control the client through a web browser without needing to install it on your portable device. You can interact with the Plant SCADA client from anywhere in real-time.

---

**Note:** Availability of a Plant SCADA floating license is verified every time a user attempts to connect to a Plant SCADA client using Access Anywhere. For more information, see [Licensing](#).

---

## Getting Started

To use Plant SCADA Access Anywhere to remotely connect to Plant SCADA clients, you need to:

1. Install Plant SCADA and check that you are able to connect to the Plant SCADA client.
2. Install and configure the Access Anywhere Server on the computer where Plant SCADA is installed.  
See [Installing and Configuring the Access Anywhere Server](#).
3. Configure the Windows Firewall on the computer where the Access Anywhere Server is installed.  
See [Configuring Windows Firewall](#).
4. Configure Remote Desktop Service (RDS) on the computer where the Access Anywhere Server is installed.

See [Configuring Remote Desktop Service](#).

5. Configure user access to the Plant SCADA clients on the computer where the Access Anywhere Server is installed.

See [Configuring User Access to Plant SCADA Clients](#).

If your Plant SCADA Access Anywhere Server will be accessed by computers outside the operations network, it is recommended that you also install the **Access Anywhere Secure Gateway**.

For more information, see [Installing and Configuring the Access Anywhere Secure Gateway](#).

After you have completed the steps listed above, you will be able to connect to a Plant SCADA client using Plant SCADA Access Anywhere.

See [Connecting Access Anywhere to a Plant SCADA Client](#).

---

**Note:** This guide only provides an overview of the steps described above. Detailed information for each step is available in different guides, a reference to which is provided at the end of each section. Look for the heading *For more information*.

## Licensing

Plant SCADA Access Anywhere is licensed for use with the following Plant SCADA / Citect SCADA versions:

- 2018
- 2018 R2
- 2020 R2
- 2023
- 2023 R2.

In order to run Plant SCADA Access Anywhere, licenses are required for the following:

- **Microsoft Remote Desktop Service (RDS)**

Every remote desktop session opened with a browser using Plant SCADA Access Anywhere consumes an RDS license. Confirm that the required number of RDS licenses is available to users.

- **Plant SCADA License(s)**

Access Anywhere requires Plant SCADA to have a control/view-only client license available for each Access Anywhere client session.

## For more information...

Refer to the following topics in the *Plant SCADA Access Anywhere Server Installation and Configuration Guide*:

- [Overview](#)
- [Configuring the Plant SCADA Access Anywhere Server](#)

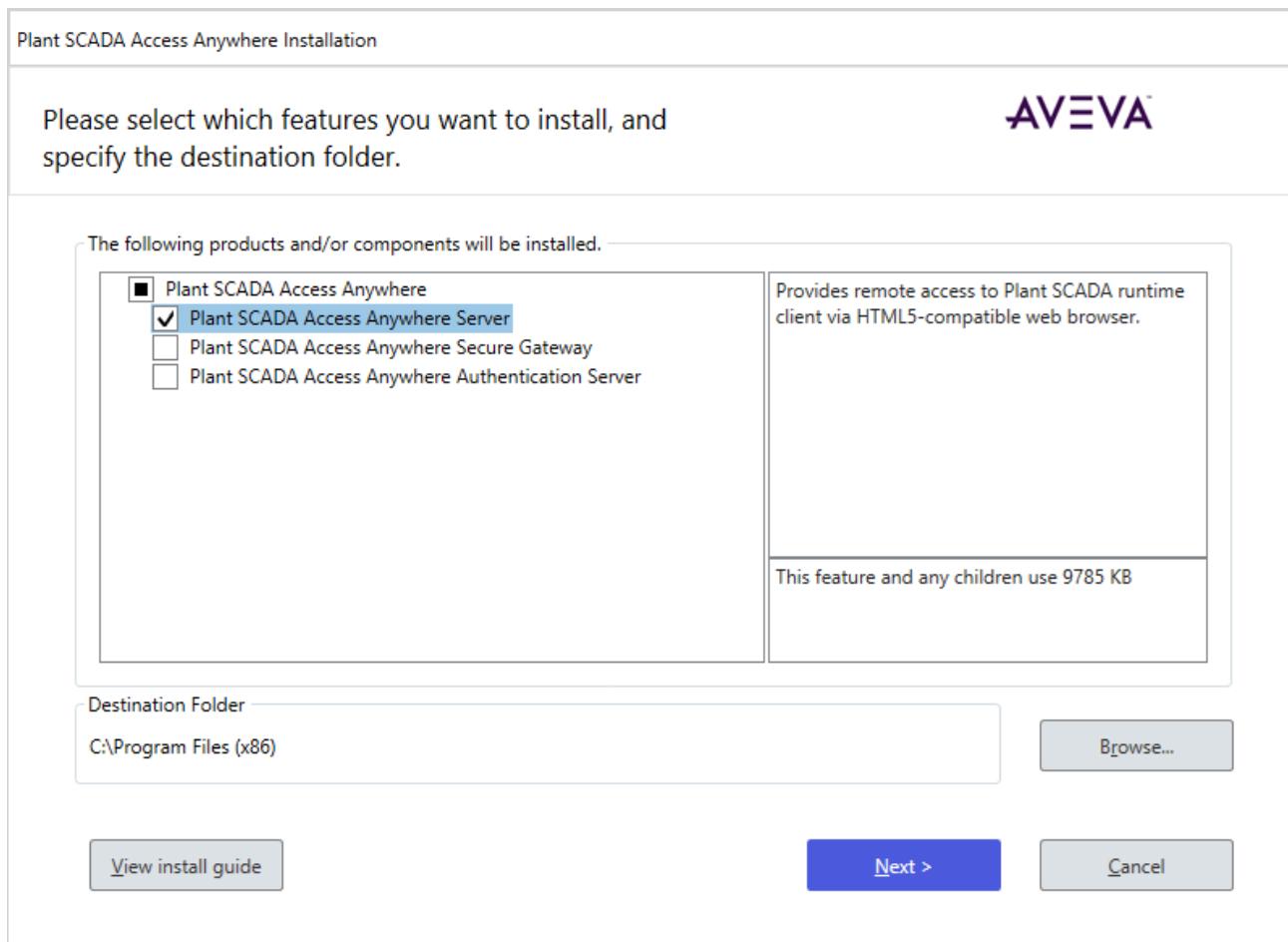
## Installing and Configuring the Access Anywhere Server

A basic installation of Plant SCADA Access Anywhere Server usually takes about five minutes. Check that all installation prerequisites have been met before starting the installation procedure.

**Note:** Before you install Plant SCADA Access Anywhere 2023, you need to uninstall any earlier versions. This means an existing server configuration file will no longer be available. If you want to migrate your existing server settings to the new version, see the topic [Migrate an Existing Configuration from an Earlier Version](#) in the *Plant SCADA Access Anywhere Server Installation and Configuration Guide* before you upgrade.

### To install a Plant SCADA Access Anywhere Server:

1. Run the Access Anywhere installer file setup.exe from your installation media.
2. Select **Plant SCADA Access Anywhere Server** when presented with the options to select the installation component.



3. Click **Next** through every dialog box, accept the EULA and then click **Finish**.
4. Configure the Plant SCADA Access Anywhere Server using the Configuration Console.

AVEVA Plant SCADA Access Anywhere Server 2.0 Configuration

General	Performance	Communication	Acceleration	Security	Logging	Advanced
---------	-------------	---------------	--------------	----------	---------	----------

Plant SCADA Access Anywhere Server service state: Running

Plant SCADA Access Anywhere Server status: Active

Number of sessions: 0

Started at: 16/11/21 21:29:38

Start Server

Stop Server

---

The Plant SCADA Access Anywhere Server and the Ericom Licensing Server run as a [Windows service](#). When the Plant SCADA Access Anywhere Server service is stopped, Plant SCADA Access Anywhere Clients will not be able to connect to this host with RDP acceleration. Any active accelerated connection will be dropped (sessions will be disconnected). Regular RDP connections to this host will not be affected.

Changing certain settings, such as **Communication** and **Logging** take effect only after the services are restarted. After changing such settings, apply the changes, stop the services and then start them again.

## For more information...

Refer to the following topics in the *Plant SCADA Access Anywhere Server Installation and Configuration Guide*:

- [Installation](#)
- [Configuring the Plant SCADA Access Anywhere Server](#)

## Configuring Windows Firewall

If the Windows Firewall is enabled on the computer where the Access Anywhere Server is installed, configure it to enable the Plant SCADA Access Anywhere client connection.

1. Open the **Windows Control Panel** and then **Windows Firewall**.
2. Select **Advanced Settings** and select **Inbound Rules**.
3. Click **New Rule**.
4. Select **Port** and click **Next**.
5. Enter the specific port: 8080. Click **Next**.
6. Select to apply the rule on the **Domain**, **Private** and **Public** networks. Click **Next**.
7. Assign a name for the rule.
8. Click **Finish**.

## For more information...

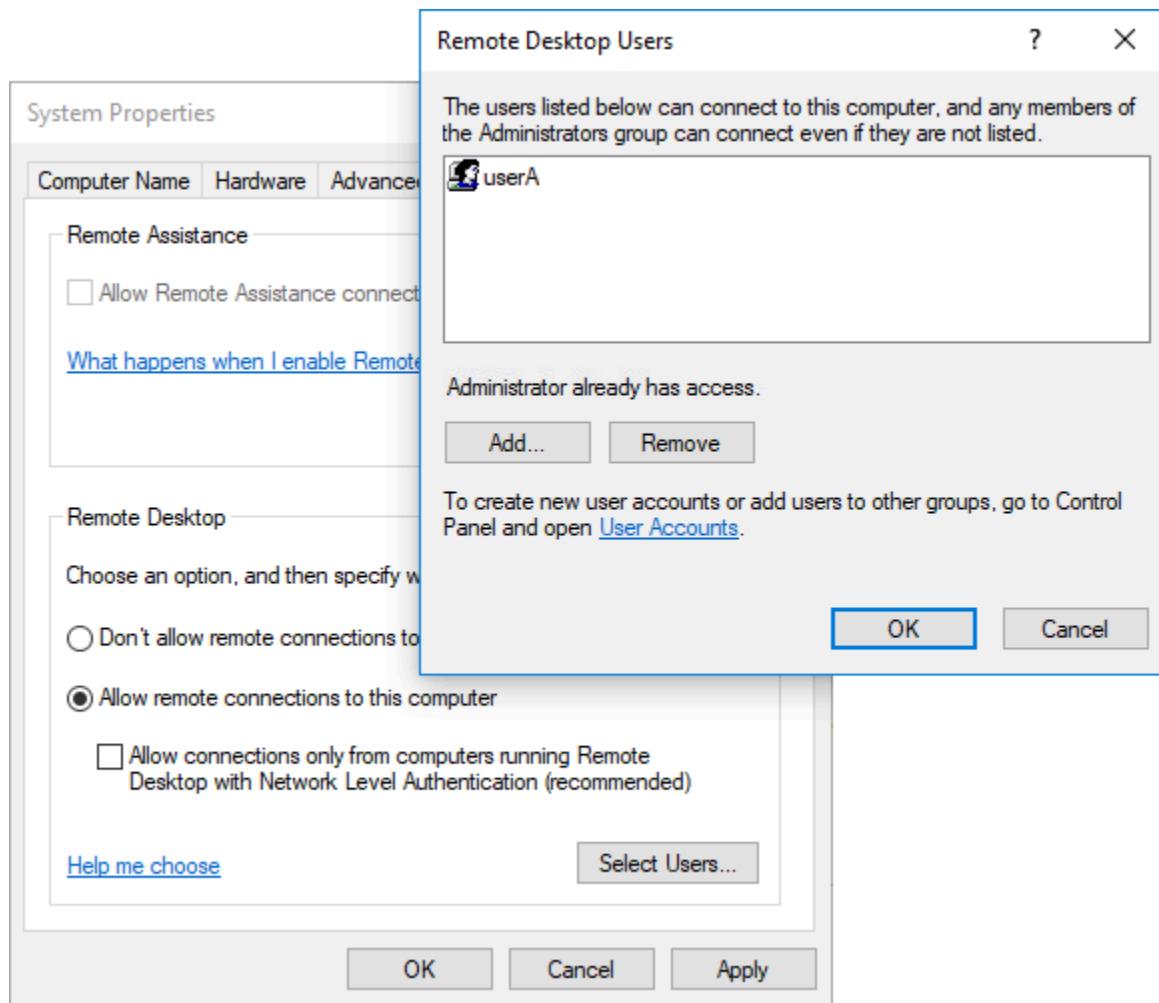
Refer to the following topics in the *Plant SCADA Access Anywhere Server Installation and Configuration Guide*:

- [Before You Install](#)

## Configuring Remote Desktop Service

Complete these steps on the computer on which the Plant SCADA Access Anywhere Server is installed:

1. Navigate to the Control Panel. Select **System and Security | System**.
2. Select the **Advanced system settings** option on the left. The System Properties dialog is displayed.
3. Click the Remote tab.
4. Select the **Allow remote connections to this computer** option.



5. Click **Select Users**. The Remote Desktop Users dialog is displayed.
6. Add the required users and click **OK**.

**Note:** For more details about configuring RDS, search for Tech Note TN000031136 on the Knowledge Base page of the AVEVA Knowledge and Support Center. You will need to register with the site before you can

---

access the document.

---

## For more information...

Refer to the following topics in the *Plant SCADA Access Anywhere Server Installation and Configuration Guide*:

- [Configuring the Plant SCADA Access Anywhere Server](#)

## Configuring User Access to Plant SCADA Clients

Add local users and domain groups to the Plant SCADA Access Anywhere groups to assign access privileges to the Plant SCADA client(view-only client or control client). To do this:

1. Add users to the Windows user groups added by the installer ('SCADA.AnywhereControl' or 'SCADA.AnywhereView').  
**Note:** If you have recently upgraded Plant SCADA Access Anywhere from a version earlier than 2020 R2, you will need to manually migrate the users in the existing "VjcaControl" and "VjcaView" Windows user groups to these new groups.
2. Navigate to the Control Panel. Select **Administrative Tools | Computer Management**. Locate the Plant SCADA Access Anywhere groups, right-click on the required group.
3. Select **Add to Group** from the context menu and add any users who require access to that group.

**Note:** If you are using Plant SCADA 2020 R2 (or later), the members of the Plant SCADA Access Anywhere user groups must also be included in Plant SCADA's SCADA.RuntimeUsers group to access the runtime display client.

---

## For more information...

Refer to the following topics in the *Plant SCADA Access Anywhere Server Installation and Configuration Guide*:

- [Configuring the Plant SCADA Access Anywhere Server](#)

## Installing and Configuring the Access Anywhere Secure Gateway

A basic installation of Plant SCADA Access Anywhere Secure Gateway usually takes about five minutes, similar to the Plant SCADA Access Anywhere Server. Check that the installation prerequisites have been met before starting the installation procedure.

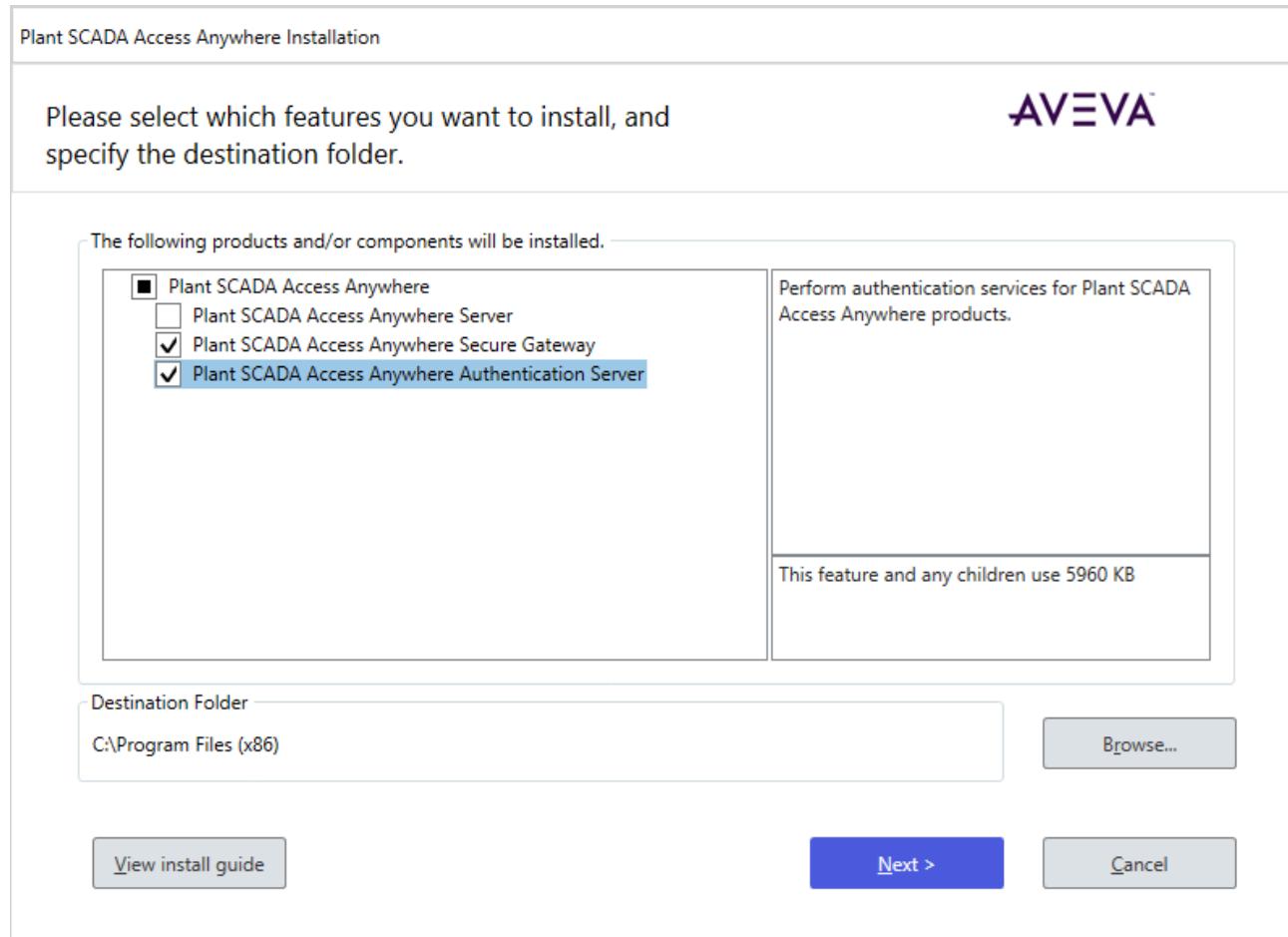
**Note:** Before you install Plant SCADA Access Anywhere 2023, you need to uninstall any earlier versions. This means an existing Secure Gateway configuration will no longer be available. If you want to migrate your existing Secure Gateway configuration to the new version, see the topic [Migrate an Existing Configuration from an Earlier Version](#) in the *Plant SCADA Access Anywhere Server Installation and Configuration Guide* before you upgrade.

---

### To install Plant SCADA Access Anywhere Secure Gateway:

1. Run the Plant SCADA Access Anywhere installer file setup.exe from your installation media.
2. Select Plant SCADA Access Anywhere **Secure Gateway** and/or Plant SCADA Access Anywhere **Authentication Server** when presented with the options to select the installation components.

These components can be installed on separate computers if required. If your Secure Gateway Server needs to be installed outside your operations network, it is recommended that the Authentication Server be installed separately within the operations network.



3. Click **Next** through the dialog boxes, accept the EULA and then click **Finish**.
4. Configure the Plant SCADA Access Anywhere Server using the Configuration Portal.

The screenshot shows the 'Server Information' page of the AVEVA Plant SCADA Secure Gateway. The left sidebar has a tree view with 'Administrator', 'Logout', 'Gateway Settings', 'Dashboard' (selected), 'Security', 'Web Server', 'Authentication Server', 'Log Settings', 'Download', 'Admin', and 'About'. The main area has sections for 'Server Status' and 'Server Activity'. 'Server Status' shows Start Time (Tuesday, 19 October 2021 4:56:44 PM), Up Time (1 Days, 18 Hours, 47 Minutes, 53 Seconds), and SSL Certificate (NOT Trusted. Has a private key). 'Server Activity' has two tables: 'Total Sessions' and 'Gateway Session Distribution'. The 'Total Sessions' table shows current and peak values for Connections under validation (0/5), Gateway Sessions (0/1), Web Server Connections (1/6), and Admins (1/1). The 'Gateway Session Distribution' table shows current and peak values for Native Sessions (0/1), HTTPS Sessions (0/0), and WebConnect Sessions (0/0). At the bottom are 'Restart Server' and 'Refresh' buttons.

	Current	Peak
Connections under validation	0	5
Gateway Sessions	0	1
Web Server Connections	1	6
Admins	1	1

	Current	Peak
Native Sessions	0	1
HTTPS Sessions	0	0
WebConnect Sessions	0	0

You can then connect the Secure Gateway to an Access Anywhere Server (see [Connecting to a Plant SCADA Access Anywhere Server](#)).

## For more information...

Refer to the following topics in the *Plant SCADA Access Anywhere Secure Gateway Installation and Configuration Guide*:

- [Installation](#)
- [Configuring the Secure Gateway](#)

## Connecting to a Plant SCADA Access Anywhere Server

The Secure Gateway needs to be configured to point to your Plant SCADA Access Anywhere Server in order to use SSL or accessing Plant SCADA. To do this:

1. Edit the config.js file located on the computer where the Secure Gateway is installed. This file is located in the following folder:

\Program Files (x86)\AVEVA Plant SCADA Access Anywhere\Plant SCADA Access Anywhere Secure Gateway\WebServer\PlantAccessAnywhere\

2. Find the commented line:

```
// address:"", // Address of server
```

3. Uncomment the line by removing the initial "//".

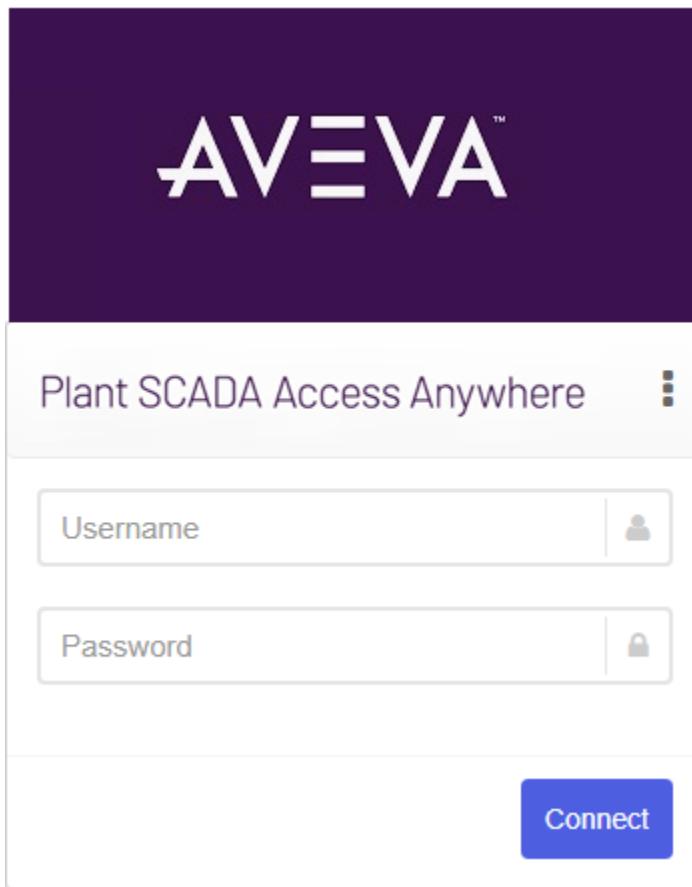
4. With the quotation marks, type the name or IP address of the computer on which the Plant SCADA Access Anywhere Server is installed.

For example:

```
address: "VD-CA-SRV", // Address of server
```

Access to the Plant SCADA Access Anywhere Server is routed through the Secure Gateway node. The following page appears when you navigate to the following address.

<https://<Name or IP Address of computer on which Secure Gateway is installed or IP Address>/>



The setup steps completed above will route the connection through to the Plant SCADA Access Anywhere Server using the supplied credentials.

**Note:** A page refresh and clearing the cookies on every client are required each time the **Address** property in the config.js is changed.

## For more information...

Refer to the following chapters in the *Plant SCADA Access Anywhere Secure Gateway Installation and Configuration Guide*:

- *Post-Installation*

## Connecting Access Anywhere to a Plant SCADA Client

Connection to a Plant SCADA client is achieved through a Plant SCADA Access Anywhere web client in the following ways:

- Through the Secure Gateway
- Directly to the Plant SCADA Access Anywhere Server.

### Connecting via the Secure Gateway

Open an HTML5 compliant browser and point it to the URL of the computer on which the Secure Gateway is installed:

`https://server:port/`

### Connecting Directly to the Plant SCADA Access Anywhere Server

If the web client is on the SCADA network, you can connect directly to the Plant SCADA Access Anywhere Server. Open an HTML5 compliant browser and point it to the URL of the Plant SCADA Access Anywhere Server:

`http://<computer name or IP Address>:8080/`

This URL will automatically redirect to the following URL:

`http://<computer name>:8080/PlantAccessAnywhere/start.html`

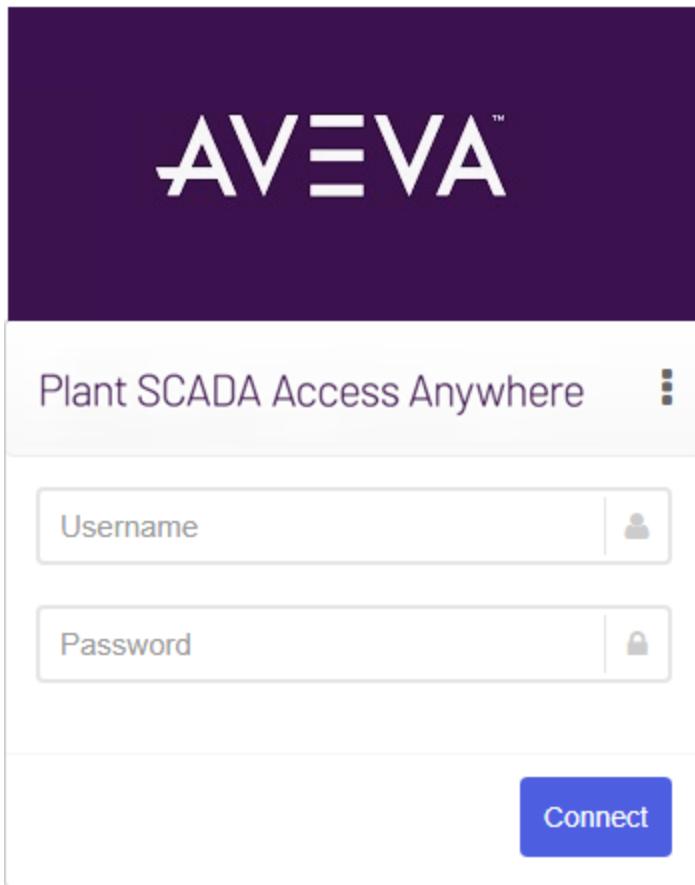
Specify the Plant SCADA Access Anywhere Server port in the URL to tell the browser to use the web server that is built-in to the Plant SCADA Access Anywhere Server service. HTTPS may also be used.

---

**Note:** If you are running Plant SCADA Access Anywhere on an IPv6 network and you attempt to connect to the Access Anywhere Server using an IPv6 address in Internet Explorer, the connection will not be successful. Use a different browser, or check with your system administrator if a hostname has been configured for the Access Anywhere Server.

### Logging on to Plant SCADA

1. After the Plant SCADA Access Anywhere Server web page appears, enter the user credentials. This user needs to belong to the 'SCADA.AnywhereControl' or 'SCADA.AnywhereView' Windows user group.



- Once you have entered your credentials, click **Connect**.

The connection dialog appears momentarily while the web browser connects to the RDP host where the Plant SCADA Access Anywhere Server is installed.

The Plant SCADA client is launched at the remote node.

---

**Note:** Once connected, closing the Plant SCADA client will log off and end the session. Closing the browser will close the Plant SCADA client, log off and end the Access Anywhere session.

---

## For more information...

Refer to the following chapters in the *Plant SCADA Access Anywhere Web Client User Guide*:

- [Viewing Plant SCADA with Plant SCADA Access Anywhere](#)

## Access Anywhere Installation and Configuration Guide

Use AVEVA Plant SCADA Access Anywhere to access AVEVA Plant SCADA clients hosted on Terminal Servers with HTML5 compatible web browsers. Follow the instructions in this guide to begin using AVEVA Plant SCADA Access Anywhere.

This guide assumes knowledge of the following:

- AVEVA Plant SCADA

- Enabling and configuring Remote Desktop Services (RDS) on Windows® operating systems
- Firewall configuration
- Web server administration.

Terminology used in this guide includes the following:

- **RDP** - Remote Desktop Protocol. A remote desktop protocol developed by Microsoft®. RDP is a standard component of Microsoft Windows.
- **RDP Host** - a Windows system that can be remotely accessed using Microsoft RDP, such as a Terminal Server (RDS Session Host) with remote access enabled.
- **RDS** - Remote Desktop Services is one of the components of Microsoft Windows that allow a user to take control of a remote computer or virtual machine over a network connection.
- **HTML5** - a new update to the HTML specification. Extends HTML with new features and functionality for communication, display, etc.
- **WebSocket** - a bi-directional, full-duplex communication mechanism introduced in the HTML5 specification.
- **SSL** - Secure Sockets Layer. A cryptographic protocol that provides encrypted communications over the Internet.

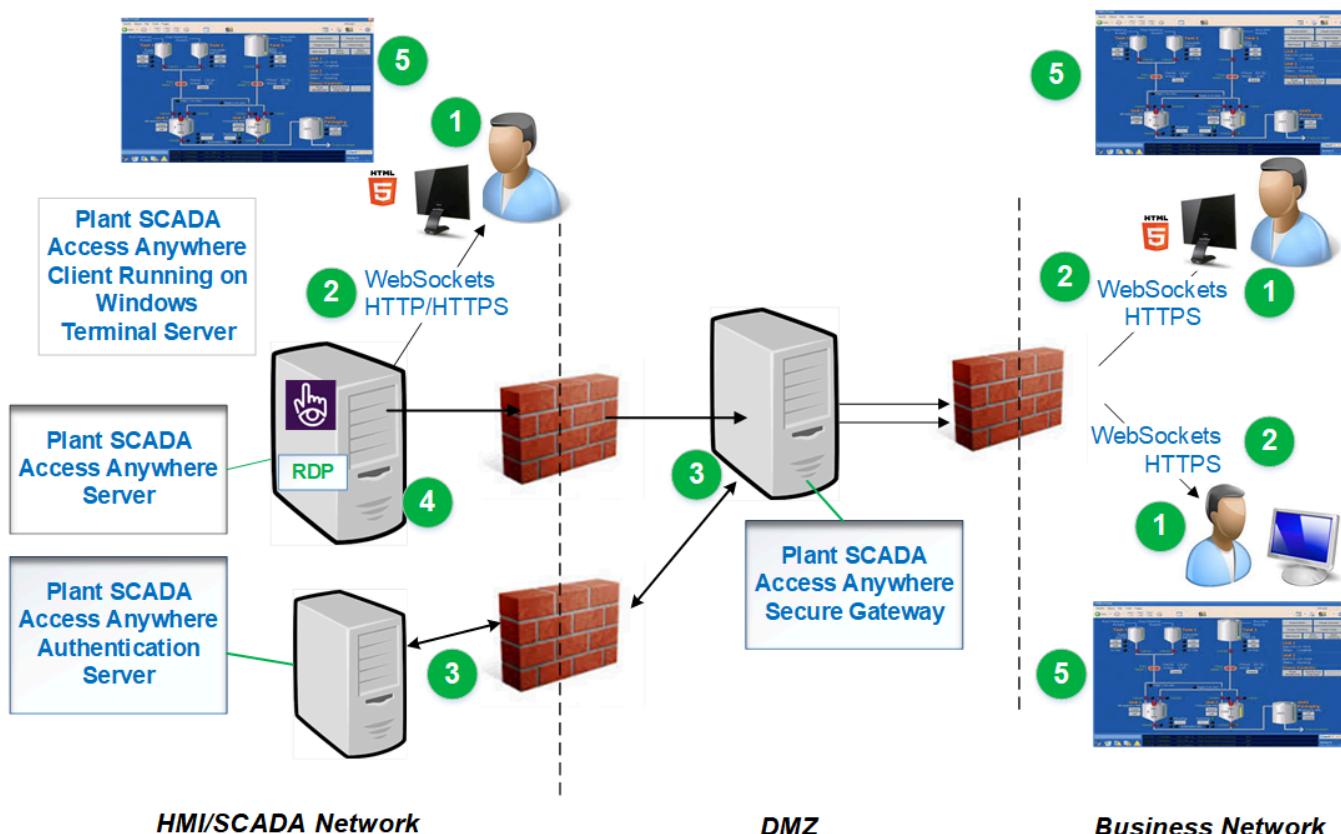
## Overview

Plant SCADA Access Anywhere provides remote access to Plant SCADA clients using any HTML5 web browser running on a desktop computer or a mobile device.

Any browser that supports an HTML5 canvas can be used as the client to view Plant SCADA. HTML5 WebSockets are typically required for Plant SCADA Access Anywhere.

## Architecture

The following diagram illustrates how the different components of Plant SCADA Access Anywhere work together:



Reference	Description
1	Initiate a connection from the client device by directing the browser to the Plant SCADA Access Anywhere start page hosted on the web server ( <a href="https://&lt;computer name&gt;:8080/">https://&lt;computer name&gt;:8080/</a> ). The <i>Start.html</i> page is displayed in the web browser using HTTP/HTTPS.
2	The browser opens a WebSocket connection to the Plant SCADA Access Anywhere Server, which is running on the RDP host itself.
3	If the optional Plant SCADA Access Anywhere Secure Gateway is installed, a Plant SCADA Access Anywhere Server browser session will connect through it. When the Secure Gateway is used, user access is managed by the Plant SCADA Access Anywhere Authentication Server.
4	The Plant SCADA Access Anywhere Server translates the WebSocket communication to and from RDP, thus establishing a connection from the browser to the RDP host itself.

Reference	Description
5	The browser then displays the content of the remote Plant SCADA client.

**Note:** Any Plant SCADA Access Anywhere Server browser sessions that originate from a location beyond the SCADA network will connect through the optional Plant SCADA Access Anywhere Secure Gateway. If the client is on the SCADA network, it can connect directly to the Plant SCADA Access Anywhere Server or via the Secure Gateway.

For more information about the Access Anywhere Secure Gateway, refer to the [Access Anywhere Secure Gateway Installation and Configuration Guide](#).

**Important:** Using the Secure Gateway to connect to your SCADA system from an external network may expose your SCADA system to unauthorized access. It is recommended that you use the Secure Gateway in conjunction with other measures to help protect your system.

This is the recommended architecture to remotely access Plant SCADA clients running on an HMI/SCADA network from an untrusted business network.

- The Plant SCADA Access Anywhere Server (WebSocket Server) is installed on the same RDP host where the AVEVA Plant SCADA client runs. The server includes a collection of web resources (HTML files, CSS, JavaScript, images, etc.).
- The Plant SCADA Access Anywhere Secure Gateway is an optional server installed separately on a computer in a DMZ to access Plant SCADA behind a firewall.
- The Plant SCADA Access Anywhere Authentication Server performs authentication services for Plant SCADA Access Anywhere. It is recommended that the Authentication Server be installed separately to the Secure Gateway within the SCADA network.

## RDP Compression and Acceleration

Plant SCADA Access Anywhere provides RDP compression and acceleration technology to improve remote client performance over a network and Internet. There are three main features of RDP technology:

- Image compression  
Images are compressed before they are transmitted to a browser for rendering. A higher compression value results in lower image quality, less impact to the network and faster update.
- Packet shaping  
Packet shaping is a computer network traffic management technique that delays some or all datagrams to reduce latency and increase usable network bandwidth.
- Whole frame rendering  
Whole frame rendering updates the display as a whole rather than in blocks, as performed by standard RDP. The benefit of whole frame rendering is especially noticeable when watching video over slow network connections. Coupled with the other optimization features, whole frame rendering results in a smoother video display on a browser.

## Before You Install

Prior to installing Plant SCADA Access Anywhere, confirm the following:

- The computer that will host the Plant SCADA Access Anywhere Server is running an operating system supported by Plant SCADA.
- The host computer's firewall is configured to permit inbound and outbound network traffic on port 8080.
- Remote Desktop Services is configured on the host computer.

---

**Important:** Plant SCADA Access Anywhere leverages RDP and translates RDP to WebSockets. RDP access needs to be enabled on the computer hosting Plant SCADA Access Anywhere. For more information, search for Tech Note 8956 on the **Tech Note, FAQ** page of the AVEVA Knowledge and Support Center. You will need to register with the site before you can access the document.

---

## Prerequisites

The Plant SCADA Access Anywhere Server has been tested on the following Microsoft® operating systems:

- Windows Server 2012 R2
- Windows Server 2016
- Windows Server 2019.

The Access Anywhere Server needs to be installed on the RDP server where Plant SCADA resides. It will have minimal impact on the RDP host's performance and scalability.

---

**Important:** Microsoft Windows Server should not be configured as a Domain Controller.

---

The Access Anywhere Server includes a built-in web server. This includes a copy of the Access Anywhere web components.

---

**Note:** If you are running Plant SCADA Access Anywhere on an IPv6 network, it is recommended you configure a hostname that refers to the Access Anywhere Server. A hostname ensures the server is compatible with all supported browsers. If you attempt to connect to the Access Anywhere Server using a standard format IPv6 address in Internet Explorer, the connection will not be successful.

---

## Configuring Firewalls

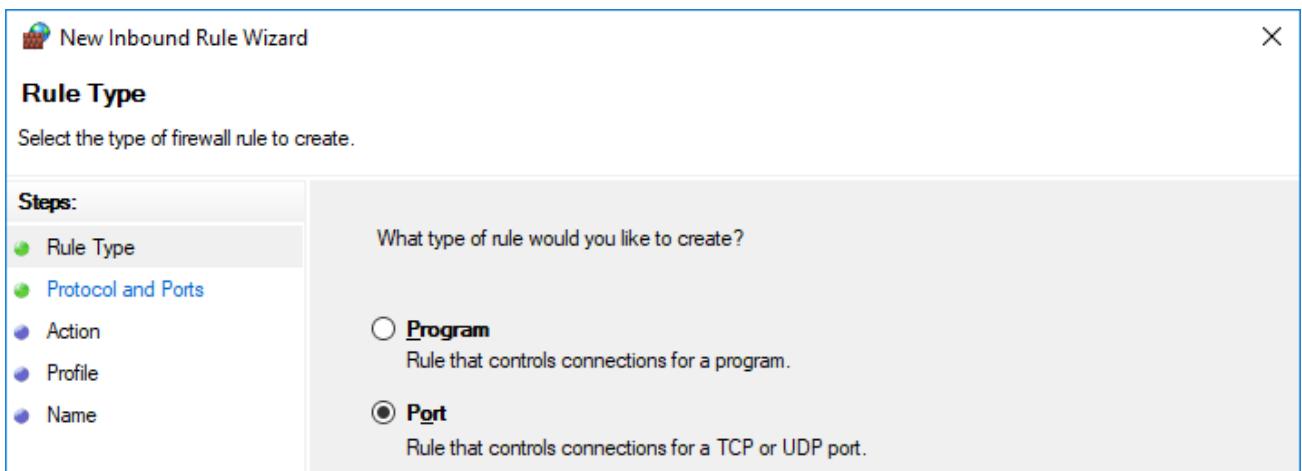
By default, a client (browser) connects to a Plant SCADA Access Anywhere Server using port 8080 for both encrypted and unencrypted WebSocket communication. This port number can be changed using the Plant SCADA Access Anywhere Server Configuration utility.

To enable direct connection from the client to the Plant SCADA Access Anywhere Server (without using the Secure Gateway), the server needs to be directly accessible from the client using port 8080.

If the Windows firewall is enabled on the same computer where the Plant SCADA Access Anywhere Server is installed, configure it to enable the Plant SCADA Access Anywhere client connection.

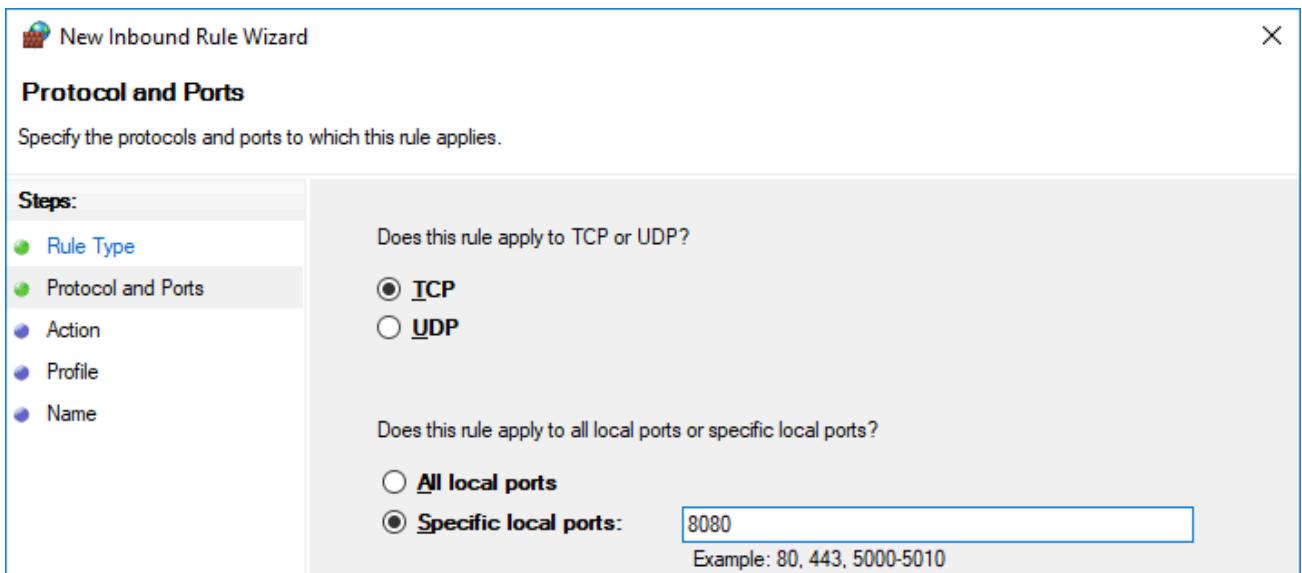
1. Open the **Windows Control Panel** and then **Windows Firewall**.
2. Select **Advanced Settings** and select **Inbound Rules**.

3. Click **New Rule**.



4. Select **Port** and click **Next**.

5. Enter the specific port: 8080.



6. Click **Next** and select **Allow the connection**.

 New Inbound Rule Wizard X

**Action**

Specify the action to be taken when a connection matches the conditions specified in the rule.

**Steps:**

- Rule Type
- Protocol and Ports
- Action**
- Profile
- Name

What action should be taken when a connection matches the specified conditions?

**Allow the connection**  
This includes connections that are protected with IPsec as well as those are not.

**Allow the connection if it is secure**  
This includes only connections that have been authenticated by using IPsec. Connections will be secured using the settings in IPsec properties and rules in the Connection Security Rule node.

[Customize...](#)

7. Click **Next** and select to apply the rule on the **Domain, Private** and **Public** networks.

 New Inbound Rule Wizard X

**Profile**

Specify the profiles for which this rule applies.

**Steps:**

- Rule Type
- Protocol and Ports
- Action
- Profile**
- Name

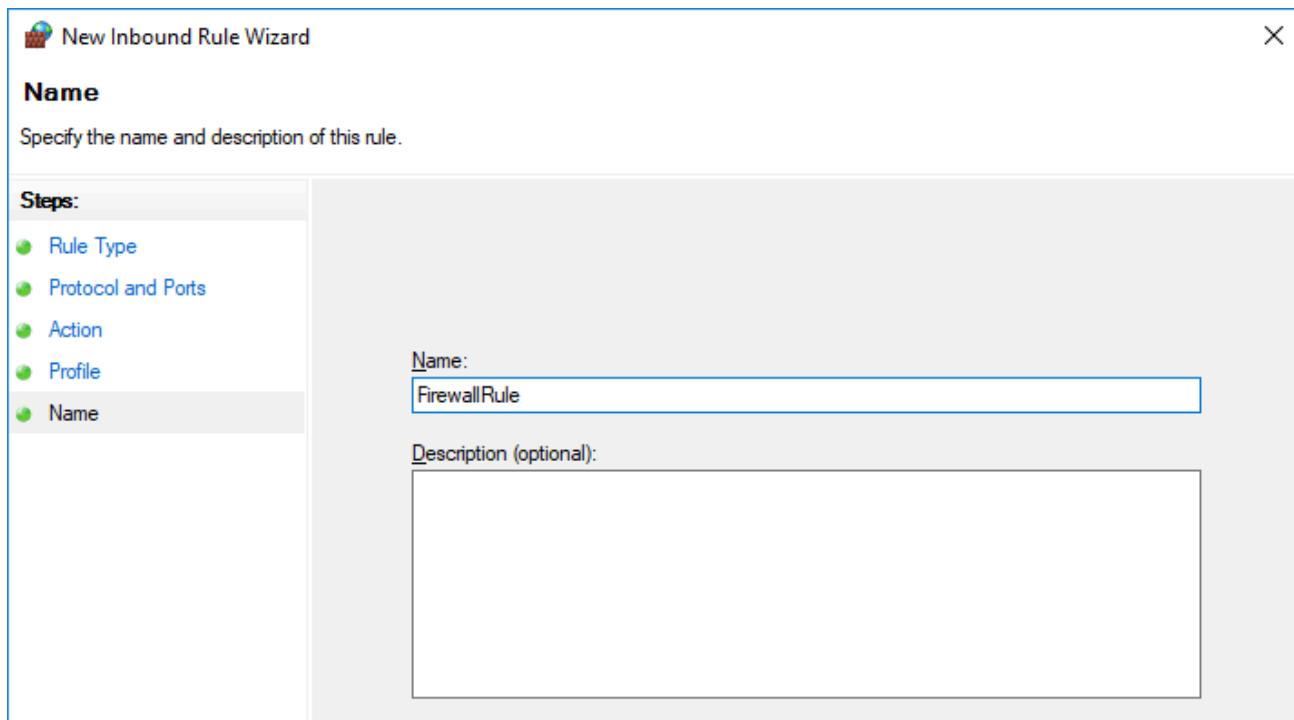
When does this rule apply?

**Domain**  
Applies when a computer is connected to its corporate domain.

**Private**  
Applies when a computer is connected to a private network location, such as a home or work place.

**Public**  
Applies when a computer is connected to a public network location.

8. Click **Next**, assign a name for the rule.



9. Click **Finish**.

## Binding Service to Every Network Interface

In a virtual network environment, Plant SCADA Access Anywhere Server should use every virtual network interface, rather than just one virtual Network Interface Controller (NIC). Network interfaces used by the Access Anywhere Server need to be accessible to the target group of users.

## Verifying the Current Network Interface Configuration

As a quick test of your current network configuration, run Microsoft PowerShell and enter the following command:

```
RESOLVE-DNSNAME dnsname
```

For example:

```
PS C:\Users\user1> resolve-dnsname itaatest
```

## Installation

Plant SCADA Access Anywhere Server is the server-side service that translates RDP into WebSocket communication. The Plant SCADA Access Anywhere Server is installed on the RDP host.

The Plant SCADA Access Anywhere client interface, running in the browser, connects to this service using WebSockets directly or through the Secure Gateway.

For more information about the Secure Gateway, refer to the [Access Anywhere Secure Gateway Installation and Configuration Guide](#).

**Note:** Plant SCADA Access Anywhere supports silent installation of its components using a response file. See

---

Unattended Silent Installation.

---

## Installing Plant SCADA Access Anywhere Server

The Plant SCADA Access Anywhere installer allows you to install the Plant SCADA Access Anywhere Server and the Access Anywhere Secure Gateway.

**Note:** The Access Anywhere Server and Access Anywhere Secure Gateway components **cannot** be installed on the same computer.

---

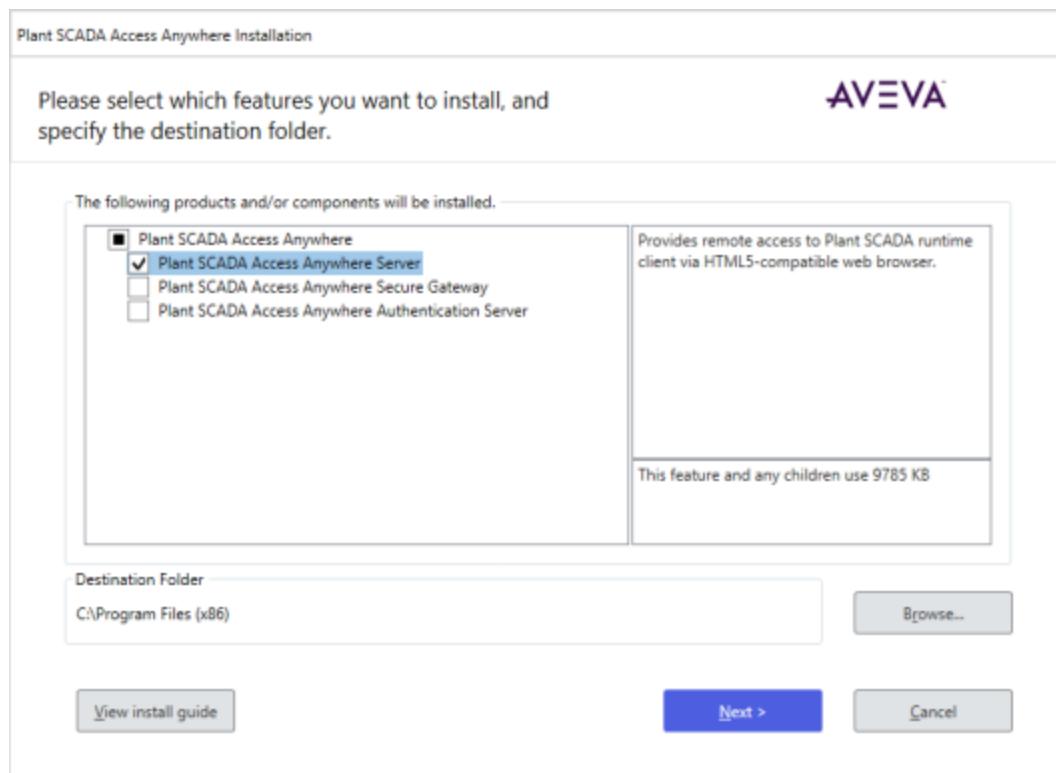
To install Plant SCADA Access Anywhere Server:

1. Run the *Setup.exe* file. The Plant SCADA Access Anywhere splash screen appears.



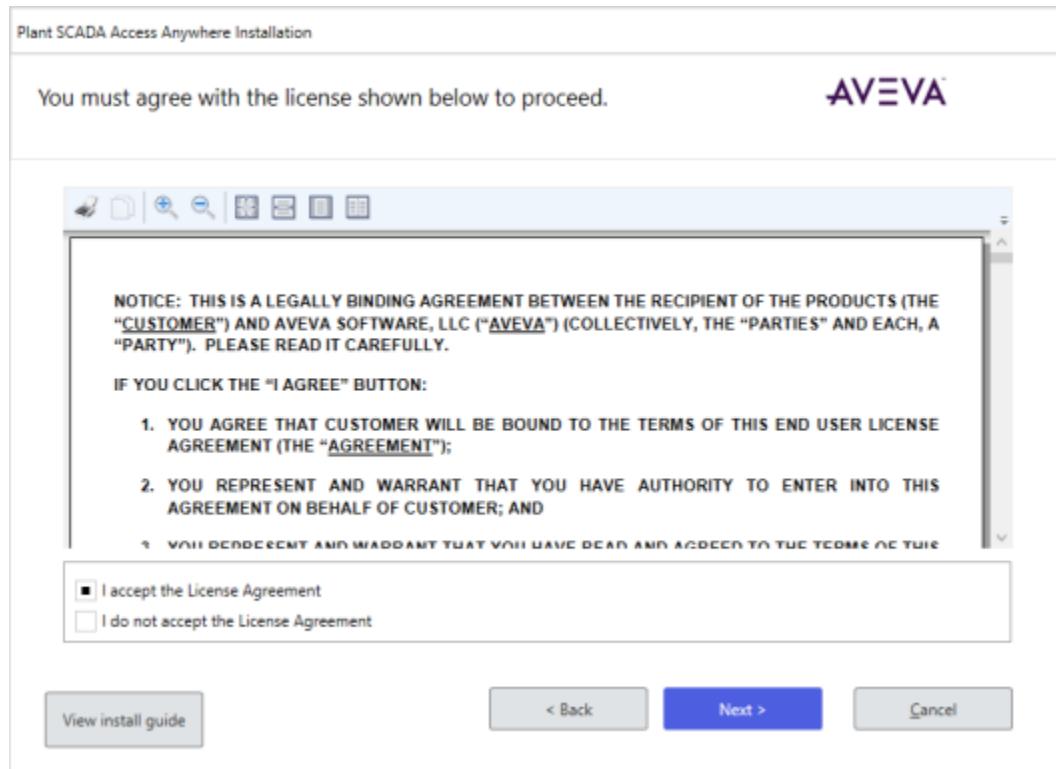
The message "Please wait for configuration to complete" appears as the installer prepares to start the installation.

2. Select the Plant SCADA Access Anywhere **Server** option.



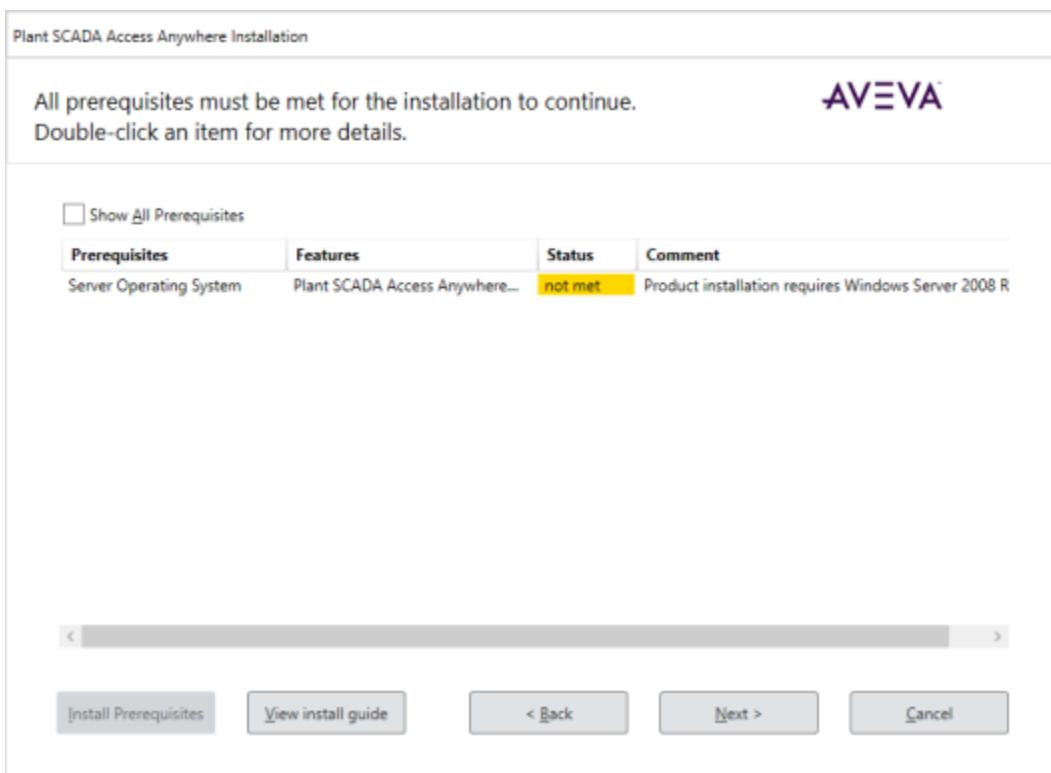
**Note:** A Plant SCADA Access Anywhere Server cannot be installed on the same computer as a Plant SCADA Access Anywhere Secure Gateway.

- Click **Next**. The License Agreement screen appears. Read the license agreement carefully, and select the **I accept the License Agreement** option to continue. Selecting **I do not accept the License Agreement** will stop you from proceeding with the installation.

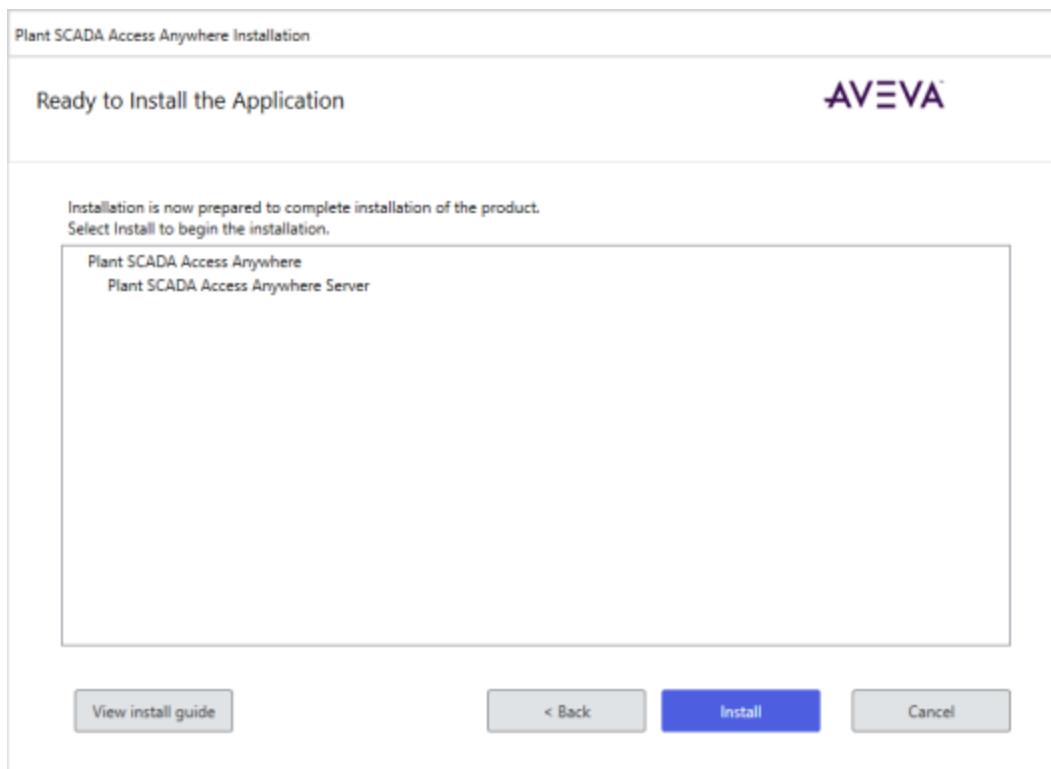


4. Click **Next**. The Installation Prerequisites screen appears.

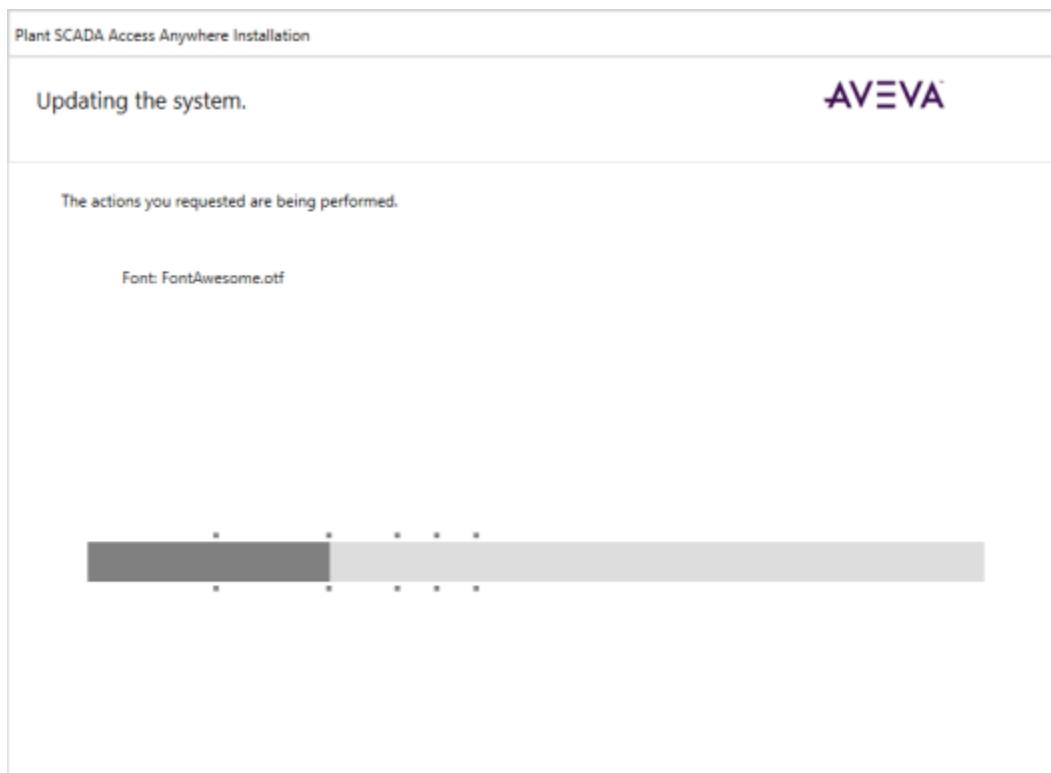
The installation program checks whether the installation prerequisites are met, and shows the missing prerequisites.



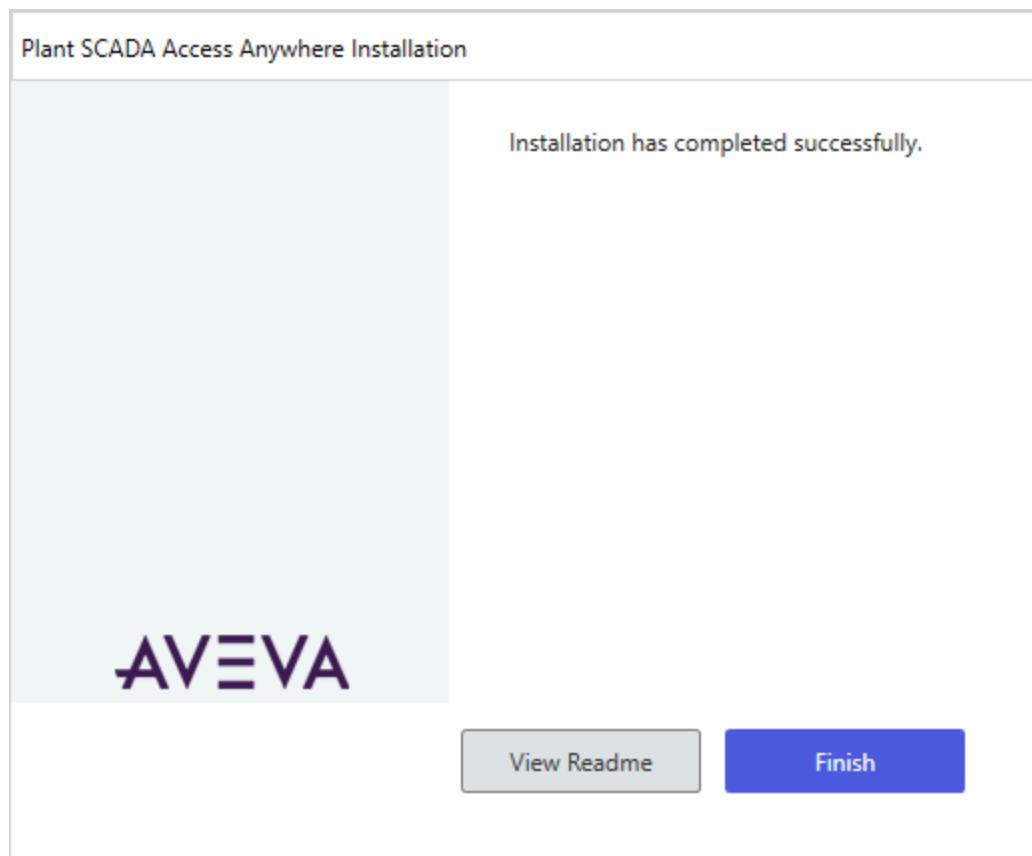
5. To view a complete list of prerequisites, select the **Show All Prerequisites** option.  
Click **Install Prerequisites** to install the missing prerequisites.
6. Click **Next**. The list of components selected for installation is displayed.



7. Click **Install**. The progress bar appears.



After the installation is complete, the Installation Complete dialog box appears. Click **View Readme** to open the *Readme.html* file, which contains details about Plant SCADA Access Anywhere features, or click **Finish**.



The Plant SCADA Access Anywhere Server service is configured to run automatically on system startup. If the service is stopped or is unable to listen on its default port (8080), clients will not be able to connect to that host.

Configure firewalls and proxies between the end-point devices and the server-side component to allow communication using port 8080, or use the Plant SCADA Access Anywhere Secure Gateway (see the [Access Anywhere Secure Gateway Installation Guide](#)).

---

**Note:** Plant SCADA Access Anywhere Server cannot be installed on systems where the host name contains non-English characters.

---

## Unattended Silent Installation

The installation of Plant SCADA Access Anywhere can be executed via a command line that calls a response file. A response file defines the components that will be installed during a silent installation. It can be used to install the following components:

- Plant SCADA Access Anywhere Server
- Plant SCADA Access Anywhere Secure Gateway
- Plant SCADA Access Anywhere Authentication Server (supports a Secure Gateway).

---

**Note:** The Access Anywhere Server and Access Anywhere Secure Gateway components should not be installed on the same computer. The Secure Gateway is an optional server installed separately on a computer in a DMZ to allow access to Plant SCADA from behind a firewall.

---

You can create your own response files as a simple text-based file (for example,

AccessAnywhereServerInstall.txt).

### Install Command

You can then execute a response file using the following command:

```
Setup.exe /silentnoreboot "<Complete path to Response File>"
```

A full absolute path needs to be provided. For example:

```
Setup.exe /silentnoreboot "C:\PlantSCADAAccessAnywhere\MyResponseFiles\AccessAnywhereServerInstall.txt"
```

### Uninstall Command

You can also use a command to run a silent uninstall of all Access Anywhere components:

```
Setup.exe /silentuninstall <Access Anywhere product ID>
```

For example:

```
Setup.exe /silentuninstall {XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}
```

The product ID for Access Anywhere can be retrieved from the Setup.xml file, located in the InstallFiles directory within the installation media.

Search for the ID value within the <mainproduct> tags.

---

**Note:** Silent modification of an existing installation is not supported. You need to uninstall all previously installed Access Anywhere components (either using the silent uninstall command or Windows Control Panel) before you can use a response file to run a silent installation.

If you want to install the Access Anywhere Secure Gateway and the Access Anywhere Authentication Server on the same computer, you need to include both in the same response file. See the example below for a Secure Gateway including the Authentication Server.

## Examples

The following examples provide the text you can use to create a response file.

- **Plant SCADA Access Anywhere Server**

```
<responsefile>
<install>
  FeatureForm.SFeatureList=Plant SCADA Access Anywhere.HTML5_Server_Files
  FeatureForm.SInstallDir=C:\Program Files (x86)
</install>
</responsefile>
```

- **Access Anywhere Secure Gateway and the Authentication Server**

```
<responsefile>
<install>
  FeatureForm.SFeatureList=Plant SCADA Access Anywhere.SecurityServer_Files,Plant SCADA
  Access Anywhere.AuthenticationServer_Files
  FeatureForm.SInstallDir=C:\Program Files (x86)
</install>
</responsefile>
```

- **Access Anywhere Secure Gateway**

```
<responsefile>
<install>
FeatureForm.SFeatureList=Plant SCADA Access Anywhere.SecurityServer_Files
FeatureForm.SInstallDir=C:\Program Files (x86)
</install>
</responsefile>
```

- **Access Anywhere Authentication Server**

```
<responsefile>
<install>
FeatureForm.SFeatureList=Plant SCADA Access Anywhere.AuthenticationServer_Files
FeatureForm.SInstallDir=C:\Program Files (x86)
</install>
</responsefile>
```

## Repairing an Installation

You can use the installation program to repair corrupt files of installed Plant SCADA Access Anywhere components.

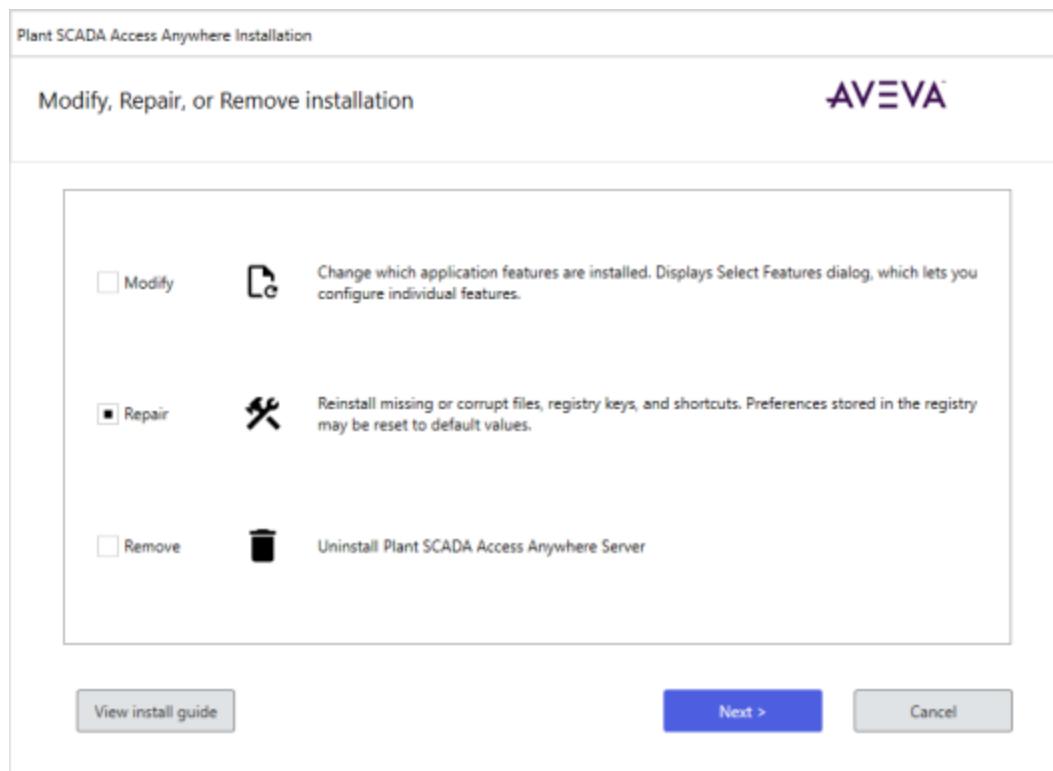
---

**Note:** Before you can repair an installation, you need to have access to the location where the installation files are stored.

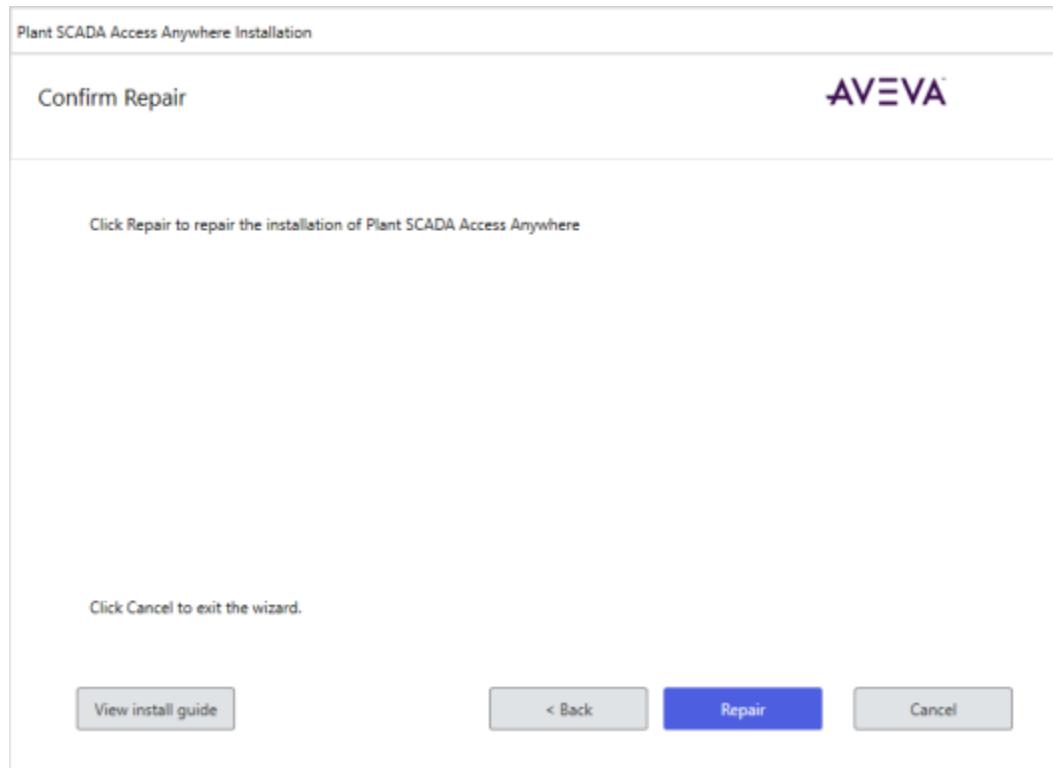
---

**To repair an installation:**

1. From the **Windows Start** menu, select **Control Panel | Programs | Programs and Features | Uninstall a Program**. The programs installed on your computer are listed.
2. Select the Plant SCADA Access Anywhere component you wish to repair. Click **Uninstall/Change**. The installer program appears.
3. Select **Repair**, and then click **Next**.



4. Click **Repair**.



5. Installed Plant SCADA Access Anywhere components are repaired, and a message confirming this appears.
6. Click **Finish** to close the dialog box.

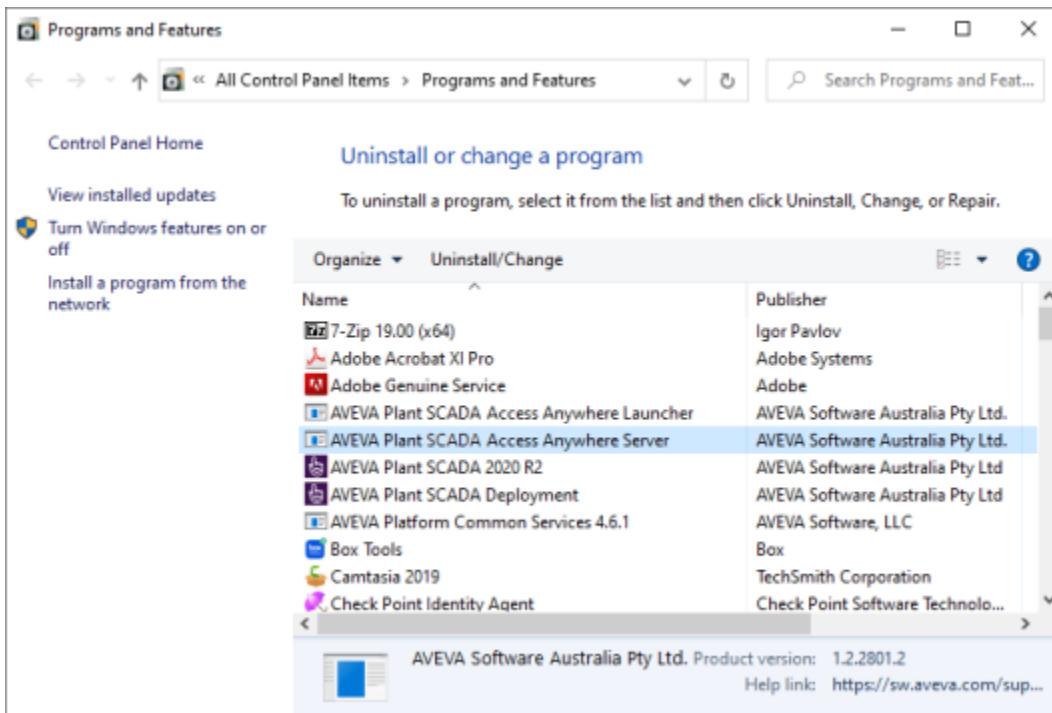
## Uninstalling Plant SCADA Access Anywhere

You may wish to remove Plant SCADA Access Anywhere components. This section provides instructions for uninstalling Plant SCADA Access Anywhere components.

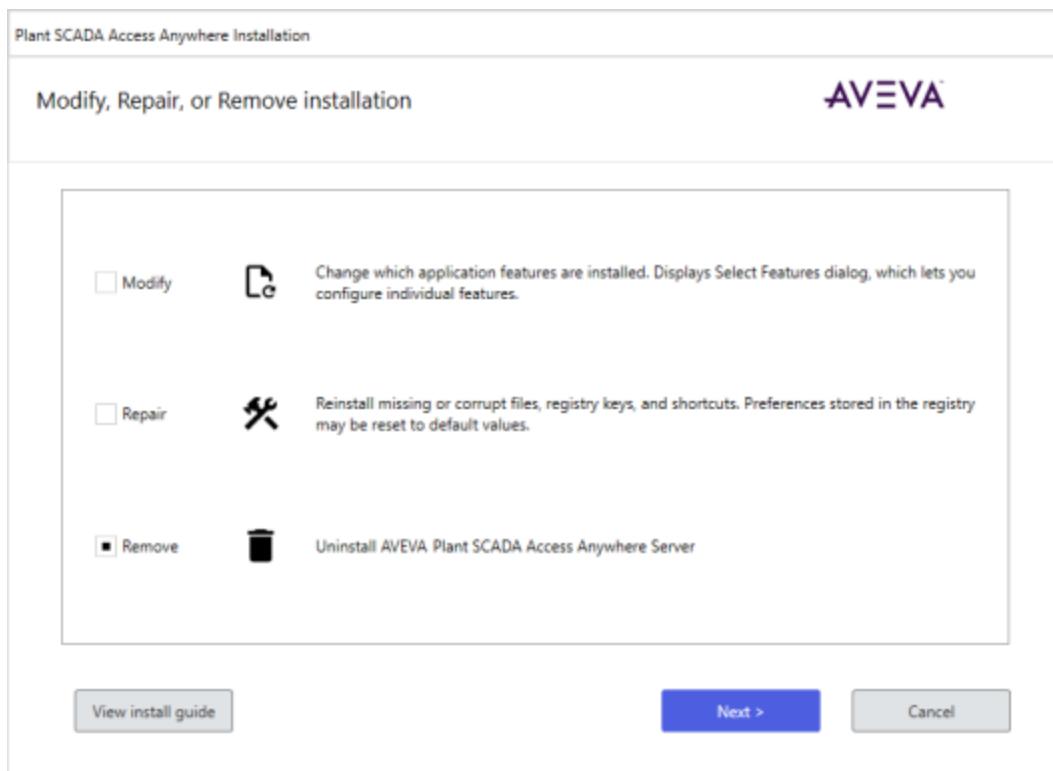
**Note:** Plant SCADA Access Anywhere supports silent uninstallation of its components using a response file. See [Unattended Silent Installation](#).

### To uninstall Plant SCADA Access Anywhere components:

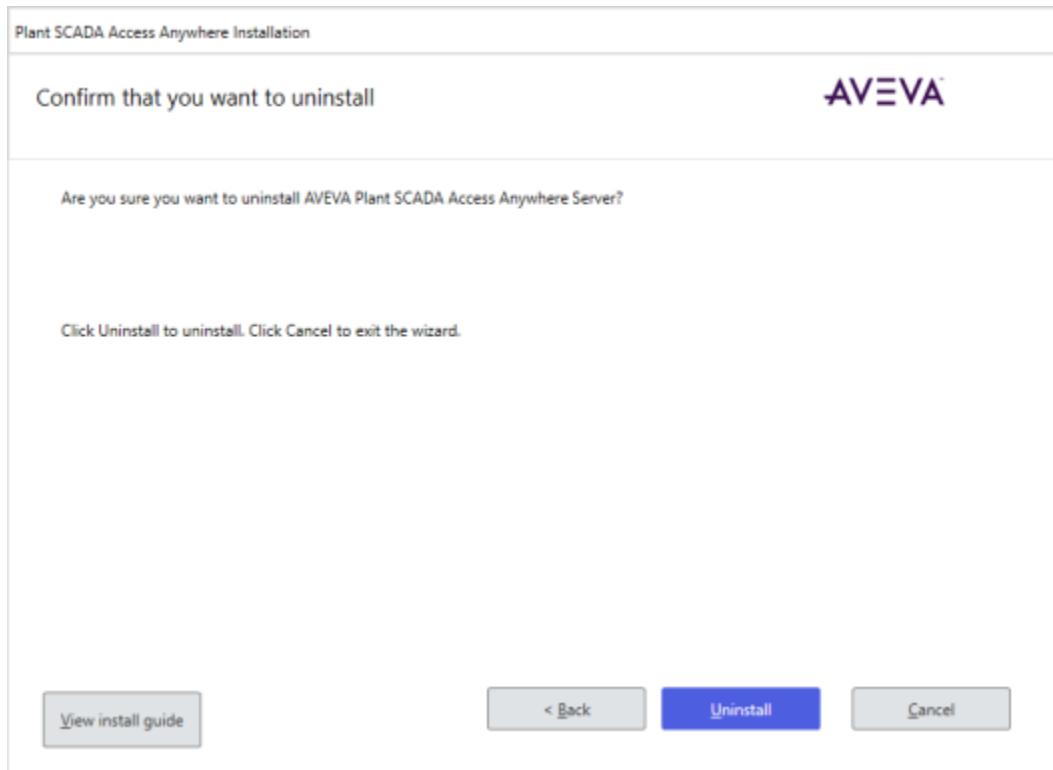
1. From the Windows Start menu, select **Control Panel | Programs | Programs and Features | Uninstall a Program**. The programs installed on your computer are listed.
2. Locate the Plant SCADA Access Anywhere Server component to uninstall, and click to select the component.



3. Click **Uninstall/Change**. The Modify, Repair or Remove Installation dialog box appears.
4. Select **Remove**.



5. Click **Uninstall** and then click **Next**. A screen confirming the uninstallation appears.



**Note:** This will remove only the selected component. To remove every component installed with the Access Anywhere Secure Gateway (the Access Anywhere Secure Gateway Client Components), run `setup.exe` again and perform a complete uninstall. Some files may still exist in the installation folders after the uninstallation.

## Upgrading to a New Version

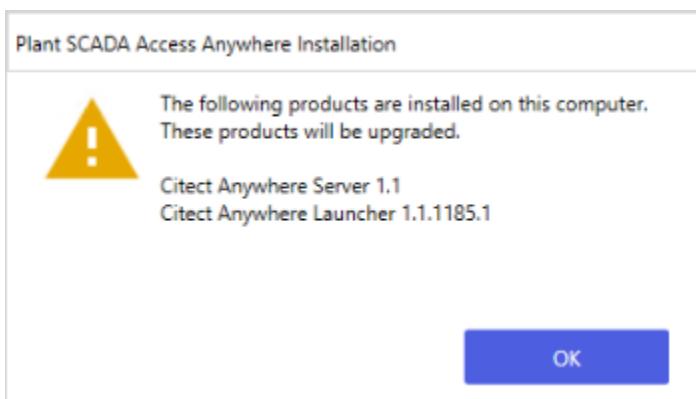
**Note:** When you upgrade Plant SCADA Access Anywhere, the existing server configuration is overwritten. If you want to migrate your existing server settings to the new version, see the topic [Migrate an Existing Configuration from an Earlier Version](#) before you upgrade.

To upgrade to the latest version of Plant SCADA Access Anywhere Server, follow these steps:

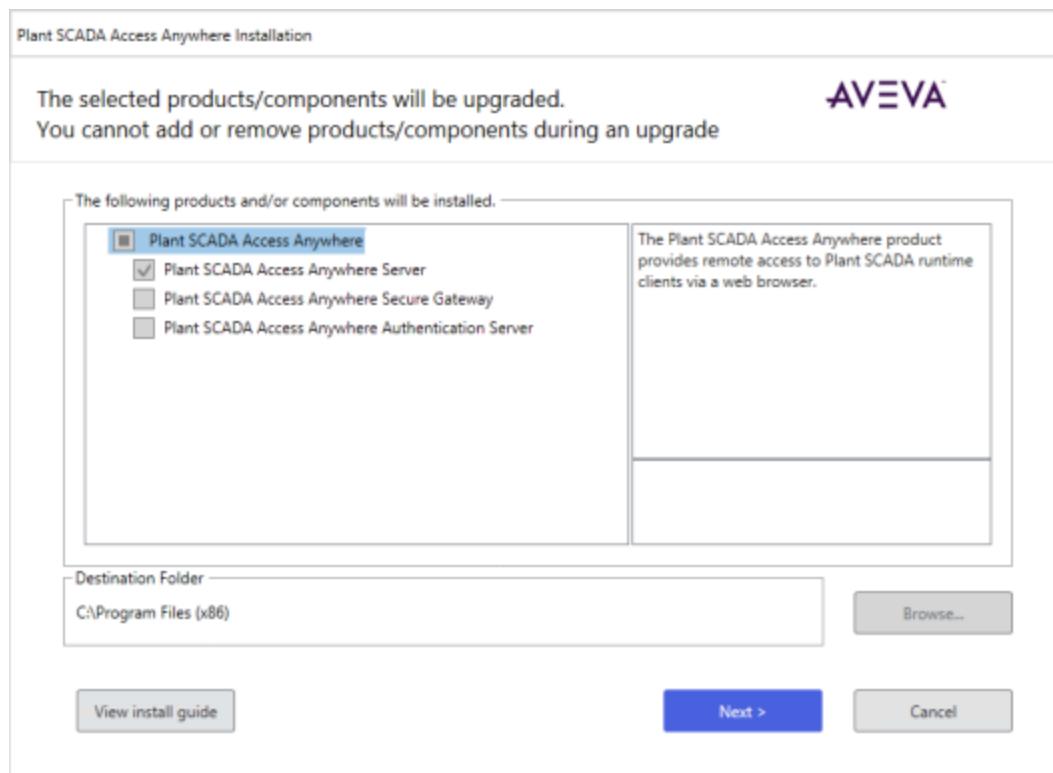
1. Run the *Setup.exe* file. The Plant SCADA Access Anywhere splash screen appears.



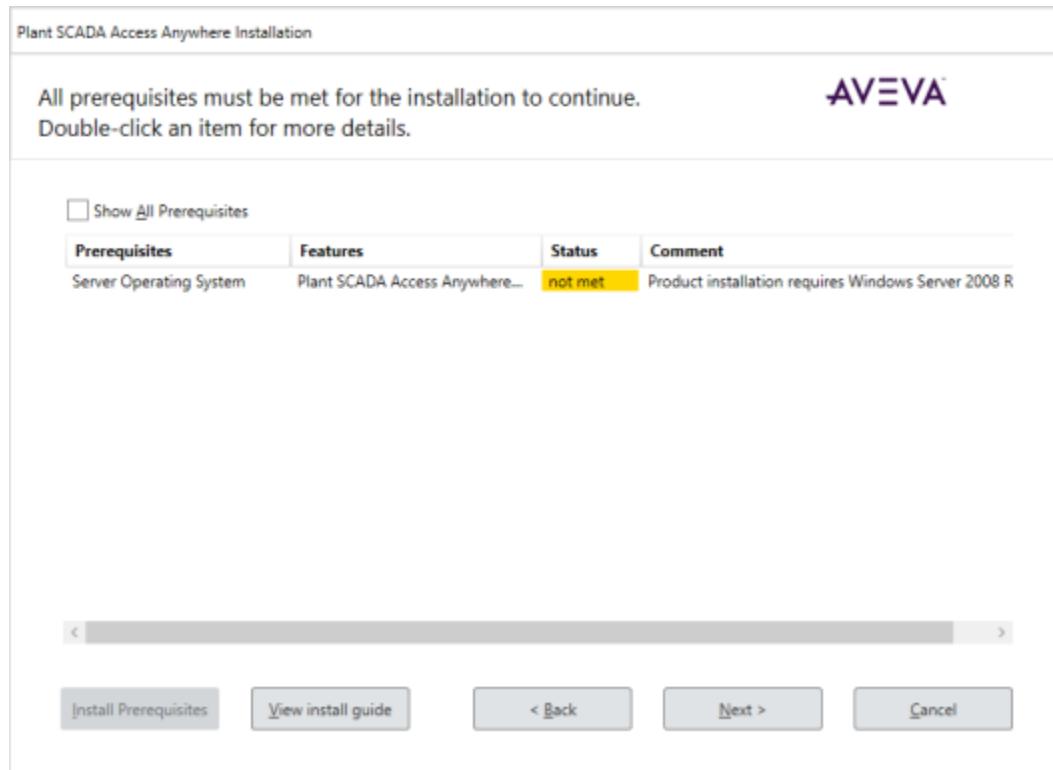
2. The following screen is then displayed.



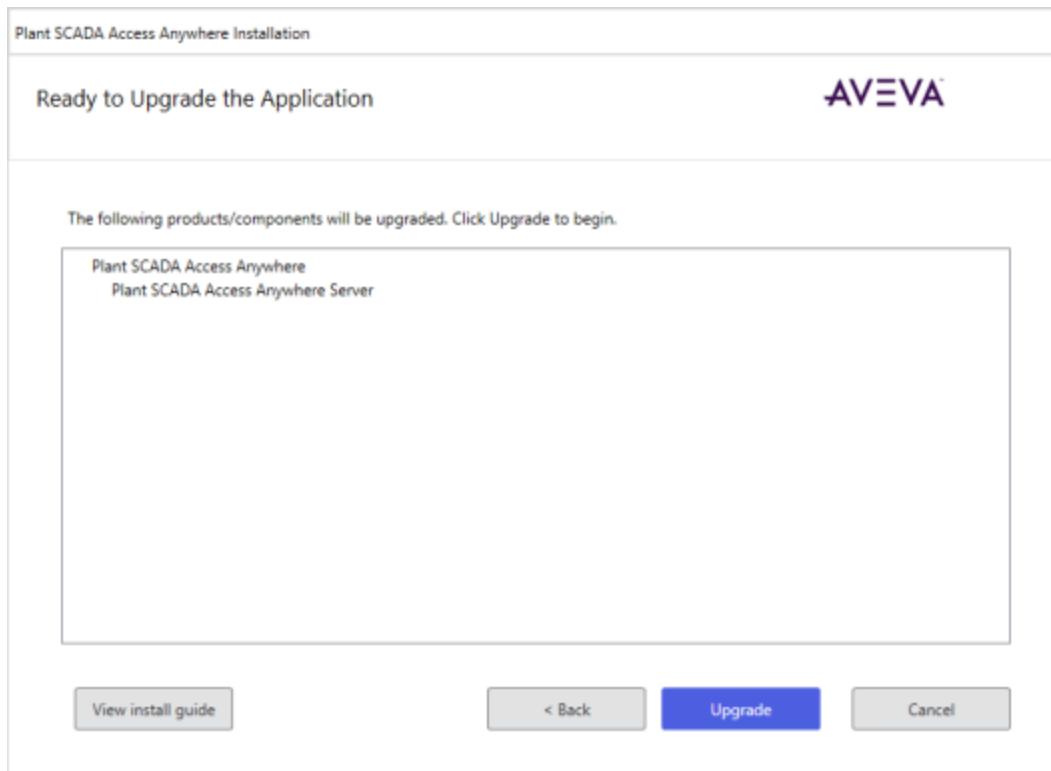
3. Click **OK**. The next screen appears with details of the Plant SCADA Access Anywhere components that will be upgraded.



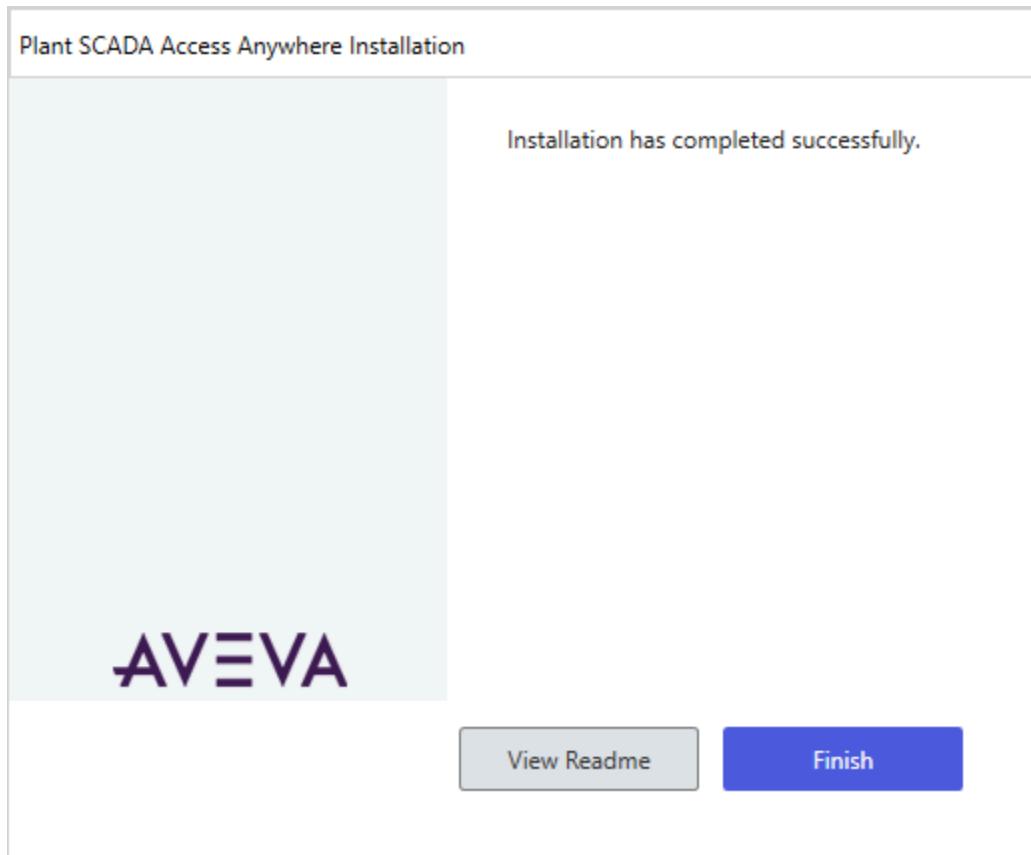
4. Click **Next**. If the prerequisites for the new version are not installed, the following screen appears.



5. Click **Install Prerequisites**. Once the prerequisites are installed, a screen with the list of the prerequisites is displayed. Click **Next** on that screen. The following screen is displayed.



6. Click **Upgrade**. The upgrade process starts, and a progress bar is displayed. The following screen appears when the upgrade completes successfully.



7. Click **Finish** to close the installer.

**Note:** Unexpected dialogs may appear the first time you log into the Web Client following an upgrade.

- If you see a "PowerTerm WebConnect Terminal Server Agent" dialog that states the system cannot find the file specified, you need to restart the Plant SCADA Access Anywhere Server.
- If an "Ericom URL Handler" dialog appears, click OK. A "How do you want to open this file?" dialog will appear. Select the option that includes a path to "PtTSAgent.exe". A third dialog will appear stating "Application not found". When you click OK on this dialog, the issue will be resolved.

## Migrate an Existing Configuration from an Earlier Version

When you upgrade, the existing Plant SCADA Access Anywhere Server configuration file is no longer used. If you want to migrate your existing server settings to the new version, you need to manually merge some information into the upgraded configuration files.

**Note:** With the release of Plant SCADA Access Anywhere 2.0, the default installation location changed from '\Program Files (x86)\Schneider Electric\Citect Anywhere' to '\Program Files (x86)\AVEVA Plant SCADA Access Anywhere'.

To migrate an existing Plant SCADA Access Anywhere Server configuration, perform the following steps.

1. Make a backup of the configuration files.

For the Plant SCADA Access Anywhere Server, this will include the following files:

- \Program Files (x86)\AVEVA Plant SCADA Access Anywhere\Launcher\SE.Scada.AnywhereLauncher.exe.config
- \Program Files (x86)\AVEVA Plant SCADA Access Anywhere\Server\WebServer\PlantAccessAnywhere\config.js

**Note:** If you have used a custom certificate for the Plant SCADA Access Anywhere Server, you will need to export a backup of the server configuration settings as a registry file as certificate information is not included in the configuration files listed above. You can view the current SSL certificate details on the **Security** tab of the [The Configuration Console](#). You can also export the configuration settings as a registry file (.reg) using the console's **Advanced** tab.

If you want to migrate an existing Access Anywhere Secure Gateway, you should also back up the following files before performing an upgrade:

- \Program Files (x86)\AVEVA Plant SCADA Access Anywhere\Plant SCADA Access Anywhere Authentication Server\EricomAuthenticationServer.exe.config
- \Program Files (x86)\AVEVA Plant SCADA Access Anywhere\Plant SCADA Access Anywhere Authentication Server\EricomAuthenticationServer\_Tenants.xml
- \Program Files (x86)\AVEVA Plant SCADA Access Anywhere\Plant SCADA Access Anywhere Secure Gateway\WebServer\PlantAccessAnywhere\config.js
- \Program Files (x86)\AVEVA Plant SCADA Access Anywhere\Plant SCADA Access Anywhere Secure Gateway\EricomSecureGateway.exe.config

The process for migrating the Secure Gateway configuration is explained in the [Access Anywhere Secure Gateway Installation and Configuration Guide](#).

2. Upgrade the software. Follow the procedure described in Upgrading to a New Version, then return to step 3.
3. Manually merge the settings from the files listed below into the new version.
4. Restart the affected services.

- **SE.Scada.AnywhereLauncher.exe.config**

Merge the 'applicationSettings' section from old config file to new config file, if applicable.

- **PlantAccessAnywhere\config.js**

Merge the defaults dictionary from the old config.js to new config.js file, if applicable.

- **"VjcaControl" and "VjcaView" Windows user groups**

Manually migrate any users in the existing "VjcaControl" and "VjcaView" Windows user groups to the new groups installed by Plant SCADA Access Anywhere 2.0 ('SCADA.AnywhereControl' or 'SCADA.AnywhereView').

## Plant SCADA Access Anywhere Web Component

The web component contains the resources that are used by the web browser to display an interface for users to use to connect to their remote application. These resources include HTML pages, JavaScript and CSS files and graphic images. Review the chapter on [Advanced Configuration](#) to modify the appearance and behavior of the web component interface.

---

**Note:** If you are running Plant SCADA Access Anywhere on an IPv6 network, it is recommended you configure a hostname that refers to the Access Anywhere Server. A hostname ensures the server is compatible with all supported browsers. If you attempt to connect to the Access Anywhere Server using a standard format IPv6 address in Internet Explorer, the connection will not be successful.

---

## Installing the Web Component

The Plant SCADA Access Anywhere web component is automatically installed along with the Access Anywhere Server. The web component may be found in the Access Anywhere Server folder:

**\Program Files (x86)\AVEVA Plant SCADA Access Anywhere\ Server\WebServer\PlantAccessAnywhere**

## WebSocket Connections

This section describes connection communication between WebSockets to both remote desktops and to the Plant SCADA Access Anywhere Secure Gateway.

## WebSocket Communication to Remote Desktops

The Plant SCADA Access Anywhere Server installation includes a self-signed certificate for SSL connections. It is recommended self-signed certificates are only used for testing purposes within a domain. Self-signed certificates will result in insecure connection warning messages in the user's web browser. They will also stop iOS devices from connecting to the Secure Gateway.

For production systems, use either a trusted certificate purchased from a certificate authority (for example, DigiCert), or a domain-issued trusted certificate provided by your IT administrator. When using a domain-issued certificate, the domain trusted root certificates need to be distributed to every device that connects to Plant SCADA Access Anywhere.

---

**Important:** A signed certificate needs to have a private key associated with it. A .CER file may not have a private key. Use a signed certificate that includes a private key, which usually has a .PFX extension.

---

**Note:** The DNS address of the Plant SCADA Access Anywhere Server or Secure Gateway server needs to match

the certificate name. If a wildcard certificate is being used, the domain needs to match. For example, if the certificate is for \*.example.com the server name needs to end with example.com.

To use a trusted certificate on to the Plant SCADA Access Anywhere Server, perform the following procedures.

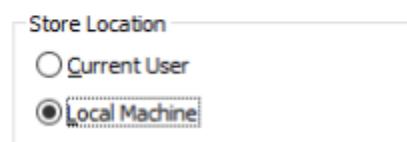
### Import the Certificate on to the Plant SCADA Access Anywhere Server

1. Locate the certificate file provided by your Plant SCADA Access Anywhere or domain administrator.
2. Double click the certificate file, or right-click and select **Import**. The Certificate Import Wizard is displayed.

#### Welcome to the Certificate Import Wizard

This wizard helps you copy certificates, certificate trust lists, and certificate revocation lists from your disk to a certificate store.

A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.



To continue, click **Next**.

3. Select **Local Machine** and click **Next**.
4. On the next screen, check that the path to the certificate is correct.

#### File to Import

Specify the file you want to import.

File name:

D:\Certificates\my-ca-server.pfx

[Browse...](#)

Note: More than one certificate can be stored in a single file in the following formats:

Personal Information Exchange- PKCS #12 (.PFX,.P12)

Cryptographic Message Syntax Standard- PKCS #7 Certificates (.P7B)

Microsoft Serialized Certificate Store (.SST)

5. Click **Next**.
6. On the next screen, enter the password for the certificate's private key.

**Private key protection**

To maintain security, the private key was protected with a password.

Type the password for the private key.

Password:

\*\*\*\*\*|

Display Password

7. Click **Next**.

8. On the next screen, specify the location where the certificate will be stored.

**Certificate Store**

Certificate stores are system areas where certificates are kept.

Windows can automatically select a certificate store, or you can specify a location for the certificate.

Automatically select the certificate store based on the type of certificate

Place all certificates in the following store

Certificate store:

Browse...

9. Click **Next**.

10. On the next screen, click **Finish** to import the certificate.

### Configure the Certificate on the Plant SCADA Access Anywhere Server

1. On the Plant SCADA Access Anywhere Server, open Windows Control Panel.

2. Search for "Certificates".

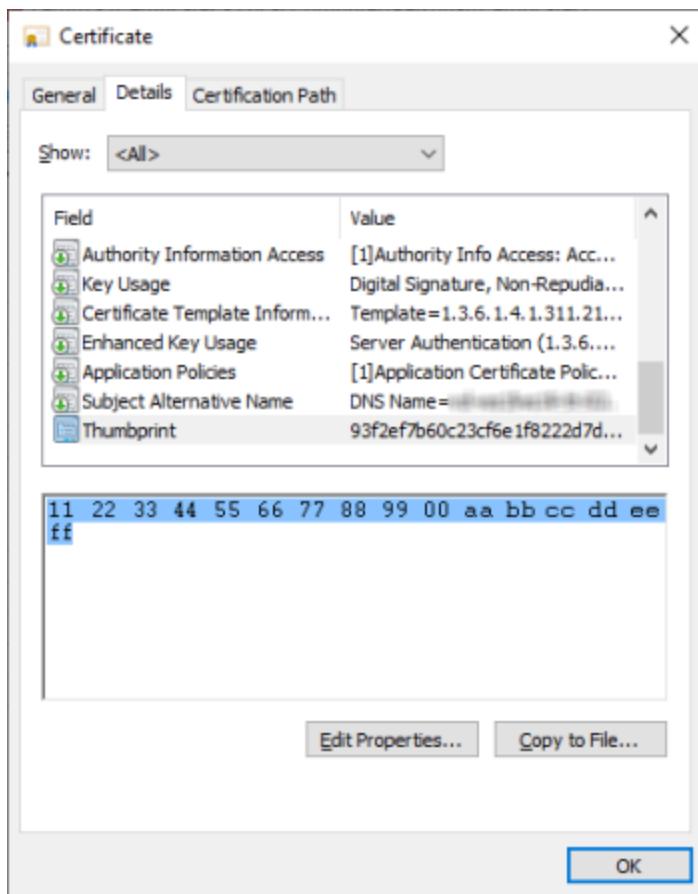
3. Select **Manage computer certificates** under Administrative Tools.

4. Locate the correct certificate in the Certificate Manager.

This will depend on where the certificate was imported. Typically this will be under 'Personal\Certificates'.

5. Double click the certificate, then select the **Details** tab.

6. Locate the **Thumbprint** property, then select the value and copy it to the clipboard.



7. Launch the Plant SCADA Access Anywhere Server Configuration tool.
8. Go to the **Security** tab.
9. Paste the thumbprint from step 6 into the **Certificate Thumbprint** field.

#### SSL Certificate

**Friendly Name:** Anywhere Signed Certificate  
**SAN:** DNS Name=localhost, DNS Name=<REDACTED>, DNS Name=<REDACTED>  
**Thumbprint:** 11 22 33 44 55 66 77 88 99 00 aa bb cc dd ee ff  
**Issued By:** CN=<REDACTED>  
**Valid From:** 2019/07/25 10:00:00

**Issued To:** <REDACTED>  
**Valid To:** 2021/07/24 10:00:00

#### Change Certificate

To change the above certificate, enter a new certificate's thumbprint below (eg: f5 a8 e6 a2 a8 9c 55 2a 8d f6 b7 d7 e7 22 ...).

Certificate Thumbprint:

Note: this change will only take effect after you click apply AND restart Access Server.

10. Click **OK** or **Apply**.
11. Restart the Plant SCADA Access Anywhere Server.

## WebSocket connections via Plant SCADA Access Anywhere Secure Gateway

When using the Plant SCADA Access Anywhere Secure Gateway, the connection between the Plant SCADA Access

Anywhere Server browser client and the Plant SCADA Access Anywhere Secure Gateway can be encrypted. Refer to the [Access Anywhere Secure Gateway Installation and Configuration Guide](#) for instructions.

## Configuring the Plant SCADA Access Anywhere Server

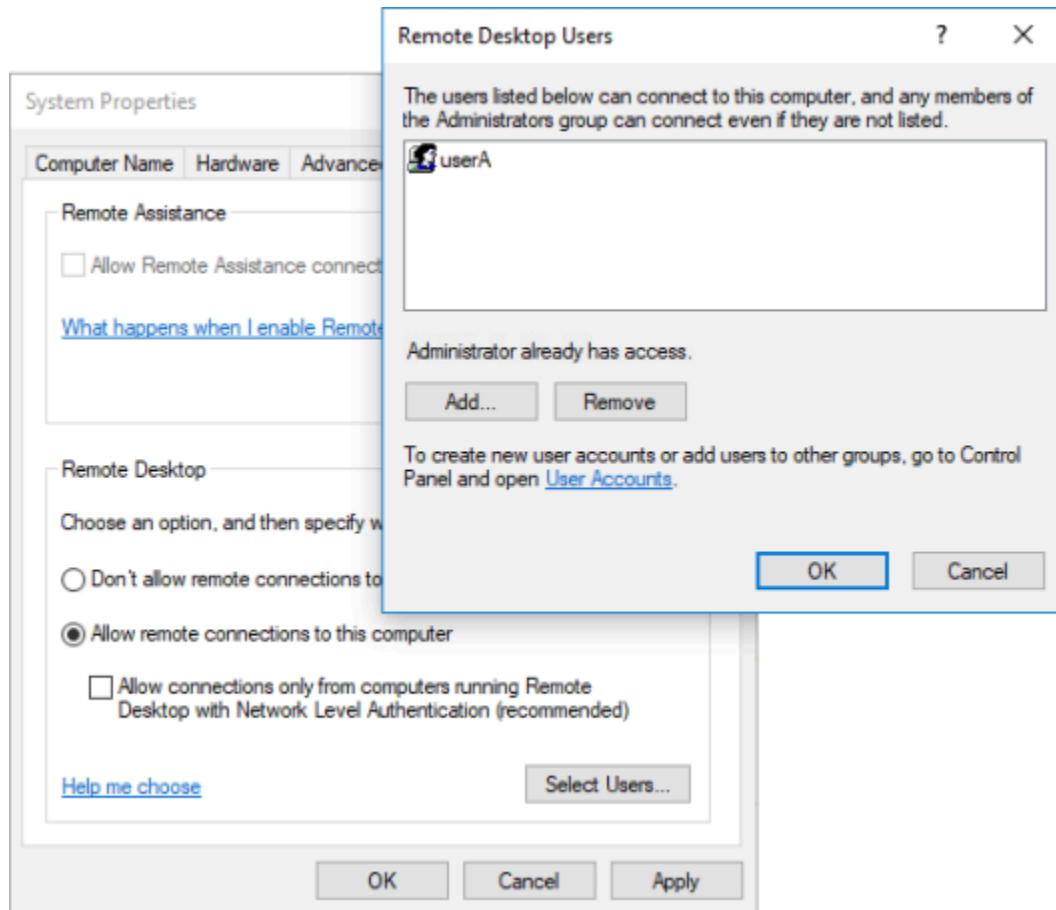
This chapter provides detailed information about configuring your Plant SCADA Access Anywhere Server. This involves the following tasks:

- Configure settings for the Plant SCADA Access Anywhere Server service in the [Configuration Console](#).
- [Configuring User Access](#) for Plant SCADA.

## Configuring Remote Desktop Services

To sign in remotely, users or groups to which users belong need to be added to the Remote Desktop Users group. Complete these steps on the computer on which the Plant SCADA Access Anywhere Server is installed:

1. Navigate to the Control Panel. Select **System and Security | System**.
2. Select the **Advanced system settings** option on the left. The System Properties dialog is displayed.
3. Click the **Remote** tab.
4. Select the **Allow remote connections to this computer** option.



5. Click **Select Users**. The Remote Desktop Users dialog is displayed.
6. Add the required users and click **OK**.

**Note:** For more details about configuring RDS, search for Tech Note 8956 on the [Tech Note, FAQ](#) page of the [AVEVA Knowledge and Support Center](#). You will need to register with the site before you can access the document.

## Configuring User Access

Depending upon their access, users can launch a AVEVA Plant SCADA view-only client or a control client.

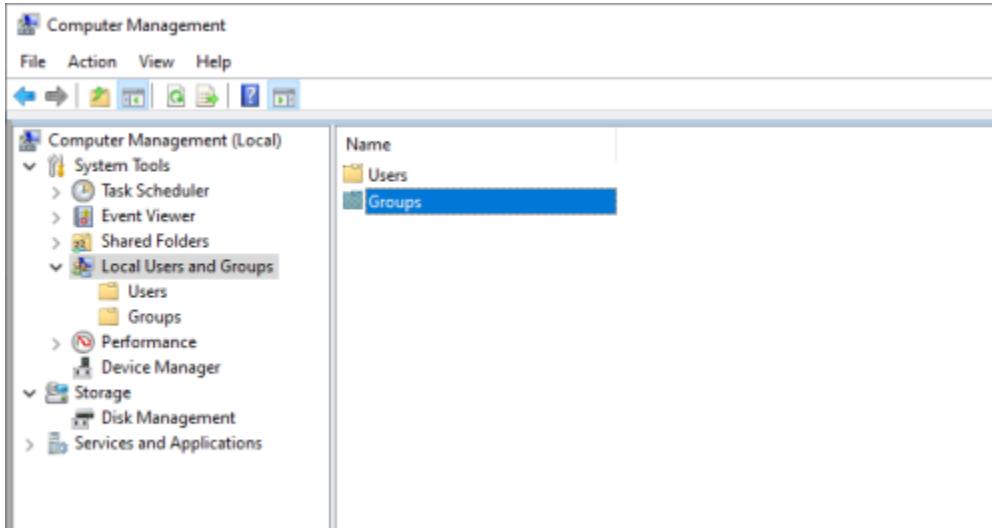
- **View-only client** - a computer configured with view-only access to the AVEVA Plant SCADA runtime system. No control of the system is possible, but access to data monitoring is available.
- **Control client** - an interface between the AVEVA Plant SCADA runtime system and an operator. If you are using AVEVA Plant SCADA on a network, every AVEVA Plant SCADA computer (on the network) is a control client.

Access to the client type is granted through two special Windows user groups ('SCADA.AnywhereControl' or '[SCADA.AnywhereView') created by the installer on the computer where the Plant SCADA Access Anywhere Server is installed.

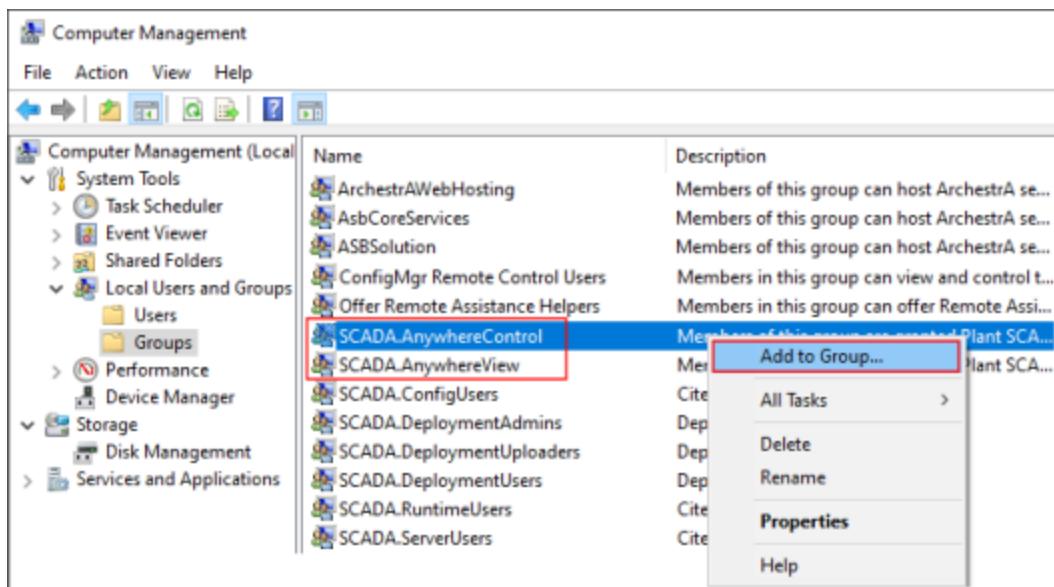
**Note:** If you have recently upgraded to Plant SCADA Access Anywhere 2.0, you will need to manually migrate the users in the existing "VjcaControl" and "VjcaView" Windows user groups to the new 'SCADA.AnywhereControl' and 'SCADA.AnywhereView' groups.

Local users and domain groups need to be added to the required group manually. To do this:

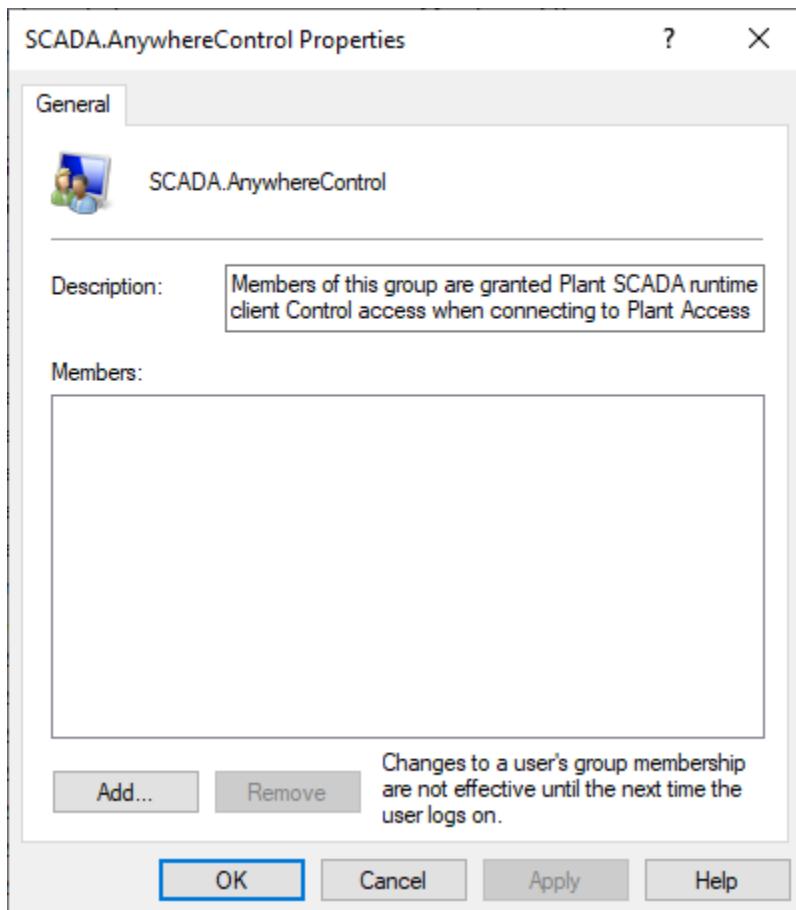
1. Navigate to the Control Panel. Select **Administrative Tools | Computer Management**.
2. In the left pane, navigate to **System Tools | Local Users and Groups**.



3. In the right pane, double-click on **Groups** to expand the node.
4. Locate the Plant SCADA Access Anywhere groups, and right-click on the group to which you want to add a user.



5. Select **Add to Group** from the context menu. The group properties dialog box appears.



6. Click **Add**. The Select User dialog box appears.
7. In the **Enter the object name to select** box, type the name of the user you want to add.
8. Click **Check Names**. Once the user name is verified, click **OK**. The group properties dialog box appears.
9. Click **Apply**. The user is added to the selected group.

---

**Note:** If you are using Plant SCADA 2020 R2 (8.30), the members of the Plant SCADA Access Anywhere user groups must also be included in Plant SCADA's SCADA.RuntimeUsers group to access the runtime display client.

---

## The Configuration Console

The Server Configuration console presents a series of tabs that enable an administrator to configure various settings for the server service.

Double-click *ServerConfiguration.hta* file, located in the following folder:

\Program Files (x86)\AVEVA Plant SCADA Access Anywhere\Server

Alternatively, select AVEVA Plant SCADA Access Anywhere | Plant SCADA Access Anywhere **Server Configuration** from the list of programs on the Windows Start menu.

In general, changing Access Anywhere Server configuration is not required. The default settings are recommended.

---

**Important:** It is recommended that end users are not given access to the computer on which the Access Anywhere Server is installed.

---

The following sections describe the different configuration tabs of the Access Anywhere Server.

- [General Tab](#)
- [Performance Tab](#)
- [Communication Tab](#)
- [Acceleration Tab](#)
- [Security](#)
- [Logging Tab](#)
- [Advanced \(for Administrator User Only\)](#).

### General Tab

This page provides functions to start and stop the Plant SCADA Access Anywhere Server service. For certain configuration changes, a service restart is required. This page also displays the number of active Plant SCADA Access Anywhere Server client sessions connected to this computer.

AVEVA Plant SCADA Access Anywhere Server 2.0 Configuration

General Performance Communication Acceleration Security Logging Advanced

Plant SCADA Access Anywhere Server service state: **Running**

Plant SCADA Access Anywhere Server status: **Active**

Number of sessions: **0**

Started at: **16/11/21 21:29:38**

**Start Server**

**Stop Server**

The Plant SCADA Access Anywhere Server and the Ericom Licensing Server run as a [Windows service](#). When the Plant SCADA Access Anywhere Server service is stopped, Plant SCADA Access Anywhere Clients will not be able to connect to this host with RDP acceleration. Any active accelerated connection will be dropped (sessions will be disconnected). Regular RDP connections to this host will not be affected.

Changing certain settings, such as **Communication** and **Logging** take effect only after the services are restarted. After changing such settings, apply the changes, stop the services and then start them again.

Whenever the Plant SCADA Access Anywhere Server service is restarted, every session on the server will be disconnected.

## Performance Tab

The Performance tab displays current performance statistics related to Plant SCADA Access Anywhere connections.

AVEVA Plant SCADA Access Anywhere Server 2.0 Configuration

General Performance Communication Acceleration Security Logging Advanced

### Server to Client communication

Number of sessions: **0**

Average compression ratio: **61 %**

Total data received from host: **12 MB**

Total data sent to client: **4 MB**

Real-time cumulative performance information for all sessions since Plant SCADA Access Anywhere Server was started. Counters are reset when the Plant SCADA Access Anywhere Server service is restarted. Display is automatically updated approximately once every 10 seconds.

## Communication Tab

This tab provides options to change the Plant SCADA Access Anywhere Server port and the address of the host computer running RDP.

When using a Plant SCADA Access Anywhere Server listening port other than the default (8080), the port number needs to be explicitly specified in the client address field (i.e., `http://<computer name>:5678/`).

When running Plant SCADA Access Anywhere Server on a computer with multiple network cards, change the

RDP host address from localhost to the IP or DNS address of the network card that has RDP access to the system. A change to either setting requires a service restart. This can be done via the General tab or using the Windows Service Manager.

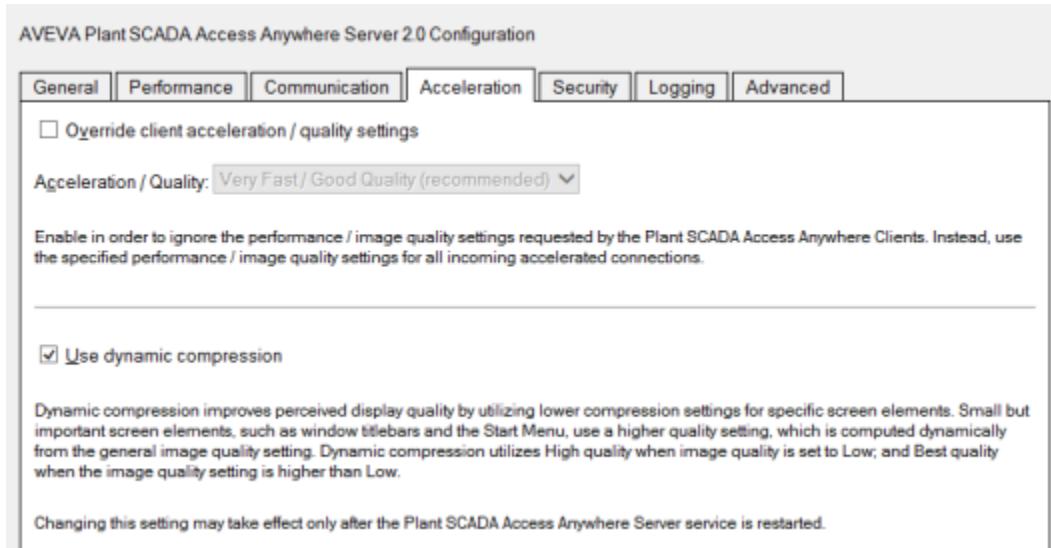


## Acceleration Tab

This tab provides options to change the Acceleration/Quality level and disable dynamic compression. When the **Override client acceleration / quality settings** option is selected, every session using the configured settings, and every client setting are ignored. When selecting or clearing this setting, the service needs to be restarted for the change to take effect. When the setting is enabled, changing the acceleration level does not require a service restart, but active users need to reconnect to use the new setting.

Dynamic Compression identifies small graphical objects on the screen (such as toolbar icons, task bar icons, Start Menu icons, etc.) and compresses them. Setting the image quality to low will result in maximum compression, minimizing the impact on the network. Other graphical objects are compressed at the selected quality. This provides the visual impression of a high quality remote desktop session.

By default, this feature is enabled. To disable, clear the **Use dynamic compression** box.



## Security

This page configures the Plant SCADA Access Anywhere Server security settings.

AVEVA Plant SCADA Access Anywhere Server 2.0 Configuration

General Performance Communication Acceleration Security Logging Advanced

## Plant SCADA Access Anywhere supports strong SSL encryption

Encrypt Plant SCADA Access Anywhere communication:

By default Plant SCADA Access Anywhere uses the same security settings as Microsoft RDP - if RDP is encrypted then Plant SCADA Access Anywhere will be encrypted. If RDP is not encrypted then Plant SCADA Access Anywhere will not be encrypted either. Set to **Always** for Plant SCADA Access Anywhere to always encrypt regardless of the RDP settings.

**Data transmitted from the clients to the server, including user credentials, is always encrypted regardless of Plant SCADA Access Anywhere and RDP security settings.**

For best performance and lowest load on the server set the RDP Security Level to Low (for 2003/XP also set the Security Layer to RDP Security Layer). This setting can be changed using the RDS (TS) Session Host Configuration or using Local Computer / Group Policies. After performing this change, modify setting above to **Always** if encryption is required.

**Note:** Plant SCADA Access Anywhere provides integrated 128-bit SSL encryption. For optimal performance, set the host's RDP Security Encryption level to Low and change the **Encrypt Plant SCADA Access Anywhere communication** to **Always**. Using this configuration, Plant SCADA Access Anywhere SSL encryption will be used instead of the RDP encryption. Avoid setting this if users will be connecting directly to RDP regularly, as those sessions will end up using Low encryption.

## Logging Tab

This tab provides options to enable/disable certain logging features. Technical Support may request a debugging log for diagnostic purposes. The debugging log is enabled here.

AVEVA Plant SCADA Access Anywhere Server 2.0 Configuration

General Performance Communication Acceleration Security Logging Advanced

Log level:

Maximum log file size:  MB

Maximum number of log file backups to keep:

Changing this setting may take effect only after Plant SCADA Access Anywhere Server service or/and Ericom Licensing Server service are restarted.

Log files will be generated in a subfolder called *Logs* under the Plant SCADA Access Anywhere installation folder. The current log files are called *AccessAnywhere.log* and *LicenseManager.log*. When a log file reaches its maximum allowed size it will be backed up, and new log file will be created.

For Plant SCADA Access Anywhere Server, Backups of previous logs will have the format *AccessAnywhereServer.bck-xxx.log*, where an *xxx* value of *007* is the most recent backup.  
 For Ericom Licensing Server, Backups of previous logs will have the format *LicenseManager.xxx.log*, where an *xxx* value of *000* is the most recent backup.

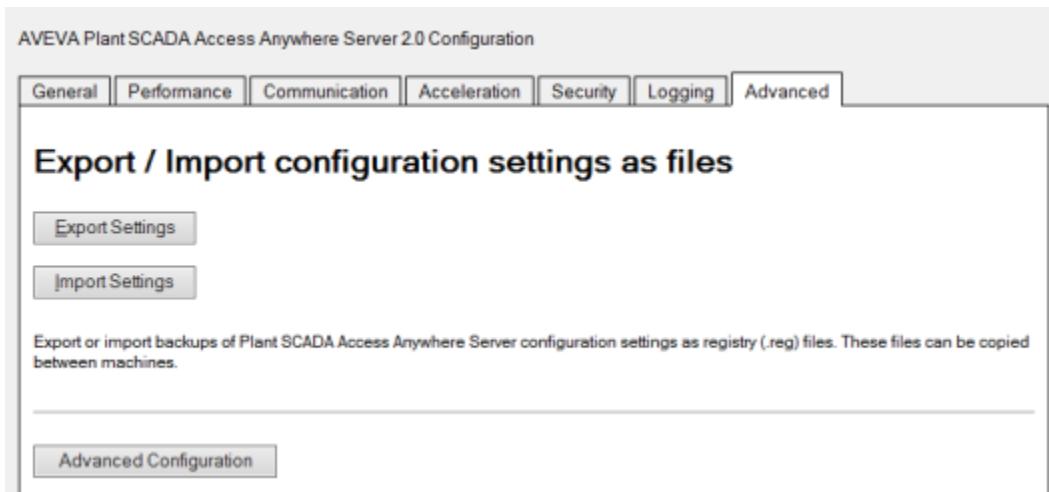
Generate debugging log (for AVEVA support)

Changing these settings may take effect only after Plant SCADA Access Anywhere Server service or/and Ericom Licensing Server service are restarted.

## Advanced

This page provides access to advanced Server settings that are stored in the system's Registry.

- **Export Settings:** Exports the Plant SCADA Access Anywhere Server Registry key to the user's home folder (i.e., My Documents).
- **Import Settings:** Imports previously saved Plant SCADA Access Anywhere Server Registry settings.
- **Advanced Configuration:** Adds every configurable Registry key setting to the Registry. By default, only settings that are changed from the default value are saved into the Registry.



## Configuring Mobile Devices

This chapter contains information about using Plant SCADA Access Anywhere on mobile devices. The following information is covered here:

- [Supported Browsers](#)
- [Logging On to Plant SCADA Access Anywhere](#)
- [Automatic Display Resize](#)

## Supported Browsers

With Plant SCADA Access Anywhere, users can access remote AVEVA Plant SCADA from HTML5 compatible web browsers on any device including smart phones, tablets, and laptop computers. To start a session, navigate to the *start.html* file that is installed on the Plant SCADA Access Anywhere Server. To do this, point a browser to the Plant SCADA Access Anywhere Server URL:

*http://machineaddress:8080*

## Browsers Tested with Plant SCADA Access Anywhere

- Internet Explorer 11
- Microsoft Edge

- Google Chrome
- Safari on Apple iOS

Multiple Plant SCADA Access Anywhere sessions can be opened in different tabs within the web browser, or in different browser windows. When a session is not in use (its tab or window is not displayed) it will reduce its CPU and memory utilization.

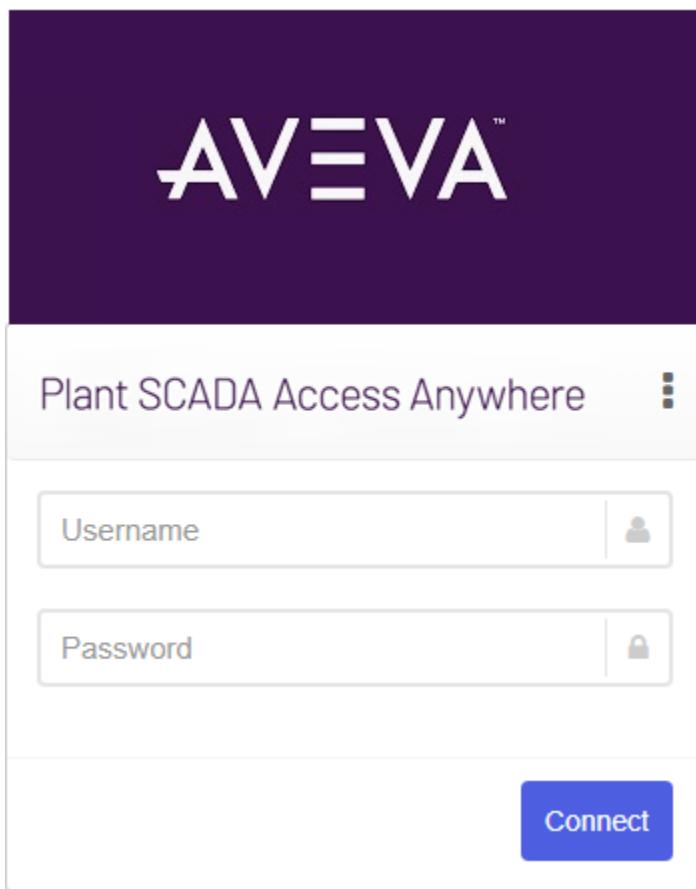
**Note:** If you are running Plant SCADA Access Anywhere on an IPv6 network and you attempt to connect to the Access Anywhere Server using an IPv6 address in Internet Explorer, the connection will not be successful. Use a different browser, or check with your system administrator if a hostname has been configured for the Access Anywhere Server.

## Logging On to Plant SCADA Access Anywhere

To log on to the Plant SCADA Access Anywhere Connection Web Page, follow these steps:

**Note:** If you have any trouble remotely connecting to the Plant SCADA Access Anywhere environment, see [Checking Connectivity](#) in the *Plant SCADA Access Anywhere Web Client User Guide* for more information.

1. Navigate to `http://<Access Anywhere Server Node Name>:8080/`. The logon dialog appears.



2. Enter the connection parameters (see the table below).
3. Tap or click **Connect** to initiate the connection. The progress indicator is displayed before connection is established.

If the Plant SCADA session does not start, and any notification appears, then contact the Access Anywhere Administrator.

Connection Parameters	Description
User Name	Credentials to log on to the RDP host. It can optionally contain domain specification, for example, domain\user. If it is not specified, you will be prompted for credentials by the RDP host. The user needs to be a Windows user and a member of the 'SCADA.AnywhereControl' or '[SCADA.AnywhereView] Windows group. They also need to be a valid Plant SCADA user.
Password	Corresponding password for the user name. This value should not be saved for future connections. If it is not specified, you will be prompted for credentials by the RDP host.

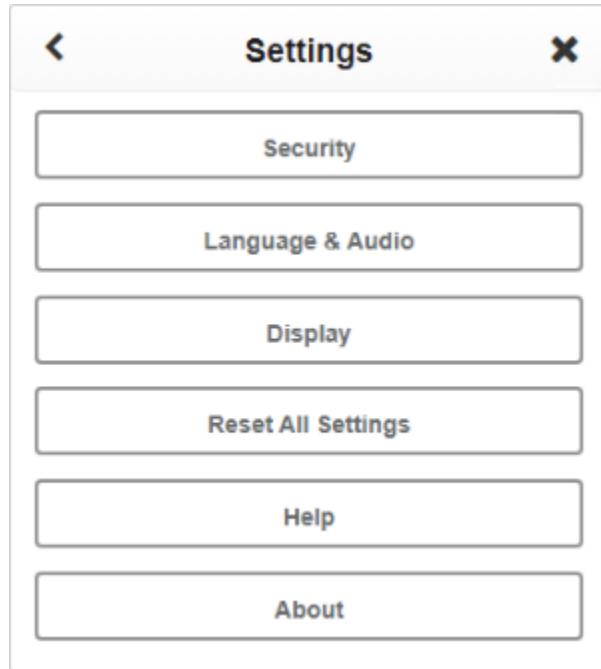
**Note:** When using a Secure Gateway, the **User Name** and **Password** fields are mandatory, otherwise they are optional.

## Configuring Advanced Settings

To configure advanced settings:



- Click the icon. The following tabs appear as shown below:



- Click on each tab to view the settings available for configuration. For example, to view Display settings, click the **Display** tab.

To return to the settings dialog, click . To close the Settings dialog and return to the Logon dialog, click .

3. Complete the required settings.

## Security Settings

Option(s)	Description
Enable SSL encryption for remote session	Select this option to allow the client to use Secure Socket Layer (SSL) encrypted WebSocket communication to the Plant SCADA Access Anywhere Server. As a default, this option is selected.

## Language & Audio Options

Option(s)	Description
Display Language	Select the language in which you wish to view the Plant SCADA Access Anywhere interface. The Display Language only applies to the logon web page, and is not propagated to the Plant SCADA client.
Keyboard Locale	Select the keyboard region to be used in the session (keyboard_locale). If you select a non-English keyboard locale, your computer will need to be configured to use a matching keyboard in the Windows <b>Region &amp; Language</b> settings. Some keyboards may not work correctly with Internet Explorer or Edge browsers.
Use keyboard scan-codes	Enables scan codes. Certain applications use scan codes and will require this setting to be enabled.
Remote Audio Playback	Configure where the session's sound will play: local computer, remote computer, or do not play. Audio playback is not supported with Internet

Option(s)	Description
	Explorer version 11.

## Display Options

Option(s)	Description
Acceleration / Quality	The acceleration, or degree of quality can be specified by selecting options from a drop down list. Faster acceleration will result in lower quality images.
Screen Resolution	Sets the resolution size of the Plant SCADA Access Anywhere session. Select a value from the drop down list of values. For example: "800 x 600". Select a screen resolution that matches your AVEVA Plant SCADA project. The default may be configured by the administrator to already match your AVEVA Plant SCADA Project.
Automatic session resize	Enables/disables automatic display resizing. By default, automatic resizing is selected (enabled). This means whenever the browser window is resized, the Plant SCADA Access Anywhere session will automatically adjust itself to the new dimensions. See <a href="#">Automatic Display Resize</a> .

**Note:** For details about the version of Plant SCADA Access Anywhere you are running, click the **About** tab in the Settings dialog.

Click **Reset All Settings** to revert to the default settings.

Depending upon the privileges your administrator has assigned, a Plant SCADA View-only client or Control client will be launched. A View-only client is a computer configured with view-only access to the AVEVA Plant SCADA runtime system. No control of the system is possible, but access to data monitoring is available. A Control client is the interface between the Plant SCADA runtime system and an operator. If you are using Plant SCADA on a network, every AVEVA Plant SCADA computer (on the network) is a control client.

Access to either client depends upon the permissions granted to you by your administrator. For more information, refer to the Plant SCADA Access Anywhere Installation and Configuration Guide.

**Note:** Click the **Help** tab in the Settings dialog to view the [Plant SCADA Access Anywhere Web Client User Guide](#).

## Automatic Display Resize

By default, Plant SCADA Access Anywhere supports automatic display resize. Whenever a browser window is resized, the Plant SCADA Access Anywhere session automatically adjusts itself to the new dimensions (assuming the AVEVA Plant SCADA Runtime window is set to full screen).

To resize a browser window, drag any corner of the browser window and release it when the desired dimensions are reached. If a browser is placed into full screen mode, the Plant SCADA Access Anywhere session will automatically expand to the full screen. Be aware that resizing is subject to the display limitations of the Plant SCADA runtime system.

You can enable/disable automatic resizing via the **Display** settings for the Plant SCADA Access Anywhere Web Client, accessible via the advanced settings on the Logon dialog.

## Using Gestures on Client Portable Devices

### Google Chromebooks

Plant SCADA Access Anywhere operates on Google Chromebook and Chromebox just like it does with a Google Chrome browser. Here are some tips to keep in mind when using Access Anywhere with a Chromebook or Chromebox.

Function	Description
Mouse Left-click	Click the Chromebook trackpad with one finger.
Mouse Right-click	Click the Chromebook trackpad with two fingers
Scrolling a document or website	Drag two fingers on the Chromebook trackpad up or down to scroll
Configure Chromebook	In the address field: <i>chrome://settings</i>

### Chrome Keyboard

The Chromebook keyboard lacks several keys that are used by Windows. ChromeOS provides standard mappings that use existing keys with the ALT button to represent certain missing keys. Plant SCADA Access Anywhere supports these key combinations:

Command	Key Combination
Delete (DEL)	ALT+Backspace
Page Up	ALT+Up
Page Down	ALT+Down
Home	CTRL+ALT+Up
End	CTRL+ALT+Down

In addition, Plant SCADA Access Anywhere provides special non-standard mappings for additional key combinations on ChromeOS.

Command	Key Combination
F1	CTRL+1
F2, ...	CTRL+2, ...
ALT+TAB	ALT+`
ALT+SHIFT+TAB	ALT+SHIFT+`
CTRL+Home	CTRL+ATL+Left
CTRL+End	CTRL+ALT+Right

## Tablet and Smartphones

Plant SCADA Access Anywhere can operate on tablets or smartphones with an HTML5 compliant browser (see list of browsers in Plant SCADA Access Anywhere Readme). Browser versions that have been tested and their specific behaviors are detailed in the *Plant SCADA Access Anywhere Web Client User Guide*.

When you design Plant SCADA clients for use with Plant SCADA Access Anywhere, remember that touch devices have different interface requirements and capabilities than a keyboard and mouse. For example, input animations should not invoke a Plant SCADA or Windows keyboard, as mobile devices have their own.

With existing AVEVA Plant SCADA clients that make use of mouse events and keys or key combinations without a supported equivalent, you may want to modify your application to use alternate application events.

The following list provides gestures available when using Plant SCADA Access Anywhere from a tablet or smartphone device where a physical keyboard and mouse is not available. Functionality will vary across different devices and certain commands may not be available.

- Single Tap performs a left click.
- Single long Tap performs a right-click.
- Tap + Hold + Drag performs a select then drag/scroll function.
- Double Tap, or tapping once with two fingers, performs double-click.
- Tap with three fingers sends Back command to a remote browser.
- Swipe down with three fingers is Page Up.
- Swipe up with three fingers is Page Down.
- Drag left or right with three fingers performs a left arrow and right arrow respectively.
- Tap the keyboard icon (upper right-hand corner of window) to open/close the virtual keyboard.
- Swipe and pinch gestures will apply to the Plant SCADA Access Anywhere session (i.e. zoom in with pinch in).
- (iOS only) When saving a Plant SCADA Access Anywhere icon to the iOS desktop, the shortcut will open the Plant SCADA Access Anywhere session full-screen mode. The browser's toolbar will be hidden and there will be more remote desktop area available.

## HTTPS Mode

For environments where WebSockets support is not available, Plant SCADA Access Anywhere can work in HTTPS

mode so that communication will be sent via HTTPS only. HTTPS mode will only be used if WebSockets is not available. WebSockets will be used when available as it will provide better performance. HTTPS mode is required when using SSL VPNs that only proxy HTTPS traffic.

To enable HTTPS Mode, the Plant SCADA Access Anywhere Secure Gateway is required. The Plant SCADA Access Anywhere Server web pages need to be delivered using the web server built into the Plant SCADA Access Anywhere Secure Gateway (files are located under the Webserver/PlantAccessAnywhere folder). Carry out the following steps to enable Plant SCADA Access Anywhere for HTTPS support.

1. Install Plant SCADA Access Anywhere Server on the desired RDP Host.
2. Install the Secure Gateway in a separate machine located in a DMZ. The Secure Gateway needs to be installed on a server that is accessible by the target end-user group(s).
3. To connect to the Plant SCADA Access Anywhere Server using HTTPS - enter the Plant SCADA Access Anywhere URL of the Secure Gateway (the Secure Gateway includes the Plant SCADA Access Anywhere web component)  
`https://<securegatewayaddress>/PlantAccessAnywhere/start.html`
4. Enter the parameters for the target Plant SCADA Access Anywhere Server in the start.html page.
5. Upon connection, if HTTPS mode is active a '-' symbol will then be shown as a prefix to the address in the browser tab.

## Advanced Configuration

Plant SCADA Access Anywhere also easily integrates with other web pages and portals. The application can accept configuration settings from other pages or directly from a web server. These settings can also be displayed in the Plant SCADA Access Anywhere start page for the user to view and modify, or trigger an automatic connection.

### Static Configuration of Config.js

An administrator can modify configuration settings for Plant SCADA Access Anywhere by editing the config.js file that is installed as part of the Plant SCADA Access Anywhere web component. This is a JavaScript file that can be modified using any text editor. It is located in the following directory:

**\Program Files (x86)\AVEVA Plant SCADA Access Anywhere\Server\WebServer\PlantAccessAnywhere\config.js**

---

**Note:** Create a backup of the original *config.js* file before making any changes. This will make it easy rollback to the original configuration.

Many of the settings in the file have the following format:

*name: value*

A value can be a number, a flag (true or false), or text enclosed in quotes. Some settings are prefixed by a double slash // which means they are disabled. Remove the double slash in order to set a value for the setting. Javascript rules apply in this file, certain characters need to be escaped (i.e. backslash). Once the settings are configured, save the file and the next user will have the new settings applied.

Refer to the [Settings Table](#) for a description of each setting.

## Settings Table

The *config.js* file contains the following configuration settings. Setting names are case sensitive. When settings are specified using cookies, their name are prefixed by *EAN\_*.

Setting Name	Description
overrideSaved	<b>False</b> (default), settings that the user changes are preserved between sessions and override values set in <i>config.js</i> . Change to <b>True</b> for <i>config.js</i> to override preserved settings.
onlyHTTPS By	By default, Plant SCADA Access Anywhere first attempts to connect using WebSockets. If the Secure Gateway is used with Plant SCADA Access Anywhere, the connection will fall back to HTTPS when WebSockets is not available. If this setting is true, HTTPS is used immediately.
noHTTPS	By default, Plant SCADA Access Anywhere first attempts to connect using WebSockets. If the Secure Gateway is used with Plant SCADA Access Anywhere, the connection will fall back to HTTPS when WebSockets is not available. If this setting is true, only WebSockets will be used and HTTPS fallback will be disabled.
hidden	A comma- or space-separated list of field names as they appear in <i>config.js</i> . For example "username, password, domain". The listed fields will be hidden so that the user will not be able to modify them. To hide a button, such as the Advanced button, prefix the button text with the word show. For example, "showAdvanced, showAbout" hides both the Advanced and About buttons. Hidden variables ignore previously saved settings.
settings (URL parameter only)	Name of a Configuration Group to be used.
wsport	The default WebSocket port that will be used by the client. The value specified in the file (8080 by default) will be used for both encrypted and unencrypted WebSocket communication. The user can override this value by explicitly specifying a port address in the client UI. For backward compatibility with older versions of Plant SCADA Access Anywhere Server, this behavior can be modified. If singlePort is set to false then the port value specified is only for encrypted communication. The value specified in the file plus

Setting Name	Description
	one (8081 by default) will be used for unencrypted WebSocket communication.
gwport	The default gateway port that will be used if it is not explicitly specified in the address field.
dialogTimeoutMinutes	Timeout period, in minutes, after which an inactive dialog is automatically closed and the session is logged off. This is only relevant for dialogs that have a logoff button.
sessionTimeoutMinutes	Timeout period, in minutes, after which an inactive session is disconnected. This timeout is reset whenever user clicks on the keyboard or a mouse button. The default value is 0, which disables this feature.
specialkeys	Enables support for special RDP key combination commands, such as CTRL+ALT+END which starts the Windows NT Security dialog box (similar to local CTRL+ALT+DEL).
chromeKeys	<b>true</b> (default) support special ChromeOS keys combinations
showDownload	<b>true</b> displays a link in the connection dialog to download the Plant SCADA Access Anywhere Server installer.
clipboardSupport	<b>true</b> (default) enables clipboard functionality; <b>false</b> disables it.
clipboardTimeoutSeconds	The delay duration before the clipboard image automatically fades out.
clipboardUseFlash	<b>true</b> (default) uses Flash when available for one-click copy into local clipboard.
clipboardKey	Key to open clipboard paste dialog, set to <b>false</b> to disable.
console	<b>false</b> (default) set true to enable RDP console mode.
settingsURL	<b>false</b> (default) set <b>true</b> to enable RDP console mode.
endURL	URL to open to after the Plant SCADA Access Anywhere session has ended (# value closes window). If there is a prefix with the symbol ^ then this sets the value of window.location instead of top.location. This

Setting Name	Description
	is useful when the Plant SCADA Access Anywhere session is embedded in a frame.
address	Address of Plant SCADA Access Anywhere Server. Leave blank for the standard configuration.
fulladdress	Address of RDP host. Leave blank for the standard configuration.
username	Username to pass into the Plant SCADA Access Anywhere session.
password	Password to pass into the Plant SCADA Access Anywhere session (entered as clear text in <i>config.js</i> file)
domain	Domain to pass into the Plant SCADA Access Anywhere session.
remember	<b>False</b> (default) determines whether the user's password will be saved in the Plant SCADA Access Anywhere page for future use. Set to <b>true</b> to enable password saving (not recommended for kiosk usage).
encryption	<b>False</b> determines if encryption will be enabled from the Plant SCADA Access Anywhere client to the server
blaze_acceleration	<b>True</b> determines if RDP acceleration will be used
blaze_image_quality	Sets the quality level using a numeric For example: 40 (fair quality), 75, 95 (best).
resolution	Sets the resolution size of the Plant SCADA Access Anywhere session. The value set needs to be a valid option under the Plant SCADA Access Anywhere screen resolution setting. For example: "1024,768" For Full Screen, use: screen.
use_gateway	<b>False</b> (default), set to true to use a Secure Gateway for remote access.
gateway_address	Defines the address and port of the Secure Gateway For example: secure.acme.com:4343
useScancodes	No longer in use, see <i>convert_unicode_to_scancode</i> .
convert_unicode_to_scancode	<b>False</b> (default), set to <b>True</b> when using certain applications that send characters as scancodes (i.e. VMware vSphere Client, any application where you

Setting Name	Description
	may have face challenges typing text). This setting will generate scancodes based on the selected locale.
leaveMessage	The message displayed to the user after they navigate away from an active session.
audiomode	0, enables audio redirection (default) 1, play audio on remote computer 2, disables audio redirection
name	Defines a custom string for the connection name. By default, the RDP Host address is used.
minSendInterval	Specifies the minimum duration between mouse position messages sent from the client when the mouse button is pressed. Units is milliseconds.

**Note:** In some cases, the local browser needs to be closed and reopened before changes take effect. The local browser cache may also need to be cleared.

## Defining Configuration Groups

Every user shares the configuration settings defined in the *config.js* configuration file. It is possible to specify special settings that will override the global settings for certain groups of users. Multiple configuration groups are defined in the configuration file.

For example, if the Marketing group will have clipboard redirection and printing enabled, change *config.js* as follows:

```
var defaults = { //this already exists in the file
...
  "Marketing":{ //Bold text are new additions
remember:false,
audiomode:0
},
};
```

**Note:** The double quotes surrounding Marketing needs to be identical. It may be necessary to delete them and re-type them if the text was copied from another source.

Also, the last setting of the configuration group should not have a ',' at the end. This comma will be placed after the closing bracket '}'.

In the URL to be used by the Marketing group, add the **settings** parameter:

<http://<computer name>:8080/PlantAccessAnywhere/start.html? settings=Marketing>

## Settings Precedence

When a Plant SCADA Access Anywhere client starts, it reads configuration information from a variety of sources. If two or more sources contain different values for the same setting, the value that Plant SCADA Access Anywhere will use is determined by the following precedence order:

### Highest precedence to Lowest precedence

- URL parameters
- Cookies
- Saved settings from previous session
- *config.js*

For example, if the gateway\_address is specified to be "server1" in config.js but "server2" in a cookie (EAN\_gateway\_address), then the value "server2" will be used.

If the setting override Saved is set to true in *config.js*, then any settings predefined in the config.js file will override previously used settings, and the precedence order will change slightly:

### Highest Precedence to Lowest Precedence

- URL parameters
- Cookies
- Saved settings from previous session
- *config.js*

---

**Note:** These settings become effective only after the user starts a new session. In some cases, the local browser needs to be closed and reopened before changes become effective. The local browser cache may also need to be cleared.

---

## Passing Credentials using Form POST

User credentials may be passed to Plant SCADA Access Anywhere using the form POST method. This functionality is used to provide SSO (single sign-on) from an outside source that has already authenticated the user (such as an SSL VPN).

The Plant SCADA Access Anywhere Secure Gateway is required in order to use form POST with Plant SCADA Access Anywhere. Refer to the [Access Anywhere Secure Gateway Installation Guide](#) manual for detailed instructions.

## Embedding Plant SCADA Access Anywhere in an iFrame

To embed Plant SCADA Access Anywhere within a third-party web page using the iframe mechanism, simply place an iframe tag within the containing page, and have the iframe's SRC attribute reference the Plant SCADA Access Anywhere URL.

For example:

```
<body>
  <h1>Embedded Plant SCADA Access Anywhere</h1>
  <iframe src="http://127.0.0.1:8080/PlantAccessAnywhere/start.html" style="width:1024px;
  height:768px"></iframe>
</body>
```

When a Plant SCADA Access Anywhere session ends, it can be configured to send the browser to a specified URL using the endURL setting.

- Specify a simple URL to redirect the iframe.

- Prefix the URL with ^ to redirect the iframe's parent (container).
- Prefix the URL with \$ to redirect the top-most container.
- Specify # and the URL will close the browser tab.

## SSL VPN Configuration

Plant SCADA Access Anywhere is compatible with many SSL VPNs. An SSL VPN that does not support WebSockets will require the Secure Gateway as well. Juniper IVE version 7.4 supports WebSockets, so the Secure Gateway is not required.

Plant SCADA Access Anywhere has been tested with Juniper's SA SSL VPNs and the documentation in this section will be based off Juniper's administration pages. Configuration with other third-party SSL VPN appliances will be similar to the procedures described here (difference are mostly in terminology).

### Web Proxy with Juniper 7.4 (or later)

Juniper version 7.4 (or later) supports WebSockets. Plant SCADA Access Anywhere links are published in the Juniper's web interface as web applications. To publish a new Access Anywhere connection, go to the Juniper Admin page and do the following:

1. Go to **Resource Profiles | Web | New Web Application Resource Profile**.
2. Enter the **Name** of the Plant SCADA Access Anywhere connection that the users should see.
3. Enter the Plant SCADA Access Anywhere URL in the **Base URL** box.
4. Click **Save and Continue**.
5. In the Roles dialog add every role that should have access to the Plant SCADA Access Anywhere link and click **Save Changes**.
6. In the Bookmarks tab, enter the desired label for the connection.
7. When you log into Juniper, a Plant SCADA Access Anywhere link will be displayed under the Web bookmarks section (i.e. Plant SCADA Access Anywhere). Click on the link to connect to a Plant SCADA client published with Plant SCADA Access Anywhere.

### Web Proxy with Older Juniper Versions

Juniper versions prior to 7.4, and many other SSL VPNs will not support native WebSockets. Such SSL VPNs require HTTPS Mode (see HTTPS Mode chapter in this guide) to run Plant SCADA Access Anywhere. HTTPS mode is enabled by installing the Secure Gateway.

To use Plant SCADA Access Anywhere and the Secure Gateway:

1. Go to **Resource Profiles | Web | New Web Application Resource Profile**.
2. Enter the **Name** of the Plant SCADA Access Anywhere connection that the users should see.
3. Enter the Gateway's Plant SCADA Access Anywhere URL address as the Base URL. Click **Save**.
4. Go to **Autopolicy: Web Access Control**.
5. Edit the automatically entered address and delete the subfolder "Plant SCADA Access Anywhere".

Instead of [https://GWaddress.com:443/PlantAccessAnywhere/\\*](https://GWaddress.com:443/PlantAccessAnywhere/*) the correct resource is

[https://GWaddress.com:443/\\*](https://GWaddress.com:443/*)

6. Click **Save and Continue**
7. In the Roles dialog, add every role that should have access to the Plant SCADA Access Anywhere link and click **Save Changes**.
8. When you log into Juniper - the Plant SCADA Access Anywhere link will be displayed under the Web bookmarks section (i.e. Plant SCADA Access Anywhere Connection to RDP Host). Simply click on the link to connect to an application or desktop published with Plant SCADA Access Anywhere.

**Note:** If the link is not translating properly, check that there is not a Passthrough Proxy policy defined for the Gateway Server (where the web component is hosted).

## Network Connect

Juniper's Network Connect mode opens a VPN tunnel to the private network. When using Network Connect, simply enter the Plant SCADA Access Anywhere parameters as if they were on the private network.

## Access Anywhere Secure Gateway Installation Guide

AVEVA Plant SCADA Access Anywhere can be used to access AVEVA Plant SCADA clients hosted on Terminal Servers with HTML5 compatible web browsers. Plant SCADA Access Anywhere connects to Plant SCADA remotely via the Plant SCADA Access Anywhere Secure Gateway. This guide provides information about installing and configuring the Secure Gateway.

This guide assumes knowledge of the following:

- AVEVA Plant SCADA
- Enabling and configuring RDP on Windows operating systems
- Firewall configuration
- Web server administration

Terminology used in this guide includes the following:

- **DMZ** (demilitarized zone) - physical or logical subnetwork that contains and exposes an organization's external services to a larger untrusted network.
- **HTML5** - a new update to the HTML specification. Extends HTML with new features and functionality for communication, display, etc.
- **RDP** - Remote Desktop Protocol. A remote desktop protocol developed by Microsoft. RDP is a standard component of Microsoft Windows.
- **RDP Host** - a Windows system that can be remotely accessed using Microsoft RDP, such as a Terminal Server (RDS Session Host) with remote access enabled.
- **RDS** - Remote Desktop Services is one of the components of Microsoft Windows that allow a user to take control of a remote computer or virtual machine over a network connection.
- **SSL** - Secure Sockets Layer. A cryptographic protocol that provides protected communications over the Internet.
- **VPN** - Virtual Private Network. It enables a computer to securely send and receive data across shared or

public networks as if it were directly connected to the private network.

- **WebSocket** - a bi-directional, full-duplex communication mechanism introduced in the HTML5 specification.

## Introduction

Plant SCADA Access Anywhere Secure Gateway is a complementary component of Plant SCADA Access Anywhere that provides encrypted remote access to Plant SCADA clients.

Secure Gateway provides the following benefits.

- Accesses Plant SCADA clients running on an internal network using a single port.
- Provides access without the need to purchase, install, configure, and manage a VPN.
- Located in a perimeter network, also known as a DMZ, while other resources reside behind an internal firewall.
- Installs a single SSL digital certificate on the Secure Gateway node instead of requiring a certificate for every host that needs to be accessed.
- Compatible with HTML5 client browsers supported by Plant SCADA Access Anywhere.

## Architecture

Secure Gateway acts as a gateway between users in remote locations and Plant SCADA clients running in a control network. The server component can be installed in a DMZ to route traffic between an external business network and an internal HMI SCADA network.

---

**Important:** Using the Secure Gateway to connect to your SCADA system from an external network may expose your SCADA system to unauthorized access. It is recommended that you use the Secure Gateway in conjunction with other measures to improve the overall protection for your system.

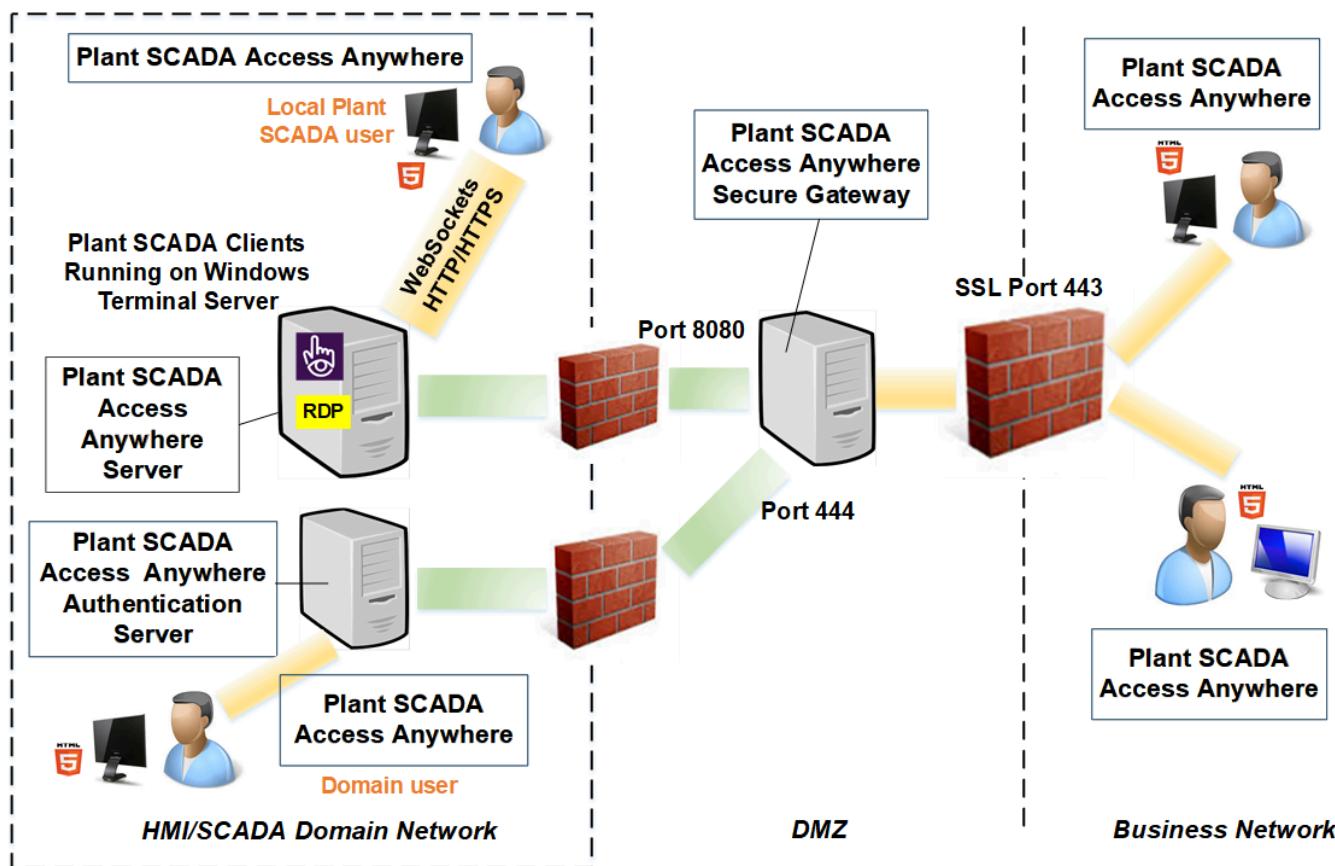
---

The Access Anywhere Secure Gateway comprises of two installed components:

- Secure Gateway Server — provides encrypted access to the Plant SCADA Access Anywhere Server
- Authentication Server — performs authentication services for Plant SCADA Access Anywhere.

These components can be installed on separate computers if required. If your Secure Gateway Server is going to be exposed to computers outside your SCADA domain network, it is recommended that the Authentication Server be installed separately within the SCADA domain network.

The following diagram illustrates how a Secure Gateway Server provides remote access to Plant SCADA clients.



Web traffic from an external business network is tunneled through an SSL-based Secure Gateway connection. User authentication occurs on the Authentication Server within the operations network.

## Installation

This chapter describes how to install the Access Anywhere Secure Gateway. It describes installation prerequisites, a step-by-step installation procedure, configuration details, and instructions to uninstall a Secure Gateway.

---

**Note:** Plant SCADA Access Anywhere supports silent installation of the Secure Gateway components using a response file. See the topic [Unattended Silent Installation](#) in the *Access Anywhere Installation and Configuration Guide*.

---

## Installation Prerequisites

The computer hosting the Secure Gateway needs to meet the following prerequisites.

- The Secure Gateway needs to be installed on a computer running one of the following Microsoft® operating systems:
  - Windows Server 2012 R2
  - Windows Server 2016
  - Windows Server 2019

- Windows 8.1
- Windows 10 (check the Microsoft website for currently supported versions).
- The following ports need to be configured on the computer where the Secure Gateway will be installed:
  - Port 443 is required between an external network and the Secure Gateway server. This is a common port that is also used by Microsoft Internet Information Services (IIS). Check for port conflicts. The port can be changed.
  - Port 8080 is required between the Secure Gateway Server and the Plant SCADA Access Anywhere Server. The port can be changed.
  - The Secure Gateway includes an HTTP proxy that listens on port 80 by default. The port can be disabled after installing the Secure Gateway.

---

**Note:** If you are running Plant SCADA Access Anywhere on an IPv6 network, it is recommended you configure a hostname that refers to the Plant SCADA Access Anywhere Server. A hostname ensures the server is compatible with all supported browsers. If you attempt to connect to the Plant SCADA Access Anywhere Server using a standard format IPv6 address in Internet Explorer, the connection will not be successful.

---

## Installing the Secure Gateway

The Plant SCADA Access Anywhere installer allows you to install the Plant SCADA Access Anywhere Secure Gateway and the Plant SCADA Access Anywhere Authentication Server. These are both required to setup a Secure Gateway, however they can be installed on separate computers.

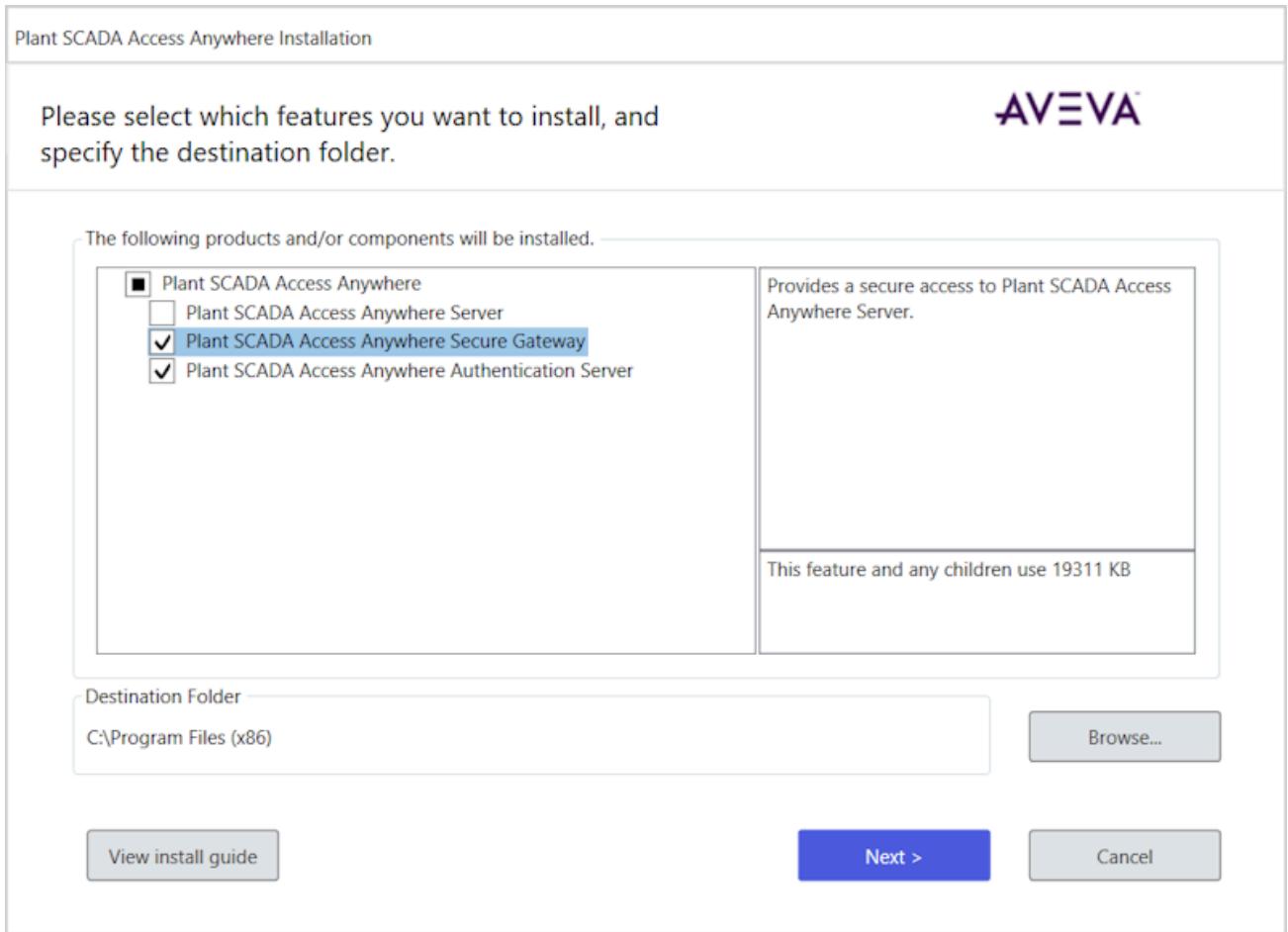
---

**Note:** Before you install Plant SCADA Access Anywhere, you need to uninstall any earlier versions. This means the existing Secure Gateway configuration will no longer be available. If you want to migrate your existing settings to the new version, see the topic [Migrate an Existing Configuration from an Earlier Version](#) before you upgrade.

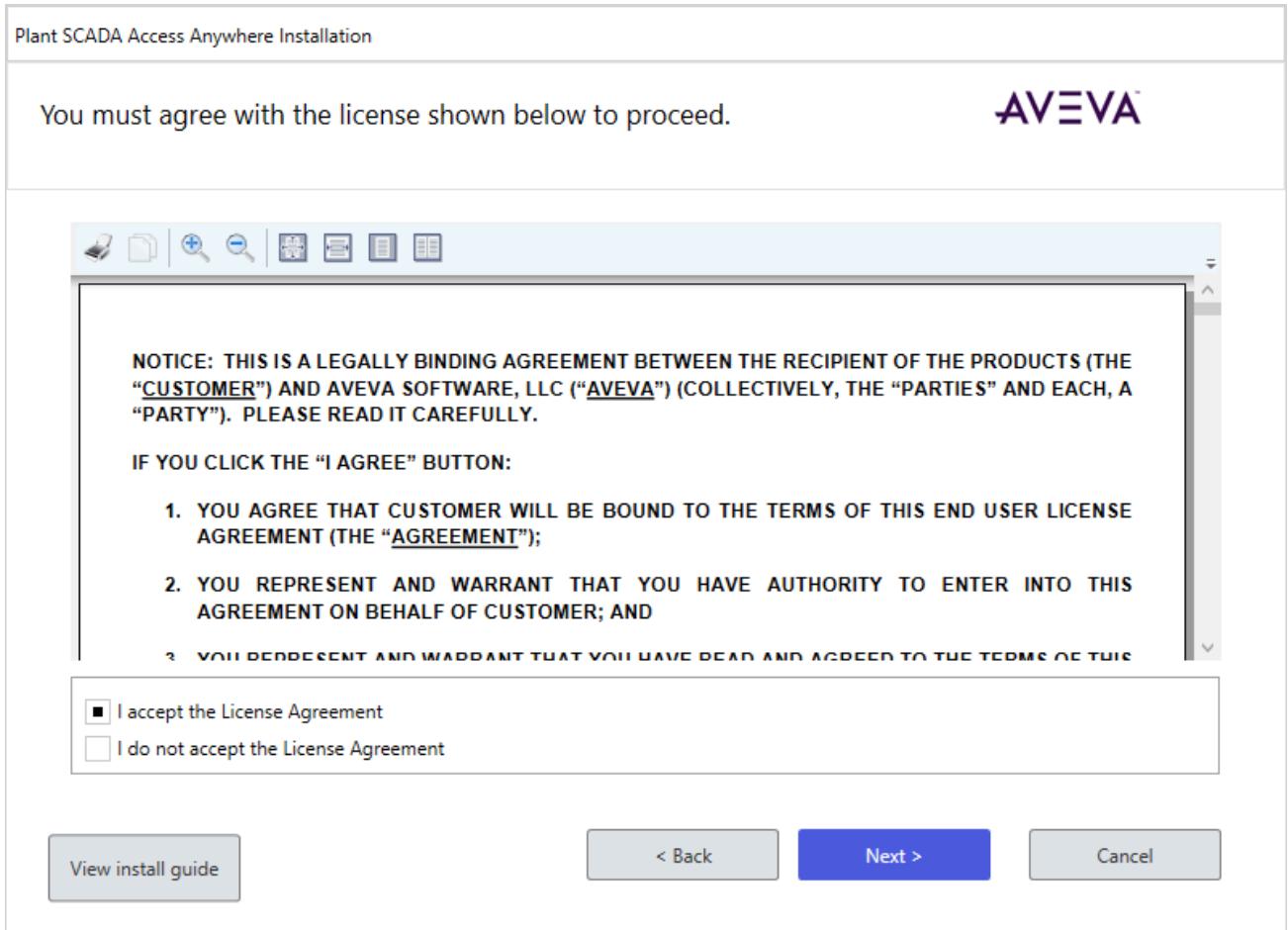
---

### To install the Plant SCADA Access Anywhere Secure Gateway components:

1. Run the *Setup.exe* file.  
The message "*Please wait for configuration to complete*" appears as the installer prepares to start the installation.
2. Select the Plant SCADA Access Anywhere **Secure Gateway** option, and/or the Plant SCADA Access Anywhere **Authentication Server** option.



3. Click **Next**. The License Agreement screen appears. Read the license agreement carefully, and select the **I accept the License Agreement** option to continue. Selecting **I do not accept the License Agreement** will stop you from proceeding with the installation.



4. Click **Next**. The Installation Prerequisites screen appears.  
The installation program checks whether the installation prerequisites are met, and shows the missing prerequisites.

Plant SCADA Access Anywhere Installation

All prerequisites must be met for the installation to continue. Double-click an item for more details.

Show All Prerequisites

Prerequisites	Features	Status	Comment
Microsoft .NET Framework...	Plant SCADA Access Anywhere	met	Product installation requires Microsoft .NET Framework
Operating System	Plant SCADA Access Anywhere...	met	Product installation requires Windows Server 2008 R

< >

[Install Prerequisites](#) [View install guide](#) [Back](#) [Next >](#) [Cancel](#)

To view a list of every prerequisite, select the **Show All Prerequisites** option.

5. Click **Install Prerequisites** to install the missing prerequisites.
6. Click **Next**. The component selected for installation is displayed.

## Plant SCADA Access Anywhere Installation

Ready to Install the Application



Installation is now prepared to complete installation of the product.  
Select **Install** to begin the installation.

Plant SCADA Access Anywhere

Plant SCADA Access Anywhere Secure Gateway

Plant SCADA Access Anywhere Authentication Server

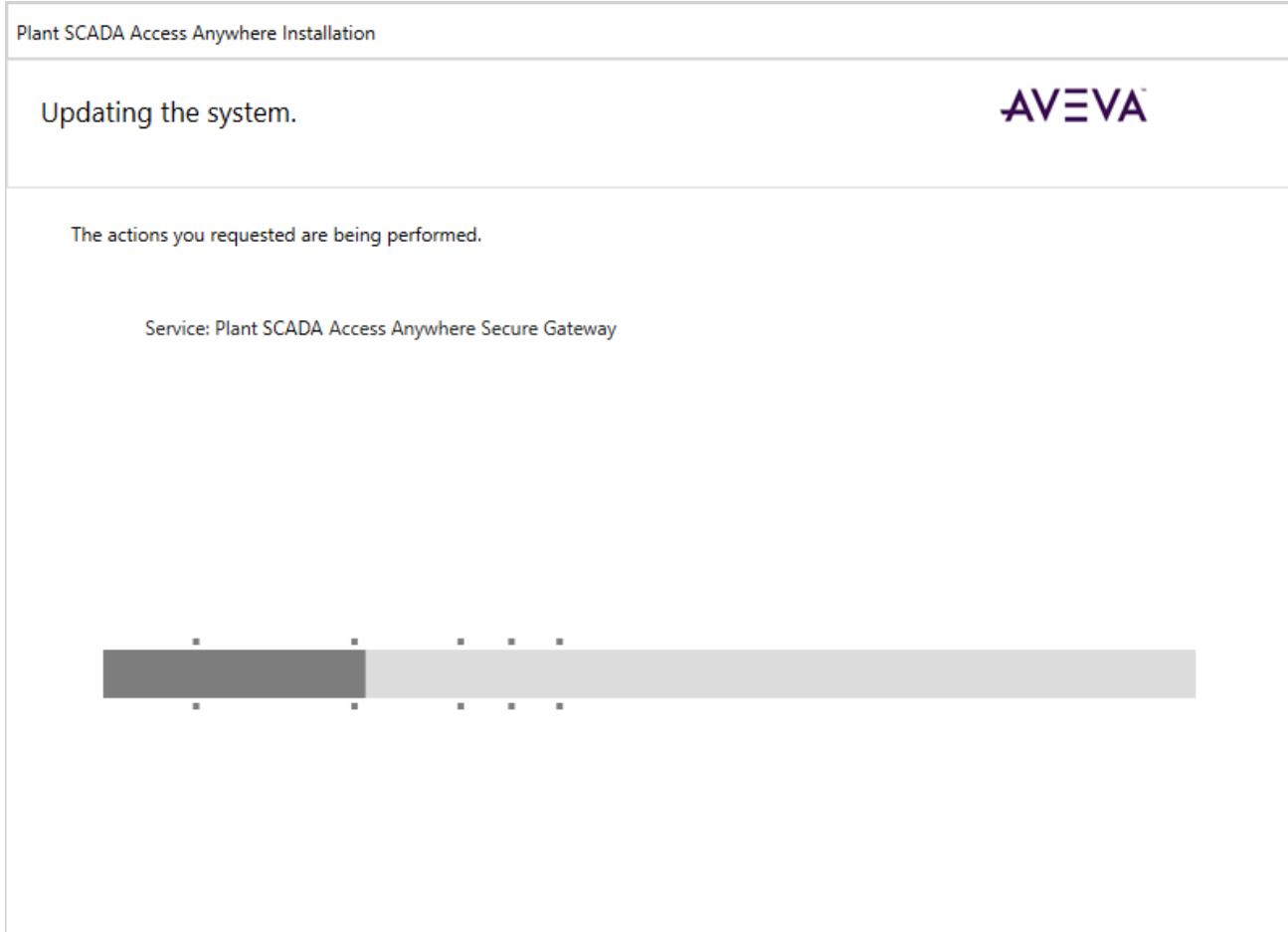
[View install guide](#)

< [Back](#)

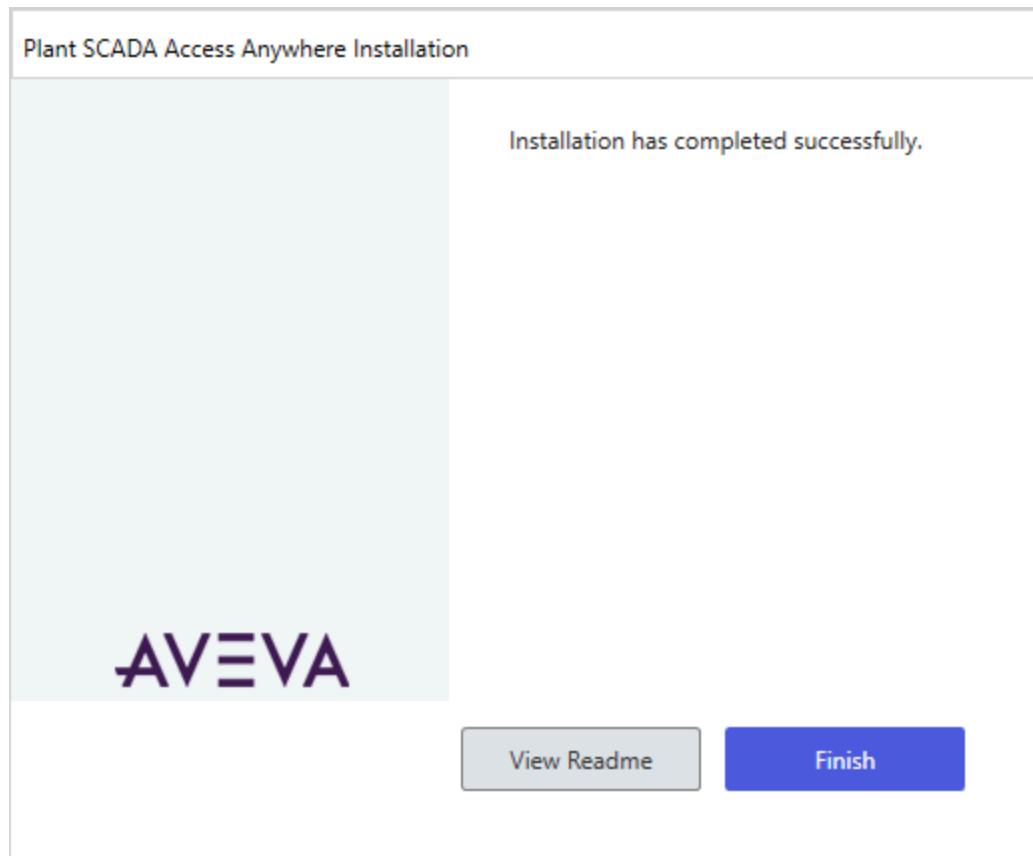
[Install](#)

[Cancel](#)

7. Click **Install**. The progress bar appears.

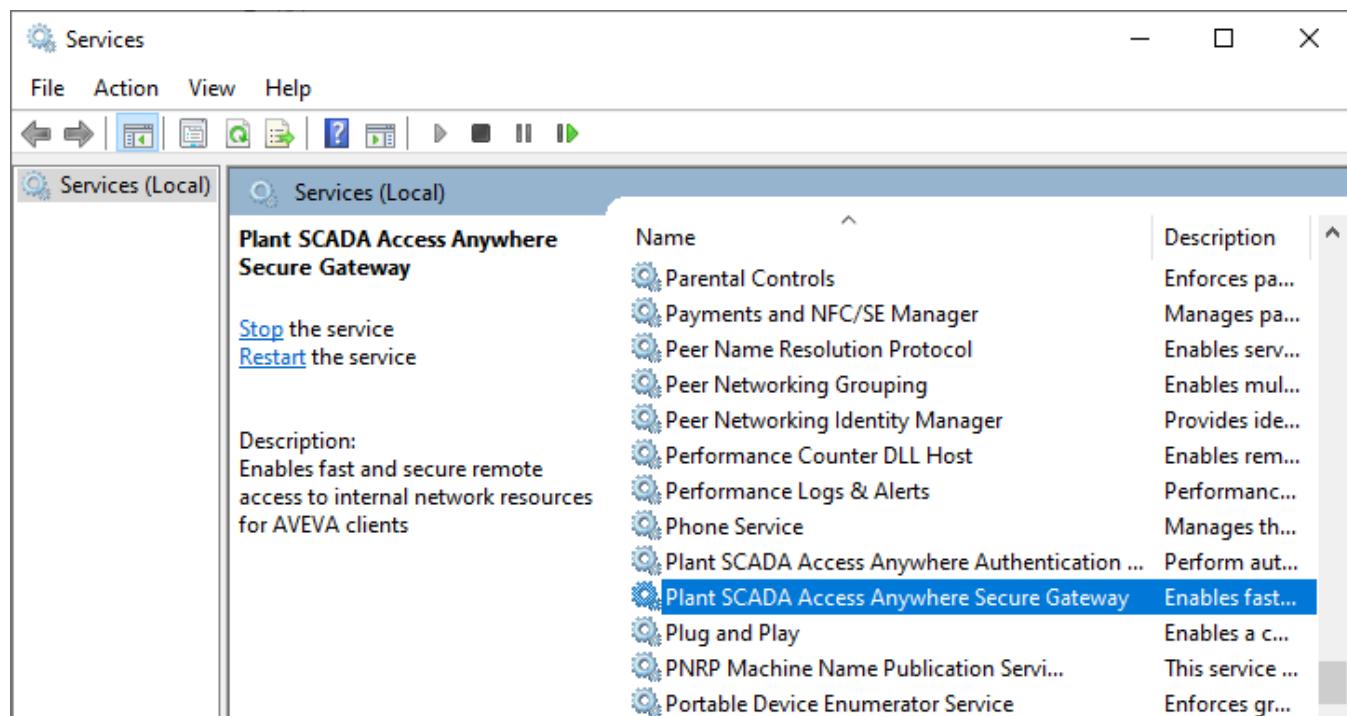


8. After the installation is complete, the Installation Complete dialog box appears. Click **View Readme** to open the *Readme.html* file, which contains details about Plant SCADA Access Anywhere features, or click **Finish**.



## Completing the Installation

The Secure Gateway runs as a service and can be stopped and restarted from the Microsoft Windows Services Manager:



The Secure Gateway service is configured to run automatically on computer startup. If the service is stopped or is unable to listen on its configured port, clients will be unable to connect to hosts through the gateway. An error message will be written into the Windows application event log.

**Important:** If Microsoft IIS or Plant SCADA are running on the same server as the Secure Gateway, check that there are no port conflicts. Change the IIS ports and/or the port on Plant SCADA's System Management Server to values other than 80 and 443. Or change the Secure Gateway port to a value other than 443 and disable the HTTP auto redirect feature after the installation. If there is a port conflict on either the HTTP or HTTPS port, the Secure Gateway cannot operate properly. For more information, see the sections [Built-In Web Server](#) on page and [HTTP Redirect](#) on page .

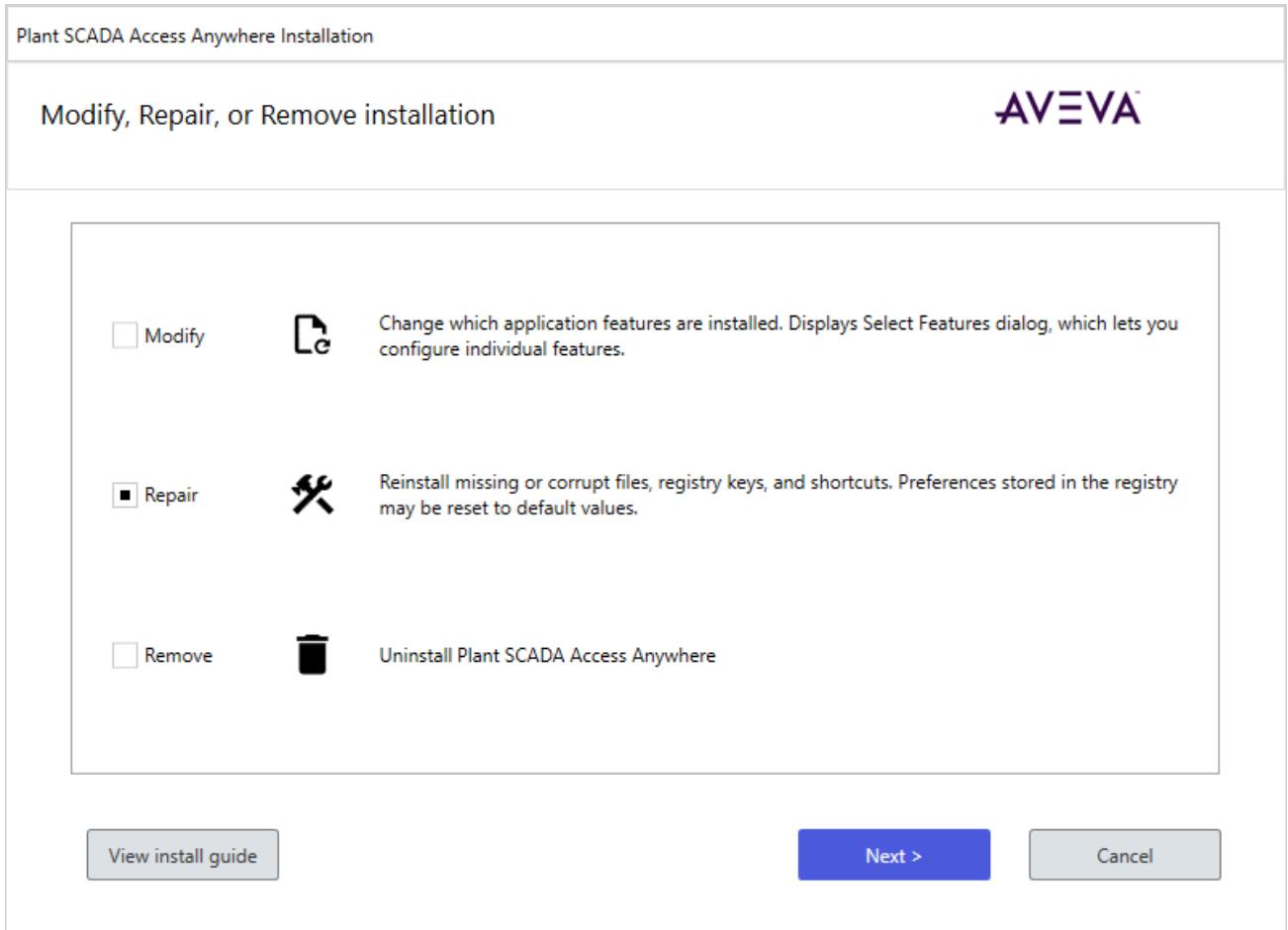
## Repairing an Installation

You can use the installation program to repair corrupt files of installed Plant SCADA Access Anywhere components.

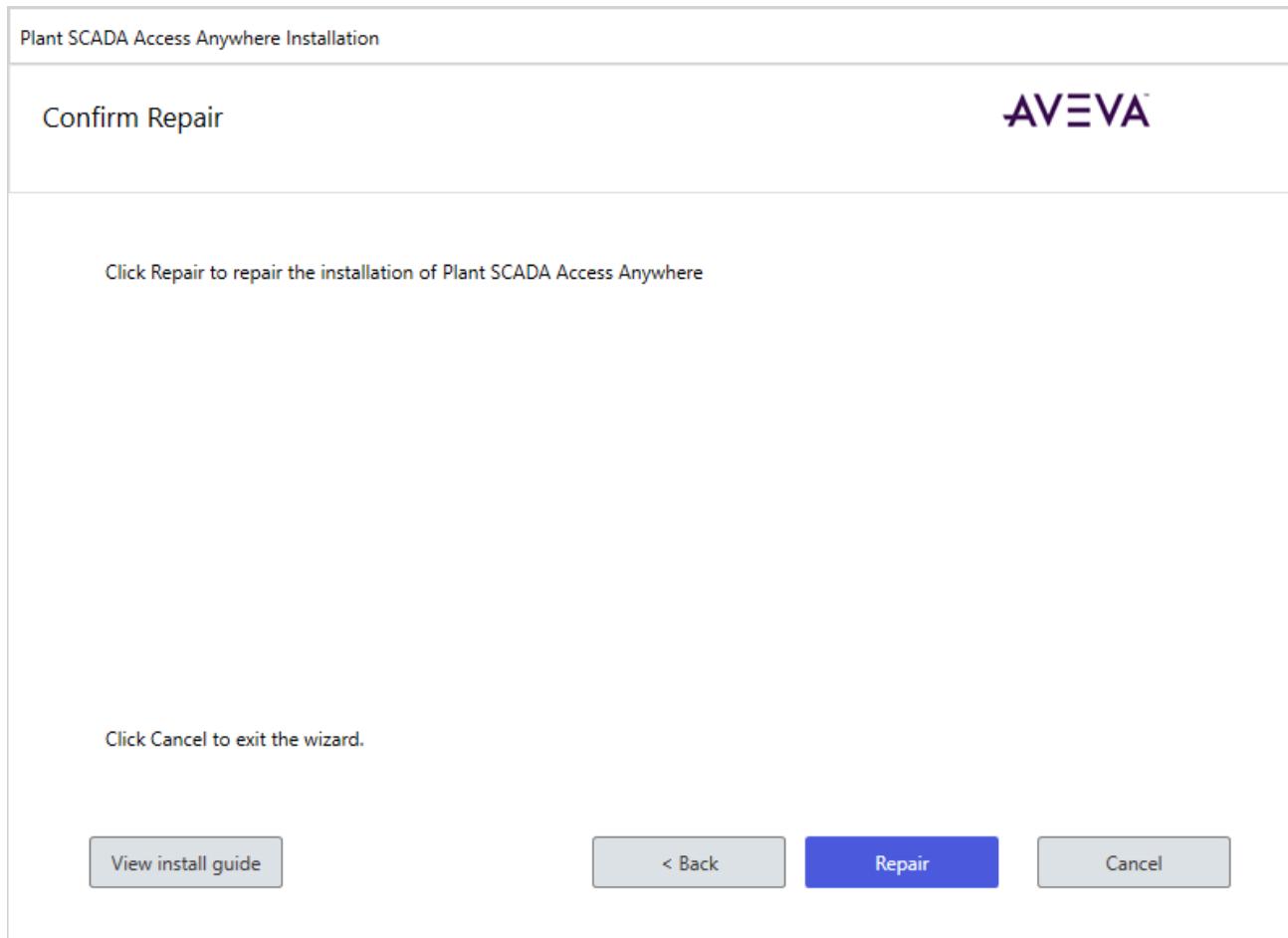
**Note:** You need to have access to the installation files before you can repair an installation.

To repair an installation:

1. From the **Windows Start** menu, select **Control Panel | Programs | Programs and Features | Uninstall a Program**. Every program installed on your computer is listed.
2. Select the Plant SCADA Access Anywhere component you wish to repair. Click **Uninstall/Change**. The installer program appears.
3. Click **Repair**, and then click **Next**.



4. Click **Repair**.



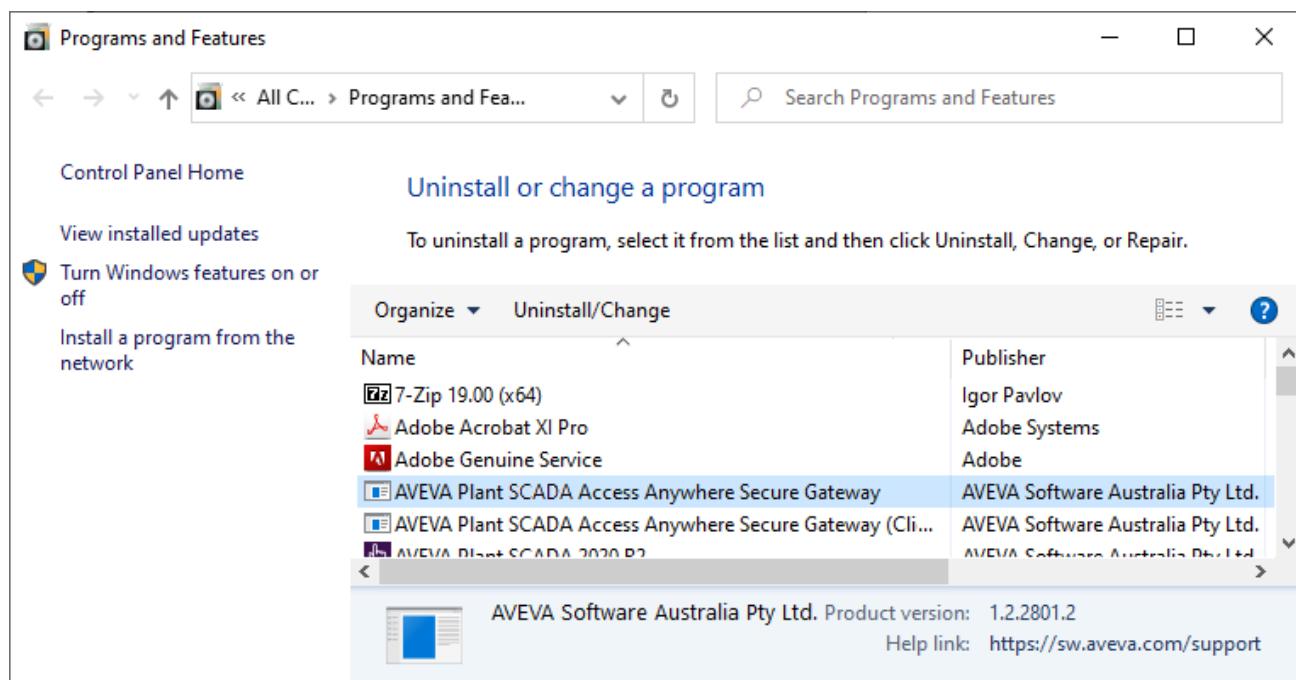
5. Installed Plant SCADA Access Anywhere components are repaired, and a message confirming successful repair appears.
6. Click **Finish** to close the dialog box.

## Uninstalling the Secure Gateway

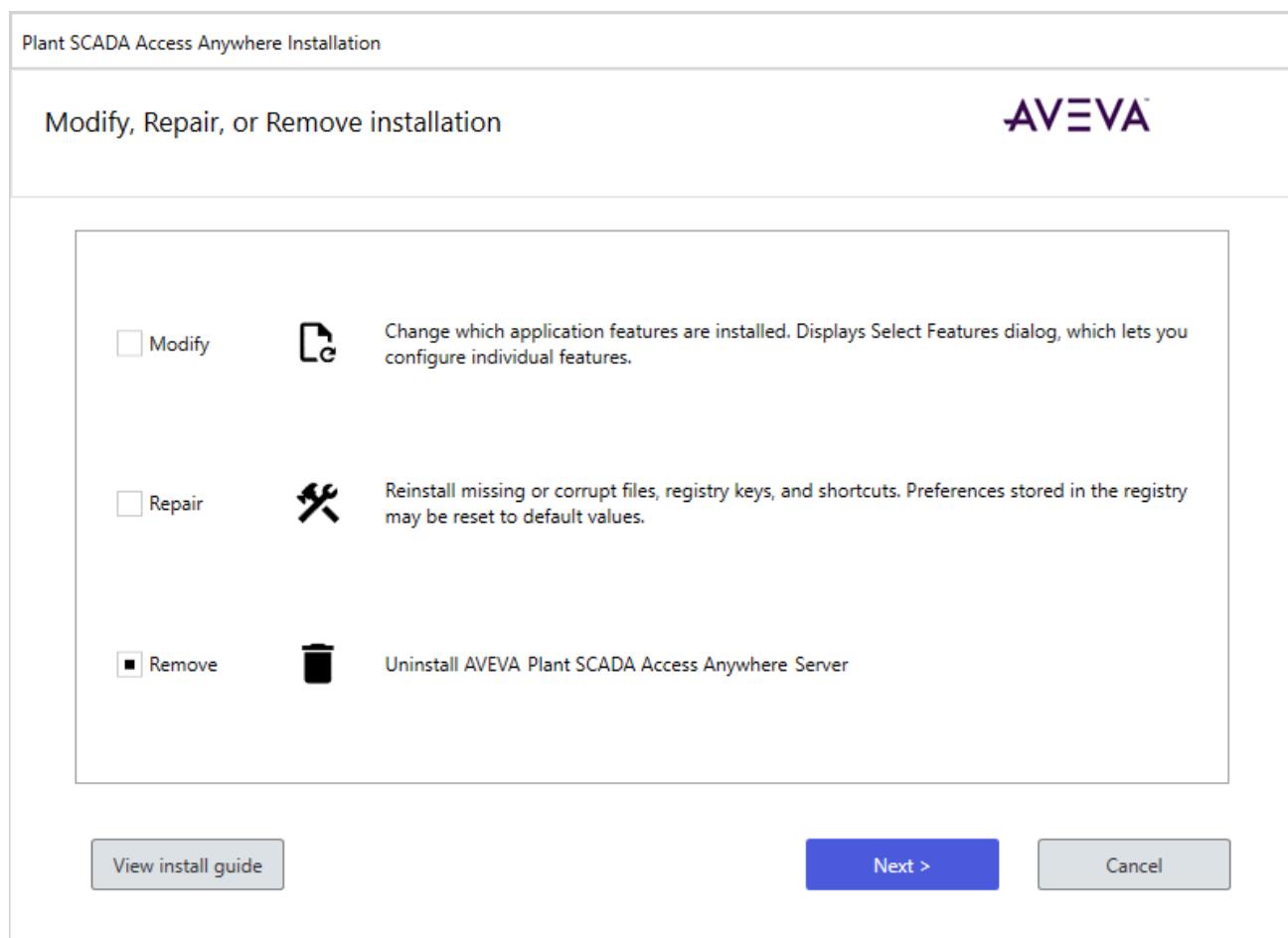
You may wish to remove Plant SCADA Access Anywhere components. This section provides instructions for uninstalling Plant SCADA Access Anywhere components.

### To uninstall Plant SCADA Access Anywhere components:

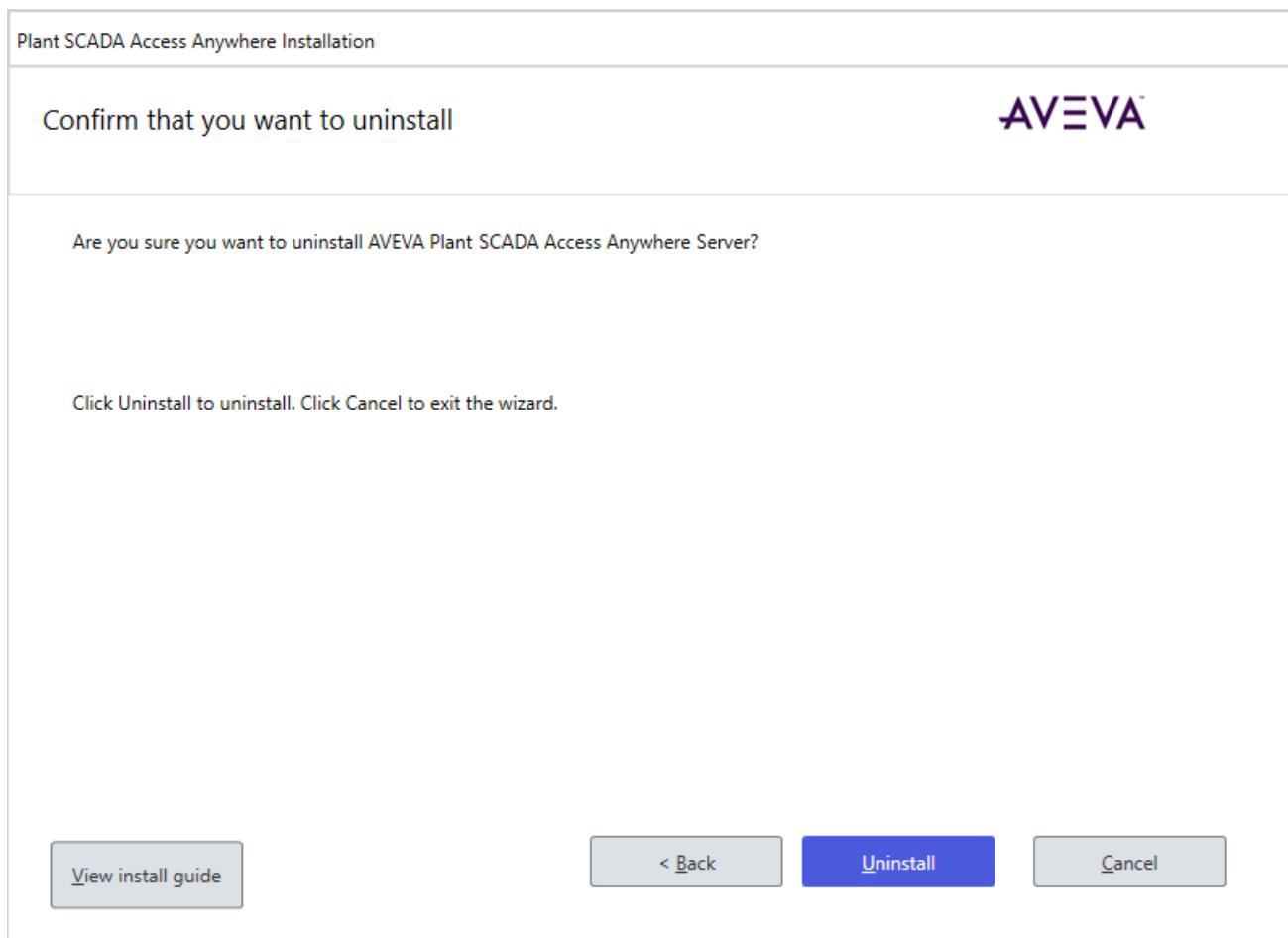
1. From the Windows Start menu, select **Control Panel | Programs | Programs and Features | Uninstall a Program**. All programs installed on your computer are listed.
2. Locate the Plant SCADA Access Anywhere Secure Gateway component to uninstall, and click to select it.



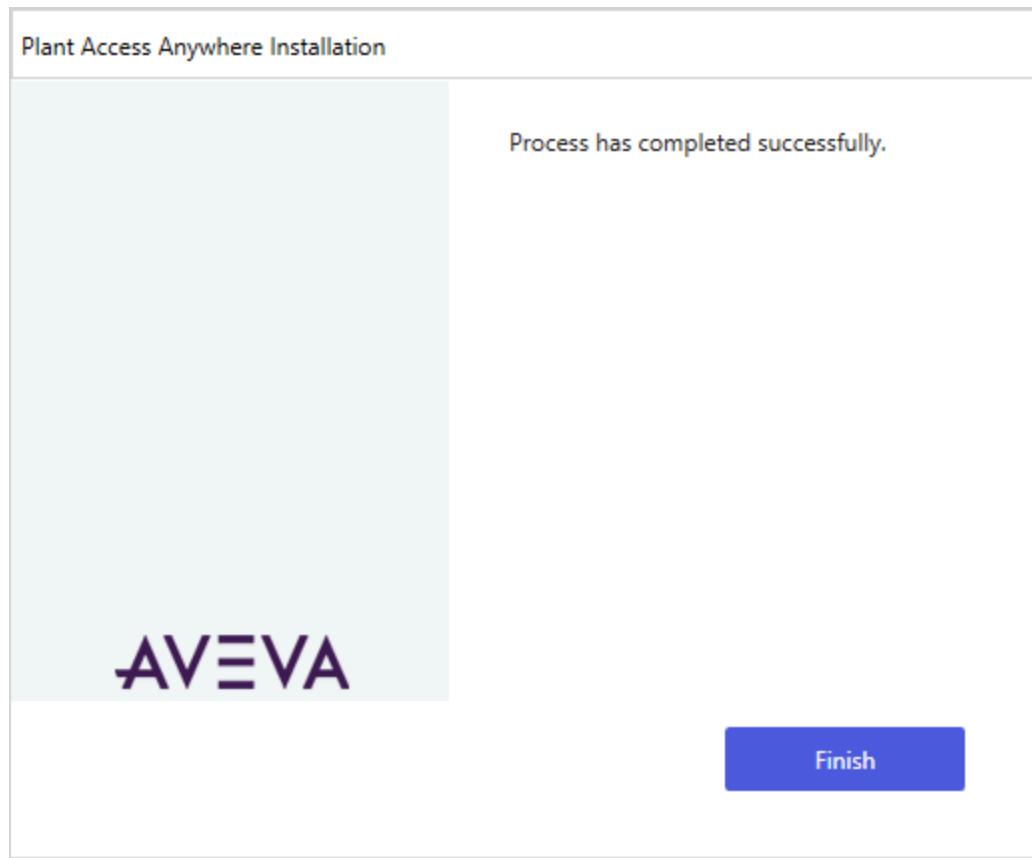
3. Click **Uninstall/Change**. The Modify, Repair or Remove Installation dialog box appears.
4. Select the **Remove** option.



5. Click **Next**. A message prompting you to confirm the uninstallation appears.



6. Click **Uninstall** and then click **Next**. A screen confirming successful uninstallation appears.



**Note:** This will remove only the selected component. To remove every component installed with the Plant SCADA Access Anywhere Secure Gateway (the Plant SCADA Access Anywhere Secure Gateway Client components), run *setup.exe* again and perform a complete uninstall. Some files may still exist in the installation folders after the uninstallation.

## Migrate an Existing Configuration from an Earlier Version

When you install a new version of Plant SCADA Access Anywhere, the existing Secure Gateway configuration file is no longer available. If you want to migrate your existing settings to the new version, you need to manually merge some information into the newly installed configuration files.

### To migrate an existing Secure Gateway configuration:

1. Make a backup of the configuration files.

For the Secure Gateway, this will include the following files:

- \Program Files (x86)\AVEVA Plant SCADA Access Anywhere\Plant SCADA Access Anywhere Authentication Server\EricomAuthenticationServer.exe.config
- \Program Files (x86)\AVEVA Plant SCADA Access Anywhere\Plant SCADA Access Anywhere Authentication Server\EricomAuthenticationServer\_Tenants.xml
- \Program Files (x86)\AVEVA Plant SCADA Access Anywhere\Plant SCADA Access Anywhere Secure Gateway\WebServer\PlantAccessAnywhere\config.js
- \Program Files (x86)\AVEVA Plant SCADA Access Anywhere\Plant SCADA Access Anywhere Secure Gateway\EricomSecureGateway.exe.config

If you also want to migrate an existing Plant SCADA Access Anywhere Server, you should also back up the following files before performing an upgrade:

- \Program Files (x86)\AVEVA Plant SCADA Access Anywhere\Launcher\SE.Scada.AnywhereLauncher.exe.config
- \Program Files (x86)\AVEVA Plant SCADA Access Anywhere\Server\WebServer\PlantAccessAnywhere\config.js

**Note:** If you have used a custom certificate for the Plant SCADA Access Anywhere Server, you will need to export a backup of the server configuration settings as a registry file as certificate information is not included in the configuration files listed above. You can view the current SSL certificate details on the **Security** tab of the Access Anywhere Server Configuration console. You can also export the configuration settings as a registry file (.reg) using the console's **Advanced** tab.

The process for migrating the Plant SCADA Access Anywhere Server is explained in the *Plant SCADA Access Anywhere Installation and Configuration Guide*.

2. Install the new version of the software, then return to step 3.
3. Manually merge the settings from the files listed below into the new version of each file.
4. Restart the affected services.

**Note:** With the release of Plant SCADA Access Anywhere 2020 R2, the default installation location changed to '\Program Files (x86)\AVEVA Plant SCADA Access Anywhere'.

- **EricomAuthenticationServer.exe.config**

Merge the following sections into the new configuration file, if applicable.

- <appSettings /> - except the key 'CertificateThumbprint' if a self-signed certificate is used.
- <logSettings />
- <translatorSettings />

- **EricomAuthenticationServer\_Tenants.xml**

Merge every section across to the new xml file, if applicable.

- **PlantAccessAnywhere\config.js**

Merge every section across to the new xml file, if applicable.

- **EricomSecureGateway.exe.config**

For the EricomSecureGateway.exe.config file, the file format has been changed. The relevant sections have been moved to a file called **EricomSecureGateway.config** (without the '.exe' extension).

To migrate applicable settings, merge the following sections to the new EricomSecureGateway.config file based on the mappings in the table below. For example, if you had specified a **Session Inactivity Timeout** of 15 minutes in the Secure Gateway **Administration** settings, it will appear in the version 1.1 "EricomSecureGateway.exe.config" file as follows:

```
<configuration>
  ...
  <sessionsSettings>
    <Admin>
      <add key="InactivityTimeoutMinutes" value="15" />
    ...
  </Admin>
  ...
</configuration>
```

To migrate the setting, copy the value of the "InactivityTimeoutMinutes" key to the following section in the new "EricomSecureGateway.Config" file:

```
<PropertySet name="ESG Configuration Parameters" version="1.0">
<Section name="">
...
<Section name="Admin">
<Property name="InactivityTimeoutMinutes" type="int" value="15" />
...
</Section>
...
</Section>
</PropertySet>
```

V 1.1 EricomSecureGateway.exe.config	V2.0 EricomSecureGateway.config
configuration.externalServersSettings.AuthenticationServer	PropertySet["ESG Configuration Parameters"].Section[""].Section["AuthenticationServer"]
configuration.externalServersSettings.WebConnectServer	PropertySet["ESG Configuration Parameters"].Section[""].Section["WebConnectServer"]
configuration.externalServersSettings.VMwareViewServer	PropertySet["ESG Configuration Parameters"].Section[""].Section["VMwareViewServer"]
configuration.sessionsSettings.Visitor	PropertySet["ESG Configuration Parameters"].Section[""].Section["Visitor"]
configuration.sessionsSettings.WebServer	PropertySet["ESG Configuration Parameters"].Section[""].Section["Sessions"].Section["InternalWebServer"]
configuration.sessionsSettings.Admin	PropertySet["ESG Configuration Parameters"].Section[""].Section["Admin"]
configuration.sessionsSettings.Gateway	PropertySet["ESG Configuration Parameters"].Section[""].Section["Sessions"].Section["Gateway"]
configuration.internalWebServerSettings.General	PropertySet["ESG Configuration Parameters"].Section[""].Section["InternalWebServer"]
configuration.internalWebServerSettings.Folders	PropertySet["ESG Configuration Parameters"].Section[""].Section["InternalWebServer"].Property["FolderList"]
configuration.mailAlertsSettings.SmtpServer	PropertySet["ESG Configuration Parameters"].Section[""].Section["MailAlert"].Section["SmtpServer"]

V 1.1 EricomSecureGateway.exe.config	V2.0 EricomSecureGateway.config
configuration.mailAlertsSettings.Mail	PropertySet["ESG Configuration Parameters"].Section[""].Section["MailAlert"].Section["Mail"]
configuration.mailAlertsSettings.Alerts	PropertySet["ESG Configuration Parameters"].Section[""].Section["MailAlert"].Section["Alerts"]
configuration.logSettings.Roles	PropertySet["ESG Configuration Parameters"].Section[""].Section["Log"].Section["Level"].Section["Role"]
configuration.logSettings.HttpDataWatch	PropertySet["ESG Configuration Parameters"].Section[""].Section["Log"].Section["Level"].Section["HttpDataWatch"]
configuration.logSettings.SessionRecording	PropertySet["ESG Configuration Parameters"].Section[""].Section["SessionRecording"]
configuration.advancedSettings	PropertySet["ESG Configuration Parameters"].Section[""].Section["Advanced"]
configuration.appSettings	PropertySet["ESG Configuration Parameters"].Section[""].Section["Network"]

## Post-Installation

Before proceeding further, the Secure Gateway needs to be manually configured to point to your Plant SCADA Access Anywhere Server. To do this:

1. Edit the *config.js* file located on the computer where the Secure Gateway is installed. This file is located in the following folder.

**\Program Files (x86)\AVEVA Plant SCADA Access Anywhere\Plant SCADA Access Anywhere Secure Gateway\WebServer\PlantAccessAnywhere\**

2. Find the commented line:

```
// address:"", // Address of server
```

3. Uncomment the line by removing the initial "//".

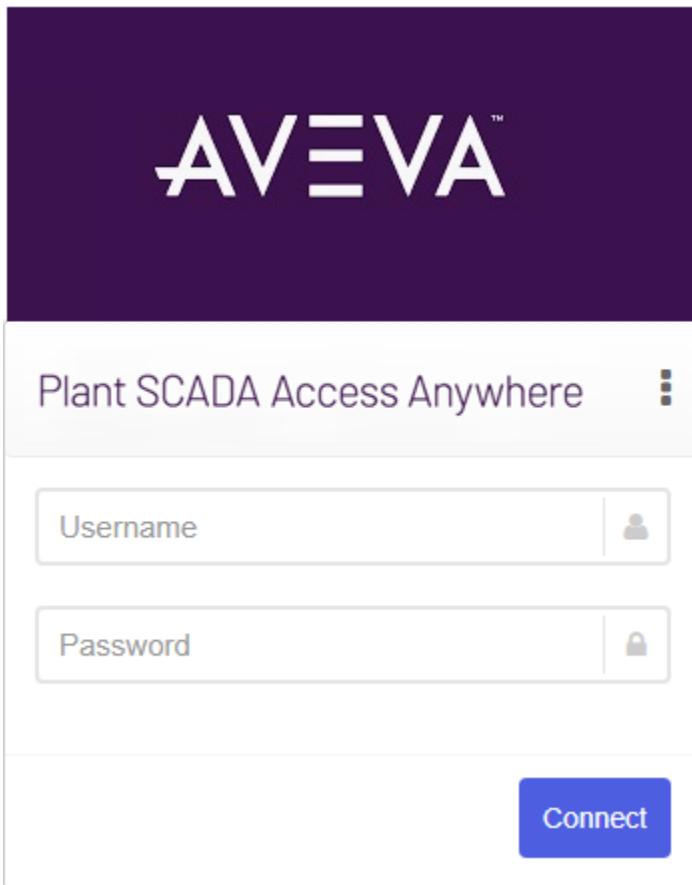
4. Within the quotation marks, type the name or IP address of the computer on which the Plant SCADA Access Anywhere Server is installed.

For example:

```
address: "VD-CA-SRV", // Address of server
```

Access to the Plant SCADA Access Anywhere Server is routed through the Secure Gateway node. When you navigate to the *https://<Name or IP Address of the computer where the Secure Gateway is installed>/*, the

following page appears:



This login dialog will function differently depending on the value of the 'Authentication Server' settings, discussed in more detail later.

In brief, if the Secure Gateway is configured to use an Authentication Server, it will authenticate users before passing the credentials on to the Plant SCADA Access Anywhere Server.

If the Secure Gateway is not configured to use an Authentication Server, the credentials will be passed directly to the Plant SCADA Access Anywhere Server for authentication.

Although subtle, these settings may make a big difference if the Secure Gateway and Plant SCADA Access Anywhere Servers are located in different domains, or have other differences in their authentication capabilities.

When logging in, you will need to prefix your username with your domain name, or, if there is a common domain name, you can enter it in the config.js file, under the 'Domain' entry, for example:

DOMAIN: "domain.com", // Optional domain for RDP Session...

The Administration page for the Authentication Server, showing the current status, and the domain name that it is authorizing against, can be viewed at <https://localhost:7443/Admin> with a local administrator login.

---

**Note:** A page refresh and clearing cookies on every client are required each time the **Address** property in the config.js is changed.

---

## Configuring the Secure Gateway

The Plant SCADA Access Anywhere Secure Gateway can be configured via the included Configuration Portal, or by manually editing the *EricomSecureGateway.exe.config* file. Both of these methods are detailed below.

## Configuration Portal

To access the Configuration Portal page, go to Windows **Start** menu | AVEVA Plant SCADA Access Anywhere | **Administration Portal**.

Alternatively, use a web browser and navigate to the Secure Gateway's configuration URL:

<https://localhost/admin/login.html>

Logging on to the Configuration Portal is accessible only to members of the local Administrators group of the Plant SCADA Access Anywhere Secure Gateway server. Logons are audited in the Secure Gateway log file.

Administrators are strongly encouraged to enforce a strong password policy for Secure Gateway administrators.

Plant SCADA Access Anywhere

Secure Gateway

### Configuration Portal

User name

Password

Login

To log out of the Configuration Portal, click **Logout**.

The screenshot shows the Configuration Portal interface. On the left is a sidebar with navigation links: Administrator (selected), Logout (highlighted with a red box), Gateway Settings, Dashboard (selected), Security, Web Server, Authentication Server, Log Settings, Download, Admin, and About. The main content area is titled "Server Information". It contains two sections: "Server Status" and "Server Activity".

**Server Status**

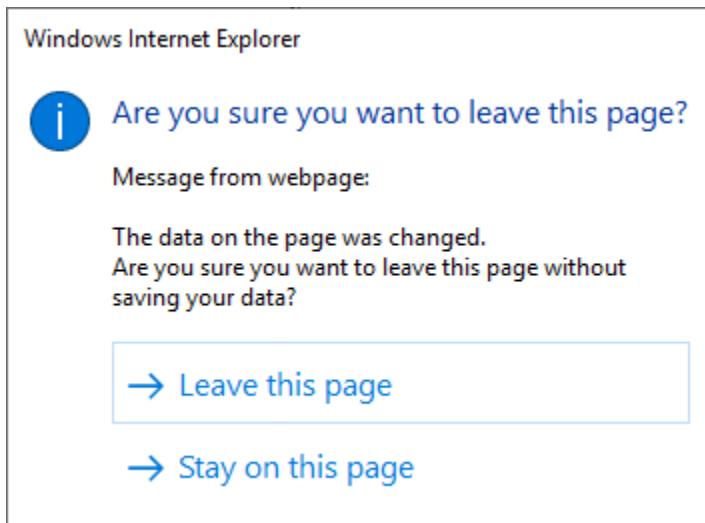
Start Time	Tuesday, 16 November 2021 9:29:48 PM
Up Time	0 Days, 15 Hours, 17 Minutes, 9 Seconds
SSL Certificate	NOT Trusted. Has a private key.

**Server Activity**

**Total Sessions**

	Current	Peak
Connections under validation	0	5
Gateway Sessions	0	1
Web Server Connections	1	5
Admins	1	1

After making changes to any settings, click **Save**. If a different page is selected and the settings are not saved, a warning dialog will appear. Click **Leave this Page** to continue and cancel any changes. Click **Stay** on this page to return to the current page to save changes.



## Dashboard

Secure Gateway **Configuration Dashboard** displays useful statistics related to the Secure Gateway operation. Open this page to view server uptime, SSL certificate status, session activity, and to restart the Secure Gateway Server service.

- Administrator
- [Logout](#)
- [Gateway Settings](#)
- [Dashboard](#)
- Security
- Web Server
- Authentication Server
- Log Settings
- Download
- Admin
- About

## Server Information

### Server Status

Start Time	Tuesday, 16 November 2021 9:29:48 PM
Up Time	0 Days, 15 Hours, 9 Minutes, 48 Seconds
SSL Certificate	NOT Trusted. Has a private key.

### Server Activity

#### Total Sessions

	Current	Peak
Connections under validation	0	5
Gateway Sessions	0	1
Web Server Connections	1	5
Admins	1	1

#### Gateway Session Distribution

	Current	Peak
Native Sessions	0	1
HTTPS Sessions	0	0
WebConnect Sessions	0	0

[Restart Server](#) [Refresh](#)

## Security

The Security section shows which **Secured Port** is being used by the Secure Gateway and which **SSL Certificate** is in use. There are also additional SSL configuration options.

For production systems, it is recommended that you use either a trusted certificate purchased from a certificate authority (for example, DigiCert), or a domain-issued trusted certificate provided by your IT administrator. For more information refer to [SSL Certificates](#).

The screenshot shows the 'Security' configuration page. On the left, a sidebar menu includes 'Administrator', 'Logout', 'Gateway Settings' (selected), 'Dashboard', 'Security' (selected), 'Web Server', 'Authentication Server', 'Log Settings', 'Download', 'Admin', and 'About'. The main content area has a heading 'Security' and a note: 'By forcing SSL policy errors on clients you might lock yourself out of the admin. Use with care!'. It states that fields marked with \* are mandatory. Configuration options include a 'Secured Port' set to 443, an 'SSL Certificate' dropdown set to 'Ericom Secure Gateway self-signed certificate', and several checkboxes for SSL handshake ignore errors. At the bottom are 'Save' and 'Cancel' buttons.

## Web Server

The Secure Gateway has a built-in web server to host web pages for Plant SCADA Access Anywhere. The built-in Web server cannot be disabled and listens on the Secure Gateway port.

---

**Note:** If you are running Plant SCADA Access Anywhere on an IPv6 network, it is recommended you configure a hostname that refers to the Access Anywhere Server. A hostname ensures the server is compatible with all supported browsers. If you attempt to connect to the Access Anywhere Server using a standard format IPv6 address in Internet Explorer, the connection will not be successful.

---

Refer to the chapter [Built-In Web Server](#) for more information.

## Log Settings

In this section, the Log Settings can be configured. Consult with a support engineer on which settings to enable.

The screenshot shows the 'Basic Log Settings' configuration page. On the left, a sidebar menu includes 'Administrator', 'Logout', 'Gateway Settings' (selected), 'Dashboard', 'Security', 'Web Server', 'Authentication Server', 'Log Settings' (selected), 'Download', 'Admin', and 'About'. The main content area has a title 'Basic Log Settings' and a subtitle 'Change gateway log level to fine-tune the amount of information written to the log per functionality.' It lists five log levels with dropdown menus: Visitor (Info), Session (Info), WebServer (Info), Admin (Info), and Statistics (Info). Below the list are 'Save' and 'Cancel' buttons.

## Built-In Authentication Server

The Secure Gateway includes an Authentication Server that provides a layer of protection by authenticating end-users before they can access any internal resource.

The Authentication Server can be enabled / disabled from the Configuration Portal. The Authentication Server is installed locally and is employed to authenticate users.

---

**Note:** In order to use the Authentication Server, you need to restart your computer after you install the Secure Gateway.

---

**Note:** The Authentication Server can only be configured for one domain at a time.

Use the Configuration page to modify settings for the Authentication Server:

The screenshot shows the 'Authentication Server' configuration page. The sidebar menu is identical to the previous screenshot. The main content area has a title 'Authentication Server' and a subtitle 'When enabled and both address and port are specified, the gateway will use the Authentication Server to authenticate the user before connecting the client to a resource inside the organization.' It shows a checked checkbox 'Enabled', an 'Address' input field containing 'localhost', and a 'Port' input field containing '444'. Below the form are 'Save' and 'Cancel' buttons.

The configuration settings are stored in the file *EricomSecureGateway.exe.config*. The user configurable settings are located under the <ExternalServerSettings><AuthenticationServer> section and defined in the following table.

Setting	Description
Address	The address that the Authentication Server will bind to.
Port	This is the numerical value of the port that the Authentication Server listens over. Check that no other services on the system are using the same port. A port conflict will interfere with the operation of the Authentication Server.

Any configuration of the Authentication Server itself needs to be done through its Web-Portal, which is by default located at <https://localhost:7443>.

The configuration settings are stored in the file *EricomAuthenticationServer.exe.config*. The user configurable settings are located under the <appsettings> section.

## Download - Obtaining Log Files

This is the section from where you can download the current log file. It also contains a link to the Viewer application that is required to read the log files.

The screenshot shows the 'Download' section of the Admin interface. On the left, there's a sidebar with links: Administrator, Logout, Gateway Settings, Dashboard, Security, Web Server, Authentication Server, Log Settings, Download (which is highlighted with a blue background), Admin, and About. The main area has a title 'Download' and three links: 'Plant SCADA Access Anywhere Secure Gateway Log File', 'Plant SCADA Access Anywhere Secure Gateway Log viewer', and 'Ericom Secure Gateway Administration Manual'.

## Admin

This section allows you to set Inactivity Timeout settings, and define a whitelist of allowed client IP addresses.

The screenshot shows the 'Administration Settings' page. On the left, a sidebar menu includes 'Administrator', 'Logout', 'Gateway Settings' (with 'Dashboard', 'Security', 'Web Server', 'Authentication Server', 'Log Settings', 'Download'), 'Admin' (which is selected), and 'About'. The main content area has a heading 'Administration Settings' and a note about enabling Admin lockdown. It includes fields for 'Session Inactivity Timeout (Minutes)' (set to 5), 'Enable Admin Whitelist By IP Address' (checkbox checked), and two input fields for 'Allowed IPv4 Addresses' and 'Allowed IPv6 Addresses'.

Administrator

Logout

Gateway Settings

Dashboard

Security

Web Server

Authentication Server

Log Settings

Download

Admin

About

## Administration Settings

By enabling Admin lockdown and setting the IP addresses below you restrict the ability to login and configure gateway. Use with care!

If you lock yourself out of the admin by failing to add your own IP address, you must edit gateway configuration directly.

Allowed address list items can either be an IP address or an IP range (from-to) of networks allowed to access the gateway Admin, separated by semicolons (';').

Lockdown-related changes will only take effect on the next Admin login operation.

Fields marked with \* are mandatory.

Session Inactivity Timeout (Minutes)\*

Enable Admin Whitelist By IP Address.

Allowed IPv4 Addresses

Allowed IPv6 Addresses

## SSL Certificates

The Plant SCADA Access Anywhere Secure Gateway installation includes a self-signed certificate for SSL connections. It is recommended self-signed certificates are only used for testing purposes within a domain. Self-signed certificates will result in insecure connection warning messages in the user's web browser. They will also attempt to prevent iOS devices from connecting to the Secure Gateway.

For production systems, use either a trusted certificate purchased from a certificate authority (for example, DigiCert), or a domain-issued trusted certificate provided by your IT administrator. When using a domain-issued certificate, the domain trusted root certificates need to be distributed to every device that connects to Plant SCADA Access Anywhere.

**Important:** The signed certificate needs to have a private key associated with it. A .CER file may not have a private key. Use a signed certificate that includes a private key, which usually has a .PFX extension.

**Note:** The DNS address of the Plant SCADA Access Anywhere Server or Secure Gateway server needs to match the certificate name. If a wildcard certificate is being used, the domain needs to match. For example, if the certificate is for \*.example.com the server name needs to end with example.com.

## Configuring the SSL Certificates

To use a trusted certificate on to the Plant SCADA Access Anywhere Secure Gateway Server and the Authentication Server, perform the following procedures.

## Import the Certificate on to the Secure Gateway Server

1. Locate the certificate file provided by your Plant SCADA Access Anywhere or domain administrator.
2. Double click the certificate file, or right-click and select **Import**. The Certificate Import Wizard is displayed.

### Welcome to the Certificate Import Wizard

This wizard helps you copy certificates, certificate trust lists, and certificate revocation lists from your disk to a certificate store.

A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.

Store Location

Current User

Local Machine

To continue, click **Next**.

3. Select **Local Machine** and click **Next**.
4. On the next screen, make sure the path to the certificate is correct.

#### File to Import

Specify the file you want to import.

File name:

D:\Certificates\my-ca-server.pfx

[Browse...](#)

Note: More than one certificate can be stored in a single file in the following formats:

Personal Information Exchange- PKCS #12 (.PFX,.P12)

Cryptographic Message Syntax Standard- PKCS #7 Certificates (.P7B)

Microsoft Serialized Certificate Store (.SST)

5. Click **Next**.
6. On the next screen, enter the password for the certificate's private key.

**Private key protection**

To maintain security, the private key was protected with a password.

Type the password for the private key.

Password:

\*\*\*\*\*|

Display Password

7. Click **Next**.

8. On the next screen, specify the location where the certificate will be stored.

**Certificate Store**

Certificate stores are system areas where certificates are kept.

Windows can automatically select a certificate store, or you can specify a location for the certificate.

Automatically select the certificate store based on the type of certificate

Place all certificates in the following store

Certificate store:

Browse...

9. Click **Next**.

10. On the next screen, click **Finish** to import the certificate.

**Note:** If you are using a separate Plant SCADA Access Anywhere Authorization Server, you will need to repeat these steps to import the certificate on the computer where the Authorization Server is installed.

### Configure the Certificate on to the Secure Gateway Server

1. Browse to the Secure Gateway Administration Portal at '<https://localhost/Admin/Login.html>'.
2. Log in to the Administration Portal.
3. Go to the **Security** page.
4. Choose the certificate in the SSL Certificate drop-down.

Administrator

Logout

Gateway Settings

Dashboard

Security

Web Server

Authentication Server

Log Settings

Download

Admin

About

## Security

By forcing SSL policy errors on clients you might lock yourself out of the admin.  
Use with care!

Fields marked with \* are mandatory.

Secured Port \* 443

SSL Certificate \* Ericom Secure Gateway self-signed certificate

Ignore certificate errors while doing SSL handshake with a client.

Ignore certificate errors while doing SSL handshake with a host.

Ignore certificate errors while doing SSL handshake with a HTTP Web host (External Web Server / VMware View Server).

Enable Draining Mode.  
New connections will be disabled on the server.

Save Cancel

5. Click **Save**.

### Configure the Certificate on to the Plant SCADA Access Anywhere Authentication Server

1. Browse to the Secure Gateway Administration Portal at '<https://localhost:7443/Admin/Login.html>'.
2. Log in to the Administration Portal.
3. Go to the **Security** page.
4. Choose the certificate in the SSL Certificate drop-down.

### Security Settings » Ports and SSL certificates

Select your SSL certificate

Secured Port 7443

SSL Certificate Ericom Authentication Server self-signed certificate

Save Test Connection Discard Changes

5. Click **Save**.

## Built-In Web Server

The Secure Gateway has a built-in web server to host web pages for Plant SCADA Access Anywhere. The built-in Web server cannot be disabled and listens on the Secure Gateway port.

To configure the Web server, open the Configuration tool and navigate to the Web Server page.

The screenshot shows the AVEVA Configuration tool interface. On the left is a navigation sidebar with the following items:

- Administrator
- Logout
- Gateway Settings
- Dashboard
- Security
- Web Server** (selected)
- Authentication Server
- Log Settings
- Download
- Admin
- About

The main content area is titled "Web Server". It contains a single configuration option:

Enable non-secured port for HTTPS auto-redirect

At the bottom right of the content area are two buttons: "Save" and "Cancel".

For example, if Plant SCADA Access Anywhere is selected, when the user navigates to `https://<sg-server-address>:<port-number>/` the URL will automatically redirect to:

`https://<sg-server-address>:<port-number>/PlantAccessAnywhere/start.html`

**Note:** Using the Secure Gateway to proxy to pages other than Plant SCADA Access Anywhere is not supported.

## Connecting to the Web Server

To connect to a Plant SCADA Access Anywhere server available through the Secure Gateway Web server, open a browser and navigate to the desired URL. If a port other than 443 is being used by the Secure Gateway, it needs to be explicitly stated in the URL. For example: `https://myserver:4343/PlantAccessAnywhere/start.html`.

**Note:** If you are running Plant SCADA Access Anywhere on an IPv6 network and you attempt to connect to the Access Anywhere Server using an IPv6 address in Internet Explorer, the connection will not be successful. Use a different browser, or check with your system administrator if there is a configured hostname available for the Access Anywhere Server.

The following URLs are available by default:

`https://server:port/` OR `https://server:port/PlantAccessAnywhere/start.html`

## HTTP Redirect

The Plant SCADA Access Anywhere Secure Gateway Web server listens on port 80 by default. This way, HTTP references to the server will automatically redirect to the HTTPS URL.

**Note:** This feature only works if the Secure Gateway is listening on port 443. If it is configured to use any other port, the HTTP automatic redirect is not supported.

To enable this feature, select the option: **Enable non-secured port for HTTPS auto-redirect**.

The screenshot shows the AVEVA Web Server configuration interface. On the left is a navigation sidebar with links: Administrator, Logout, Gateway Settings, Dashboard, Security, Web Server (which is selected and highlighted in blue), Authentication Server, Log Settings, Download, Admin, and About. To the right of the sidebar, the main panel title is "Web Server". Underneath the title, there is a configuration setting: "Enable non-secured port for HTTPS auto-redirect" with a checked checkbox. At the bottom right of the main panel are two buttons: "Save" and "Cancel".

Configure this feature in the *EricomSecureGateway.exe.Config* file using:

```
<add key="EnableNonSecuredPortForHttpsAutoRedirect" value="false" />
```

## Disabling HTTP Filtering

Certain types of network traffic are blocked by firewalls. Port 443 on many firewalls is initially reserved for HTTP and HTTPS based communication. Many firewalls have a rule to filter out any non-HTTP data. Depending on what the Secure Gateway will be routing, HTTP filtering may need to be disabled on the firewall.

The Secure Gateway can proxy various types of traffic. Some are HTTP based and some are not.

## Advanced Configuration

Back up the existing *EricomSecureGateway.config* file before making any changes.

To configure the settings of the built-in Web server, open the *EricomSecureGateway.config* using a text editor. Each folder in the WebServer directory may have a default document assigned for it, and may also be restricted so that end users cannot access it.

Name	Date modified	Type	Size
AccessNow	12/10/2015 1:27 PM	File folder	
Admin	12/10/2015 1:27 PM	File folder	
Blaze	12/10/2015 1:27 PM	File folder	
SG	12/10/2015 1:27 PM	File folder	
View	12/10/2015 1:27 PM	File folder	
welcome	12/10/2015 12:59 ...	HTML Document	6 KB

## Access Anywhere Web Client User Guide

This guide describes how to use AVEVA Plant SCADA Access Anywhere to remotely connect to a Plant SCADA client by means of an HTML5-compatible web browser.

### About Plant SCADA Access Anywhere

Plant SCADA Access Anywhere enables you to remotely access a running Plant SCADA client with a mobile device including tablets and smartphones, and laptop computers.

You can view and control the client through a web browser without needing to install it on your portable device. You can interact with the Plant SCADA client from anywhere in real-time.

Plant SCADA Access Anywhere provides the following features:

- Provides user access and interaction with Plant SCADA clients from any device that has an HTML5-compatible web browser (see [Supported Browsers](#)).
- Provides access without the need to install, configure, update, or patch any software on a portable device.
- Works on platforms that support only web applications such as the Google Chrome operating system.

Plant SCADA Access Anywhere Secure Gateway provides the following alternative functions with Access Anywhere Server:

- Provides access without the need to purchase, install, configure, and manage a VPN.
- Accesses internal resources using a single Secure Gateway that is installed in a perimeter network, while other resources reside securely behind an internal firewall.
- Installs a single Secure Socket Layer (SSL) digital certificate on the Secure Gateway node instead of installing on every host that needs to be accessed.

### Supported Browsers

With Plant SCADA Access Anywhere, users can access remote Plant SCADA from HTML5 compatible web browsers on any device including smart phones, tablets, and laptop computers.

To start a session, navigate to the *start.html* file that is installed on the Access Anywhere Server. To do this, point a browser to the Access Anywhere Server URL:

*https://<computer name or IP address>:8080*

## Browsers Tested with Plant SCADA Access Anywhere

- Microsoft Edge
- Google Chrome
- Safari on Apple iOS

---

**Note:** Browser extensions and toolbars may inject JavaScript code into web pages, which can adversely impact the behavior of certain web pages. If Plant SCADA Access Anywhere is not working properly, disable or uninstall any active browser extensions or tool bars. Restart the web browser after uninstalling or disabling an extension to confirm that it is no longer active.

---

**Note:** If you are running Plant SCADA Access Anywhere on an IPv6 network and you attempt to connect to the Plant SCADA Access Anywhere Server using an IPv6 address in Internet Explorer, the connection will not be successful. Use a different browser, or check with your system administrator if there is a configured hostname available for the Plant SCADA Access Anywhere Server.

---

## Important Notes

Certain versions of the above listed browsers, such as Safari and Chrome, are in theory functionally compatible with Access Anywhere. You may be able to use them, but specific behaviors are unknown because different browser versions have not been tested with Access Anywhere. We strongly recommend that you upgrade to the most recent version.

Although Plant SCADA Access Anywhere supports the listed browsers, you should review the [Running Access Anywhere on Different Web Browsers](#) for behaviors and alternatives specific to each browser.

Multiple Access Anywhere sessions may be opened in different tabs within the web browser, or in different browser windows. When a session is not in use (its tab or window is not displayed), it will reduce its CPU and memory utilization on the server.

## Viewing Plant SCADA with Plant SCADA Access Anywhere

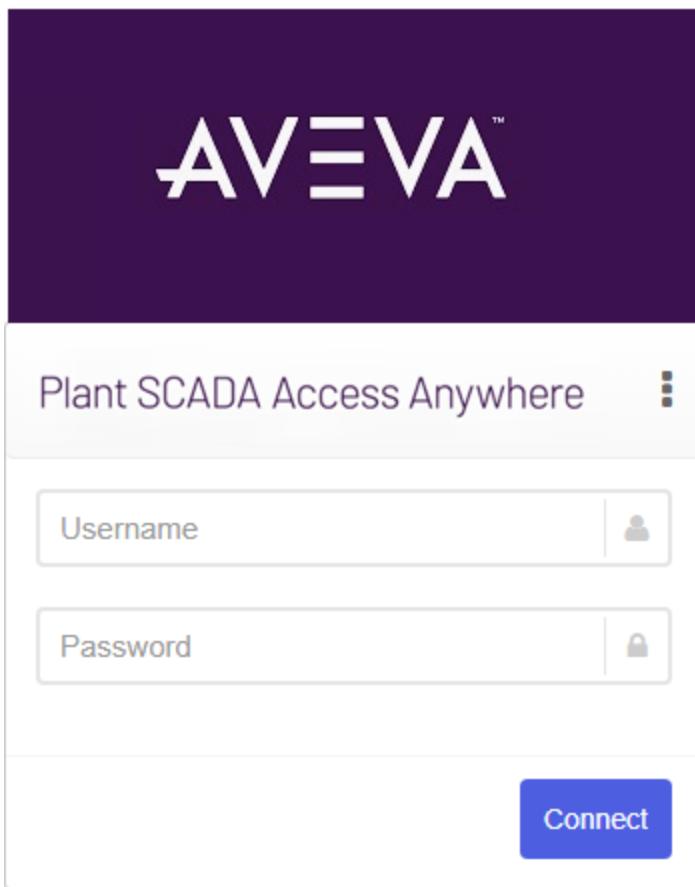
This chapter describes how to log on to Plant SCADA Access Anywhere from any device with an HTML5-compatible web browser. It includes:

- A step-by-step procedure to log on (see [Logging On to Plant SCADA Access Anywhere](#))
- How to securely connect to Access Anywhere (see [Securely Connecting to Plant SCADA Access Anywhere](#))
- Advanced connection settings.

## Logging on to Plant SCADA Access Anywhere

To log on to the Plant SCADA Access Anywhere Connection Web Page, follow these steps:

1. Navigate to <https://<Plant SCADA Access Anywhere Server Node Name>:8080/>. The Logon dialog appears.



2. Enter the connection parameters (see the table below).
3. Tap or click **Connect** to initiate the connection. The progress indicator is displayed before connection is established.

Connection Parameters	Description
User Name	Credentials to log on to the RDP host. If the Secure Gateway is in a different domain to the Authentication Server/Plant SCADA Access Anywhere Server, the domain needs to be specified (for example, domain\user).
Password	Corresponding password for the user name. This value should not be saved for future connections.

If the Plant SCADA session does not start or any notifications appear, confirm your details with your Access Anywhere Administrator. You can also check your connectivity using the Plant SCADA Access Anywhere demonstration site. See [Checking Connectivity](#).

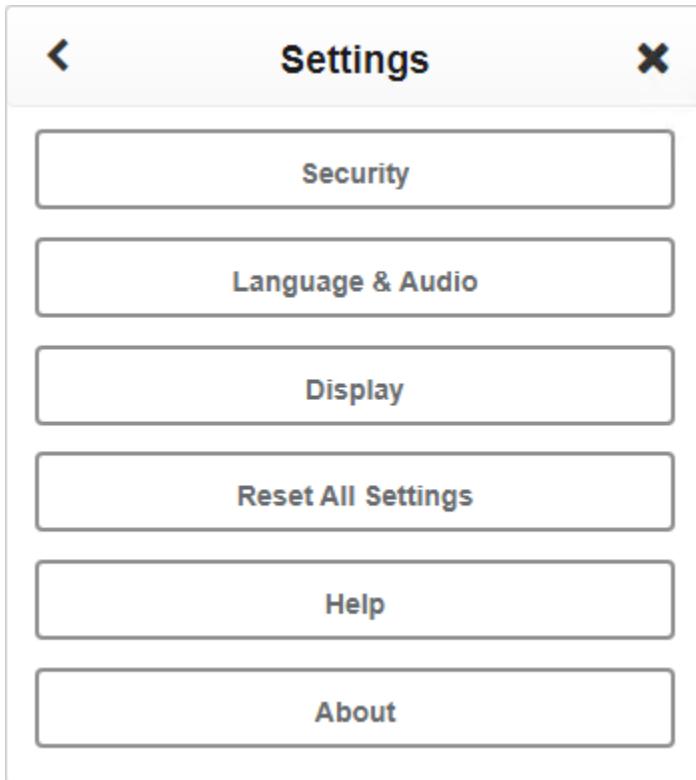
**Note:** If you are running Plant SCADA Access Anywhere on an IPv6 network and you attempt to connect to the Plant SCADA Access Anywhere Server using an IPv6 address in Internet Explorer, the connection will not be successful. Use a different browser, or check with your system administrator if there is a configured hostname available for the Plant SCADA Access Anywhere Server.

## Configuring Advanced Settings

To configure advanced settings:



1. Click . The following tabs appear as shown below:



2. Click on each tab to view the settings available for configuration. For example, to view Display settings, click the **Display** tab.



To return to the Settings dialog, click the following icon.



To close the Settings dialog and return to the Logon dialog, click the close icon.

3. Complete the required settings.

## Security Settings

Option(s)	Description
Enable SSL encryption for remote session	Select this option to allow the client to use Secure Socket Layer (SSL) encrypted WebSocket communication to the Plant SCADA Access Anywhere Server. As a default, this option is selected.

## Language & Audio Options

Option(s)	Description
Display Language	Select the language in which you wish to view the Plant SCADA Access Anywhere interface. The Display Language only applies to the logon web page, and is not propagated to the Plant SCADA client.
Keyboard Locale	Select the keyboard region to be used in the session (keyboard_locale). If you select a non-English keyboard locale, your computer will need to be configured to use a matching keyboard in the Windows <b>Region &amp; Language</b> settings. Some keyboards may not work correctly with Edge browsers.
Use keyboard scan-codes	Enables scan codes. Certain applications use scan codes and will require this setting to be enabled.
Remote Audio Playback	Configure where the session's sound will play: local computer, remote computer, or do not play. Audio playback is not supported with Internet Explorer version 11.

## Display Options

Option(s)	Description
Acceleration / Quality	The acceleration, or degree of quality can be specified by selecting options from a drop-down list. Faster acceleration will result in lower quality images.
Screen Resolution	Sets the resolution size of the Plant SCADA Access Anywhere session. Select a value from the drop-down list of values. For example: "800 x 600". Select a screen resolution that matches your Plant SCADA project. The default may be configured by the administrator to already match your Plant SCADA Project.
Automatic session resize	Enables/disables automatic display resizing. By default, automatic resizing is selected (enabled).

Option(s)	Description
	This means whenever the browser window is resized, the Plant SCADA Access Anywhere session will automatically adjust itself to the new dimensions. See "Automatic Display Resize" in the Plant SCADA Access Anywhere Server Install Guide for more information.

**Note:** For details about the version of Plant SCADA Access Anywhere you are running, click the **About** tab in the Settings dialog.

Click the **Reset All Settings** tab to revert to the default settings.

Depending upon the privileges your administrator has assigned, a Plant SCADA view-only client or control client will be launched. A view-only client is a computer configured with view-only access to the Plant SCADA runtime system. No control of the system is possible, but access to data monitoring is available. A control client is the interface between the Plant SCADA runtime system and an operator. If you are using Plant SCADA on a network, every Plant SCADA computer (on the network) is a control client.

Access to either client depends upon the permissions granted to you by your administrator. For more information, refer to the Plant SCADA Access Anywhere Installation and Configuration Guide.

**Note:** Click the **Help** tab in the Settings dialog to view the *Plant SCADA Access Anywhere Web Client User Guide*.

## Securely Connecting to Plant SCADA Access Anywhere

Plant SCADA Access Anywhere is compatible with most VPNs. SSL and VPNs that will not support WebSockets require the Secure Gateway as well. Juniper IVE version 7.4 (and later) supports WebSockets, so the Secure Gateway is not required.

Access Anywhere has been tested with Juniper's SA SSL VPNs. For more information about Juniper's VPNs, see the [SSL VPN Configuration](#) section in the *Access Anywhere Server Installation and Configuration Guide*. Configuration with other third-party SSL VPN appliances will be similar to the procedures described here (differences are mostly in terminology).

## Obtaining a Plant SCADA Runtime License

Plant SCADA requires a license to run as a display client in Plant SCADA Access Anywhere. By default, the display client will obtain the Plant SCADA license it requires from a server to which it is connected (an alarm, trend, report or I/O server).

If you want the display client to use a Plant SCADA license that is hosted locally on the computer where Plant SCADA Access Anywhere is running, you need to check the following:

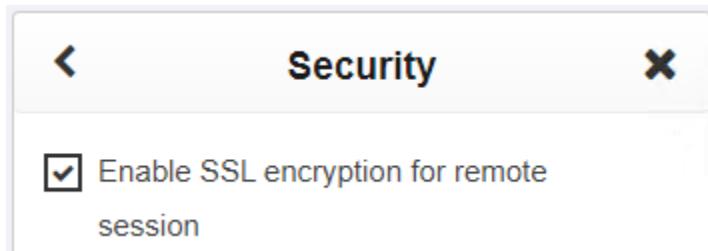
- The license needs to be configured locally on the Plant SCADA Access Anywhere computer.
- The Citect.ini file on the Plant SCADA Access Anywhere computer needs to have the [Client]UseLocalLicense parameter set to 1.
- Plant SCADA needs to be running as a Windows Service.

## Checking Connectivity

If you have any trouble connecting remotely to Plant SCADA Access Anywhere, you can connect to a Plant SCADA Access Anywhere demonstration on the following site:

<https://engage.aveva.com/operations-control-demonstrations-thank-you.html>

After registration, locate the Plant SCADA demonstration page and use the Plant SCADA Access Anywhere link to navigate to the demo site. Before you login, you will need to go to the Plant SCADA Access Anywhere client settings and turn off **Enable SSL encryption for remote session**.



If the demo site appears and you can successfully launch Plant SCADA, the browser is compatible with Plant SCADA Access Anywhere.

If the demo site works, verify that the following are true for your own system:

- You can locally connect at the Plant SCADA Access Anywhere node itself by using a browser listed in the [Supported Browsers](#) section.
- The Plant SCADA Access Anywhere service is running.
- Windows Firewall is configured correctly.
- The Plant SCADA Access Anywhere port between your browser and the Access Anywhere environment is available. The default port is 8080.
- A trusted certificate may be required for the Plant SCADA Access Anywhere Secure Gateway or the Plant SCADA Access Anywhere Server.
- Verify network connectivity.
- Verify that the client device can reach the Access Anywhere Server or the Access Anywhere Secure Gateway node. The Ping and Traceroute commands are useful in a Windows-based system. Third party tools exist for certain mobile devices to provide equivalent functionality. If you cannot reach a node by name, try using its IP address.

## Viewing a Plant SCADA Access Anywhere Session

After successfully logging on, an RDP session is launched and Plant SCADA starts at the remote machine. The Plant SCADA client appears within a browser window.

While it is connected, Plant SCADA Access Anywhere intercepts the mouse, button, and keyboard events, and transmits them to the RDP host. As a result, various keyboard keys and mouse buttons that are usually handled by the browser will behave differently. For example, clicking the F5 button normally causes the browser to reload the current page, but when using Plant SCADA Access Anywhere F5 will not reload the page. Instead, the keyboard command will be transmitted to and handled by the remote Plant SCADA client.

**Note:** In most browsers, clicking the **Back**, **Forward**, or **Reload** browser buttons will cause Plant SCADA Access Anywhere Server to display a message asking if you wish to leave the current page. If you decide to proceed, the

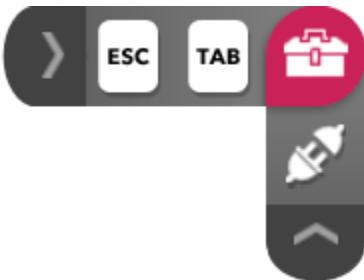
---

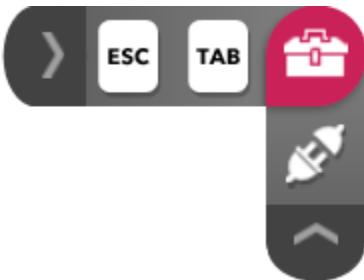
remote session will be disconnected. (Some browsers may not prompt for confirmation).

When accessed by Plant SCADA Access Anywhere, a Plant SCADA client cannot be minimized. If the remote Plant SCADA client is shut down at the server end, your session will be terminated automatically in approximately three seconds. Closing the browser in your remote session will end the RDP session.

## Plant SCADA Access Anywhere Toolbar



After connection to a Plant SCADA client is established, click  to view the Plant SCADA Access Anywhere toolbar.

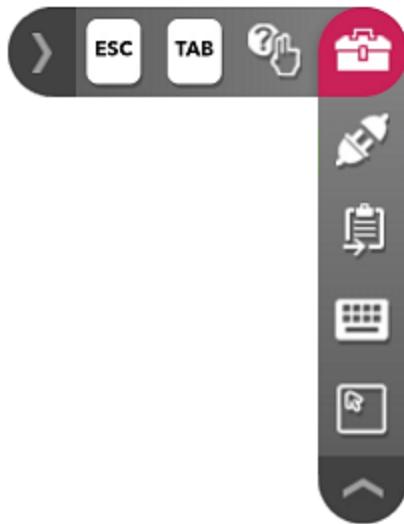


You can show or hide the branches of the toolbar by clicking the arrows at the end of each branch.

The Plant SCADA Access Anywhere toolbar includes the following buttons:

Button	Description
	Closes the Plant SCADA client
	Navigates to the next element in the tab order
	Ends the Plant SCADA Access Anywhere session

If the toolbar appears on a mobile device (such as an Android tablet or iPad), it will include additional buttons.



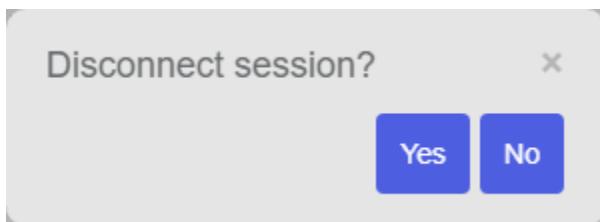
Button	Description
	Displays instructions for using gestures
	Allows you to copy text to the remote clipboard
	Displays the on-screen keyboard
	Changes the cursor operation from tap mode to drag-and-select mode.

## Ending a Plant SCADA Access Anywhere Session

To log out or disconnect, click the following:



A message appears prompting you to confirm the action.



The browser returns to the connection dialog after you click **Yes**.

No trace of a session remains on your device after ending a session. For additional protection, close the browser tab or window that previously ran the Plant SCADA Access Anywhere Server session.

## Session Auto-logoff

A session is logged off when you close the Plant SCADA client. Plant SCADA Access Anywhere Server includes an auto-logoff feature that automatically logs off the session if nothing is displayed on the screen for a specified period.

## Automatic Reconnect

Active Plant SCADA Access Anywhere sessions automatically attempt to recover from temporary network outages by reconnecting to your session. You may experience a slight delay during the reconnect attempt, but after the session is re-established, you can continue working without having to log back on to the session.

# Using Plant SCADA Access Anywhere on Portable Devices

This chapter describes the behavior and functionality of Plant SCADA Access Anywhere on different mobile devices, web browsers, and operating systems.

## Tablet and Smartphones

Plant SCADA Access Anywhere can operate on portable devices with an HTML5-compliant browser (for example, Safari on iPad). For the list of supported browsers, see [Supported Browsers](#).

**Note:** Plant SCADA Access Anywhere is supported on Android versions 4.2.2 and higher.

Mobile devices enable you to remotely access and interact with Plant SCADA clients. Touch gestures are used to perform the tasks normally done by a mouse using a desktop or laptop computer. Built-in software keyboards are used instead of physical keyboards. Certain mouse events will not have an equivalent on a touch device.

Software keyboards in mobile devices will not have F1-F12, CTRL, or ALT keys. In addition, mouse hover and tool tips are not supported on touch devices.

Be aware of these differences when using Plant SCADA Access Anywhere to remotely view the Plant SCADA client. Also, become familiar with the way your device implements touch interfaces. Keep this in mind for the design of Plant SCADA projects for use with Plant SCADA Access Anywhere. For example, Input animations should not need to invoke a Plant SCADA or Windows keyboard as mobile devices have their own.

For existing Plant SCADA clients that make use of mouse events, keys, and/or key combinations without a supported equivalent in a touch environment, you may want to modify your client to use alternate events that will work in a touch environment.

The following list provides tips on using Plant SCADA Access Anywhere on a tablet or smartphone device without a physical keyboard or mouse. Functionality will vary across different devices and some commands may not be available.

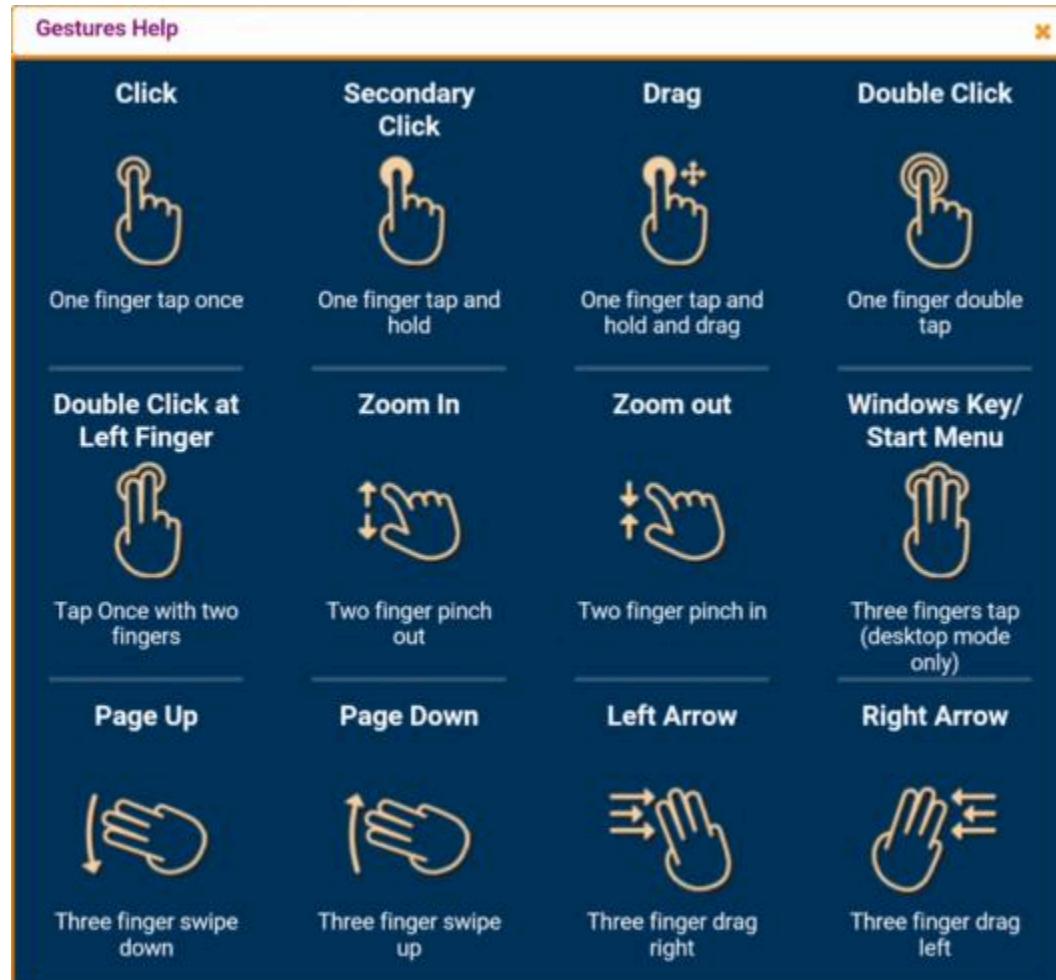
- Single Tap performs a left click.
- Single long Tap performs a right-click.
- Tap + Hold + Drag performs a select then drag/scroll function.

- Double Tap, or tapping once with two fingers, performs double-click.
- Tap with three fingers sends Back command to a remote browser.
- Swipe down with three fingers is Page Up.
- Swipe up with three fingers is Page Down.
- Drag left or right with three fingers performs a left arrow and right arrow respectively.
- Tap the keyboard icon (upper right-hand corner of window) to open/close the virtual keyboard.
- Swipe and pinch gestures will apply to the Plant SCADA Access Anywhere session (i.e. zoom in with pinch in).

**Note:** (For iOS only) when saving a Plant SCADA Access Anywhere icon to the iOS desktop, the shortcut will open the Plant SCADA Access Anywhere session in full-screen mode. The browser's toolbar will be hidden and there will be more remote desktop area available.

## Gestures Support

Plant SCADA Access Anywhere supports many different gestures when used on a touch device, such as a tablet. Tap the Gestures help icon in the Toolbar to see the complete list of supported gestures.



## Automatic Display Resize

Plant SCADA Access Anywhere supports automatic display re-size. Whenever a browser window is re-sized, the Plant SCADA Access Anywhere session automatically adjusts itself to the new dimensions (assuming the Plant SCADA Runtime window is set to full screen).

To re-size a browser window, drag any corner of the browser window and release it when the desired dimensions are reached. If a browser is placed into full screen mode, the Plant SCADA Access Anywhere session will automatically expand to the full screen. Be aware that resizing is subject to the display limitations of the Plant SCADA runtime system.

You can enable/disable automatic resizing via the **Display** settings for the Access Anywhere Web Client. See [Logging on to Plant SCADA Access Anywhere](#).

## Running Access Anywhere on Different Web Browsers

This section describes browser-specific behaviors in Plant SCADA Access Anywhere. The behaviors are grouped by operating systems. Alternative solutions are provided when available.

- [Android Operating System](#)
- [Google Chromebook](#)
- [iOS](#)

### Android Operating System

This table shows the behavior of different Android browsers.

Browser(s)	Behavior Type	Description
All	Drag and drop	You may experience difficulty dragging and dropping pop-up windows on the Android operating system. As a workaround, using a stylus to perform the drag and drop operations may improve the functionality.
Chrome	Scroll bars	Scroll bars in Plant SCADA cannot be scrolled by tapping or pressing and dragging the scroll button. Tapping on the gray area of the scroll bar itself will provide scroll functionality.
Chrome	Text display	When an Android tablet runs low on memory, text in a window can appear blurred. As a workaround, you should close the running applications and restart the device.

Browser(s)	Behavior Type	Description
Chrome	Text input	Typing double-byte language characters using the native Android software keypad are rendered as the question mark character (?) in a Plant SCADA data entry field. As a workaround, configure Plant SCADA Access Anywhere Server to use the Windows Keyboard option. Double-byte language characters can be entered correctly from the Android keypad OR by using a paired Bluetooth hardware keyboard.

## Google Chromebooks

Plant SCADA Access Anywhere operates on Google Chromebook and Chromebox just like it does with a Google Chrome browser. Here are some tips to keep in mind when using Plant SCADA Access Anywhere with a Chromebook or Chromebox.

Function	Description
Mouse Left-click	Click the Chromebook trackpad with one finger.
Mouse Right-click	Click the Chromebook trackpad with two fingers
Scrolling a document or website	Drag two fingers on the Chromebook trackpad up or down to scroll
Configure Chromebook	In the address field: <i>chrome://settings</i>

## Chromebook Keyboard

The Chromebook keyboard lacks several keys that are used by Windows. ChromeOS provides standard mappings that use existing keys with the **ALT** key to represent certain missing keys. Plant SCADA Access Anywhere supports these key combinations:

Command	Key Combination
Delete (DEL)	ALT+Backspace
Page Up	ALT+Up
Page Down	ALT+Down
Home	CTRL+ALT+Up
End	CTRL+ALT+Down

In addition, Plant SCADA Access Anywhere provides special non-standard mappings for additional key combinations on ChromeOS.

Command	Key Combination
F1	CTRL+1
F2, ...	CTRL+2, ...
ALT+TAB	ALT+`
ALT+SHIFT+TAB	ALT+SHIFT+`
CTRL+Home	CTRL+ATL+Left
CTRL+End	CTRL+ALT+Right

## iOS

This table shows the specific behavior of Safari browsers running on the iOS operating system.

Behavior	Description
Accessing Plant SCADA Access Anywhere	You may experience difficulty accessing Plant SCADA Access Anywhere. This can occur if the Private Browsing option is left in Enabled state. Private browsing is enabled if Safari bars appear black or dark instead of blue or gray. As a workaround, navigate to Settings, then Safari, and disable private browsing.
Refresh current page	When you click to refresh or navigate away from the current page while connected to the Plant SCADA Access Anywhere server, you will not be prompted to confirm your action, which may result in unwanted page navigation. There is no workaround.
Invoking Plant SCADA keyboard	A user input animation may not bring up the Plant SCADA keyboard and the alphabetic re-sizeable keyboard is not functional. The recommended workaround is to use the device's built-in keyboard.
Moving between text boxes	The Next and Previous buttons are not enabled to move between text boxes on dialog boxes with multiple text boxes. As a workaround, use a touch gesture to transfer focus.
Certificates	When connecting through the Secure Gateway, self-signed certificates are not supported on iOS. They will result in the error message "Gateway: HTTP sessions are disabled".

Behavior	Description
	This error message will also be shown when using domain-issued certificates if the domain root certificate is not installed on your device. Contact your system administrator to install the root certificate.

# AVEVA Cloud Services

AVEVA offers a range of cloud services through its common cloud platform CONNECT. You can access CONNECT online at <https://connect.aveva.com/>.

For Plant SCADA users, CONNECT provides access to AVEVA Integration Studio, an infrastructure-as-a-service virtual development environment.

Integration Studio allows you to add Plant SCADA nodes to a project template. These nodes can be used to create multiple instances of virtual machines with a fully licensed version of Plant SCADA installed.

For more information, see [Plant SCADA and AVEVA Integration Studio](#).

## Plant SCADA and AVEVA Integration Studio

AVEVA Integration Studio is an infrastructure-as-a-service virtual development environment that is delivered on AVEVA's cloud services platform CONNECT.

Integration Studio lets you run and manage software applications without the complexity of building and maintaining the physical infrastructure typically required when developing and launching software. You can create templates for virtual machines and deploy multiple instances on demand.

AVEVA Integration Studio offers the following benefits:

- Provides virtual infrastructure for AVEVA servers and software that is pre-installed, configured, and licensed
- Enables management of your projects from one consolidated virtual dashboard
- Provides a fully virtual environment for training your staff and customers
- Simplifies IT overheads, thereby delivering efficiencies and reduced costs.

For more information, see the [AVEVA Integration Studio](#) documentation.

---

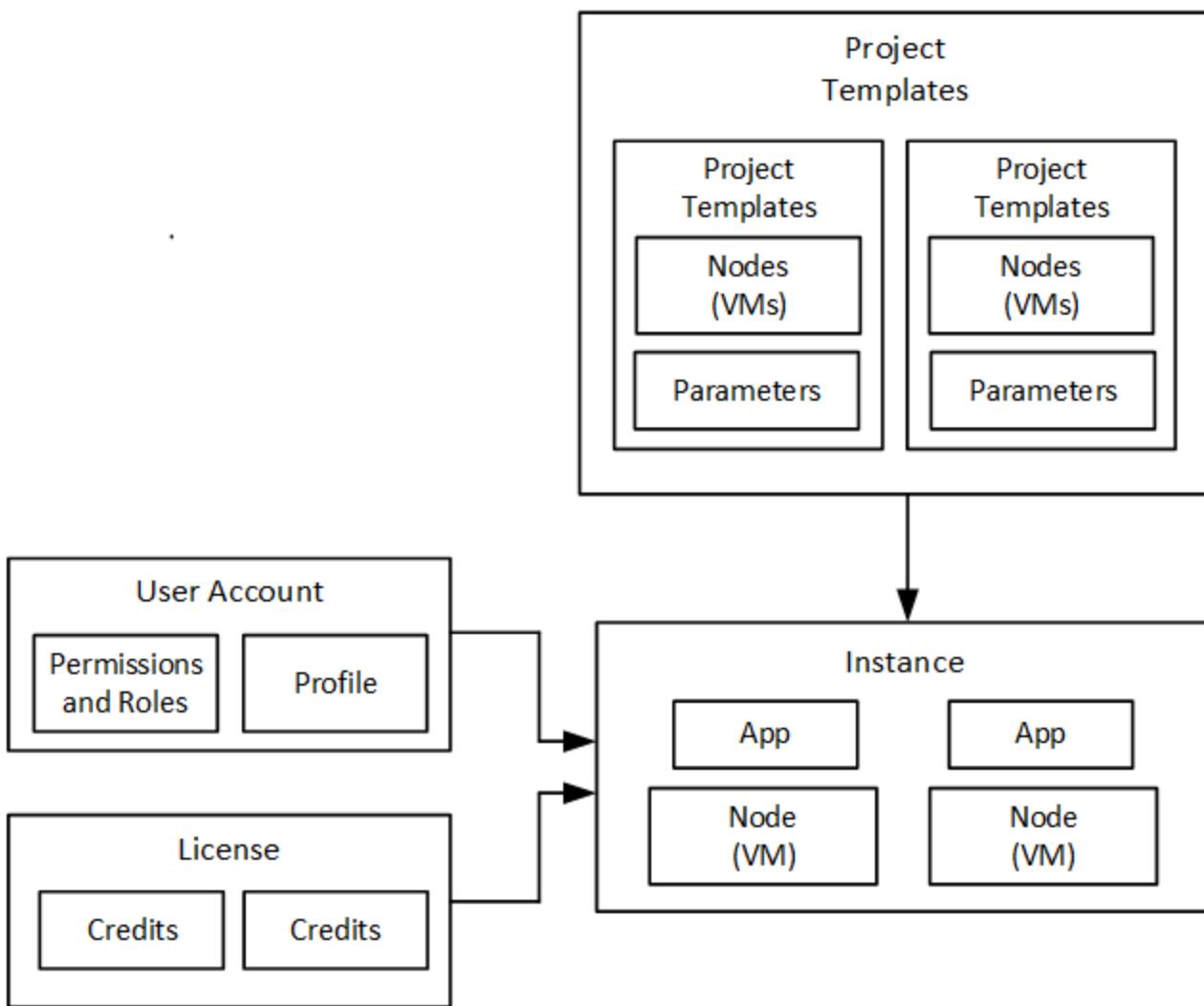
**Note:** Integration Studio is intended for use as a platform to develop and test software applications. It should not be used to operate a live production facility.

You can create collaborative simulation systems using the following components:

- **Project templates** include all of the settings needed to launch a simulation system. They define the different types of virtual machines required for a system as a set of "Nodes".
- **Instances** are the virtual machines generated from the Nodes defined in a project template. They form the simulation system accessed by your end users.
- **User accounts** give people access to AVEVA Integration Studio. Accounts are created and managed by administrators on your team.
- **Licenses and Credits** are applied to your AVEVA Integration Studio subscription. They control the number of nodes that can be created, however every node is equipped with a full unlimited license for the installed application. There is no need to apply or reconfigure licenses, and you do not need to use the AVEVA's License Manager.

---

**Note:** AVEVA Enterprise Licensing pages will still appear in Configurator on the virtual machines created by Integration Studio. You do not need to use these pages. See [Known Issues](#).



To manage AVEVA Integration Studio, you need a CONNECT account with administrative privileges and a paid subscription to Integration Studio. See [Access AVEVA Integration Studio](#).

**Note:** The following Plant SCADA features are currently not supported by Integration Studio:

- If you use the Setup Wizard to assign the role **HMI (Standalone, no networking)** to a virtual machine, it will not be able to acquire an HMI license and will run in demo mode.
- Dual network interface cards are not supported.

Installing and using Plant SCADA Access Anywhere is also not supported by Integration Studio.

## See Also

[Use Plant SCADA with AVEVA Integration Studio](#)

## Access AVEVA Integration Studio

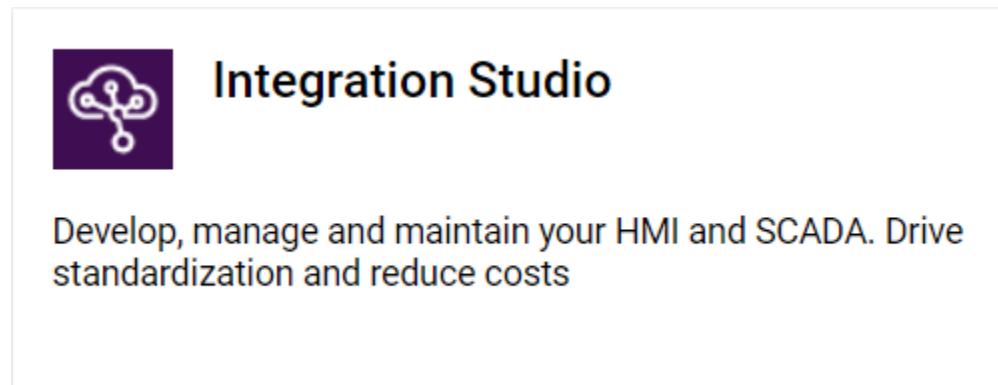
You can access AVEVA Integration Studio through CONNECT, AVEVA's common cloud platform. It is located online at <https://connect.aveva.com/>.

When you log in to CONNECT, you will see a set of folders that host the cloud services you can access (referred to

as "solutions"). This will be set up for you by the CONNECT administrator within your organization.

You will receive an email from AVEVA informing you that you have been invited to join CONNECT. You can use the link in this email to access the login page.

Your CONNECT account should provide administrator privileges and include a paid subscription to AVEVA Integration Studio. If this is the case, the following will appear when you select an appropriate host folder.



You can click on this tile to launch AVEVA Integration Studio.

**Note:** Popups need to be enabled in your browser to launch Integration Studio using this button.

## See Also

[Use Plant SCADA with AVEVA Integration Studio](#)

## Use Plant SCADA with AVEVA Integration Studio

AVEVA Integration Studio allows you to include Plant SCADA nodes in a project template. These nodes can be used to create multiple instances of virtual machines with a fully licensed version of Plant SCADA installed.

Two types of Plant SCADA nodes can be configured:

- Development Workstation
- Runtime Client.

A node based on a **Development Workstation** will install the following Plant SCADA components:

- AVEVA Common Components
- Configuration environment
- Runtime
- Deployment Server
- OPC UA Server
- Industrial Graphics Web Server
- Communications Drivers.

All installed Plant SCADA drivers are included, except for BACNET. If the BACNET driver is required, you can install it by modifying your Plant SCADA installation via the Programs and Features page in Control Panel. You can also download and install a BACNET driver pack from the Communication Drivers page at the [AVEVA™](#)

## Knowledge & Support Center.

A **Runtime Client** will only include the runtime components. You cannot run a server on this type of node.

The node types you require in a template will depend on the purpose of the simulation system you would like to create.

- **Example 1 - Virtual training room**

If you want to introduce Plant SCADA to a number of trainees, you could create a template with a single node based on a Development Workstation. This could be used to create an instance of a virtual machine for each trainee. This would provide access to Plant SCADA's development environment, and allow each trainee to build and run a project as a standalone system.

- **Example 2 - System test environment**

If you want to use a virtual simulation to test a project update for a distributed production environment, you can create a set of nodes that represent the computers included in the system you would like to test. This could include a development workstation, multiple runtime servers, a deployment server, a System Management Server and runtime display clients.

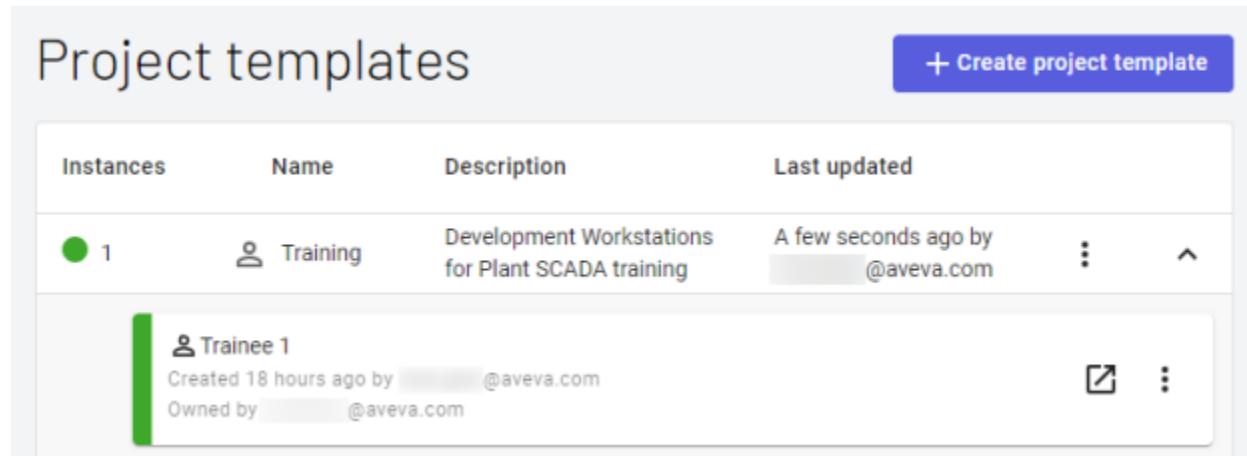
In this case, you would use the Development Workstation node type for the development machines, the deployment server, and any runtime servers. You would use the Runtime Client node type for pure runtime-only clients. You could then use these nodes to create a virtual simulation of a physical plant, providing a safe environment to deploy and test a Plant SCADA project.

---

**Note:** If you create a distributed simulation system that includes a set of redundant servers, they will not be able to operate as a redundant pair by default. If this is required, see [Configure Server Redundancy](#).

---

When you have completed the configuration of your template, you can create and launch instances from the [Project templates](#) page.



The screenshot shows a table titled "Project templates" with a "Create project template" button at the top right. The table has columns: Instances, Name, Description, and Last updated. There are two entries:

Instances	Name	Description	Last updated
1	Training	Development Workstations for Plant SCADA training	A few seconds ago by @aveva.com
	Trainee 1	Created 18 hours ago by @aveva.com Owned by @aveva.com	 

When you launch an instance, you have the option to use a **Browser** or **RDP**.

- Select **Browser** to launch the instance as a virtual machine running on a new tab in your browser.

Integration Studio will generate a unique user name and password for each virtual machine it creates. When you select the Browser option, you will be automatically logged in using this generated profile.

- Select **RDP** to view the credentials required to access the virtual machine via a remote desktop application.

## Credentials for Training

Username: [REDACTED]

Password: \*\*\*\*\* 



Before you connect to Training using RDP, make sure the firewall remote desktop protocol rules allow access from your IP address. [More information](#)

[Download RDP](#)

 OK

If you are using the Browser option and need to know your user name or password, you can go back to Integration Studio and select the RDP option to view your user credentials.

---

**Note:** An Integration Studio instance of one or more nodes is configured as a Windows workgroup. Individual instances are isolated within their own workgroups and can only communicate with nodes included within their workgroup.

---

An explanation of the steps required to create a project template is included in the Integration Studio documentation. Click on the help icon on the Integration Studio title bar to access this information.



For information that is specific to Plant SCADA node types, see [Add Plant SCADA Nodes to a Project Template](#).

---

**Note:** Virtual machines based on a Plant SCADA node will be secured by default. This has implications for the setup of both single node and distributed simulations. For example, a System Management Server is setup on each instance of a development workstation node with encryption enabled. See [Security Implications for Plant SCADA Simulations](#) for more information.

---

## See Also

[Known Issues](#)

## Add Plant SCADA Nodes to a Project Template

The process required to add nodes to a project template is described in the Integration Studio documentation. Click on the help icon on the Integration Studio title bar and search for the topic *Add a Project Template*.

The following is additional information that will help you add Plant SCADA Nodes to a template. It aligns with the steps you progress through when you create a new template.

- **Project template information**

In the **System Suite definition** field, select **2023**. This suite contains AVEVA products with 2023 branding that

are supported by Integration Studio.

- **Node Configuration**

When you add a node, select **2023-PlantSCADAR2** as the node type.

## Add a node

Node name  
Training workstation

2023-PlantSCADAR2

▼ 2023

SystemPlatform

SystemPlatformR2

PlantSCADAR2

Add

You also have the option to select a **Machine type**. This determines the level of performance supported by an instance.

## Add a node

Node name

2023-PlantSCADAR2

ⓘ Machine type  
HighPerformance - Standard\_DS3\_v2

Cancel Add

Be aware that the level of performance you select for a node will directly impact the cost of running an instance. The higher the performance, the more it will cost per hour to run.

**Note:** You should confirm which Machine Type is appropriate for a node with your CONNECT administrator.

- **Launch parameters**

Indicate if you want a node to create instances based on a **Development Workstation** or a **Runtime Client**.

See [Use Plant SCADA with AVEVA Integration Studio](#) for a description of each.

Be aware that a Development Workstation includes a Runtime Client. If you are setting up a Development Workstation there is no need to select both, though there is no implications if you do.

## See Also

[Security Implications for Plant SCADA Simulations](#)

## Security Implications for Plant SCADA Simulations

Virtual machines based on a Plant SCADA node are secured by default. This has the following implications:

- **Encryption Enabled**

Instances generated from a Development Workstation node will include a configured System Management Server (SMS) with encryption enabled. Connections to non-encrypted systems have been disabled by default.

For multi-node systems, you will need to decide which virtual machine will act as your SMS. All other development nodes will then need to be configured to point to your nominated SMS. To do this:

1. Open Configurator and go to the **System Management Server** page.
2. Select **No System Management Server connection**, then the **Configure** button.
3. When the changes have been confirmed, reconnect the virtual machine nominated as your System Management Server.

Select **Connect to an existing System Management Server** and enter the virtual machine name of the SMS.

---

**Note:** Integration Studio uses Windows workgroups. This means the limitations that apply to Plant SCADA when using workgroups will also apply here. For example, the Deployment Server must be on the same node as the SMS, and IPV6 is not supported.

- **Runtime clients in a multi-node system**

If your multi-node system includes a runtime client, the connection to the SMS will not be configured. Under these circumstances, you will need to:

1. Connect the client to your nominated SMS on a Development Workstation node.  
You will need to type in the name of the node, it will not appear in the browse list.
2. When prompted for credentials, you will need to specify the name and password of the user that connects to your nominated SMS. You can determine this by viewing the RDP connection dialog in Integration Studio.

To connect a runtime client to the Deployment Server, you will need to go through the Deployment Client page in Configurator.

When prompted for the username and password, use the credentials displayed on the RDP dialog of the node that hosts the Deployment Server. The required format of the username will be:

<node name>\<username>

If you only specify the username, the connection will not be successful.

- **Server Passwords**

The Server Authentication password is preconfigured with an automatically generated unique strong

password.

If you are running a multi-node system, you will need to set this to the same password on every node. To do this, go to the **Plant SCADA | Computer Setup** page in Configurator.

The **Deployment Server** is also preconfigured with an automatically generated unique strong password.

As an administrator, you can change the password without needing to know the generated password.

## Configure Server Redundancy

If you have used Integration Studio to create a distributed simulation system that includes a set of redundant alarm servers, they will not be able to operate as a redundant pair by default. Both will function as a primary server.

If you specifically need to enable alarm server redundancy in your simulation system, you need to perform the following steps on both virtual machines within a redundant pair.

1. Open the Command Prompt and use the **IPConfig** command to display the **Connection-specific DNS Suffix**.
2. Copy this value to the clipboard.
3. In **Control Panel**, open the **System** page.
4. Select **Change Settings** to display the System Properties dialog.
5. On the **Computer Name** tab, click the **Change** button.
6. On the **Computer Name/Domain Changes** dialog click the **More** button.  
The DNS Suffix and NetBIOS Computer Name dialog will appear.
7. In the field **Primary DNS suffix of this computer**, paste the address you copied to the clipboard.
8. Restart the virtual machine to implement these changes.

You will then need to reset the connection to the System Management Server. To do this:

1. Open Configurator.
2. Go to the **System Management Server** page.
3. Select **No System Management Server connection**, then the **Configure** button.
4. When the changes have been confirmed, reconnect the virtual machine to the System Management Server used by your simulation system.
  - If the System Management Server is on a different virtual machine, select **Connect to an existing System Management Server** and enter the required virtual machine name.
  - If the System Management is hosted locally on the same virtual machine, select **This machine is the System Management Server**.

You then need to repeat these steps for the second virtual computer in a redundant pair and reset its connection to the System Management Server.

## See Also

[Known Issues](#)

## Known Issues

### **Plant SCADA starts in demo mode following a reboot**

If you restart a virtual machine, Plant SCADA will go into demo mode the first time you launch runtime. This is because the AVEVA Plant SCADA Runtime Manager service does not pick up a license after a reboot. If you open Runtime Manager, you will see that the status of the System Services process is "Running (Demo)".

If you restart the System Services process, Plant SCADA will pick up a license correctly.

### **Uninstalling the Plant SCADA Project DBF AddIn**

Plant SCADA includes an add-in for Microsoft Excel called the **Plant SCADA Project DBF AddIn**. By default, this is installed by Integration Studio on a development workstation node.

If you modify the Plant SCADA installation on a virtual machine and remove the DBF AddIn, you cannot install it a second time. It is recommended that you do not attempt to remove the Plant SCADA Project DBF AddIn from a development workstation node.

### **Applying Monthly Updates**

Monthly updates to Plant SCADA are not automatically included in the version installed on a virtual machine by Integration Studio.

To apply an update to Plant SCADA, you will need to manually install it on each virtual machine. See [Manage Updates](#).

### **Server Redundancy**

If you create a distributed simulation system that includes a set of redundant servers, they will not be able to operate as a redundant pair by default.

If this is required, additional configuration is needed. See [Configure Server Redundancy](#).

### **AVEVA Enterprise Licensing pages in Configurator**

Licenses are managed through your AVEVA Integration Studio subscription. Every node is equipped with a local unlimited license which means you do not need to connect to a License Server or use AVEVA's License Manager.

Despite this, **AVEVA Enterprise Licensing** pages will still appear in **Configurator** on the virtual machines created by Integration Studio. This has the following impact on the functionality of these pages:

- The **Test Connection** button will not work.
- The **Secure** page will indicate that a secure connection to the License Server is not configured.

As there is no License Server, these issues will not impact your simulation system. You do not need to use these pages.

# Reference

This section includes the following reference information:

- [Cicode Reference](#)
- [Parameters](#)
- [VBA Programming Reference](#).

---

**Note:** For information regarding other methods you can use to extend Plant SCADA, see the topic [Extensibility](#).

---

## Cicode Reference

Cicode is a programming language designed for use in Plant SCADA to monitor and control plant equipment. It is a structured language similar to Visual Basic or 'C'. You need no previous programming experience to use it.

For more information see [About Cicode](#).

To access a full list of available Cicode functions, see [Cicode Function Categories](#).

## About Cicode

Cicode is a programming language designed for use with Plant SCADA. It is a structured language similar to Visual Basic or 'C'.

You can use Cicode to access real-time data (variables) in a Plant SCADA project and Plant SCADA facilities: variable tags, alarms, trends, reports, and so on. You can also interface to various components on a computer, such as the operating system and communication ports. Cicode also supports advanced features including pre-empted multitasking, multi threads, and remote procedure calls.

To access a full list of available Cicode functions, see [Cicode Function Categories](#).

Use the following sections as a quick start to using Cicode in your Plant SCADA projects:

- [Using Cicode Files](#) — Cicode can be stored in procedures called functions for multiple reuse and centralized maintenance.
- [Using Cicode Commands](#) — Cicode can be typed directly into command fields in online Plant SCADA forms.
- [Using Cicode Expressions](#) — Cicode expressions are used to display and log data for monitoring and analysis, and to trigger various elements in your system, such as alarms, events, reports, and data logging.
- [Using Cicode Functions](#) — A Cicode function is a small program, a collection of statements, variables, operators, conditional executors, and other functions. A Cicode function can perform complex tasks and give you access to Plant SCADA graphics pages, alarms, trend data, and so on.
- [Working with Commonly Used Functions](#) — Cicode has many pre-defined functions that perform a variety of tasks. The functions you will use most often are described in this topic.
- [Writing Functions](#) — Where system functionality cannot be achieved with built-in functions, you can write your own functions.

- [Cicode Editor](#) — The Cicode Editor is the code editing tool provided with Plant SCADA for writing, editing and debugging Cicode.

## See Also

[Performing Advanced Tasks](#)  
[Using Cicode Programming Standards](#)

## Using Cicode Files

Write your Cicode functions in Cicode source files, stored on your hard disk. Cicode files have a \*.CI extension.

To minimize potential future difficulties with maintaining your Cicode files, adopt a programming standard as early as possible (see [Using Cicode Programming Standards](#)). Maintain structured Cicode files, by logically grouping your Cicode functions within the files, and by choosing helpful descriptive names. For details about modular programming methods, see [Modular Programming](#).

For details about using and debugging Cicode functions, see [Formatting Functions](#) and [Debugging Cicode](#) respectively.

When you compile your Plant SCADA project, the compiler reads the functions in your Cicode source files. Your system can then use these functions in the same way as it uses built-in functions. You can use as many Cicode files as required. Cicode files reside in the same directory as your Plant SCADA project. When you back up your project, the Cicode source files in the project directory are also backed up.

## See Also

[Creating Cicode Files](#)  
[Opening Cicode Files](#)

## Restricted Cicode Keywords

The following keywords have specific function in the operation of Cicode and should not be used for function names, variable names, and so on. They are reserved for use as mathematical operators, condition executors, or as part of a function definition.

and	global	quality
bitand	if	real
bitor	int	return
bitxor	is	select
case	mod	string
cicode	module	then
CiVBA	nop	timestamp

do	not	to
else	object	var
end	or	while
for	private	
function	public	

If you use any of these words inappropriately in your Cicode, it will not be allowed by the compiler.

## Using Cicode Commands

Cicode commands extend the control element of a Plant SCADA control and monitoring system. You use commands to control your Plant SCADA system and therefore the processes in your plant.

Each command has a mechanism to activate it. Commands can be issued manually, through an operator typing a key sequence, or by clicking on a button (or object) on a graphics page. You can also configure commands to execute automatically:

- When an operator logs into or out of the runtime system
- When a graphics page is displayed or closed
- When an alarm is triggered
- In a report
- When an event is triggered

To define a Cicode command, you enter a statement (or group of statements) in the command field (Input category) for an object.

Each statement in a command usually performs a single task, such as setting a variable to a value, calculating a value, displaying a message on the screen, or running a report. For information on using variables, see the section titled [Using Variables](#).

If you want to evaluate a condition, like checking the state of your plant rather than perform an action or command upon your plant, use an expression instead. See the section titled [Using Cicode Expressions](#).

## See Also

[Setting Variables](#)

[Performing Calculations](#)

[Using Multiple Command Statements](#)

[Using Include \(Text\) Files](#)

[Getting Runtime Operator Input](#)

## Setting Variables

You can set a Variable in Plant SCADA within a Command field, an Expression field, or in a Cicode Function, by

using the mathematical 'equals' sign (=) assignment operator. The value on the right is assigned (set) to the variable on the left, as shown in the following Cicode example :

```
<VAR_TAG> = Val;
```

Where:

- <VAR\_TAG> is the name of the variable, and
- Val is the value being assigned to the variable.

## Examples

To set a digital variable (named BIT\_1) to ON (1), use the command:

```
BIT_1 = 1;
```

To set a digital variable (named BIT\_1) to OFF (0), use the command:

```
BIT_1 = 0;
```

To set a digital variable (named B1\_PUMP\_101\_M) to ON (1), use the command:

```
B1_PUMP_101_M = 1;
```

To set a digital variable (named B1\_PUMP\_101\_M) to OFF (0), use the command:

```
B1_PUMP_101_M = 0;
```

To set an analog variable (named B1\_TIC\_101\_SP) to a value of ten (10), use the command:

```
B1_TIC_101_SP = 10;
```

You can copy a variable to another by assigning (setting) the value of a variable to the value of another variable, for example:

```
B1_PUMP_101_COUNT = B1_PUMP_101_CLIMIT;
```

The value of B1\_PUMP\_101\_COUNT is set to the value of B1\_PUMP\_101\_CLIMIT only when that command is issued.

---

**Note:** The value of B1\_PUMP\_101\_CLIMIT could change immediately after, but B1\_PUMP\_101\_COUNT remains unchanged and storing the original value, until this command is issued again.

---

## See Also

[Using Cicode Commands](#)

## Performing Calculations

Mathematical calculations can be performed between variables in a Cicode statement. For example:

```
B1_TIC_101_SP = B1_TIC_101_PV + B1_TIC_102_PV - 100;
```

When this command is executed, the variable B1\_TIC\_101\_SP is set to a value that is the sum of variables B1\_TIC\_101\_PV and B1\_TIC\_102\_PV minus 100.

## See Also

[Using Cicode Commands](#)

## Using Multiple Command Statements

A single statement in a Cicode command usually performs a single task. When the Plant SCADA runtime system is in operation, the statement executes whenever the command is requested. For example, if the statement is linked to a keyboard command, the task is performed when an operator presses the keyboard key defined as that command.

To perform several tasks at the same time, you combine statements in a command property:

```
B1_PUMP_101_COUNT = B1_PUMP_101_CLIMIT;  
BATCH_NAME = "Bread";  
B1_TIC_101_SP = 10;
```

The example above uses three statements, separated by semi-colons ( ; ). The first statement sets the variable B1\_PUMP\_101\_COUNT to the value of the variable B1\_PUMP\_101\_CLIMIT; the second statement sets the variable BATCH\_NAME to the string "Bread"; and the third statement sets the variable B1\_TIC\_101\_SP to 10. Each statement is executed in order.

---

**Note:** You should separate each statement in a command with a semicolon (;). However, there are exceptions where this is not required. See [End of Line Markers](#).

The number of statements you can enter in a command property is limited only by the size of the field. However, for clarity, don't use too many statements; enter the statements into an Include File or write a Cicode Function. You then refer to the include file or call the function in the command property field.

## See Also

[Using Cicode Commands](#)

## Using Include (Text) Files

There is a maximum number of characters that you can type in a Command or Expression field (usually 128). If you need to include many commands (or expressions) in a property field, you can define a separate include file that contains the commands or expressions.

An include file is a separate and individual ASCII text file containing only one sequence of Plant SCADA commands or expressions that would otherwise be too long or complicated to type into the Command or Expression field within Plant SCADA. The include file name is entered instead, and the whole file is activated when called.

When you compile the project, the commands (or expressions) in the include file are substituted for the property field, just as if you had typed them directly into the field.

Use a text editor such as Notepad to create the text file.

Enter the name of the include file (either upper- or lower case) in the property, in the following format:

```
@<filename>
```

where *<filename>* is any valid DOS file name. Be aware that the bracket characters (< >) are part of the syntax.

You can use include files with many properties (except record names), but they are commonly used for commands and expressions, for example:

- Key sequence: F5 ENTER
- Command: @<setvars.cii>

In the above example, the `setvars.cii` include file would contain commands to be substituted for the Command property when you compile your project, for example:

```
PV12 = 10;  
PV22 = 20;  
PV13 = 15;  
PV23 = 59;  
PageDisplay("Mimic");
```

**Note:**

- Include files can not be used for genie properties.
- Do not confuse *include files* and *included projects*. Include files contain Plant SCADA commands and/or expressions and are used as substitutions in a Plant SCADA command or expression property field. Included projects are separate (usually smaller) Plant SCADA projects that can be included in another Plant SCADA project so that they appear together as one project.
- The include file name can contain a maximum of 64 characters, or 253 characters including a path, and can consist of any characters other than the semi-colon (;) or the single quote('). There is no need to include the `.cii` extension, but if the file is not in the project directory, you need to enter the full path to the file. If the file is not in the project directory, it will not be backed up with the Backup facility.
- If modifying an Include file with the Cicode Editor, when you save your changes a `.ci` file extension will be appended to the file name. Change this to a `.cii` file extension in Windows Explorer.

## See Also

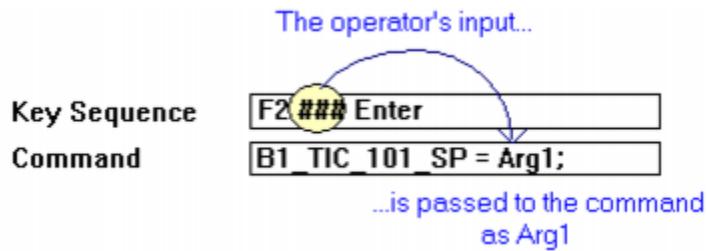
[Using Cicode Commands](#)

## Getting Runtime Operator Input

You can define a keyboard command as a key sequence, to perform a specific task each time the key sequence is pressed, for example:

- Key sequence: F2 ENTER
- Command: B1\_TIC\_101\_SP = 10;

A key sequence can include provision for the operator to enter data. In the following example, the operator can set the value of the variable B1\_TIC\_101\_SP:



The operator sends out the command by pressing the F2 key, up to three characters, and the Enter key. The three character sequence (identified by the three hash (#) characters) is called an *argument*. The argument is passed into the command (as Arg1) when the command is completed (when the operator presses the Enter key).

The operator might type:



The value 123 is passed to the command, and B1\_TIC\_101\_SP is set to 123.

It is recommended that you use a specific key (for example, Enter) to signal the end of a key sequence. If, for example, you use the key sequence F2 ####, the operator **needs to** enter 4 characters for the command to be executed - Plant SCADA waits for the fourth character. But if you use F2 #### Enter, the operator can enter between one and four characters as necessary. The command executes as soon as the Enter key is pressed.

To use more than one argument in a command, separate the arguments with commas ( , ):

- Key sequence: F2 ####,## Enter
- Command: B1\_TIC\_101\_SP = Arg1; B1\_TIC\_101\_PV = Arg2;

To set both variables, the operator can type:



The values 123 and 18 are passed to the command. B1\_TIC\_101\_SP is set to 123 and B1\_TIC\_101\_PV is set to 18.

## See Also

[Using Cicode Commands](#)

## Using Cicode Expressions

Cicode expressions are the basic elements of the Cicode language. An expression can be a constant, the value of a variable tag, or the result of a complex equation. You can use expressions to display and log data for monitoring and analysis, and to trigger various elements in your system, such as alarms, events, reports, and data logging.

You can enter a Cicode expression in any Plant SCADA editor form or graphic object that contains an expression property. Unlike a command, an expression does not execute a specific task - it is evaluated. The evaluation process returns a value that you can use to display information on the screen (for example, as a bar graph) or to make decisions. The following expression returns a result of 12:

- Numeric expression: 8 + 4

In the above example, the value of the expression is a constant (12) because the elements of the expression are constants (8 and 4).

## Displaying Data Using Expressions

In the following example, the value of the expression is the value of the variable B1\_TIC\_101\_PV. As its value changes, the value of the expression also changes. You can use this expression to display a number on a graphics page.

- Numeric expression: B1\_TIC\_101\_PV

As the expression changes, the number also changes.

Expressions can also include mathematical calculations. For example, you can add two variables together and display the combined total:

- Numeric expression: B1\_TIC\_101\_PV + B1\_TIC\_102\_PV

In this case, the value of the expression is the combined total. As the value of one variable (or both variables) changes, the value of the expression changes.

## Decision-Making

Some expressions return only one of two logical values, either TRUE(1) or FALSE(0). You can use these expressions to make decisions, and to perform one of two actions, depending on whether the return value is TRUE or FALSE. For example, you can configure a text object with appearance as follows:

- On text when: B1\_PUMP\_102\_CMD
- ON text: Pump Running
- OFF text: "Pump Stopped"

In this example, if B1\_PUMP\_102\_CMD is a digital tag (variable), it can only exist in one of two states (0 or 1). When your system is running and the value of B1\_PUMP\_102\_CMD changes to 1, the expression returns TRUE and the message "Pump Running" is displayed. When the value changes to 0, the expression returns FALSE and the message "Pump Stopped" is displayed.

## Logging Expression Data

You can log the value of an expression to a file for trending, by defining it as a trend tag:

Trend Tag Name	B1_TIC
Expression	B1_TIC_101_PV + B1_TIC_102_PV
File Name	[log]:B1_TIC

When the system is running, the value of the expression B1\_TIC\_101\_PV + B1\_TIC\_102\_PV is logged to the file [log]:B1\_TIC.

## Triggering Events Using Expressions

Logical expressions - those that return either TRUE (1) or FALSE (0) -can be used as **triggers**.

For example, you might need to log the expression in the above example only when an associated pump is running.

Trend Tag Name	B1_TIC
Expression	B1_TIC_101_PV + B1_TIC_102_PV
File Name	[log]:B1_TIC
Trigger	B1_PUMP_101_CMD

In this example, the trigger is the expression B1\_PUMP\_101\_CMD (a digital variable tag). If the pump is ON, the result of the trigger is TRUE, and the value of the expression (B1\_TIC\_101\_PV + B1\_TIC\_102\_PV) is logged. If the pump is OFF, the result is FALSE, and logging ceases.

## Using Cicode Functions

A Cicode function can perform more complex tasks than a simple command or expression allows. Functions give you access to Plant SCADA graphics pages, alarms, trend data, and so on.

Plant SCADA has several hundred built-in functions that display pages, acknowledge alarms, make calculations, and so on. You can also write your own functions to meet your specific needs.

The following topics describe ways you can use Cicode functions.

- [Calling Functions from Commands and Expressions](#)
- [Triggering Functions via Runtime Operator Input](#)
- [Evaluating Functions](#)
- [Combining Functions with Other Statements](#)
- [Passing Data to Functions \(Arguments\)](#)
- [Using String Arguments](#)
- [String Assignment](#)
- [Using the Caret Escape Sequence Character](#)
- [Using Multiple Arguments](#)
- [Using Numeric Arguments](#)
- [Using Variable Arguments](#)
- [Using Operator Input in Functions](#)
- [Returning Data from Functions.](#)

To access a full list of available Cicode functions, see [Cicode Function Categories](#).

## See Also

[Working with Commonly Used Functions](#)

[Writing Functions](#)

## Calling Functions from Commands and Expressions

You can call a function by entering its name in any command or expression property. The syntax is as follows:

Command	FunctionName ( Arg1, Arg2, ... );
---------	-----------------------------------

Where:

- *FunctionName* is the name of the function
- *Arg1, Arg2, ...* are the arguments you pass to the function

## See Also

[Using Cicode Functions](#)

### Triggering Functions via Runtime Operator Input

In the following command, the PageNext() function displays the next graphics page when the Page Down keyboard key is pressed by the Runtime operator.

<b>Key Sequence</b>	Page_Down
<b>Command</b>	PageNext();

## See Also

[Using Cicode Functions](#)

### Evaluating Functions

You can use a function in any expression. For example, the AlarmActive() function returns TRUE (1) if any alarms are active, and FALSE (0) if no alarms are active. In the following text object, either "Alarms Active" or "No Alarms Active" is displayed, depending on the return value of the expression.

<b>ON text when</b>	AlarmActive(0)
<b>ON Text</b>	"Alarms Active"
<b>OFF Text</b>	"No Alarms Active"

**Note:** Functions return a value that indicates the success of the function, or provides information on an error that has occurred. In many cases (for example, when used in a command) the return value can be ignored. You need to use the parentheses () in the function name, even if the function uses no arguments. Function names are not case-sensitive: PageNext(), pagenext() and PAGENEXT() call the same function.

## See Also

[Using Cicode Functions](#)

### Combining Functions with Other Statements

In expressions and commands you can use functions alone or in combination with other functions, operators, and so on.

The following example uses three statements:

<b>Command</b>	Report("Shift"); B1_TIC_101_PV = 10; PageDisplay("Boiler 1")
----------------	---

Each statement is executed in order. The "Shift" report is started first, the variable B1\_TIC\_101\_PV is set to 10

next, and finally, the "Boiler 1" page is displayed.

Functions combine with operators and conditional executors to give you specific control over your processes, for example, you can test for abnormal operating conditions and act on them.

## See Also

[Using Cicode Functions](#)

### Passing Data to Functions (Arguments)

The parentheses ( ) in the function name identify the statement as a function and enclose its arguments. Arguments are the values or variables that are passed into the function when it executes.

**Note:** Some functions, such as PageNext(), have no arguments. However you need to include the parentheses ( ) or Plant SCADA will not recognize that it is a function, and an error could result when the project is compiled.

## See Also

[Using Cicode Functions](#)

### Using String Arguments

Functions can require several arguments or, as in the following example, a single argument:

Command	PageDisplay("Boiler 1");
---------	--------------------------

This function displays the graphics page called "Boiler 1". Be aware that when you pass a string to a function, you need to always enclose the string in double quotes.

You can use the PageDisplay() function to display any graphics page in your system - in each case, only the argument changes. For example, the following command displays the graphics page "Boiler 2":

Command	PageDisplay("Boiler 2");
---------	--------------------------

You can use the Report() function to run a report (for example, the "Shift" report) when the command executes:

Command	Report("Shift");
---------	------------------

The following example uses the Prompt() function to display the message "Press F1 for Help" on the screen when the command executes:

Command	Prompt("Press F1 for Help");
---------	------------------------------

## See Also

[Using Cicode Functions](#)

## String Assignment

You can assign string variables in commands. For example, if BATCH\_NAME is a variable tag defined as a string data type, you can use the following command to set the tag to the value "Bread":

```
BATCH_NAME = "Bread";
```

**Note:** You need to enclose a string in double quotation marks ( " ).

## See Also

[Using Cicode Functions](#)

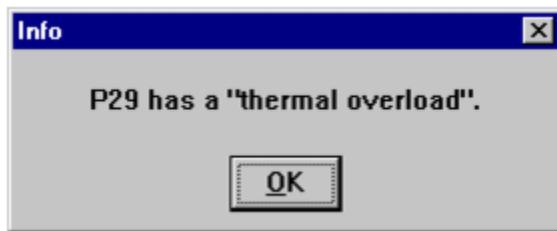
## Using the Caret Escape Sequence Character

The caret character ( ^ ) signifies a special instruction in Cicode, called an escape sequence, primarily used in the formatting of text strings. Escape sequences include formatting instructions such as new line, form feed, carriage return, backspace, horizontal and vertical tab-spaces, single and double quote characters, the caret character, and hexadecimal numbers.

Strings are commonly represented in Cicode between double quote characters ( " ) known as delimiters. If you want the string to contain a double quote character itself as part of the string, you need to precede the double quote character with the caret character ( ^" ) so that Cicode doesn't interpret the double quote in the string as the delimiter indicating the end of the string. The caret character is interpreted as a special instruction, and together with the characters immediately following it, are treated as an escape sequence instruction. See the section [Converting and Formatting Cicode Variables](#) for the list of escape sequences used in Cicode.

In the following Cicode example, both of these message functions will display the following message.

```
Message("Info", "P29 has a ^"thermal overload^".", 0);
sCurrentAlmText = "Thermal Overload";
Message("Info", "P29 has a ^""+sCurrentAlmText+"^".", 0);
```



## See Also

[Using Cicode Functions](#)

## Using Multiple Arguments

Some functions require several arguments. You need to list arguments between the parentheses, and separate each argument with a comma ( , ) as in the following example:

Command	Login("Manager", "ABC");
---------	--------------------------

The order of the arguments affects the operation of any function. The `Login()` function logs a user into your runtime system. The first argument ("Manager") indicates the name of the user, and the second argument ("ABC" ) is the user's password. If you reverse the order of the arguments, the function would attempt to login a user called "ABC" - if a user by this name does not exist, an error message displays.

## See Also

[Using Cicode Functions](#)

### Using Numeric Arguments

You can pass **numbers** (integers and floating point numbers) directly to a function, for example:

Command	AlarmAck(2, 35);
---------	------------------

## See Also

[Using Cicode Functions](#)

### Using Variable Arguments

When variables (such as real-time data) are used as arguments, the value of the variable is passed, not the variable itself. The following example uses the `DspStr()` function to display the value of a process variable at AN25:

Command	DspStr(25, "TextFont", B1_TIC_101_PV);
---------	--

In this instance, the **value** of `B1_TIC_101_PV` displays. If it is a real-time variable, the number that displays depends on its value at the time.

**Note:** If you use double quotes around variables, for example, "B1\_TIC\_101\_PV", the text string `B1_TIC_101_PV` displays, rather than the value of the variable.

## See Also

[Using Cicode Functions](#)

### Using Operator Input in Functions

You can pass operator input to functions at runtime. For example, you can define a System Keyboard Command to let the operator select a page:

Key Sequence	F10 ##### Enter
Command	PageDisplay(Arg1);

When the command executes, the page name is passed to the function as Arg1. The operator can then display any page, for example:



## See Also

[Using Cicode Functions](#)

### Returning Data from Functions

Functions return data to the calling statement (a command or expression). Some functions simply return a value that indicates whether the function was successful. For example, both the PageNext() and PageDisplay() functions return 0 (zero) if the page displays successfully, otherwise they return an error number. For a large number of simple applications, you can ignore this return value.

Some functions return data that you can use in an expression or command. For example, the Date() function returns the current date as a string. To display the current date on a graphics page, use the following expression in a text object display value property:

Numeric expression	Date();
--------------------	---------

The following example shows an entry command event for a graphics page, using a combination of two functions. The FullName() function returns the name of the user who is currently logged in to the run-time system, passing this name to the calling function, Prompt(). When the page is opened, a welcome message displays in the prompt line.

On page entry	Prompt("Hello, " + FullName())
---------------	--------------------------------

For example, if the current user is John Citizen, the message "Hello, John Citizen" displays.

## See Also

[Using Cicode Functions](#)

### Referencing an Object Using a Name at Runtime

Use the Cicode functions [DspGetAnFromName](#), [DspGetMetadataFromName](#), [DspGetAnFromNameRelative](#) and [DspGetMetadataFromNameRelative](#) to reference objects on a page, in a genie, group and symbol.

It is important to keep the following points in mind when referencing objects:

- When referencing an object within a genie, group or symbol you need to use relative path syntax. This means you can search "up" or "down" the objects within the genie, symbol and/or group.
- The name is always relative to the object you are referencing from.
- If referencing an object within a group it will search only in the group the object belongs to.
- If referencing an object at the group level it will search for the object across the defined paths.

### Using the Syntax

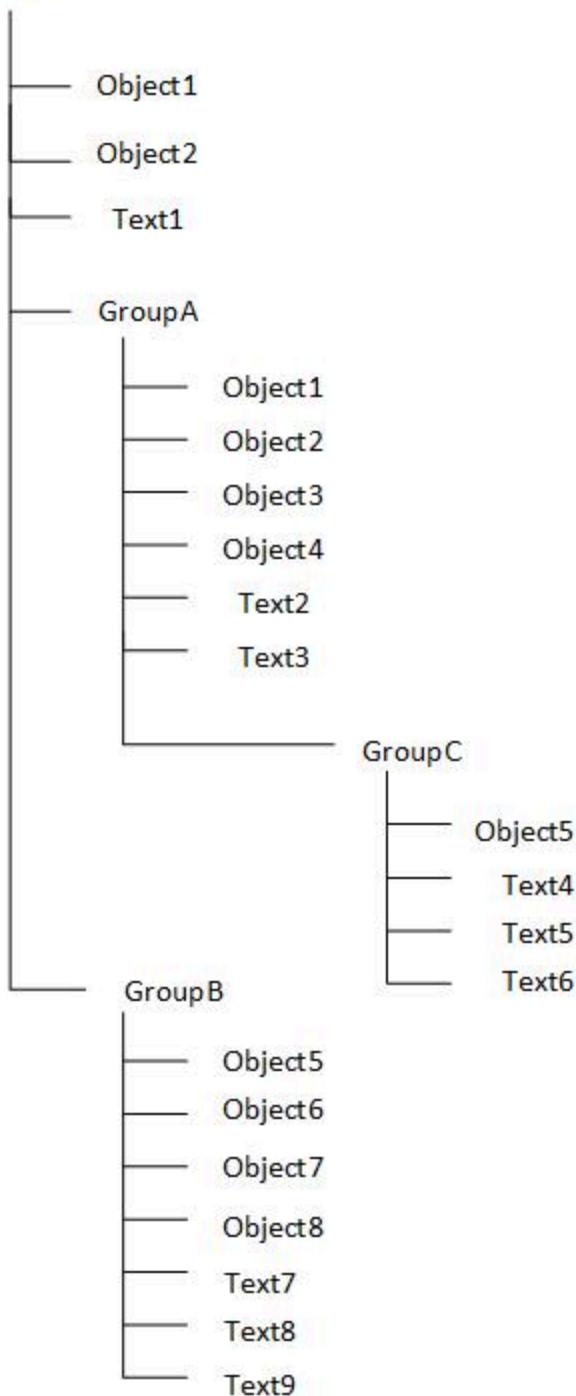
- "Name" – Used at page level. This is a fully qualified name.

- ".Name" - References itself, either in a group, genie or symbol.
- "..\Name" – References objects within a group.

## Examples

The diagram below includes objects and text on the page, and objects and text within groups A, B and C. A tree-view hierarchy has been used to illustrate how referencing can be applied.

## Page



Referenced Object	Syntax	Description
To reference "Object1" from Object2 at the root or page level	DspGetAnFromName("Object1")	Object1 is a fully qualified name. Searches at page level for this object.
To reference "Object1" from	DspGetMetadataFromName("Objec")	Object1 is a fully qualified name.

Referenced Object	Syntax	Description
Object2 at the root or page level.	t1","Pump")	Pump is the name of the metadata defined in Object1. Searches the page level for Object1 and the metadata name.
To reference "Object1" in GroupA from Object2 from the root level	DspGetAnFromName("GroupA.Object1")	"GroupA.Object2" is a fully qualified name. From the root (page) level search for Object1 within GroupA.
To reference "Object5" in GroupC from Object2 at the root level	DspGetAnFromName("GroupA.GroupC.Object5")	"GroupA.GroupC.Object5" is a fully qualified name. From the root (page)level search for Object5 within GroupC and GroupA.
To reference "Object1" in GroupA from Object4	DspGetAnFromName(".Object1")	".Object1". This means search for Object1 in the current group.
To reference "Object2" in GroupA from Object8 in Group B	DspGetAnFromName(..\..\GroupA.Object2")	"..\..\GroupA.Object2" Navigate two levels up the hierarchy, which will be root (page) level, and then search for GroupA and within Group A, Object2.
To reference "Object2" in GroupA from Text8 in Group B	DspGetAnFromNameRelative(639, "..\..\GroupA.Object2")	From AN 639 and "..\..\GroupA.Object2" navigate two levels up the hierarchy and search for Group A and within Group A, Object2. <b>Note:</b> 639 is the animation number for Text8 once pasted on the graphics page within GroupB.
To reference "Object2" in GroupA from Text8 in Group B	DspGetMetadataFromName(..\..\GroupA.Object2", "Meter")	"..\..\GroupA.Object2" Navigate two levels up the hierarchy which will be from the root (page)level, and then search for GroupA and within Group A, Object2, where the name of the metadata is "Meter".
To reference "Object1" at the root level from Object5 in GroupC	DspGetAnFromName(..\..\..\Object1")	"..\..\..\Object1" Navigate three levels up the hierarchy which will be from the root (page)level, and then search for Object1.
To reference "Object1" at the root level from Text5 in GroupC	DspGetMetadataFromNameRelative(625, "..\..\..\Object1", "Pump")	From AN 625 navigate three levels up and from the root level search for Object1 where the name of the

Referenced Object	Syntax	Description
		<p>metadata is Pump.</p> <p><b>Note:</b> 625 is the animation number for Text6 once pasted on the page within GroupC.</p>

**Note:** When referencing from an object within a group, only those objects in the group the object belongs to will be searched. For example: If you reference Object1 in Group A from Object7 in GroupB using the syntax "..\Object1", an error will be returned, as no Object1 exists in GroupB. The correct syntax to use is "..\..\GroupA.Object1".

---

## Working with Commonly Used Functions

Cicode has many functions that perform a variety of tasks. Many of these are used for building complex Plant SCADA systems. The functions you will often use are divided into six categories:

### Alarm Functions

You can use alarm functions to display alarms and their related alarm help pages, and to acknowledge, disable, and enable alarms. You can assign a privilege to each command that uses an alarm function, so that only an operator with the appropriate privilege can perform these commands. However, you should assign privileges to commands only if you have not assigned privileges to individual alarms.

- [AlarmAck](#): Acknowledges an alarm. The alarm where the cursor is positioned (when the command is executed) is acknowledged. You can also use this function to acknowledge multiple alarms.
- [AlarmComment](#): Adds a comment to the alarm summary entry at run time. The comment is added to the alarm where the cursor is positioned when the command is executed. A keyboard argument passes the comment into the function. Verify that the length of the comment does not exceed the length of the argument, or an error results.
- [AlarmDisable](#): Disables an alarm. The alarm where the cursor is positioned (when the command is executed) is disabled. You can also use this function to disable multiple alarms.
- [AlarmEnable](#): Enables an alarm. The alarm where the cursor is positioned (when the command is executed) is enabled. You can also use this function to enable multiple alarms.
- [AlarmHelp](#): Displays an alarm help page for the alarm. Each alarm in your system can have an associated help page. The help page for the alarm at the position of the cursor (when the command is executed) is displayed.

### Page Functions

With the page functions, you can display your graphics pages and the standard alarm pages.

---

**Note:** The following page functions are not supported in the server process in a multiprocessor environment. Calling page functions from the server process results in a hardware alarm being raised.

---

- [PageAlarm](#): Displays current alarms on the alarm page configured in the project.
- [PageDisabled](#): Displays disabled alarms on the alarm page configured in the project.

- **PageDisplay:** Displays a new page on the screen. The Page name or number is required as an argument. (Use the PageLast() function to go back to the last page - the page that this new page replaced).
- **PageFile:** Displays a file on the file page configured in the project.
- **PageGoto:** Displays a new page on the screen. This function is similar to the PageDisplay() function, except that if PageLast() is called, it does not return to the last page.
- **PageHardware:** Displays hardware alarms on the alarm page configured in the project.
- **PageLast:** Displays the graphics page that was displayed before the current one. You can use this function to 'step back' through the last ten pages.
- **PageNext:** Displays the next graphics page (defined in the Next Page property of the Pages form).
- **PagePrev:** Displays the previous graphics page (defined in the Prev Page property of the Pages form).
- **PageSOE:** Displays a category of sequence of events (SOE) entries on the SOE page.
- **PageSummary:** Displays summary alarm information on the alarm page configured in the project.
- **PageTrend:** Displays a standard trend page.

## Keyboard Functions

Keyboard functions control the processing of keyboard entries and the movement of the keyboard cursor on the graphics page.

- **KeyBs:** Backspaces (removes) the last key from the key command line. Use this function with a 'Hotkey' command. It is normally used to erase keyboard characters during runtime command input.
- **KeyDown:** Moves the cursor down the page to the closest animation point number (AN).
- **KeyLeft:** Moves the cursor left (across the page) to the closest animation point number (AN).
- **KeyRight:** Moves the cursor right (across the page) to the closest animation point number (AN).
- **KeyUp:** Moves the cursor up the page to the closest animation point number (AN).

## Report Functions

To run a report by operator action, use the following function:

- **Report:** Runs the report on the report server.

## Time/date Functions

The following functions return the current date and time:

- **Date:** Returns the current date as a string.
- **Time:** Returns the current time as a string.

## Miscellaneous Functions

- **Beep:** Beeps the speaker on the Plant SCADA computer.
- **FullName:** Returns the full name of the user who is currently logged in to the system.

- InfoForm: Displays the animation information form. This form displays the real-time data that is controlling the current animation.
- Login: Allows a user access to the Plant SCADA system.
- LoginForm: Displays a dialog box to allow a user to log in to the system.
- Logout: Logs the current user out of the Plant SCADA system.
- Name: Returns the user name of the user who is currently logged in to the system.
- Prompt: Displays a message on the screen. The message **String** is supplied as an argument to the function.
- Shutdown: Terminates Plant SCADA. Use this function, or the ShutdownForm() function, to shut down your system. Otherwise buffered data may be lost.
- ShutdownForm: Displays a dialog box to allow a user to shut down your Plant SCADA system.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## Writing Functions

Plant SCADA is supplied with over 600 built-in functions. One of these functions (or several functions in combination) can usually perform the required tasks in your system. However, where system functionality cannot be achieved with built-in functions, you can write your own functions.

A Cicode function is a small program: a collection of statements, variables, operators, conditional executors, and other functions.

While it is not necessary to be an experienced programmer to write simple Cicode functions, it is strongly recommended not to attempt to write large, complex functions unless you are familiar with computer programming, and have experience with Cicode. Functions are equivalent to the subroutines of BASIC and assembly language, and the subroutines and functions used in Pascal and C.

Cicode functions can have many purposes. Quite often, functions are used to store a common set of commands or statements that would otherwise require repetitious typing and messy command or expression fields.

Some functions are simple, created to avoid a long command or expression. For example, the following command increments the variable tag COUNTER:

<b>Command</b>	IF COUNTER < 100 THEN COUNTER = COUNTER + 1; ELSE COUNTER = 0; END;
----------------	--

This command would be easier to use (and re-use) if it was written as a function that can be called in the command:

<b>Command</b>	IncCounter ( );
----------------	-----------------

To be able to use the function like this, you need to write it in a Cicode file, and declare it with the FUNCTION keyword:

```
FUNCTION
IncCounter ( )
IF COUNTER < 100 THEN
COUNTER = COUNTER + 1;
```

```
ELSE
COUNTER = 0;
END
END
```

Be aware that the indented code is identical in functionality to the long command above.

By placing the command code inside a function, and using the function name in the command field as in the previous example, this function need only to be typed once. It can then be called any number of times, from anywhere in Plant SCADA that requires this functionality. Because the code exists in the one location, rather than repeated wherever needed (in potentially many places), it can be easily maintained (altered if necessary).

**Note:** This is designed specifically for editing and debugging Cicode functions.

## See Also

[Cicode Function Structure](#)

[Using Cicode Files](#)

## Cicode Function Structure

A function in Cicode can be described as a collection or list of sequential statements that Plant SCADA can perform (execute) in the logical order that they exist within the function.

A Cicode function starts with the FUNCTION statement and finishes with the END statement. Every statement that lies between the FUNCTION and END statements, will be executed by the function, when called to do so.

A typical Cicode function is structured like the following example:

```
FUNCTION
FunctionName ( )
    ! The exclamation point indicates that the rest of this line contains a comment.
    ! Further Cicode statements go here, between the function name and the END.
END
```

The line immediately following the FUNCTION statement, contains the name of the function, which is used to identify the function to Plant SCADA. This name is referred to when the function is called upon (called) to be executed (perform the statements it contains) by some other event, action, or function in Plant SCADA.

**Note:** Functions can contain statements that call other functions. These functions are then executed before returning to the rest of the statements within the calling function.

The function name has to end with parentheses ( ), which may or may not contain one or more arguments required by the function. Arguments are explained in the section titled [Function Argument Structure](#).

Every line between the function name line and the END statement line contain the statements that will be executed when the function is called in Plant SCADA. These statements are executed one at a time in logical order from top to bottom within the function. For details about function structure, see [Formatting Functions](#). For details about Cicode function syntax, see [Following Cicode Syntax](#).

For details about using comments in Cicode and in Cicode functions, see [Using Comments in Cicode](#).

## See Also

[Writing Functions](#)

## Writing Groups of Functions

To perform complex tasks you need careful design. Large, complex functions are not only more difficult to understand and debug than simple functions, but they can also hide tasks that are common to other activities.

Cicode functions allow a modular approach - complex tasks can be organized into small functions, each with a single, clear purpose. These small functions can then be called by other functions, or called directly in commands and expressions. In fact, any function can call - and be called by - any other function.

For example, you might need to write a set of functions for handling alarms. To perform any action on an alarm, you first need to know which alarm. You would identify the alarm in a separate function, and call this function from the other functions.

## See Also

[Writing Functions](#)

## Cicode Function Libraries

Cicode functions are stored within Cicode files. You can use a separate file for each stand-alone function, or group several functions together into a common file. For easy maintenance, store functions that perform related tasks in the same file - for example, store functions that act on alarm data in an **Alarms.CI** file.

---

**Note:** Every Cicode file in your project directory will be included when you compile your project.

---

## See Also

[Writing Functions](#)

## Creating a Function Outline

First, define the purpose of the function group, and create an outline of the tasks to be performed. The following example shows an outline for a group of functions that change the threshold values of analog alarms during runtime. The outline describes the workings of the function group, and is written in **pseudocode** (also called Program Design Language).

```
/*
This file contains functions to allow the operator to make runtime
changes to Analog Alarm thresholds.
This file has 4 functions. The master function calls the other
functions.
ChangeAnalogAlarmThresholds ( )
This calls in turn:
1:GetVariableTag ( )
Argument: cursor position
Displays threshold values in prompt line
Return: success or error code
*/
```

## See Also

[Writing Functions](#)

## Pseudocode

Pseudocode is a Cicode comment, enclosed between the comment markers /\* and \*/, and is ignored by the compiler. With pseudocode, you can get the logic of the function correct in a more readable structure, before you write it in Cicode syntax, leaving the pseudocode within the finished code as comments.

It is good practice to use comments as file headers at the start of each Cicode file, to describe the functions in the file - their common purpose, a broad description of how they achieve that purpose, special conditions for using them, and so on. You can also use the header to record maintenance details on the file, such as its version number and date of revision. For example:

```
/*
** FILE: Recipe Download.Ci
**
** AUTHOR: AJ Smith
**
** DATE: March 2008
**
** REVISION: 1.0 for Plant SCADA v7.1
**
** This file contains functions to allow the operator to load the
** recipe data from the SQL server to the PLC.
*/
```

Following the file header are the functions in series:

```
/*
** Main function
*/
FUNCTION
RecipeDownload ( )
! {body of function}
!
END
/*
** Function to open the SQL connection.
*/
FUNCTION
RecipeConnectSQL ( )
! {body of function}
!
END
! (and so on)
```

## See Also

[Writing Functions](#)

## Considering Tag Value Quality in Cicode

When using tag values in Cicode, it is important to consider the quality of the tag and how it is determined. Just as the quality of a displayed value should be presented to a client user, the quality of a value also needs to be reflected in Cicode calculations.

For example, the type of driver used to communicate with an I/O device will greatly impact how tag value quality is determined.

- Tags from traditional drivers may be set to a bad quality when the I/O device is not contactable (or for other errors in the driver commands).
- Tags from devices using the Driver Runtime Interface (DRI) can have their quality set individually and could be set to a bad quality for reasons specific to an I/O point.

For more information on DRI-based drivers, see [Retrieving Time-stamped Data from I/O Devices](#).

Best practice is to treat quality handling like error handling, and code for scenarios where it is required.

## I/O Quality

The quality of a tag value is essentially propagated from the quality of the I/O point from which it is sourced.

- In traditional protocols, such as Modbus, quality is driven by a single error code provided in a command response or by the field device being offline. This type of protocol does not provide the ability to nominate a quality for individual I/O points. This still holds true for Modbus over TCP as well (as used by the Plant SCADA's MODNET driver).

As the response error codes are typically a result of an invalid request, these types of issues are normally addressed in commissioning and are usually not encountered during operation. However, it is still possible to lose contact with a field device at any time which will result in all tags related to that device going into bad quality.

- In more advanced protocols, such as OPC, quality is independently applied to each specific I/O point. To accommodate this, and to also cater for other advanced functionality such as unsolicited responses from the field, a newer generation of Plant SCADA drivers was developed to use the Driver Runtime Interface (DRI).

When a DRI driver is notified from the field of an update (value, quality and timestamp), this is immediately reflected in the corresponding tag. An example of a change in quality due to operational issues could be a faulty sensor. This increases the chances of a tag not being in good quality and the need for this scenario to be covered when using tag values in Cicode.

A traditional driver uses a chained sequence of request/response transactions that result in a tag update waiting on the reply back from a field device. As an I/O device is only marked online when the corresponding field device can be contacted, reverting from standby back to primary will never result in intermediate bad quality values from the switch.

However, a DRI driver uses a chained sequence of subscribe/publish transactions allowing the driver to update a tag at any time. When reverting from standby to a primary that has just returned online, it will be up to the driver to immediately provide tag updates for a new subscription without waiting to receive values from the field. If so, these values will be marked with a quality to indicate as such and will be followed by a good quality update when the field value is received.

If the driver is receiving updates from an interim gateway style device, such as an OPC Server, it will also be able

to provide similar updates.

## Implications for Cicode

Your Cicode should handle values that are not good quality in the same way it handles other error cases.

Tag extensions can be used to select the quality component of a tag (for example, TagA.Q), and this can then be checked using the [QualityGetPart](#) Cicode function.

If the quality of a tag needs to be referenced multiple times in a Cicode function, it is best practice to assign it to a variable when first required and then keep using that variable. When a tag is assigned to a Cicode variable (for example, var = TagA), that variable will also take the quality of the tag.

Further, the quality will continue to propagate through to any additional variables that are assigned an expression containing the first variable. The quality component of these variables can be extracted using the [VariableQuality](#) Cicode function, and checked using [QualityGetPart](#) as described above.

Note, however, that if a Cicode variable is assigned the value extension of a tag (for example, var = TagA.V), then the variable will always have good quality.

## Examples

### Example 1

Your Cicode needs to handle the possibility of a tag update occurring after it has determined the quality of a tag. If quality changes due to a tag update, your Cicode may proceed using an altered quality value. This can occur because of the way the Cicode execution system operates in a multi-thread environment.

This could even happen when the tag quality check and tag usage are on the same line, for example:

```
IF QualityIsGood(MyTag.Q) AND MyTag > 100 THEN
```

The easiest way to avoid this is to only use the tag when assigning to a Cicode variable, and only use the Cicode variable from then on.

In the case of the example above, a better approach would be to assign "MyTag" to a local tag at the start of the function and use this instead.

```
Int localMyTag
IF QualityIsGood(VariableQuality(localMyTag) AND localMyTag > 100 THEN
```

```
// Following example calculates flow rate based on a PLC tag 'MyDiffPress'
// and then writes value to a memory mode Disk PLC tag 'MyFlowRate'
// Returns 1 for error when BQ on the PLC tag and forces MyFlowRate to Override mode.
INT
FUNCTION
UpdateMyFlowRate()

    INT FlowFactor = 5;
    INT SpecificGravity = 1;
    //Copy the variable tag MyDiffPress to a local variable
    INT DiffPress = MyDiffPress;
    INT FlowRate = 0;
    INT Error = 0;

    IF QualityIsGood(VariableQuality(DiffPress)) THEN
        //Do this instead of using QualityIsGood(VariableQuality(MyDiffPress))
```

```
FlowRate = FlowFactor * Sqrt(DiffPress/SpecificGravity);
MyFlowRate.OverrideMode = 0;
MyFlowRate = FlowRate;
ELSE
    Error = 1;
    MyFlowRate = 0;
    MyFlowRate.OverrideMode = 4;
END
RETURN Error;
END
```

### Example 2

The following example checks the threshold of a tag value.

```
INT
FUNCTION
IsThresholdExceeded(STRING Tag1, INT Threshold)
    INT exceeded = FALSE;
    INT myVar1 = TagReadEx(MyTag1);

    IF QualityIsGood(VariableQuality(myVar1) AND myVar1 > Threshold THEN exceeded = TRUE;
    END
    RETURN exceeded;
END
```

### Example 3

The following example logs tag values to a file.

```
FUNCTION
LogMyValue(STRING MyTag)
    INT myVar1 = TagReadEx(MyTag);
    IF QualityIsGood(VariableQuality(myVar) THEN
        LogToMyFile(MyTag + " : " + IntToStr(MyTag));
    ELSE
        LogToMyFile(MyTag + " : "No Value");
    END
END
```

## Using Comments in Cicode

It is good programming practice to include comments in your Cicode files. Comments allow you to quickly understand how a function works next time you (or another designer) need to modify it.

The Cicode compiler recognizes the following single line, C style, and C++ style comments:

```
! A single line comment
WHILE DevNext ( hDev ) DO
Counter = Counter + 1 ; ! An in-line comment
END
/* A block comment is a C-style comment, and can
extend over several lines. Block comments need to
finish with a delimiter, but delimiters at the
start of each line are optional only. */
// A double-slash comment is a C++ style comment, for example:
Variable = 42; // This is a comment
```

Single line ( ! ) and C++ style ( // ) comments can have a line of their own, where they refer to the block of statements either before or after it. It is good practice to set a convention for these comments. These comments can also be on the same line as a statement, to explain that statement only. Any characters after the ! or // (until the end of the line) are ignored by the compiler.

Block (C style) comments begin with /\* and end with \*/. These C style comments need no punctuation between the delimiters.

## See Also

[Writing Functions](#)

### Using Comments for Debugging Functions

You can use comments to help with the debugging of your functions. You can use comments to temporarily have the compiler ignore blocks of statements by changing them to comments. C style and C++ style comments can be nested, for example.

```
FUNCTION
IncCounter ( )
IF COUNTER < 100 THEN
COUNTER = COUNTER + 1 ;
/* ELSE // Comment about statement
COUNTER = 0; // Another comment
*/
END
END
```

The complete ELSE condition of the IF conditional executor will be ignored (and not execute) so long as the block comment markers are used in this example.

---

**Note:** The inline ( // ) comments have no effect within the block ( /\* and \*/ ) comments (as the whole section is now one big comment), and should remain unchanged, so that when you do remove the block comments, the inline comments will become effective again.

---

## See Also

[Writing Functions](#)

### Tag Reference/TagReadEx Behavior in Cicode Expressions

The following table describes the tag reference and TagReadEx behavior in a Cicode expression if the quality of the tag is BAD:

Tag Reference / TagReadEx syntax	Error Mode/Citect.ini settings	Cicode Expression behavior
"Tag1"	ErrSet(0) [Code]HaltOnInvalidTagData = 0	Tag ref returns a BAD quality value, Cicode expression continues, Error is set.
TagReadEx("Tag1")	ErrSet(0) [Code]HaltOnError = 0	Function returns a BAD quality value, Cicode expression continues,

Tag Reference / TagReadEx syntax	Error Mode/Citect.ini settings	Cicode Expression behavior
		Error is set.
"Tag1"	ErrSet(0) [Code]HaltOnInvalidTagData = 1	Tag ref returns a BAD quality value, Cicode expression stops.
TagReadEx("Tag1")	ErrSet(0) [Code]HaltOnError = 1	Function returns a BAD quality value, Cicode expression stops.
"Tag1"	ErrSet(1)	Tag ref returns a BAD quality value, Cicode expression continues, Error is set.
TagReadEx("Tag1")	ErrSet(1)	Function returns a BAD quality value, Cicode expression continues, Error is set .
"Tag1.V"	ErrSet(0) or ErrSet(1)	Tag ref returns a GOOD quality value, Cicode expression continues, No error is set.
TagReadEx("Tag1.V")	ErrSet(0) or ErrSet(1)	Function returns a GOOD quality value, Cicode expression continues, No error is set.

## See Also

[Writing Functions](#)

[TagReadEx](#)

[Tag Functions](#)

## Following Cicode Syntax

Some programming languages have strict rules about how the code needs to be formatted, including the indenting and positioning of the code structure. Cicode has no indenting or positioning requirements, allowing you to design your own format - provided only that you follow the correct syntax order for each statement (see [Cicode Function Syntax](#)). However, it is a good idea to be consistent with your programming structure and layout, so that it can be easily read and understood.

For details about programming standards, see the section titled [Using Cicode Programming Standards](#), which includes sections on:

- Standards for constants, variable tags, and labels
- Standards variables: declaration, scope, and naming
- Standards for functions: naming , file headers, headers
- Formatting of: declarations, statements, expressions, and functions
- Use of comments

For information on problem solving, see the sections on [Modular Programming](#), [Defensive Programming](#), [Function Error Handling](#), or [Debugging Cicode](#).

The following is an example of a simple Cicode function:

```
/*
This function is called from a keyboard command. The operator
presses the key and enters the name of the page to be displayed. If
the page cannot be displayed, an error message is displayed at the
prompt AN.
*/
INT
FUNCTION
MyPageDisplay ( STRING sPage ) ! pass in the name of the page to be displayed
! declare a local integer to hold the results of the pagedisplay function
INT Status;
! call the page Cicode pagedisplay function and store the result
Status = PageDisplay ( sPage ) ;
! determine if the page display was successful
IF Status < > 0 THEN ! error was detected
! display an error message at the prompt AN
DspError ( "Cannot Display " + sPage ) ;
END
! return the status to the caller
RETURN Status;
END
```

The rules for formatting statements in Cicode functions are simple, and help the compiler in interpreting your code.

It is good practice to use white space to make your code more readable. In the example above, the code between the FUNCTION and END statements is indented, and the statement within the IF THEN conditional executor is further indented to make the conditions and actions clear. Develop a pattern of indentation - and stick to it. Extra blank lines in the code make it easier to read (and understand).

## See Also

[Writing Functions](#)

## Cicode Function Syntax

In the following function syntax example:

- Every placeholder shown inside arrow brackets ( <placeholder> ) should be replaced in any actual code with the value of the item that it describes. The arrow brackets and the word they contain should not be included in the statement, and are shown here only for your information.
- Statements shown between square brackets ( [ ] ) are optional. The square brackets should not be included in the statement, and are shown here only for your information.

Cicode functions have the following syntax:

```
[ <Scope> ]
[ <ReturnDataType> ]
FUNCTION
<FunctionName> ( <Arguments> )
```

```
<Statement> ;
<Statement> ;
<Statement> ;
RETURN <ReturnValue> ;
END
```

Where:

- <Scope> = Scope Statement: optional, PRIVATE or PUBLIC, default PUBLIC, no semicolon. See the section titled [Function Scope](#).
- <ReturnDataType> = Return Data Type Statement: optional and one of INT, REAL, STRING, OBJECT, QUALITY, or TIMESTAMP. No default, no semicolon. If no return type is declared, the function cannot return any data. See the section titled [Declaring the Return Data Type](#).
- FUNCTION = FUNCTION Statement: required, indicates the start of the function, [Restricted Cicode Keywords](#), no semicolon. See the section titled [Declaring Functions](#).
- <FunctionName> = Name statement: required, up to 32 ASCII text characters, case insensitive, no spaces, no reserved words, no default, no semicolon. See the section titled [Naming Functions](#).
- ( <Arguments> ) = Argument statement: surrounding brackets required even if no arguments used, if more than one argument - each need to be separated by a comma, can contain constants or variables of INT or REAL or STRING or QUALITY or TIMESTAMP data type, default can be defined in declaration, can be spread over several lines to aid readability, no semicolon. See the section titled .
- <Statement> = Executable Statement: required, one or more executable statements that perform some action in Plant SCADA, often used to manipulate data passed into the function as arguments, semicolon required.
- RETURN = Return Statement: optional, used to instruct Cicode to stop executing the function and return to the calling function, keyword, no semicolon. If a ReturnDataType was specified then the Return Statement should include a ReturnValue.
- <ReturnValue> = Return Value Statement; required if RETURN Statement used in function, need to be either a constant or a variable, the data type need to have been previously declared in the function Return Data Type Statement - or does not return a value, semicolon required. See the section titled [Returning Values from Functions](#).
- END = END Statement: required, indicates the end of the function, keyword, no semicolon. See the section titled [Declaring Functions](#).

## See Also

[End of Line Markers](#)

[Following Cicode Syntax](#)

[Writing Functions](#)

## End of Line Markers

Most statements within the function are separated by semicolons ( ; ) but some exceptions exist. The FUNCTION and END Statements (the start and end of the function) have no semicolons, nor does the Scope or Return Data Type Statements, nor any statement that ends with a [Restricted Cicode Keywords](#).

Where a statement is split over several lines (for example, within the IF THEN conditional executor), each line ends with a semicolon - unless it ends in a reserved word.

## See Also

[Writing Functions](#)

## Function Scope

The optional Scope Statement of a function (if used), precedes all other statements of a function declaration in Cicode, including the FUNCTION Statement.

The scope of a function can be either PRIVATE or PUBLIC, and is declared public by default. That is, if no Scope Statement is declared, the function will have public scope.

Both PRIVATE and PUBLIC are Cicode keywords and as such, are reserved.

A private scope function is only accessible (can be called) within the file in which it is declared. You call a private function using the Cicode functions [TaskNew](#), [TaskNewEx](#) and [TaskCall](#).

Public scope functions can be shared across Cicode files, and can be called from pages and Plant SCADA databases (for example, Alarm.dbf).

Because functions are public by default, to make a function public requires no specific declaration. To make a function private however, you need to prefix the FUNCTION Statement with the word **PRIVATE**.

```
PRIVATE
FUNCTION
FunctionName ( <Arguments> )
<Statement> ;
<Statement> ;
<Statement> ;
END
```

## See Also

[Writing Functions](#)

## Declaring the Return Data Type

For information about the RETURN Statement, see the section titled [Returning Values from Functions](#).

The optional Return Data Type Statement of a function (if used), follows the optional Scope Statement (if used), and precedes the FUNCTION Statement declaration in Cicode.

The return data type of a function can be only one of six possible data types: INT (32 bits), REAL (64 bits), STRING (255 bytes), OBJECT (32 bits), QUALITY or TIMESTAMP (64 bits). If no Return Data Type Statement is declared, the function will not be able to return any type of data.

INT, REAL, STRING, OBJECT, QUALITY and TIMESTAMP are Cicode keywords and as such, are reserved.

---

**Note:** In the following function syntax example, every placeholder shown inside arrow brackets (`<placeholder>`) should be replaced in the actual code with the value of the item that it describes. The arrow brackets and the word they contain should not be included in the statement, and are shown only for your information.

To declare the data type that will be returned to the calling code, prefix the FUNCTION Statement with one of the Cicode data type keywords, in the `<ReturnDataType>` placeholder in the following example.

```
<ReturnDataType>
FUNCTION
```

```
FunctionName ( <Arguments> )
<Statement> ;
<Statement> ;
<Statement> ;
END
```

The following example returns an integer of value 5:

```
INT
FUNCTION
FunctionName ( <Arguments> )
<Statement> ;
INT Status = 5;
<Statement> ;
RETURN Status;
END
```

If the RETURN Statement within the function encounters a different data type to that declared in the return data type statement, the value is converted to the declared return data type.

In the example below, the variable Status is declared as a real number within the function. However, Status is converted to an integer when it is returned to the caller, because the data type of the return was declared as an integer type in the return data type statement:

```
INT ! declare return value as integer
FUNCTION
FunctionName ( <Arguments> )
<Statement> ;
REAL Status = 5; ! declare variable as a REAL number
<Statement> ;
RETURN Status; ! returned as an integer number
END
```

If you omit the return data type, the function does not return a value.

## See Also

[Writing Functions](#)

## Declaring Functions

The required FUNCTION Statement follows the optional Scope Statement (if used) and the optional Return Data Type Statement (if used), and precedes any other statements of a function declaration in Cicode. Everything between it and the END Statement, contains the function.

Both FUNCTION and END are Cicode keywords and, as such, are reserved.

You declare the start of a function with the FUNCTION Statement, and declare the end of a function with the END Statement:

```
FUNCTION
<FunctionName> ( <Arguments> )
<Statement> ;
<Statement> ;
<Statement> ;
END
```

The FUNCTION Statement needs to be followed by the Name Statement, then the Argument Statement, before any code statements that will be processed by the function.

For information on the Name and Argument Statements, see the sections titled [Naming Arguments and Function Argument Structure](#).

The code (as represented by the <Statement> placeholders) located between the FUNCTION and END Statements, will be executed (processed by the function) when called to do so.

Functions can execute a large variety of statements, and are commonly used to process and manipulate data, including the arguments passed when the function was called, plant-floor and other Plant SCADA data, Windows data, and so on. Plant SCADA provides many built-in functions. For more information, see the section titled [Working with Commonly Used Functions](#).

## See Also

[Writing Functions](#)

### Naming Functions

The required name statement follows the FUNCTION Statement and precedes the arguments statement in a Plant SCADA function. The function name is used elsewhere in Plant SCADA to activate (call) the function to have it perform the statements it contains.

Replace the <FunctionName> placeholder in the following function example with an appropriate name for your function. See the section [Function Naming Standards](#) for details.

```
FUNCTION
<FunctionName> ( <Arguments> )
<Statement> ;
<Statement> ;
<Statement> ;
END
```

You can use up to 32 ASCII text characters to name your functions. You can use any valid name except for a [Restricted Cicode Keywords](#). The case is ignored by the Plant SCADA compiler, so you can use upper and lower case to make your names clear. For example, MixerRoomPageDisplay is easier to read than mixerroompagedisplay or MIXERROOMPAGEDISPLAY.

```
FUNCTION
MixerRoomPageDisplay ( <Arguments> )
<Statement> ;
<Statement> ;
<Statement> ;
END
```

Your functions take precedence over any other entity in Plant SCADA with the same name:

- **Variable tags.** When you call a function by the same name as a variable tag, the function has precedence. The variable tag can not be referred to because the function executes each time the name is used.
- **Built-in functions.** You can give your function the same name as any built-in Cicode function. Your function takes precedence over the built-in function - the built-in function cannot be called. Because built-in Cicode functions cannot be changed, this provides a method of 'modifying' any built-in function to suit an application. For example, you might want to display the message "Press F1 for Help" whenever you display a page. You could simply write a new function called PageDisplay(). The body of the function would be the statements that display the page and prompt message:

```
Prompt ( "Press F1 for Help" ) ;PageDisplay ( <Arguments> ) ;
```

Your function is invoked whenever you use the function name in Plant SCADA.

## See Also

[Writing Functions](#)

### Function Argument Structure

The optional Arguments Statement follows the required FUNCTION Statement and precedes the executable statements of a function in Cicode.

**Note:** The maximum number of arguments you can have in a function is 128.

When you call a function, you can pass one or more arguments to the function, enclosed within the parentheses () located after the function name statement. Replace the <Arguments> placeholder in the following function example with your Argument Statement.

```
FUNCTION
FunctionName ( <Arguments> )
<Statement> ;
<Statement> ;
<Statement> ;
END
```

For your function to perform tasks with data, it requires accessibility to the data. One way to achieve this, is to pass the data directly to the function when the function is being called. To enable this facility, Cicode utilizes arguments in its function structure. An argument in Cicode is simply a variable that exists in memory only as long as its function is processing data, so the scope of an argument is limited to be local only to the function.

Arguments cannot be arrays.

Arguments are variables that are processed within the body of the function only. You cannot use an argument outside of the function that declares it.

As arguments are variables used solely within functions, they needs to be declared just as you would otherwise declare a variable in Cicode. See the section titled [Declaring Variable Properties](#). An argument declaration requires a data type, a unique name, and may contain an initial value which also behaves as the default value for the argument.

---

**Note:**

In the following function syntax example:

- Every placeholder shown inside arrow brackets ( <placeholder> ) should be replaced in any actual code with the value of the item that it describes. The arrow brackets and the word they contain should not be included in the statement, and are shown here only for your information.
- Statements shown between square brackets ( [ ] ) are optional. The square brackets should not be included in the statement, and are shown here only for your information.

Cicode function argument statements have the following syntax:

```
<ArgumentDataType>
<ArgumentName>
[ = <InitialDefaultValue> ]
```

Where:

- <ArgumentDataType> = Argument Data Type Statement: required, INT, REAL, STRING, OBJECT, QUALITY, or TIMESTAMP. See the section titled [Declaring Argument Data Type](#).

- <ArgumentName> = Argument Name Statement: required, up to 32 ASCII text characters, case insensitive, no spaces, no reserved words. See the section titled [Naming Arguments](#).
- <InitialDefaultValue> = Argument Initialization Statement: optional, preceded by equals (=) assignment operator, a value to assign to the argument variable when first initialized, needs to be the same data type as that declared in the argument <ArgumentDataType> parameter, defaults to this value if no value passed in for this argument when the function was called.

See the section titled [Setting Default Values for Arguments](#).

The Argument Statement in a Cicode function can have only one set of surrounding parentheses ( ), even if no arguments are declared in the function.

If more than one argument is used in the function, each needs to also be separated by a comma.

Argument Statements can be separated over several lines to aid in their readability.

When you call a function, the arguments you pass to it are used within the function to produce a resultant action or return a value. For information on passing data to functions, see the section titled [Passing Data to Functions \(Arguments\)](#). For information on returning results from functions, see the section titled [Returning Data from Functions](#).

Arguments are used in the function and referred to by their names. For instance, if we name a function AddTwoIntegers, and declare two integers as arguments naming them FirstInteger and SecondInteger respectively, we would end up with a sample function that looks like the following:

```
INT
FUNCTION
AddTwoIntegers ( INT FirstInteger, INT SecondInteger )
INT Solution ;
Solution = FirstInteger + SecondInteger ;
RETURN Solution ;
END
```

In this example, the function would accept any two integer values as its arguments, add them together, and return them to the caller as one integer value equal to the summed total of the arguments values passed into the function.

This functionality of passing values into a function as arguments, manipulating the values in some way, then being able to return the resultant value, is what makes functions potentially very powerful and time saving. The code only needs to be written once in the function, and can be utilized any number of times from any number of locations in Plant SCADA.

## See Also

[Writing Functions](#)

## Declaring Argument Data Type

If an argument is listed in a Cicode function declaration, the Argument Data Type Statement is required, and is listed first before the required Argument Name Statement and the optional Argument Initialisation Statement.

The argument data type of a function can be only one of six possible data types:INT (32 bits), REAL (64 bits), STRING (255 bytes), OBJECT (32 bits), QUALITY (64 bits) or TIMESTAMP (64 bits).

INT, REAL, STRING, OBJECT, QUALITY and TIMESTAMP are Cicode keywords and as such, are reserved.

---

**Note:** In the following function syntax example:

- 
- Every placeholder shown inside arrow brackets (**<placeholder>**) should be replaced in any actual code with the value of the item that it describes. The arrow brackets and the word they contain should not be included in the statement, and are shown here only for your information.
  - Statements shown between square brackets ([ ]) are optional. The square brackets should not be included in the statement, and are shown here only for your information.
- 

To declare the argument data type that will be used in the function, you need to prefix the Argument Name Statement with one of the Cicode data type keywords, in the **<ArgumentDataType>** placeholder in the following example.

```
FUNCTION
FunctionName ( <ArgumentDataType> <ArgumentName> [ =
<InitialDefaultValue> ] )
<Statement> ;
<Statement> ;
<Statement> ;
END
```

The Argument Statement in a Cicode function needs to have only one set of surrounding parentheses () brackets, even if no arguments are declared in the function.

If more than one argument is used in the function, each needs to also be separated by a comma.

Argument Statements can be separated over several lines to aid in their readability.

## See Also

[Writing Functions](#)

## Naming Arguments

If an argument is listed in a Cicode function declaration, the Argument Name Statement is required, and is listed second, after the required Argument Data Type Statement, and before the optional Argument Initialization Statement.

The argument name is used only within the function to refer to the argument value that was passed into the function when the function was called. The name of the argument variable should be used in the executable statements of the function in every place where you want the argument variable to be used by the statement.

---

**Note:** In the following function syntax example:

- Every placeholder shown inside arrow brackets (**<placeholder>**) should be replaced in any actual code with the value of the item that it describes. The arrow brackets and the word they contain should not be included in the statement, and are shown here only for your information.
  - Statements shown between square brackets ([ ]) are optional. The square brackets should not be included in the statement, and are shown here only for your information.
- 

Replace the **<ArgumentName>** placeholder in the following function example with an appropriate name for your Argument variable. See the section titled [Function Argument Structure](#) for details.

```
FUNCTION
FunctionName ( <ArgumentDataType> <ArgumentName> [ = <InitialDefaultValue> ] )
<Statement> ;
<Statement> ;
<Statement> ;
END
```

You can use up to 32 ASCII text characters to name your arguments. You can use any valid name except for a

**Restricted Cicode Keywords.** The case is ignored by the Plant SCADA compiler, so you can use upper and lower case to make your names clear. For example, iPacketQty is easier to read than ipacketqty or IPACKETQNTY.

```
FUNCTION
FunctionName ( INT iPacketQty )
<Statement> ;
<Statement> ;
<Statement> ;
END
```

To refer to the argument (in the body of your function) you use the name of the argument in an executable statement:

```
INT
FUNCTION
AddTwoIntegers ( INT FirstInteger, INT SecondInteger )
INT Solution ;
Solution = FirstInteger + SecondInteger ;
RETURN Solution ;
END
```

## See Also

[Writing Functions](#)

## Setting Default Values for Arguments

If an argument is listed in a Cicode function declaration, the Argument Initialisation Statement is optional, and if used, is listed last in the Argument Statement after the required Argument Data Type and the Argument Name Statements. The Argument Initialization Statement needs to be preceded by an equals ( = ) assignment operator.

**Note:** In the following function syntax example:

- Every placeholder shown inside arrow brackets ( <placeholder> ) should be replaced in any actual code with the value of the item that it describes. The arrow brackets and the word they contain should not be included in the statement, and are shown here only for your information.
- Statements shown between square brackets ( [ ] ) are optional. The square brackets should not be included in the statement, and are shown here only for your information.

Replace the <InitialDefaultValue> placeholder in the following function example with an appropriate value for your Argument variable.

```
FUNCTION
FunctionName (<ArgumentDataType> <ArgumentName> [ =
<InitialDefaultValue> ])
<Statement> ;
<Statement> ;
<Statement> ;
END
```

The default value for an argument needs to be of the same data type as declared for the argument in the Argument Data Type Statement.

You assign a default argument variable value in the same manner that you assign a Cicode variable value, by using the equals ( = ) assignment operator. For example:

```
FUNCTION
PlotProduct ( INT iPackets = 200 , STRING sName = "Packets" )
<Statement> ;
```

```
<Statement> ;
<Statement> ;
END
```

If you assign a default value for an argument, you may omit a value for that argument when you call the function, because the function will use the default value from the declaration. To pass an empty argument to a function, omit any value for the argument in the call. For example, to call the PlotProduct function declared in the previous example, and accept the default string value of "Packets", a Cicode function call would look like:

```
PlotProduct(500)
```

Be aware that the second argument for the function was omitted from the calling code. In this instance, the default value for the second argument ( "Packets" ) would remain unchanged, and so would be used as the second argument value in this particular function call.

If you do call that function and pass in a value for that argument in the call, the default value is replaced by the argument value being passed in. However, the arguments are reinitialized every time the function is called, so each subsequent call to the function will restore the default values originally declared in the function.

If the function has more than one argument and none of them are explicitly declared in the function declaration, the default values for the undeclared arguments will be used.

For more information on function calls, callers, and calling, see the section titled [Calling Functions from Commands and Expressions](#).

Argument Statements can be separated over several lines to aid in their readability.

## See Also

[Writing Functions](#)

## Returning Values from Functions

Many of the built-in Cicode functions supplied with Plant SCADA return a data value to their calling statement. Mathematical functions return a calculated value. The Date() and Time() functions return the current date and time. Other functions, like PageDisplay(), perform an action, and return a value indicating either the success of the action or the type of error that occurred.

You can also use return values in your own functions, to return data to the calling statement. The return value is assigned in the RETURN Statement:

The optional RETURN Statement of a function (if used), needs to be placed in the executable Statements section of a Cicode function between the FUNCTION and END Statements. Because the RETURN Statement is used to return data values that have usually been manipulated by the function, they are usually placed last just before the END Statement.

```
<ReturnDataType>
FUNCTION
FunctionName ( <Arguments> )
<Statement> ;
<Statement> ;
<Statement> ;
RETURN <ReturnValue> ;
END
```

The RETURN Statement consists of the RETURN keyword, optionally followed by a value to be returned, and finished with the semicolon (;) [End of Line Markers](#).

The RETURN value needs to be of the same data type as was declared in the Return Data Type Statement at the start of the function declaration. The return data type of a function can be only one of six possible data types: INT (32 bits), REAL (64 bits), STRING (255 bytes), OBJECT (32 bits), QUALITY or TIMESTAMP (64 bits). If no Return Data Type Statement is declared, the function will not be able to return any type of data.

If the RETURN Statement within the function encounters a different data type to that declared in the Return Data Type Statement, the value is converted to the declared return data type. For information about the Return Data Type Statement, see the section titled [Declaring the Return Data Type](#).

FUNCTION, INT, REAL, STRING, OBJECT, QUALITY, and TIMESTAMP are Cicode keywords and as such, are reserved.

**Note:** In the following function syntax example every placeholder shown inside arrow brackets (*<placeholder>*) should be replaced in any actual code with the value of the item that it describes. The arrow brackets and the word they contain should not be included in the statement, and are shown here only for your information.

To declare the value that will be returned to the calling code, you need to replace the *<ReturnValue>* placeholder in the following example with an appropriate data value to match the Return Data Type as declared in the function.

```
<ReturnDataType>
FUNCTION
FunctionName ( <Arguments> )
<Statement> ;
<Statement> ;
RETURN <ReturnValue> ;
END
```

The following example returns an integer of value 5:

```
INT
FUNCTION
FunctionName ( <Arguments> )
<Statement> ;
INT Status = 5; <Statement> ;
RETURN Status;END
```

The RETURN statement passes a value back to the calling procedure (either another function, command or expression). Outside of the function, the return value can be read by the calling statement. For example, it can be used by the caller as a variable (in a command), or animated (in an expression).

## See Also

[Writing Functions](#)

## Using Variables

A variable is a named location in the computer's memory where data can be stored. Cicode variables can store the basic data types (such as strings, integers, and real numbers) and each variable is specific for its particular data type. For example, if you set up a Cicode variable to store an integer value, you cannot use it for real numbers or strings.

**Note:** Each data type uses a fixed amount of memory: integers use 4 bytes of memory, real numbers use 8 bytes, and strings use 1 byte per character. PLC INT types use only 2 bytes.

The computer allocates memory to variables according to the data type and the length of time you need the variable to be stored.

Real-time variables (such as PLC variables) are already permanently stored in database files on your hard disk. Any variable you use in a database field command or expression needs to be defined as a variable tag, or the compiler will report an error when the system is compiled.

Working with variables involves the following:

- Declaring Variable Properties
- Declaring the Variable Data Type
- Naming variables
- Setting Default Variable Values
- Using Variable Scope
- Using Database Variables.

---

**Note:** Cicode variables can handle a wide range of Plant SCADA variable tag data types. For example, a Cicode variable of INT data type can be used to store I/O device data types: BCD, BYTE, DIGITAL, INT, LONG, LONGBCD, and UINT.

---

## See Also

- [Variable Declaration Standards](#)  
[Variable Naming Standards](#)  
[Variable Scope Standards](#)  
[Using Arrays](#)

## Declaring Variable Properties

You need to declare each variable used in your functions (except for variables that are configured as variable tags). In the declaration statement, you specify the name and data type of the variable. You can also set a default value for the variable.

## See Also

- [Using Variables](#)

## Declaring the Variable Data Type

You can use variables of the following data types:

INT	Integer (32 bits)	-2,147,483,648 to 2,147,483,647
REAL	Floating point (64 bits)	-1.7E308 to 1.7E+308 with 15 digits of precision
STRING	Text string (255 bytes maximum, including null termination character)	ASCII (null terminated)

OBJECT	ActiveX control object reference (32 bits)	
QUALITY	Represents the Plant SCADA quality (64 bits)	QUAL_GOOD, QUAL_BAD, QUAL_UNCR
TIMESTAMP	Represents the number of 100-nanosecond intervals since January 1, 1601 (64 bits)	

If you want to specify a digital data type, use the integer type. Digital types can either be TRUE(1) or FALSE(0), as can integer types.

**Note:**

- Cicode may internally store floating point values as 64 bit to minimize rounding errors during floating point calculations.
- When comparing REAL values in Cicode it is recommended you use the Round() function before doing the comparison.

## QUALITY Data Type

The QUALITY data type is a new data type in Cicode which incorporates the Plant SCADA quality. The QUALITY data type and the Cicode quality labels can be used in Cicode expressions.

The operators allowed for the QUALITY data type are:

- Assignment operator: =.
- Relational operators: =, <>.

The assignment operation also allows for the QUALITY data type.

**Example:**

```
QUALITY q1;
QUALITY q2;

q1 = q2;
q1 = Tag1.Field.Q;

//the following expression will generate a compiler error as a tag //element can be
modified only as a whole
Tag1.Field.Q = q1;
```

A set of Cicode functions are provided which allow quality fields to be initialized, a specific quality field to be extracted, and other operations on the QUALITY data type. Conversion between the QUALITY data type and other Cicode data types is not allowed. Direct conversion from Quality to string will return an empty string.

**Example:**

```
//this will generate a compiler error
INT n = Tag1.Q;
```

## TIMESTAMP Data Type

The TIMESTAMP data type represents the date and time as a 64-bit value by specifying the number of 100-nanosecond intervals since January 1, 1601. This data type is always in Coordinated Universal Time (UTC).

The operators allowed for the TIMESTAMP data type are:

- Assignment operator: =.
- Relational operators: =, <>, <, >, <=, >=.

### Example:

```
TIMESTAMP t1;
TIMESTAMP t2;
t1 = Tag1.T;
t1 = t2;
IF t1 < Tag2.T THEN
// insert code here
END
STRING sStr;
REAL Result;
INT x, y;
OBJECT hObject;
```

The first 32 characters of a variable name needs to be unique.

### [Using Variables](#)

## Naming Variables

Throughout the body of the function, the variable is referred to by its name. You can name a variable any valid name except for a reserved word, for example:

```
STRING sStr;
REAL Result;
INT x, y;
OBJECT hObject;
```

The first 32 characters of a variable name needs to be unique.

## See Also

[Using Variables](#)

[Variable Naming Standards](#)

## Setting Default Variable Values

When you declare variables, you can set them to an initial (startup) value; for example:

```
STRING Str = "Test";
REAL Result = ;
INT x = 20, y = 50;
```

## See Also

[Using Variables](#)

### Using Variable Scope

Scope refers to the accessibility of a function and its values. A Cicode variable can be defined as any one of three types of scope - global, module, and local. By default, Cicode variables are module scope, unless they are declared within a function.

Variables have the following format:

```
DataType Name [=Value];
```

### Global Variables

A global Cicode variable can be shared across all Cicode files in the system (as well as across include projects). They cannot be accessed on pages or databases (for example, Alarm.dbf).

Global Cicode variables are prefixed with the keyword **GLOBAL**, and needs to be declared at the start of the Cicode file. For example:

```
GLOBAL STRING sDefaultPage = "Mimic";
INT
FUNCTION
MyPageDisplay(STRING sPage)
INT iStatus;
iStatus = PageDisplay(sPage);
IF iStatus <> 0 THEN
PageDisplay(sDefaultPage);
END
RETURN iStatus;
END
```

The variable **sDefaultPage** could then be used in any function of any Cicode file in the system.

---

**Note:** Use global variables sparingly if at all. If you have many such variables being used by many functions, finding bugs in your program can become time consuming. Use local variables wherever possible. Global Cicode STRING types are 256 bytes.

---

### Module Variables

A module Cicode variable is specific to the file in which it is declared. This means that it can be used by any function in that file, but not by functions in other files.

By default, Cicode variables are defined as module, therefore prefixing is not required (though a prefix of **MODULE** could be added if desired). Module variables should be declared at the start of the file. For example:

```
STRING sDefaultPage = "Mimic";
INT
FUNCTION
MyPageDisplay(STRING sPage)
INT Status;
Status = PageDisplay(sPage);
IF Status <> 0 THEN
PageDisplay(sDefaultPage);
```

```
END
RETURN Status;
END
INT
FUNCTION
DefaultPageDisplay()
PageDisplay(sDefaultPage);
END
```

**Note:** Use module variables sparingly if at all. If you have many such variables being used by many functions, finding bugs in your program can become time-consuming. Use local variables wherever possible.

## Local Variables

A local Cicode variable is only recognized by the function within which it is declared, and can only be used by that function. You need to declare local variables before you can use them.

Any variable defined within a function (that is, after the function name) is a local variable, therefore no prefix is needed. Local variables are destroyed when the function exits.

Local variables take precedence over global and module variables. If you define a local variable in a function with the same name as a global or module variable, the local variable is used; the global/module variable is unaffected by the function. This situation should be avoided, however, as it is likely to cause confusion.

### Local Variables and Variable Tags

Local variables have limited functionality compared with variable tags. Limitations are:

- Qualities of Override, OverrideMode, ControlMode and Status elements are showing Bad with extended substatus QUAL\_EXT\_INVALID\_ARGUMENT. Writing to the elements returns error Invalid argument passed (274).
- Values of Override, OverrideMode, ControlMode and Status elements are showing 0.
- Respective timestamps and quality of Field, Valid and default elements are the same.
- Field, Valid and default elements can be read.
- Field and default elements can be written.

## See Also

[Using Variables](#)

[Variable Scope Standards](#)

## Using Database Variables

You can use any variable that you have defined in the database (with the Variable Tags form) in your functions. To use a database variable, specify the tag name:

<Tag>

Where *Tag* is the name of the database variable.

For example, to change the value of the database variable "LT131" at run time, you would use the following statement in your function:

LT131=1200; !Changes the value of LT131 to 1200

## See Also

[Using Variables](#)

## Using Arrays

A Cicode variable array is a collection of Cicode variables of the same data type, in the form of a list or table. You name and declare an array of variables in the same way as any other Cicode variable. You can then refer to each element in the array by the same variable name, with a number (index) to indicate its position in the array. Working with arrays involves the following:

### Declaring Array Properties

Arrays have several properties that you need to declare to the compiler along with the array name: data type, size and dimension. You can also set default values for individual elements of the array. An array declaration has the following syntax:

```
DataType Name[Dim1Size,{Dim2Size},{Dim3Size}]{=Values};
```

### Declaring the Array Data Type

As with any other Cicode variable, arrays can have four Data Types:

INT	Integer (32 bits)
REAL	Floating point (64 bits)
STRING	Text string (255 bytes)
OBJECT	ActiveX object (32 bits)
QUALITY	Plant SCADA Quality (64 bits)
TIMESTAMP	Date and Time (64 bits)

### Naming Arrays

Throughout the body of a Cicode function, a Cicode variable array is referred to by its name, and individual elements of an array are referred to by their index. The index of the first element of an array is 0 (that is a four element array has the indices 0,1,2, and 3). You can name a variable any valid name except for a [Restricted Cicode Keywords](#); for example:

```
STRING StrArray[5]; ! list
REAL Result[5][2]; ! 2-D table
INT IntArray[4][3][2]; ! 3-D table
```

### Declaring the Variable Array Size

You need to declare the size of the array (the number of elements the array contains, from 1 to 32,767), for

example:

```
STRING StrArray[5];
```

This single dimension array contains 5 elements. The compiler multiplies the number of elements in the array by the size of each element (dependent upon the Data Type), and allocates storage for the array in consecutive memory locations.

You cannot declare arrays local to a function. However, they can be declared as Module (that is at the beginning of the Cicode file), or Global. When referring to the array within your function, take care to remain within the size you set when you declared the array. The example below would cause an error:

```
STRING StrArray[5];
...
StrArray[10] = 100;
...
```

The compiler allows storage for 5 strings. By assigning a value to a 10<sup>th</sup> element, you cause a value to be stored outside the limits of the array, and you could overwrite another value stored in memory.

## Setting Default (Initial) Array Values

When you declare an array, you can (optionally) set the individual elements to an initial (or start-up) value within the original declaration statement. For instance, naming a string array "ArrayA", sizing it to hold 5 elements, and initializing the array with string values, would look like the following example:

```
STRING ArrayA[5] = "This", "is", "a", "String", "Array";
```

This array structure would contain the following values:

```
ArrayA[0] = "This"
ArrayA[1] = "is"
ArrayA[2] = "a"
ArrayA[3] = "String"
ArrayA[4] = "Array"
```

## Passing Array Elements as Function Arguments

To pass a Cicode variable array element to a Cicode function, you need to provide the element's address; for example:

```
/* Pass the first element of ArrayA. */
MyFunction (ArrayA[0])
/* Pass the second element of ArrayA. */
MyFunction (ArrayA[1])
/* Pass the fifth element of ArrayA. */
MyFunction (ArrayA[4])
```

## Using One-dimensional Arrays

To use a one-dimensional array:

```
STRING ArrayA[5] = "This", "is", "a", "String", "Array";
This array sets the following values:
ArrayA[0] = "This"
ArrayA[1] = "is"
ArrayA[2] = "a"
ArrayA[3] = "String"
```

```
ArrayA[4] = "Array"
```

## Using Two-dimensional Arrays

To use a two-dimensional array:

```
REAL ArrayA[5][2]=1,2,3,4,5,6,7,8.3,9.04,10.178;
```

This array sets the following values:

ArrayA[0][0]=1	ArrayA[0][1]=2
ArrayA[1][0]=3	ArrayA[1][1]=4
ArrayA[2][0]=5	ArrayA[2][1]=6
ArrayA[3][0]=7	ArrayA[3][1]=8.3
ArrayA[4][0]=9.04	ArrayA[4][1]=10.178

## Using Three-dimensional Arrays

To use a three-dimensional array:

```
INT ArrayA[4][3][2]=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,  
16,17,18,19,20,21,22,23,24;
```

This array sets the following values:

ArrayA[0][0][0]=1	ArrayA[0][0][1]=2	ArrayA[0][1][0]=3
ArrayA[0][1][1]=4	ArrayA[0][2][0]=5	ArrayA[0][2][1]=6
ArrayA[1][0][0]=7	ArrayA[1][0][1]=8	ArrayA[1][1][0]=9
ArrayA[1][1][1]=10	ArrayA[1][2][0]=11	ArrayA[1][2][1]=12
ArrayA[2][0][0]=13	ArrayA[2][0][1]=14	ArrayA[2][1][0]=15
ArrayA[2][1][1]=16	ArrayA[2][2][0]=17	ArrayA[2][2][1]=18
ArrayA[3][0][0]=19	ArrayA[3][0][1]=20	ArrayA[3][1][0]=21
ArrayA[3][1][1]=22	ArrayA[3][2][0]=23	ArrayA[3][2][1]=24

You use arrays in your functions in the same way as other variables, but arrays have special properties that, in many situations, reduce the amount of code you need to write.

## Using Array Elements in Loops

You can set up loops that deal efficiently with arrays by incrementing the index number. The following example shows a method of initializing an array:

```
REAL Array[10]
```

```
:  
FOR Counter = 0 TO 9 DO  
Array[Counter] = 0  
END  
RETURN Total  
:
```

## Using the Table (Array) Functions

Cicode has built-in functions for processing Cicode variable arrays:

- To perform calculations (max, min, total, etc.) on array elements.
- To look up the index number of an array element.
- To shift the elements of an array left or right.

---

**Note:** Plant SCADA also provides a set of array functions that support associations with animation objects. See [Array Functions](#).

---

## See Also

[Variable Declaration Standards](#)

[Using Cicode Files](#)

## Using Cicode Macros

Cicode has the following macros:

- **IFDEF:** Determines one of two possible outcomes based on the existence of a specified non-alarm tag at compile time. Use one of the macros below for alarm tags.
- **IFDEFAdvAlm:** Determines one of two possible outcomes based on the existence of a specified advanced alarm tag at compile time.
- **IFDEFAnaAlm:** Determines one of two possible outcomes based on the existence of a specified analog alarm tag at compile time.
- **IFDEFDigAlm:** Determines one of two possible outcomes based on the existence of a specified digital alarm tag at compile time.
- **IFDEFHresAlm :** Determines one of two possible outcomes based on the existence of a specified time stamped alarm tag at compile time.

## See Also

[Macro Arguments](#)

### IFDEF

The IFDEF macro allows you to define two possible outcomes based on whether or not a specified tag exists within a project at the time of compiling. The macro can be implemented anywhere a simple expression is used,

including fields within relevant Plant SCADA dialogs.

The macro was primarily created to avoid the "Tag not found" compile error being generated whenever a genie was missing an associated tag. By allowing a "0" or "1" to be generated within the **Hidden When** field of a Genie's properties, elements could simply be hidden if a required tag was missing, allowing the genie to still be pasted onto a graphics page.

The macro accepts three arguments: the first specifies the tag that requires confirmation, the second defines the outcome if the tag exists, the third defines the outcome if it does not exist. In the case of a genie being pasted on a graphics page, the IFDEF function would be configured as follows in the **Hidden When** field of the object properties dialog:

```
IFDEF("Bit_1",0,1)
```

If the tag "Bit\_1" is defined in the tag database, the value in the **Hidden When** field will be 0. If Bit\_1 is undefined, the value will be 1. Since the object is hidden when the value is TRUE (1), the object will be hidden when Bit\_1 is undefined. See Hiding Graphics Objects for details.

Beyond this purpose, the IFDEF macro can be broadly used as a conditional variable. The [*<value if defined>*] and [*<value if not defined>*] arguments can support any variable, expression, or constant. The [*<value if defined>*] argument is optional; if you leave it blank it will generate the current variable. You can also use nested IFDEF macros.

---

**Note:** As different types of alarms can share the same name, you have to use a variation of IFDEF to check for the existence of alarm tags. See IFDEFAnaAlm for analog alarms, IFDEFDigAlm for digital alarms, or IFDEFAdvAlm for advanced alarms.

---

## Syntax

```
IFDEF(TagName, [<value if defined>], <value if not defined>)
```

## Return Value

If the tag specified in the first argument exists, the value defined by the second argument is returned. This could be a variable, expression, or constant, or the current tag value if the argument has been left blank. If the specified tag does not exist, the variable, expression, or constant defined by the third argument is returned.

## Example

```
! Generate the tag value if tag "Bit_1" is defined
! Generate an empty string if "Bit_1" is not defined
IFDEF("Bit_1","","")
! Generate a zero value (0) if tag "Bit_1" is defined
! Generate a true value (1) if "Bit_1" is not defined
IFDEF("Bit_1",0,1)
```

For more examples of how to implement the IFDEF macro, see Tech Note TN5672 via the AVEVA Knowledge and Support Center.

## Related Macros

[IFDEFAnaAlm](#), [IFDEFDigAlm](#), [IFDEFAdvAlm](#), [IFDEFHresAlm](#)

## See Also

[Macro Arguments](#)

[Using Cicode Macros](#)

## IFDEFAdvAlm

Based on the IFDEF macro, IFDEFAdvAlm allows you to define two possible outcomes based on whether or not a specified advanced alarm tag exists within a project at the time of compiling. The macro can be implemented anywhere a simple expression is used, including fields within relevant Plant SCADA dialogs.

The macro accepts three arguments: the first specifies the advanced alarm tag that requires confirmation, the second defines the outcome if the alarm exists, the third defines the outcome if it does not exist.

**Note:** As different types of alarms can share the same name, you have to use a variation of IFDEF to check for the existence of alarm tags. See [IFDEFAnaAlm](#) for analog alarms, or [IFDEFDigAlm](#) for digital alarms.

## Syntax

**IFDEFAdvAlm**(TagName, [<value if defined>], <value if not defined>)

## Return Value

If the advanced alarm tag specified in the first argument exists, the value defined by the second argument is returned. This could be a variable, expression, or constant, or the current tag value if the argument has been left blank. If the specified alarm does not exist, the variable, expression, or constant defined by the third argument is returned.

## Example

```
! Generate tag value if advanced alarm "AdvAlarm_1" is defined
! Generate an empty string if "AdvAlarm_1" is not defined
IFDEFAdvAlm("AdvAlarm_1","","")
! Generate a zero value (0) in Hidden When field if advanced alarm
"AdvAlarm_1" is defined
! Generate a true value (1) in Hidden When field if "AdvAlarm_1"
is not defined
IFDEFAdvAlm("AdvAlarm_1",0,1)
```

For more examples of how to implement the IFDEF macro, see Tech Note TN5672 via the AVEVA Knowledge and Support Center.

## Related Macros

[IFDEF](#), [IFDEFAnaAlm](#), [IFDEFDigAlm](#), [IFDEFHresAlm](#)

## See Also

[Macro Arguments](#)

## Using Cicode Macros

### IFDEFAnaAlm

Based on the IFDEF macro, IFDEFAnaAlm allows you to define two possible outcomes based on whether or not a specified analog alarm tag exists within a project at the time of compiling. The macro can be implemented anywhere a simple expression is used, including fields within relevant Plant SCADA dialogs.

The macro accepts three arguments: the first specifies the analog alarm tag that requires confirmation, the second defines the outcome if the alarm exists, the third defines the outcome if it does not exist.

**Note:** As different types of alarms can share the same name, you have to use a variation of IFDEF to check for the existence of alarm tags. See [IFDEFDigAlm](#) for digital alarms, or [IFDEFAdvAlm](#) for advanced alarms.

### Syntax

**IFDEFAnaAlm**(TagName, [<value if defined>], <value if not defined>)

### Return Value

If the analog alarm tag specified in the first argument exists, the value defined by the second argument is returned. This could be a variable, expression, or constant, or the current tag value if the argument has been left blank. If the specified alarm does not exist, the variable, expression, or constant defined by the third argument is returned.

### Example

```
! Generate tag value if analog alarm "AnaAlarm_1" is defined
! Generate an empty string if "AnaAlarm_1" is not defined
IFDEFAnaAlm("AnaAlarm_1","", "")
! Generate a zero value (0) in Hidden When field if analog alarm
"AnaAlarm_1" is defined
! Generate a true value (1) in Hidden When field if "AnaAlarm_1"
is not defined
IFDEFAnaAlm("AnaAlarm_1",0,1)
```

For further examples of how to implement the IFDEF macro, see Tech Note TN5672 via the AVEVA Knowledge and Support Center.

### Related Macros

[IFDEF](#), [IFDEFDigAlm](#), [IFDEFAdvAlm](#), [IFDEFHresAlm](#)

### See Also

[Macro Arguments](#)

[Using Cicode Macros](#)

## IFDEFDigAlm

Based on the IFDEF macro, IFDEFDigAlm allows you to define two possible outcomes based on whether or not a specified digital alarm tag exists within a project at the time of compiling. The macro can be implemented anywhere a simple expression is used, including fields within relevant Plant SCADA dialogs.

The macro accepts three arguments: the first specifies the digital alarm tag that requires confirmation, the second defines the outcome if the alarm exists, the third defines the outcome if it does not exist.

**Note:** As different types of alarms can share the same name, you have to use a variation of IFDEF to check for the existence of alarm tags. See [IFDEFAnaAlm](#) for analog alarms or [IFDEFAdvAlm](#) for advanced alarms.

## Syntax

```
IFDEFDigAlm(TagName, [<value if defined>], <value if not defined>)
```

## Return Value

If the digital alarm tag specified in the first argument exists, the value defined by the second argument is returned. This could be a variable, expression, or constant, or the current tag value if the argument has been left blank. If the specified alarm does not exist, the variable, expression, or constant defined by the third argument is returned.

Example

```
! Generate tag value if digital alarm "DigAlarm_1" is defined
! Generate an empty string if "DigAlarm_1" is not defined
IFDEFDigAlm("DigAlarm_1","", "")
! Generate a zero value (0) in Hidden When field if digital alarm
"DigAlarm_1" is defined
! Generate a true value (1) in Hidden When field if "DigAlarm_1"
is not defined
IFDEFDigAlm("DigAlarm_1",0,1)
```

For more examples of how to implement the IFDEF macro, see Tech Note TN5672 via the AVEVA Knowledge and Support Center.

## Related Macros

[IFDEF](#), [IFDEFAnaAlm](#), [IFDEFAdvAlm](#), [IFDEFHresAlm](#)

## See Also

[Macro Arguments](#)

[Using Cicode Macros](#)

## IFDEFHresAlm

Based on the IFDEF macro, IFDEFHresAlm allows you to define two possible outcomes based on whether or not a specified time stamped alarm tag exists within a project at the time of compiling. The macro can be implemented anywhere a simple expression is used, including fields within relevant Plant SCADA dialogs.

The macro accepts three arguments: the first specifies the digital alarm tag that requires confirmation, the second defines the outcome if the alarm exists, the third defines the outcome if it does not exist.

**Note:** As different types of alarms can share the same name, you have to use a variation of IFDEF to check for the existence of alarm tags. See [IFDEFAnaAlm](#) for analog alarms or [IFDEFAdvAlm](#) for advanced alarms.

## Syntax

```
IFDEFHresAlm(TagName, [<value if defined>], <value if not defined>)
```

## Return Value

If the time stamped alarm tag specified in the first argument exists, the value defined by the second argument is returned. This could be a variable, expression, or constant, or the current tag value if the argument has been left blank. If the specified alarm does not exist, the variable, expression, or constant defined by the third argument is returned.

## Example

```
! Generate tag value if time stamped alarm "TimestampedAlarm_1" is defined
! Generate an empty string if "TimestampedAlarm_1" is not defined
IFDEFHresAlm("TimestampedAlarm_1",,, "")
! Generate a zero value (0) in Hidden When field if digital alarm "TimestampedAlarm_1" is
defined
! Generate a true value (1) in Hidden When field if "TimestampedAlarm_1"
is not defined
IFDEFHresAlm("TimestampedAlarm_1",0,1)
```

For more examples of how to implement the IFDEF macro, see Tech Note TN5672 via the AVEVA Knowledge and Support Center.

## Related Macros

[IFDEF](#), [IFDEFAnaAlm](#), [IFDEFDigAlm](#), [IFDEFAdvAlm](#)

## See Also

[Macro Arguments](#)  
[Using Cicode Macros](#)

## Macro Arguments

The Cicode macros use the following arguments.

## TagName

The name of the tag you would like the IFDEF macro to confirm the existence of. The Plant SCADA compiler will check the current project database for a tag matching this name.

## [<value if defined>]

Defines the outcome of the macro if the specified tag exists in the current project. This argument is optional, which means you can:

- Generate any variable, constant, or expression.
- Generate the current value for the specified tag by leaving the argument blank.

## <value if not defined>

Defines the outcome of the macro if the specified tag does not exist in the current project. This will generate any variable, constant, or expression, including a blank string (" ") if you want nothing to be presented.

## See Also

[Using Cicode Macros](#)

## Converting and Formatting Cicode Variables

Plant SCADA provides the following functions for converting variables.

- IntToStr: converts an integer variable into a string
- IntToReal: converts an integer into a floating-point variable
- RealToStr: converts a floating-point variable into a string
- StrToInt: converts a string into an integer variable
- StrToReal: converts a string into a floating-point variable

You can convert data types without using these Cicode functions, but the result of the format conversion might not be what you expect. If you want more control over the conversion process, use the appropriate Cicode functions.

---

**Note:** Variables of type *object* cannot be converted to any other type.

---

When variables are automatically converted, or when the return value from a function call is converted, specific rules apply.

## Converting Variable Integers to Strings

To convert an integer variable to a string:

```
IntVar=5;  
StringVar=IntVar;
```

The value of StringVar is set to "5".

The format of the string is specified when the variable is defined in the database. However you can override this default format with the string format (:) operator, and use the # format specifier to set a new format. For example:

```
IntVar=5;
```

```
StringVar=IntVar:####
```

The value of StringVar = " 5 ". (The '#' formatting characters determine the size and number of decimal places contained in the string, that is a length of 4 with no decimal places.)

## Converting Variable Integers to Real Numbers

When calling statements such as comparison or arithmetic operations that involve a mixture of variables of REAL and INT data types, there is no need to explicitly convert the variable of INT data type to REAL data type using the IntToReal function. Plant SCADA will automatically convert variables of INT data type to REAL data type before the operation is carried out.

However, if the expression only consists of variables of INT data type, the result of the expression will remain as INT data type and be subjected to its limitations such as integer overflow and wraparound. This does not change even if the expression is to be assigned to a variable of REAL data type.

If you want to increase the range of the expression, you will need to convert at least one of its operands to REAL data type using the IntToReal function.

### Example

```
INT nVar1 = 1000000001;
INT nVar2 = 2000000002;
REAL rVar1 = 1.000001;
REAL rVar2 = 0.0;

// rVar2 = 1000000002.000001, IntToReal is not needed when operands are mixture of INT and
REAL data types
rVar2 = nVar1 + rVar1;

// rVar2 = -1294967293.0, result is wrapped around due to range overflow of integer data
type
rVar2 = nVar1 + nVar2;

// rVar2 = 3000000003.0, result is prompted to REAL data type when one of the INT operands
is converted
rVar2 = nVar1 + IntToReal(nVar2);
```

## Converting Real Numbers to Strings

To convert a real number variable to a string:

```
RealVar=5.2;
StringVar=RealVar;
```

The value of StringVar is set to "5.2".

---

**Note:** Unpredictable results may occur if you use large numbers with a large number of decimal places.

The format of the string is specified when the variable is defined in the database. However you can override this default format with the string format (:) operator, and use the # format specifier to set a new format. For example:

```
StrTag1=RealTag1:#####.###
```

The value of StringVar = " 5.200 ". (The '#' formatting characters determine the size and number of decimal places contained in the string, that is a length of 10 including a decimal point and three decimal places.)

## Converting Strings to Integers

To convert a string variable to an integer:

```
StringVar="50.25";
IntVar=StringVar;
```

The value of IntVar is set to 50. If StringVar contains any characters other than numeric characters, IntVar is set to 0.

## Converting Strings to Real Numbers

To convert a string variable to a real number:

```
StringVar="50.25";
RealVar=StringVar;
```

The value of RealVar is set to 50.25. If StringVar contains any characters other than numeric characters, RealVar is set to 0.

## Formatting Text Strings

A string in Cicode is represented as text positioned between double quote ( " ) delimiters. For example:

```
"This is my text string."
```

A string value can be assigned to a string variable. For example:

```
STRING sMyStringVariable;
sMyStringVariable = "This is my text string.;"
```

More than one string can be joined together (concatenated) using the Cicode 'plus' mathematical operator ( + ). For example:

```
STRING sMyStringVariable;
sMyStringVariable = "This is my text string." + "This is my second
text string.;"
```

The two strings would be joined together and assigned to the string variable sMyStringVariable. However, if subsequently displayed somehow, like in the following MESSAGE example, the concatenated string would look wrong because there is no space character positioned between the string sentences.

```
STRING sMyStringVariable;
sMyStringVariable = "This is my text string." + "This is my second
text string.;"
```

```
MESSAGE("String Concatenation Example",sMyStringVariable,32);
```



To overcome this potential formatting problem, you could include an extra space as the last character in the strings, or include the space as a third string in the concatenation. For example:

```
sMyStringVariable = "This is my text string. " + "This is my
second text string. ";
```

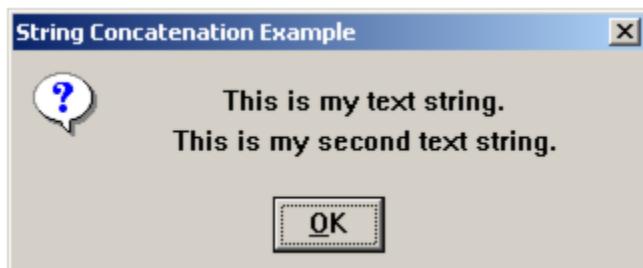
or

```
sMyStringVariable = "This is my text string." + " " + "This is my  
second text string. ";
```

However, these are considered poor programming practices and not recommended. Instead, you can use special string formatting commands known as *escape sequences*.

If the two strings (as used in the previous example), were formatted using appropriate escape sequences positioned within the strings, and subsequently displayed somehow, like in the following MESSAGE example, the concatenated string would look different, For example:

```
STRING sMyStringVariable;  
STRING s.NewLine = "^n";  
sMyStringVariable = "This is my text string." + s.NewLine + "This  
is my second text string. " ;  
MESSAGE("String Concatenation Example",sMyStringVariable,32);
```



Strings and string variables can also be concatenated as in the previous example. Be aware of how the newline escape sequence ( ^n ) was assigned to the string variable s.NewLine, and how this value was concatenated between the other strings and assigned to the string variable sMyStringVariable for display in the MESSAGE function.

## Escape Sequences (String Formatting Commands)

Cicode supports several escape sequences that you can use in text strings for custom formatting of the string. By using the appropriate Cicode escape sequences listed below you can format the string display to do such things as divide onto separate lines at specific positions, insert tab spaces, insert quotes, or to display Hexadecimal numbers.

Cicode escape sequences are preceded by a caret ( ^ ) character. The caret character is interpreted as a special instruction, and together with the characters immediately following it, are treated as an Cicode escape sequence formatting command. The escape sequences used in Cicode are:

^b	backspace
^f	form feed
^n	new line
^t	horizontal tab
^v	vertical tab
^'	single quote
^"	double quote

<code>^^</code>	caret
<code>^r</code>	carriage return
<code>^0xhh</code>	where <i>hh</i> is a hexadecimal number (for example, <code>^0x1A</code> )

## Working with Operators

With Cicode, you can use the data operators that are standard in a large number of programming languages: mathematical, bit, relational, and logical operators.

### See Also

[Using Mathematical Operators](#)

[Using Bit Operators](#)

[Using Relational Operators](#)

[Using Logical Operators](#)

[Order of Precedence of Operators](#)

### Using Mathematical Operators

Standard mathematical operators allow you to perform arithmetic calculations on numeric variables - integers and floating point numbers.

Operator	Description
<code>+</code>	Addition
<code>-</code>	Subtraction
<code>*</code>	Multiplication
<code>/</code>	Division
<code>MOD</code>	Modulus (Remainder)

### Example

The following are examples of mathematical operators

Command	<code>PV12 = PV10 + PV11;</code>
Comment	PV12 is the sum of PV10 and PV11
Command	<code>Counter = Counter - 1;</code>

Comment	The value of Counter is decreased by 1
Command	PV12 = Speed * Counter;
Comment	PV12 is the product of Speed and Counter
Command	Average = Total / ShiftHrs;
Comment	Average is Total divided by ShiftHrs
Command	Hold = PV12 MOD PV13;
Comment	If PV12 = 10 and PV13 = 8, Hold equals 2 (the <b>remainder</b> when PV12 is divided by PV13)

**Note:** Cicode uses the standard order of precedence, that is multiplication and division are calculated before addition and subtraction. In the statement  $A=1+4/2$ , 4 is divided by 2 before it is added to 1, and the result is 3. In the statement  $A=(1+4)/2$ , 1 is first added to 4 before the division, and the result is 2.5.

You can also use the addition operator (+) to concatenate (join) two strings.

Operator	Description
+	Concatenate

## Example

Command	Message = "For info see " + "Supervisor";
Comment	Message now equals "For info see Supervisor"

## See Also

[Working with Operators](#)

## Using Bit Operators

With a bit operator, you can compare the corresponding bits in two numeric expressions. (A bit is the smallest unit of data a computer can store.)

Operator	Description
BITAND	AND
BITOR	OR
BITXOR	Exclusive OR

For example:

Command	Tag3 = Tag1 BITAND Tag2;
Command	Tag3 = Tag1 BITAND 0xFF;
Command	Tag3 = Tag1 BITOR Tag2;
Command	Tag3 = Tag1 BITXOR Tag2;

## See Also

[Working with Operators](#)

## Using Relational Operators

Relational operators describe the relationship between two values. The relationship is expressed as one value being larger than, the same as, or smaller than another. You can use relational operators for both numeric and string variables, however you can only test variables of the same type. A numeric variable cannot be compared with a string variable.

Operator	Description
=	Is equal to
<>	Is not equal to
<	Is less than
>	Is greater than
<=	Is less than or equal to
>=	Is greater than or equal to

For example:

Command	IF Message = "Alarm Active" THEN ...
Expression	PV12 <> PV10;
Command	IF (Total + Count) / Avg < 10 THEN ...
Expression	Counter > 1;
Command	IF PV12 <= PV10 THEN ...
Expression	Total >= Shift * Hours;

## See Also

[Working with Operators](#)

## Using Logical Operators

With logical operators, you can test several conditions as either TRUE or FALSE.

Operator	Description
AND	Logical AND
OR	Logical OR
NOT	Logical NOT

Examples:

Command	Result = (PV12 = 10 AND PV13 = 2);
Comment	If PV12 equals 10 and PV13 equals 2 then Result is TRUE(1)
Expression	Motor_1 AND Motor_2;
Comment	If both Motor_1 and Motor_2 are TRUE, that is Digital bits are 1 or ON, then the expression is TRUE
Expression	PV12 = 1 OR PV13 > 2 OR Counter <> 0;
Comment	If either PV12 equals 1 or PV13 is greater than 2 or Counter is not equal to 0, then the expression is TRUE
Command	Result = (Motor1_Ol OR Motor2_Ol);
Comment	If either Motor1_Ol or Motor2_Ol is TRUE, that is Digital bit is 1 or ON, then Result is TRUE (1)
Command	IF NOT PV12 = 10 THEN ...
Comment	If PV12 does not equal 10 then the result is TRUE. This is functionally identical to IF PV12 <> 10 THEN ...
Expression	NOT Tag_1;
Comment	This expression is TRUE if Tag_1 = 0. This is commonly used for testing digital variables

## See Also

[Working with Operators](#)

## Order of Precedence of Operators

Operators have a set of rules that govern the order in which operations are performed. These rules are called the

order of precedence. The precedence of Cicode operators from highest to lowest is:

1.	()
2.	NOT
3.	*, /, MOD
4.	:
5.	+, -
6.	>, <, <=, >=
7.	=, <>
8.	AND
9.	OR
10.	BITAND, BITOR, BITXOR

## See Also

[Working with Operators](#)

## Working with Conditional Executors

The statements that control decisions and loops in your functions are called conditional executors. Cicode uses four conditional executors:

- If
- For
- While
- Select case.

## See Also

[Formatting Executable Statements](#)

[Setting IF ... THEN Conditions](#)

[Using FOR ... DO Loops](#)

[Using WHILE ... DO Conditional Loops](#)

[Using the SELECT CASE statement](#)

## Setting IF ... THEN Conditions

The IF statement executes one or more statements based on the result of an expression. You can use If in one of

two formats: If Then and If Then Else.

```
If Expression Then
Statement(s);
END
-OR-
If Expression Then
Statement(s);
Else
Statement(s);
END
```

When you use the **If Then** format, the statement(s) following are executed only if the expression is TRUE, for example:

```
INT Counter;
IF PV12 = 10 THEN
Counter = Counter + 1;
END
```

In this example, the Counter increments only if the tag PV12 is equal to 10, otherwise the value of Counter remains unchanged. You can include several statements (including other IF statements), within an IF statement, for example:

```
INT Counter;
IF PV12 = 10 THEN
Counter = Counter + 1;
IF Counter > 100 THEN
Report("Shift");
END
END
```

In this example, the report runs when the Counter increments, that is when PV12 = 10, and the value of the counter exceeds 100.

You can use the **If Then Else** format for branching. Depending on the outcome of the expression, one of two actions are performed, for example:

```
INT Counter;
IF PV12 = 10 THEN
Report("Shift");
ELSE
Counter = Counter + 1;
END
```

In this example, the report runs if PV12 is equal to 10 (TRUE), or the counter increments if PV12 is anything but 10 (FALSE).

## See Also

[Working with Conditional Executors](#)

## Using FOR ... DO Loops

A For loop executes a statement or statements a specified number of times.

```
FOR Variable=Expression To Expression DO
Statement(s);
END
```

The following function uses a For loop:

```
STRING ArrayA[5] = "This", "is", "a", "String", "Array";
INT
FUNCTION
DisplayArray()
INT Counter;
FOR Counter = 0 TO 4 DO
Prompt(ArrayA[Counter]);
Sleep(15);
END
END
```

This function displays the single message "This is a String Array" on the screen one word at a time pausing for 15 seconds between each word.

## See Also

[Working with Conditional Executors](#)

## Using WHILE ... DO Conditional Loops

A While loop executes a statement or statements in a loop as long as a given condition is true.

```
WHILE Expression DO
Statement(s);
END
```

The following code fragment uses a WHILE loop:

```
INT Counter;
WHILE DevNext(hDev) DO
Counter = Counter + 1;
END
/* Count the number of records in the device (hDev) */
```

Be careful when using WHILE loops in your Cicode functions: WHILE loops can cause excessive loading of the CPU and therefore reduce system performance. If you use a WHILE loop to loop forever, you should call the Cicode function `Sleep()` so that Plant SCADA can schedule other tasks. The `Sleep()` function increases the performance of your Plant SCADA system if you use many WHILE loops.

## See Also

[Working with Conditional Executors](#)

## Nested Loops

You can "nest" one loop inside the other. That is, a conditional statement can be placed completely within (nested inside) a condition of another statement.

## See Also

[Working with Conditional Executors](#)

## Using the SELECT CASE statement

The select case statement executes on several groups of statements, depending on the result of an expression. SELECT CASE statements are a more efficient way of writing code that would otherwise have to be done with nested IF THEN statements.

```
SELECT CASE Expression
CASE CaseExpression1,CaseExpression2
Statement(s);
CASE CaseExpression3 TO CaseExpression4
Statement(s);
CASE IS >CaseExpression5,IS<CaseExpression6
Statement(s);
CASE ELSE
Statement(s);
END SELECT
```

Where **CaseExpressionn** is any one of the following forms:

- expression
- expression TO expression

Where the TO keyword specifies an inclusive range of values. The smaller value needs to be placed before TO.

- IS <relop> expression.

Use the IS keyword with relational operators (**<relop>**). Relational operators that may be used are <, <=, =, <>, >, >= .

If the Expression matches any CaseExpression, the statements following that CASE clause are executed up to the next CASE clause, or (for the last clause) up to the END SELECT. If the Expression matches a CaseExpression in more than one CASE clause, only the statements following the first match are executed.

The CASE ELSE clause is used to indicate the statements to be executed if no match is found between the Expression and any of the CaseExpressions. When there is no CASE ELSE statement and no CaseExpressions match the Expression, execution continues at the next Cicode statement following END SELECT.

You can use multiple expressions or ranges in each CASE clause. For example, the following line is valid:

```
CASE 1 To 4, 7 To 9, 11, 13, Is > MaxNumber
```

You can also specify ranges and multiple expressions. In the following example, CASE matches strings that are exactly equal to "everything", strings that fall between "nuts" and "soup" in alphabetical order, and the current value of "TestItem":

```
CASE "everything","nuts" To "soup",TestItem
```

SELECT CASE statements can be nested. Each SELECT CASE statement needs to have a matching END SELECT statement.

For example, if the four possible states of a ship are Waiting, Berthed, Loading, and Loaded, the Select Case statement could be run from a button to display a prompt detailing the ship's current state.

```
select case iStatus
CASE 1
Prompt("Waiting");
CASE 2
Prompt("Berthed");
CASE 3
Prompt("Loading");
CASE 4
Prompt("Loaded");
CASE Else
```

```
Prompt("No Status");
END SELECT
```

## See Also

[Working with Conditional Executors](#)

## Performing Advanced Tasks

This section introduces and explains event handling, Plant SCADA tasks, Plant SCADA threads, how Plant SCADA executes, and multitasking - including foreground and background tasks, controlling tasks, and pre-emptive multitasking.

### Handling Events

Cicode supports event handling. You can define a function that is called only when a particular event occurs. Event handling reduces the overhead that is required when event trapping is executed by using a loop. The following example illustrates the use of the OnEvent() function:

```
INT
FUNCTION MouseCallback()
INT x, y;
DspGetMouse(x,y);
Prompt("Mouse at "+x:"#####", "+y:#####");
RETURN 0;
END
OnEvent(0,MouseCallback);
```

The function MouseCallBack is called when the mouse is moved - there is no need to poll the mouse to check if it has moved. Plant SCADA watches for an event with the OnEvent() function.

Because these functions are called each time the event occurs, you should avoid complex or time consuming statements within the function. If the function is executing when another call is made, the function can be blocked, and some valuable information may be lost. If you do wish to write complex event handling functions, you should use the queue handling functions provided with Cicode.

### How Plant SCADA Executes

Your multi-tasking operating system gives Plant SCADA access to the CPU through threads. However, this access time is not continuous, as Plant SCADA needs to share the CPU with other applications and services.

---

**Note:** Be careful when running other applications at the same time as Plant SCADA. Some applications place high demands on the CPU and reduce the execution speed of Plant SCADA.

---

The Plant SCADA process has many operations to perform, including I/O processing, alarm processing, display management, and Cicode execution - operations that are performed continuously. And, because Plant SCADA is a real-time system, it needs to perform the necessary tasks within a minimum time - at the expense of others. For this reason, Plant SCADA is designed to be multitasking, so it can efficiently manage its own tasks.

Plant SCADA performs its tasks in a specific order in a continuous loop (cycle). Plant SCADA's internal tasks are scheduled at a higher priority than that of Cicode and have access to the CPU before the Cicode. For example, the Alarms, Trends, and I/O Server tasks all get the CPU before any of your Cicode tasks. The reports are scheduled at the same priority as your Cicode. Plant SCADA background spoolers and other idle tasks are lower

priority than your Cicode.

For Cicode, which consists of many tasks, Plant SCADA uses round-robin single priority scheduling. With this type of scheduling each task has the same priority. When two or more Cicode tasks exist, they each get a CPU turn in sequence. This is a simple method of CPU scheduling.

**Note:** If a Cicode task takes longer than its designated CPU time to execute, it is preempted until the next cycle - continuing from where it left off.

---

## Multitasking

Multitasking is when you can run more than one task at the same time. Windows supports this feature at the application level. For example you can run MS-Word and MS-Excel at the same time.

Plant SCADA also supports multitasking internally; that is you can tell Plant SCADA to do something, and before Plant SCADA has completed that task you can tell Plant SCADA to start some other task. Plant SCADA will perform both tasks at the same time. Plant SCADA automatically creates the tasks, leaving you to call the functions.

Multitasking is a feature of Plant SCADA not the operating system. Many applications cannot do this, for example if you start a macro in Excel, while that macro is running you cannot do any other operation in Excel until that macro completes.

A multitasking environment is useful when designing your Cicode. It allows you to be flexible, allowing the operator to perform one action, while another is already taking place. For example, you can use Cicode to display two different input forms at the same time, while allowing the operator to continue using the screen in the background.

## Foreground and Background Tasks

Cicode tasks (or threads) can be executing in either foreground or background mode. A foreground task is one that displays and controls animations on your graphics pages. Any expression (not a command) entered in a property field (that is Text, Rectangle, Button, etc.) is executed as a foreground task. Any other commands and expressions are executed in background mode.

The difference between a background and foreground task is that a background task can be pre-empted. That is, if system resources are limited, the task (for example, the printing of a report) can pause to allow a higher priority task to be executed. When the task is completed (or when system resources become available) the original task resumes. Foreground tasks are the highest priority and can not be pre-empted.

## Controlling Tasks

You can use the Task functions to control the execution of Cicode tasks, and use the Plant SCADA Kernel at runtime to monitor the tasks that are executing. Since Plant SCADA automatically creates new tasks (whenever you call a keyboard command, etc.), schedules them, and destroys them when they are finished, users rarely need to consider these activities in detail.

Sometimes it is desirable to manually 'spawn' a new task. For example, suppose your Cicode is polling an I/O Device (an operation which needs to be continuous), but a situation arises that requires operator input. To display a form would temporarily halt the polling. Instead you can spawn a new task to get the operator input, while the original task continues polling the device.

**Note:** The TaskNew Cicode function is used to spawn new tasks.

---

## Pre-emptive Multitasking

Cicode supports pre-empted multitasking. If a Cicode task is running, and a higher priority task is scheduled, Plant SCADA will suspend the original task, complete the higher priority task and return to the original task.

Preemption is supported between Cicode threads and other internal processes performed by Plant SCADA. You can, therefore, write Cicode that runs forever (for example, a continuous while loop) without halting other Cicode threads or Plant SCADA itself. For example:

```
INT FUNCTION MyLoopFunction()
WHILE TRUE DO
// Whatever is required in the continuous loop
Sleep(1); // Optional
END
END
```

In the above example, the function `Sleep()` is used to force preemption. The `Sleep()` function is optional, however it will reduce the load on the CPU, because the loop is suspended each second (it will not repeat at a high rate).

## Cicode Editor

You use the Cicode Editor to write, edit, and debug your Cicode code. The Cicode Editor behaves similarly to other code editing tools like Microsoft Dev Studio, and contains many advanced editing features such as:

- Dockable windows and toolbars.
- Syntax highlighting - color highlighting of syntax functions.
- IntelliSense AutoPrompt - function definition tooltips.
- IntelliSense AutoComplete - automatic inline prompting and completion of functions with their parameters.
- AutoCaseCorrect - automatic case correction of function keywords.
- AutoIndent - automatic indent alignment of code.
- AutoScroll - automatic mouse middle button support.
- Drag and Drop - copy or move of selected text.
- Bookmark and Breakpoint indicator bar - single click set and reset of bookmarks and breakpoints.
- Keyboard Shortcuts support.

Click the "Launch Cicode Editor" button in Activity Bar of the Plant SCADA Studio to start the Cicode Editor.

Cicode files are stored as text files. For more information see [Using Cicode Files](#).

**Note:** Be careful not to confuse a Cicode file (\*.ci) with an Include file (\*.cii).

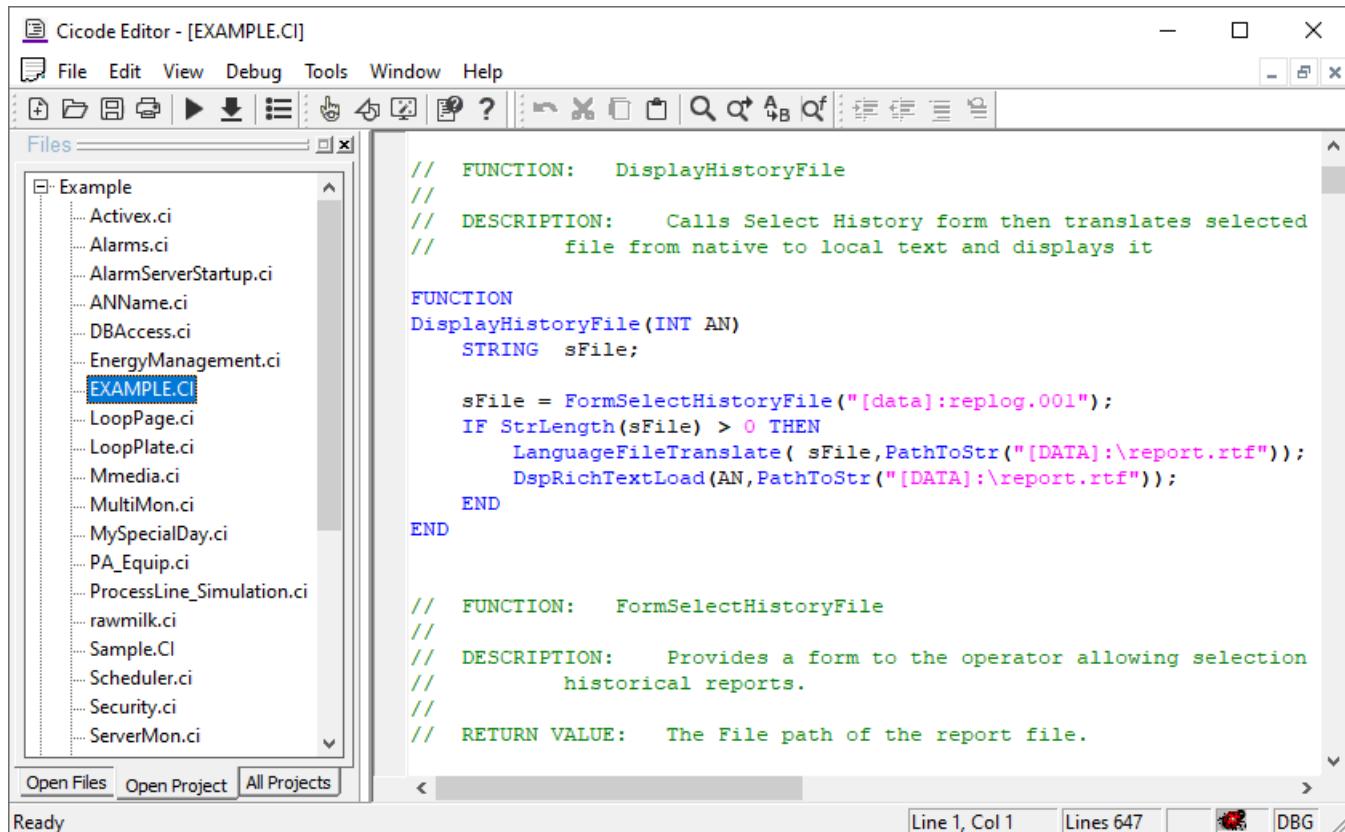
You could use any text editor to view or edit the Cicode files, however, the Cicode Editor provides integrated views specific to Cicode. As well as the features listed above, it includes:

- Breakpoint window
- Output window
- Global Variable Window
- Stack window
- Thread window

- Compile Errors window
- VBA Watch window
- Files window

To minimize potential future problems with maintaining your Cicode files, you should adopt a programming standard as early as possible, as discussed in the section [Using Cicode Programming Standards](#). Maintain structured Cicode files, by logically grouping your Cicode functions within the files, and by choosing helpful descriptive names.

Modular programming methods are discussed in the section [Modular Programming](#). Cicode functions are introduced in the section titled [Using Cicode Functions](#). Suggestions for debugging your Cicode is included in the section titled [Debugging Cicode](#).



```
// FUNCTION:    DisplayHistoryFile
//
// DESCRIPTION:   Calls Select History form then translates selected
//                 file from native to local text and displays it

FUNCTION
DisplayHistoryFile(INT AN)
    STRING sFile;

    sFile = FormSelectHistoryFile("[data]:replug.001");
    IF StrLength(sFile) > 0 THEN
        LanguageFileTranslate( sFile, PathToStr("[DATA]:\report.rtf"));
        DspRichTextLoad(AN,PathToStr("[DATA]:\report.rtf"));
    END
END

// FUNCTION:    FormSelectHistoryFile
//
// DESCRIPTION:   Provides a form to the operator allowing selection
//                 historical reports.
//
// RETURN VALUE:  The File path of the report file.
```

## Creating Cicode Files

### To create a new Cicode file:

1. Start the Cicode Editor.
2. Choose **File | New**, or click **New**.

Save the Cicode file after creating it. The file is only stored on disk after you save it.

## See Also

[Cicode Editor](#)

### Creating Functions

#### To create a new Cicode function:

1. Start the Cicode Editor.
2. Choose **File | New**, or click **New**.
3. Type in your new Cicode function in the blank space, or at the end of the file. Format the Cicode function correctly, following the documented syntax.
4. Save the Cicode file.

## See Also

[Cicode Editor](#)

### Saving Files

#### To save a Cicode file:

1. Choose **File | Save**, or click **Save**.
2. If the file is new, you will be prompted by the Save as dialog. Plant SCADA automatically suggests a name.
3. Type in a new name in the **File name** field.
4. Click **Save** to save the file, or **Cancel** to abort the save.

To save your Cicode file under a new name, choose **Save as** instead of **Save**. The original file will remain in your project under the original filename, until you delete it. All source files in your project directory will be included when you compile your project.

## See Also

[Cicode Editor](#)

[Creating Cicode Files](#)

[Opening Cicode Files](#)

### Opening Cicode Files

#### To open a Cicode file:

1. Start the Cicode Editor.
2. Choose **File | Open**, or click **Open**.
3. Select a file from the list. You can use the dialog controls to open other projects and directories.

4. Click the **Open** button to open the file, or **Cancel** to abort.

---

**Note:** Double clicking on any Cicode file (\*.ci) in the explorer will launch the Cicode Editor and open the Cicode file. However, be careful not to confuse a Cicode file with an Include file (\*.cii).

---

## See Also

[Cicode Editor](#)  
[Saving Files](#)  
[Creating Cicode Files](#)

## Deleting Cicode Files

### To delete a Cicode file:

1. Run the Cicode Editor.
2. **Choose File | Open**, or click the **Open** button.
3. Select the target file from the list. You can use the dialog controls to open other projects and directories.
4. Press the **Delete** key.
5. Click the **Yes** button to confirm delete, or **No** to abort.
6. Click the **Cancel** button to close the Open form.

## See Also

[Cicode Editor](#)  
[Creating Cicode Files](#)  
[Opening Cicode Files](#)

## Finding Text in Cicode Files

### To find text in a Cicode file:

1. Choose **Edit | Find**, or click the **Find** button.
  2. Complete the Find dialog, filling in the **Find what** field.
  3. Click the **Find Next** button to begin searching, or **Cancel** to abort.
- The search is performed down the file from the cursor. Hits are highlighted.

## See Also

[Cicode Editor](#)

## Compiling Cicode Files

### To compile Cicode:

1. Run the Cicode Editor.
2. Choose **File | Compile**, or click the Compile button.

---

**Note:** You cannot compile Cicode functions individually. When you compile Plant SCADA, it automatically compiles the entire contents of the project.

---

## See Also

[Cicode Editor](#)

[Creating Cicode Files](#)

[Opening Cicode Files](#)

## Viewing Errors Detected by the Cicode Compiler

### To view errors detected by the Cicode compiler:

Do either of the following:

- From the **Compile Errors** in the **File** menu of the Project Editor, click **Goto**. This launches the Cicode Editor and opens the appropriate file at the correct line.
- Choose **View | Compile Errors**, then double-click the compile error you want to view.

## See Also

### Find User Function

If you need to locate a Cicode function in your Cicode source files, you can use the **Find User Function** option.

### To locate a function within a Cicode source file:

1. Open the Cicode file you would like to search.
2. From the **Edit** menu, select **Find User Function**.
3. The Find User Function dialog box will open.
4. In the **Find** field, enter a function name (or part of a function name). If required, select the **Find Exact Match** check box.
5. Click **OK**.

A list of functions that match your search criteria will be displayed.

---

**Note:** If you leave the **Find** field empty and click **OK**, a full list of functions appears in the list.

---

### To edit the Cicode file that contains a function:

- Within the search results, select a function and click **Edit**.

Or:

- Within the search results, double-click on a function.

The file containing the selected function will open in the Cicode Editor.

## See Also

[Cicode Editor](#)

### Cicode Editor Windows

The [Cicode Editor](#) has several editing and debug windows that you can use to display information about running Cicode and VBA.

The available Cicode Editor windows are:

- [Breakpoint Window](#)
- [Output Window](#)
- [Global Variable Window](#)
- [Stack Window](#)
- [Thread Window](#)
- [Compile Errors Window](#)
- [Citect VBA Watch window](#)
- [Files window.](#)

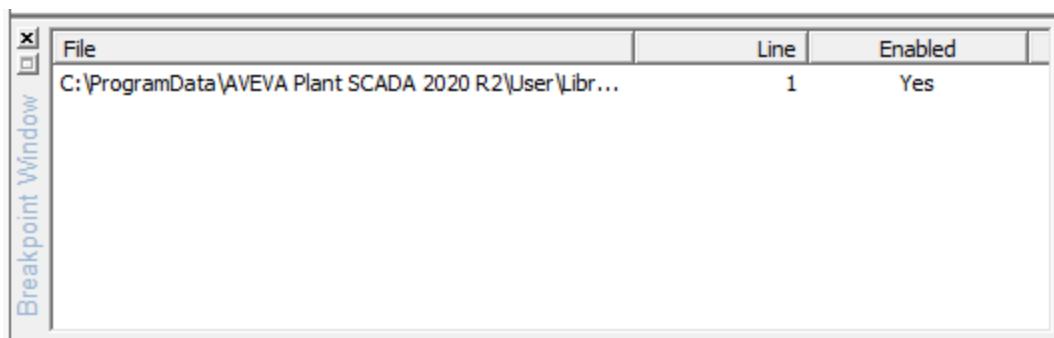
You can display a particular window by selecting it from the **View** menu, or you can use the buttons on the View toolbar.

## See Also

[Docking Windows and Toolbars](#)

### Breakpoint Window

The Breakpoint Window is used to list all breakpoints that are currently set within the project. Double clicking an item in the list loads the file into the editor and jumps to the breakpoint position. Right-clicking an item allows the enable/disable/removal of the list item.



The Breakpoint Window has the following fields:

- **File:** the full name and location of the code file in which the breakpoint exists.
- **Line:** the line number (in the code file) where the breakpoint is located.
- **Enabled:** indicates if the breakpoint is enabled or not. **Yes** indicates it is active, **No** indicates it is not.

## See Also

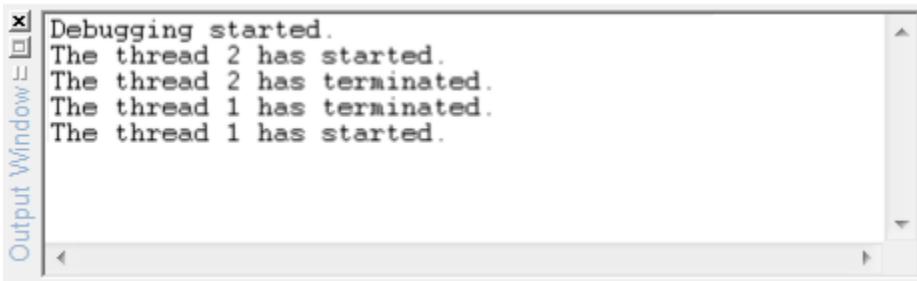
[Docking Windows and Toolbars](#)

[Cicode Editor Windows](#)

## Output Window

The Output Window lists the output messages sent by Plant SCADA during debugging. It states when threads start and terminate, and if a break occurs. This window will show messages sent by the TraceMsg() function.

The Output window shows entries in the order that they occur:



**Note:** you need to be in debug mode to view the messages.

## See Also

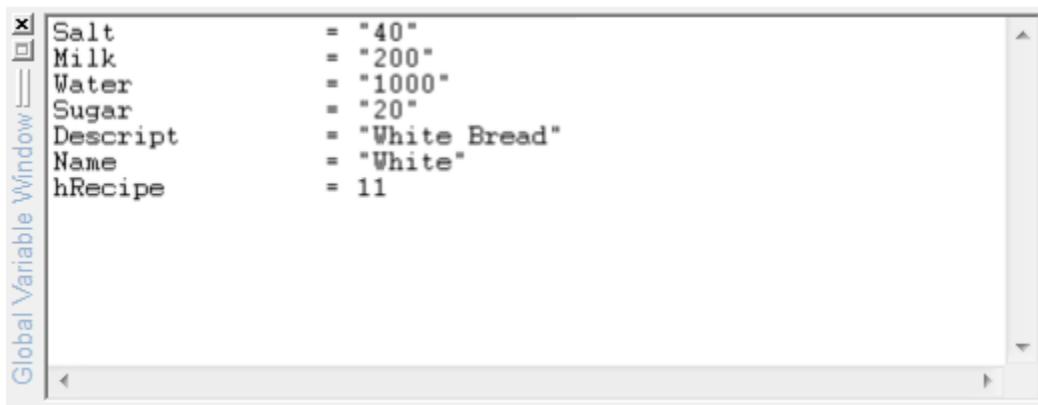
[Docking Windows and Toolbars](#)

[Cicode Editor Windows](#)

## Global Variable Window

The Global Variables Window lists the names and values of all global variables processed to date in the running project during debugging. A global variable is added to the list when it is first assigned a value. Each time the

Global variable is processed, its value will be updated in the Global Variable Window.



**Note:** you need to be in debug mode to view global variable values in this window.

## See Also

[Docking Windows and Toolbars](#)

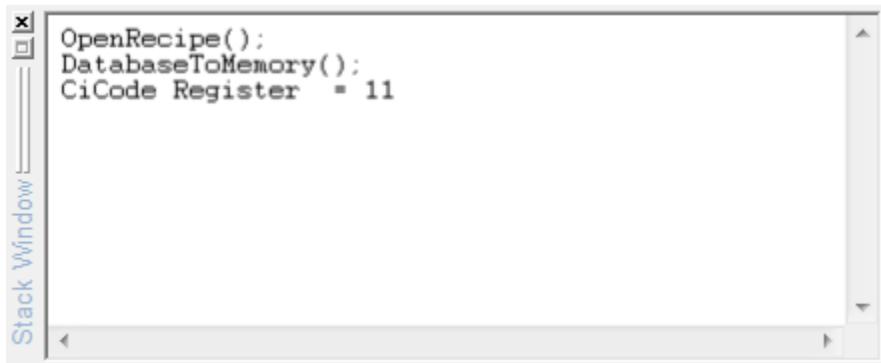
[Cicode Editor Windows](#)

## Stack Window

The Call Stack window lists the stack values of the current thread. The stack consists of the functions called (including the arguments), any variables used in the functions, and return values. This is especially useful during debugging to trace the origin of the calling procedures.

A stack is a section of memory that is used to store temporary information. For example, when you call a Cicode function, the variables used inside the function exist only as long as the function runs.

To view the values of arguments and variables in a procedure, place a breakpoint within the procedure under watch. When that breakpoint is reached, the Stack Window will display the current call stack of the procedure containing the breakpoint. The values of the stack are updated as the values change.



**Note:** you need to be in debug mode to view this window.

## See Also

[Docking Windows and Toolbars](#)

## Cicode Editor Windows

### Thread Window

The Threads History window displays the history for Cicode threads.

Name	HND	User	CPU	State	CPU_Time	Poll...	Slice	Use %
_TagSimulateTask	1	(null)	00	Sleep	00:00:00.127	250	559	0.0
_TagSimulateTask	2	(null)	00	Sleep	00:00:00.100	250	546	0.0
_LibControl_Process	3	(null)	02	Sleep	00:00:10.553	250	14...	4.6
_LibControl_Process	5	(null)	01	Sleep	00:00:07.387	250	24...	6.0
_DspLogo_Process	6	(null)	00	Wait	00:00:00.017	250	11	0.2
_TagSimulateTask	8	(null)	00	Sleep	00:00:00.126	250	542	0.1
_TagSimulateTask	10	(null)	00	Sleep	00:00:00.114	250	545	0.0

The Thread Window has the following fields:

- **Name:** The name of the Cicode thread. This is the name of the function called to start the thread (from the TaskNew() function for example).

If you click on the Name of the Cicode thread, you will make the selected thread the current focus of the Debugger. The Debugger will change the display to show the source of the new thread.

**Note:** If the thread was not started from TaskNew(), the Name shown will be **Command**.

- **Hnd:** The Cicode thread handle.
- **CPU:** The amount of CPU the Cicode thread is currently using, as a percentage of the total CPU usage. Cicode is efficient and this value should be quite small (0-25%). If this value is large it can indicate a problem with the **Note:** you need to be in debug mode to view this window.
- **State:** The state of the Cicode thread. The states are defined as follows:
  - **Ready:** The Cicode is ready to be run.
  - **Sleep:** Suspended using the Sleep() function.
  - **Run:** The thread is running.
- **CPU\_Time:** The total amount of CPU time that the Cicode thread has consumed. This tracks how much CPU time the thread has used over its lifetime.

**Note:** You need to be in debug mode to view this window.

### See Also

[Docking Windows and Toolbars](#)

[Cicode Editor Windows](#)

### Compile Errors Window

The Compile Errors window lists any code errors that have occurred during compile. You can double-click on the file name in the list, to open that code file in the Cicode Editor, and jump to the line of code that caused the compile error.

## See Also

[Docking Windows and Toolbars](#)

[Cicode Editor Windows](#)

## Citect VBA Watch Window

When in debugging mode, you can use the VBA Watch window to watch the value of any VBA variables in the current scope. Click in the **Variable** column and type in the name of the variable under watch. As it comes into scope, its value is updated and appears in the **Value** column.

**Note:** You need to be in debug mode to view this window.

## See Also

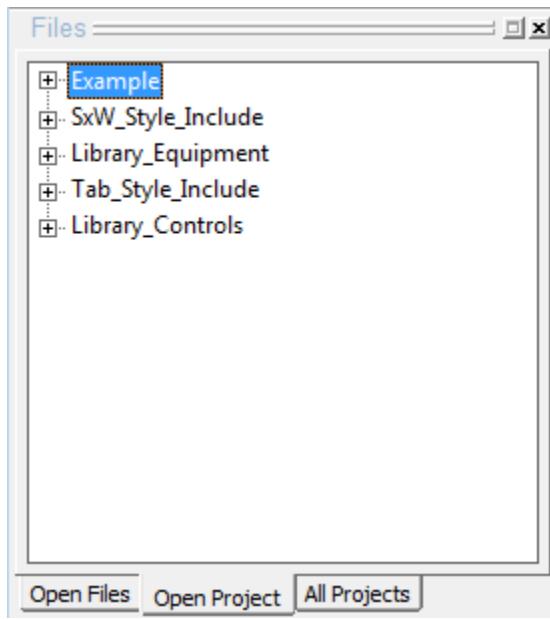
[Docking Windows and Toolbars](#)

[Cicode Editor Windows](#)

## Files Window

The Files window contains three tabs.

- The **All Projects** tab displays a tree hierarchy view of all projects and their Cicode and VBA files available within Plant SCADA project.
- The **Open Project** tab displays a tree hierarchy view of the currently selected project, and all included projects. The currently selected project will be the top entry.
- The **Open Files** tab lists the names of all files currently open for editing in the Cicode Editor.



**Note:** Clicking any of the file names displayed in the tree will open that file in the editor and give it the focus.

## See Also

[Docking Windows and Toolbars](#)

[Cicode Editor Windows](#)

## Cicode Editor Options

The [Cicode Editor](#)'s error handling behavior is controlled through the Cicode Editor Options dialog. This dialog can be used to specify:

- How errors are handled in running Cicode
- The circumstances that will start the debugger
- How the debugger behaves when in debug mode.

There are two tabs on the dialog:

- [Windows and Bars Tab](#)
- [Options Tab](#)

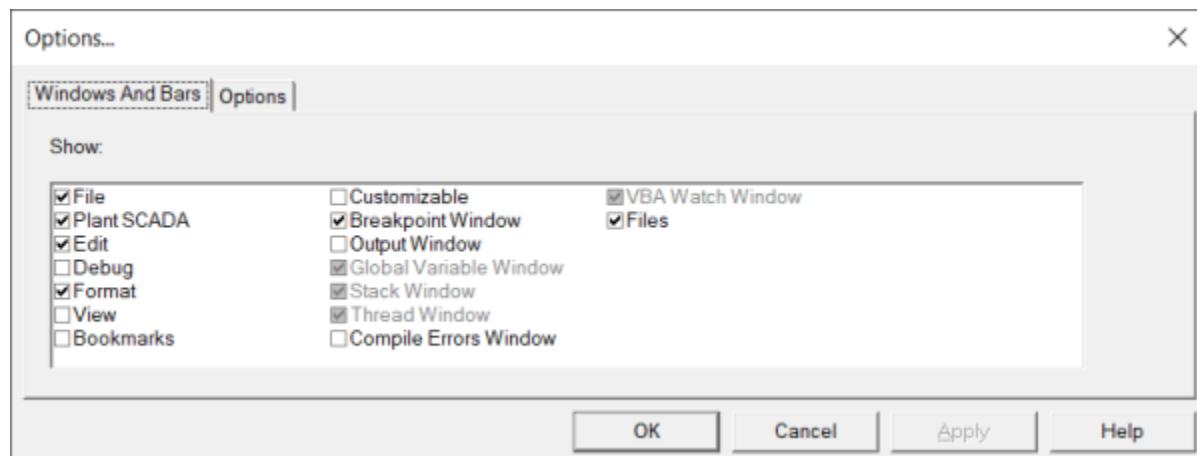
### To view/hide the Cicode Editor Options dialog:

1. Run the Cicode Editor.
2. On the **Debug** menu, select **Options**.  
Or:  
Press Ctrl + T.

## See Also

[Debugging Cicode](#)

## Windows and Bars Tab



The Windows and Bars tab displays the current display state of the listed Toolbars and Debug Windows within

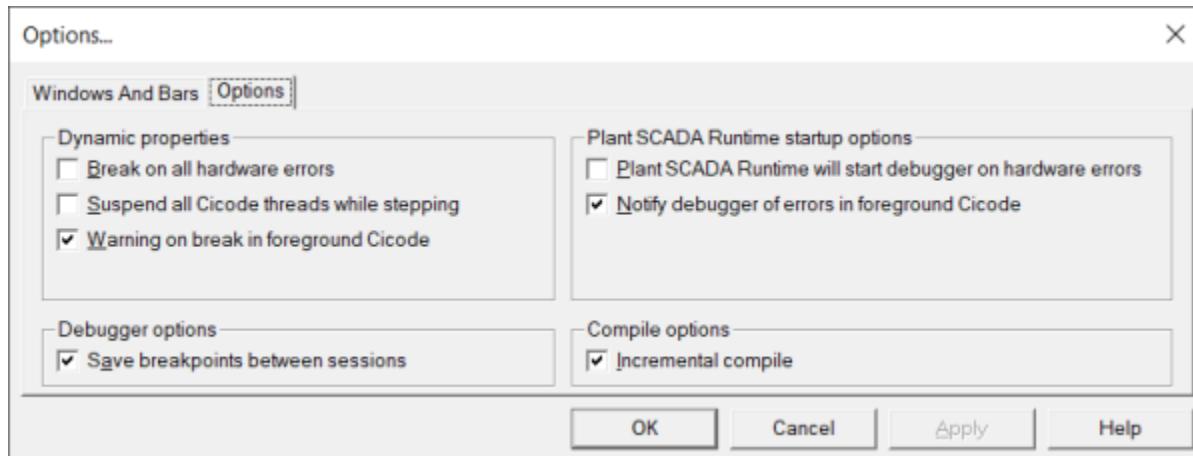
the Cicode Editor. A check mark in the checkbox next to the Window or Toolbar name enables the display of that Window or Toolbar in the Cicode Editor. A grayed-out checkbox indicates that the window is disabled (presently unable to be displayed). For example: Many of the debug windows which display the active state of project Cicode variables are disabled when a Cicode project is not running, and therefore the Cicode Editor cannot be in debug mode).

**Note:** Right-click in the toolbar area to view a menu of available toolbars and debug windows. For a description the buttons, see [Cicode Editor](#).

## See Also

[Cicode Editor Options](#)

### Options Tab



The Options properties tab has the following features:

### Dynamic Properties

- **Break on all hardware errors**

Stops a Cicode thread if a hardware error is detected. A Cicode error will be generated and the thread will terminate (without executing the rest of the function).

- **Suspend all Cicode threads while stepping**

All Cicode threads will be suspended while the debugger is stepping (or when the debugger reaches a breakpoint, or the user performs a manual break). If you try to run any Cicode thread at such a time (by pressing a button at runtime, and so on), the **Command paused while in debug mode** message will display in the runtime prompt line.

This option allows better isolation of any software errors that are detected, especially those that occur when your Cicode thread interacts with other threads. Foreground Cicode cannot be suspended and will continue running when this option is set.

**Note:** This option will help prevent all new Cicode threads from running (including keyboard and touch commands), and should not be used on a running plant.

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use the following option during normal plant or process operations:

- Suspend all Cicode threads while stepping.

This option is only for use during system testing and commissioning.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

- **Warning on break in foreground Cicode**

If a break point is 'hit' in a foreground Cicode task, the **Foreground Cicode cannot break (343)** error message is generated, and will be displayed on the Hardware Alarm page. Disable this option to stop the alarm message from displaying.

## Plant SCADA Runtime Startup Options

- **Plant SCADA Runtime will start debugger on hardware error**

Plant SCADA will automatically start the debugger when a Cicode generated hardware error is detected. The debugger will display the Cicode source file, and mark the location of the error.

**Note:** This option will interrupt normal runtime operation, and should only be used during testing and commissioning of systems.

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use the following option during normal plant or process operations:

- Plant SCADA will start debugger on hardware errors.

This option is only for use during system testing and commissioning.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

- **Notify debugger of errors in foreground Cicode**

Plant SCADA will automatically start the debugger if an error is detected in a foreground task. The debugger will display the Cicode source file, and mark the location of the error.

This option is overridden by the **Plant SCADA Runtime will start debugger on hardware errors** option. That is, if the above option is disabled, then this option is disabled also.

**Note:** Foreground Cicode cannot be suspended. The break point will be marked, but you will not be able to step through the function.

## Debugger Options

- **Save breakpoints between sessions**

Save the location and states of breakpoints between running sessions of the Cicode Editor and Debugger. This means breakpoints inserted using the Cicode Editor can later be recalled when an error is detected - even though the file (and application) has been closed.

## Compile Options

- **Incremental compile**

Enables the incremental compilation of the project.

## See Also

[Cicode Editor Options](#)

## Docking Windows and Toolbars

The windows and toolbars of the [Cicode Editor](#) can be docked or free floating within the editing and debugging environment.

## Toolbars

Toolbars are docked by default within the toolbar area at the top of the Cicode Editor.

To show/hide a toolbar, right-click within the toolbar area and make a selection from the menu that appears.

## Windows

Windows are docked by default in the document display area of the Cicode Editor (beneath the toolbar).

- **Docked windows** are those that fit within the Cicode Editor display area by docking (attaching) themselves to an internal edge of the display area. Docked windows cannot be resized manually, and will share the display space with the Editor toolbars and other docked windows.
- **Floating windows** are those that are not docked to the editor, nor are necessarily constrained by the editor boundaries. Floating windows can be resized manually, and are subject to layering (Z-order), in which they can be partly or wholly obscured by another window.

Windows and toolbars can be moved about in the Cicode Editor environment by clicking and dragging the titlebar of a window. Docking behaviour is by default, and can be overridden by holding down the CTRL key during the drag-and-drop to force the window or bar to be free floating.

The position of the mouse during the drop action determines which side the window or toolbar docks to. Docking outlines of the window or toolbar are displayed with gray lines during the drag action to indicate the potential docked position.

## See Also

[Cicode Editor Windows](#)

## Debugging Cicode

To help you locate Cicode errors, you can switch the Cicode Editor to debug mode to analyze running Cicode. You can toggle debugging on and off as required, but Plant SCADA needs to be running for the debugger to work.

---

**Note:** The Cicode Editor cannot debug foreground Cicode. A break in a foreground Cicode will result in the **Foreground Cicode cannot break** message.

---

### To switch to debug mode:

1. Run the Cicode Editor.
2. From the **Debug** menu, select **Start Debugging**.

Or:

Click the **Toggle Debug** button on the toolbar.

Or:

To start a debugging session without the project being recompiled, select **Start Debugging Without Compile** from the **Debug** menu. You can only use this option if you have already compiled the project at least once.

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Do not use the following command to start Plant SCADA if you have uncompiled project changes that are necessary for the safe operation of your plant and process.

- Start Debugging Without Compile

This option is only intended to assist debugging an already compiled project.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

---

If the current project is not running, Plant SCADA will automatically launch it. The bug icon in the bottom right-hand corner is green when debugging.

---

**Note:** You can use the parameter [\[Debug\]CodeDebug](#) to enable Cicode debugging for a specific server process. You cannot debug a process that is running as a Windows™ service.

---

## Debugging a Function

### To debug a function:

1. Run the Cicode Editor.
2. Open the file containing the function you wish to debug.
3. Switch to debug mode (see above).

If the current project is not running, Plant SCADA will automatically launch it. The bug icon in the bottom right-hand corner is green when debugging.

4. Insert a breakpoint where you want to start debugging.
  5. From the **View** menu, select any debug windows you want to use. If you are unsure, you can use them all.
  6. Initiate the thread by calling the function. You can do this directly from the Cicode window in the Kernel, or by using a function, etc.
- The function will break at the specified breakpoint. You can then use the step tools to step through and trace your function.
7. Click the Toggle Debug button to stop debugging, or choose **Debug | Stop Debugging**.

## Using Breakpoints

To debug a function, you need to first be able to stop it at a particular point in the code. You can place a breakpoint on any line in the source code functions. Breakpoints may be inserted or removed while editing or debugging without the need for them to be saved with the file.

### To insert/remove a breakpoint:

1. Open the Cicode Editing window.
2. Position the cursor on the line where you want the breakpoint to be placed or removed.
3. Click the Debug indicator bar. Alternatively, you can press **F9** or choose **Debug | Insert/Remove Breakpoint**.

The breakpoint appears as a large red dot at the beginning of the line.

You can also enable or disable breakpoints you have inserted into your Cicode.

### To enable/disable a breakpoint:

1. Open the Cicode Editing Window.
2. Position the cursor on the line where the breakpoint is located.
3. Press **Ctrl + F9**, or choose **Debug | Enable/Disable Breakpoint**.

A disabled breakpoint appears as a large dark gray (disabled) dot at the beginning of the line.

---

**Note:** You can also use the **DebugBreak** Cicode function to halt a process thread.

---

## Start Debugging on Hardware Error

For a detected hardware error to halt a function, you need to have the following options enabled:

1. **Break on all hardware errors.**  
Or:  
2. **Break on hardware errors in active thread.**

These are available on the **Options** tab of the Cicode Editor Options dialog (**Debug | Options**).

When the break occurs, the default Cicode Editor will be launched (if it is not open already), with the correct code file, function, and break point line displayed. To launch the debugger in this case, you also need to have the **Plant SCADA will start debugger on hardware errors** option set.

## Function Error Handling

Once you have halted a thread, the debugger marks the position in the code with an arrow. Now you can step through the function, line by line, and watch what happens in the debug window (see below). The following tools are provided in the Cicode Editor, to control stepping through functions.

Step Into	Advances the current Cicode thread by one statement. If the statement is a user defined function, the debugger steps into it (the pointer jumps to the first line of the source code).
Step Over	Advances the current Cicode thread by one statement. If the statement is a user defined function, the debugger steps over it (the function is not expanded).
Step Out	Advances to the end of the current function and return. If there is no calling function, the thread terminates.
Continue	Re-starts normal execution of the current Cicode thread. If there are no more breaks, the thread terminates normally.

## Cicode Errors

This section describes the error codes used by Plant SCADA in the following situations:

- [Hardware/Cicode Errors](#)
- [Cicode and General Errors](#)
- [MAPI Errors](#)

### Hardware/Cicode Errors

Plant SCADA 'traps' system errors automatically. When Plant SCADA detects a system error, it generates a hardware alarm, and the corresponding error message is placed in the alarm description. Each error has an associated (unique) error number.

You can use the IsError() function to get the number of the last error. Alternatively, you can trap and process errors within your user functions. Use the ErrSet() function to enable or disable error trapping. Plant SCADA 'traps' system errors automatically. When a system error occurs, Plant SCADA generates a hardware alarm, and the corresponding error message is placed in the alarm description. Each error has an associated (unique) error number. Number 0 means no error, errors start from 1.

Range	Source	Cause
1 - 31	PLC or I/O Device Generic errors	The I/O device is reporting an error, or Plant SCADA is experiencing the reported error trying communicate

Range	Source	Cause
		with an I/O device. Often caused by incorrect configuration or poor cabling. For a detailed list of these errors, see the <i>Error Messages</i> topic in the <b>Topology   I/O Devices   Troubleshooting Device Communications</b> section of the Plant SCADA help.
256 -511	General	<p>General errors are wide ranging, from animation to server problems. However, there are two main causes of general errors:</p> <ol style="list-style-type: none"> <li>1. Device External devices such as printers, databases, and files can cause many different hardware errors since they are beyond the control of Plant SCADA. Often the device itself is improperly configured or non-existent.</li> <li>2. Cicode Cicode errors are generated when your project configuration calls a Cicode function in an invalid way, or when a Cicode function returns an error or does illegal operations. For a detailed list of these errors see <a href="#">Cicode and General Errors</a>.</li> </ol>
382 - 383	Invalid tag data	This will indicate that the tag data validation process has identified a discrepancy with the tag index values. For more information, see the topic <i>Validating Distributed Project Data for Tag-based Drivers</i> in the <b>Topology   I/O Devices</b> section of the Plant SCADA help.

You can use the `IsError()` function to get the number of the last error. Alternatively, you can trap and process errors within your user functions. Use the `ErrSet()` function to enable or disable error trapping.

## Cicode and General Errors

The table below describes the general errors in Cicode.

Error No.	Error title	Description
<b>0</b>	No Error	No error
<b>32</b>	The unit the tag is on has not been connected to yet	When the error for a tag has specific values the tag test can show "#WAIT" for this error. This occurs for scheduled or Dial-up devices.
<b>53</b>	The tag is not known by the server being talked to at this time	The reason for a tag being unknown by the I/O or alarm system may not be that it does not exist, just that the server it is on is yet to come up and tell us about it. However, more often than not, it is because it does not exist.
<b>55</b>	Tag value is limited	Tag value is limited to be within range.
<b>256</b>	General software error	An internal Plant SCADA software error is detected. Contact Technical Support for this product and provide details on what causes the error.
<b>257</b>	Value is out of range	A numeric value is out of range. An out-of-range value has been passed to a function, or an array index is off the end of an array, or a value that is outside of the specified engineering scale has been assigned to a I/O Device variable. You can disable range checking on PLC variables with the CodeSetMode() function.
<b>258</b>	Buffer has been overrun	A buffer has been overrun. More data has been passed to a function than it can write to its temporary buffers. Try again by calling the function twice, with half the data in each call.
<b>259</b>	Array has been overrun	An array passed to a function is too small for the data requested. Define a larger array or reduce the maximum data size requested.

Error No.	Error title	Description
<b>260</b>	Path does not exist	The specified path to a device or file does not exist. During a function call (that tried to open a file), a non-existent path was specified. Call the function again with the correct path.
<b>261</b>	File does not exist	The specified file or device does not exist. During a function call (that tried to open a file), the file could not be found. Call the function again with the correct file name. This error will also be detected if you try to call TrnDelHistory() on a file that has not been added using TrnAddHistory().
<b>262</b>	Cannot open file	The specified file cannot be opened. During a function call (that tried to open a file), the file could not be opened. There may be a mode error (for example, from trying to open a read-only file in write mode), or the file may be open by others, or the operating system resources may be too low to open the file.  Check that the file does exist (use File Manager), and that you have the correct rights to open it. (Check with your network supervisor that you have correct rights to open the file).
<b>263</b>	Cannot read file	The specified file cannot be read. Either an error has been detected during a read operation, or the end of file was unexpectedly found, or a disconnection from the file server occurred, or the operating system is out of resources.
<b>264</b>	Cannot write to file	The specified file cannot be written to. During a function call (that tried to write to a file), a write error has been detected. There could be a

Error No.	Error title	Description
		disk full error, or a disconnection from of the file server may have occurred, or the operating system is out of resources.
<b>265</b>	Invalid file type	An attempt was made to open a file of the wrong type, for example, you tried to open an ASCII file as a dBASE file.
<b>266</b>	Field not found in file	The specified field does not exist in the device or database. A function that is trying to access an individual field in a database cannot find that field. Check that you have specified the correct field name and database name.
<b>267</b>	File mode is invalid	An operation has been attempted on a file or device that is of the wrong mode, for example, you tried to perform a seek on a printer device.
<b>268</b>	Key not found in file	The requested key was not found when a key search was performed on a database device, that is the record specified on an indexed search cannot be found. Either the record does not exist or you have specified the wrong key.
<b>269</b>	Bad handle specified	A bad handle has been passed to a function. You have called a function that requires a device handle, font handle, window handle, etc., but you passed a number that does not associate with a read device, font, or window (for example, you called WinGoto(100) when no window with the handle "100" exists). Check where the handle or number was retrieved from, and verify that it is the same handle. This error may also be detected if you have closed or destroyed the resource and you then try to access it.

Error No.	Error title	Description
<b>271</b>	No more free handles left	All the available file handles have been used, that is too many files or databases are open at the same time. Open fewer files at one time or increase the number of file handles in the [CtEdit]DBFiles parameter.
<b>272</b>	Out of memory	Plant SCADA is out of memory. Increase the amount of memory in the computer or use smaller databases.
<b>273</b>	Divide by zero	An attempt has been made to divide a number by zero.
<b>274</b>	Invalid argument passed	An invalid argument has been passed to a Cicode function. This is a general error message and is generated when arguments passed to a function are out of range or are invalid. Check the value of arguments being passed to the function. If arguments are input directly from the operator, you should check that the correct arguments are being passed to the function.
<b>275</b>	Overflow	A calculation has resulted in a numeric value overflow. Check for operations that will generate large numbers.
<b>276</b>	No privilege for operation	A user has requested an operation for which he or she has no privilege.
<b>277</b>	Not in correct area	A user has requested data that does not belong to the current user area.
<b>278</b>	Report already busy	A request has been made to run a report that is already running. You can get the current state of a report with the RepGetControl() function. You can ignore this error message (because the report is already

Error No.	Error title	Description
		running).
<b>279</b>	Report is late for execution	The report cannot run at the rate requested in the configuration. An attempt could have been made to run a report too frequently, and the required data cannot be read from the I/O Device(s) in time for the next report.
<b>280</b>	Invalid report ID specified	The specified report name does not exist, or the user has no privilege to run the report, or the report is not in the current user area. Check the name of the report and the current user's privilege and areas.
<b>281</b>	No server could be found	The specified Plant SCADA server cannot be found. Either the server is not running or there is some disturbance in the network. Check that the network is set up correctly, and you are using the same Server Name on both the client and server.
<b>282</b>	Foreground Cicode cannot block	You cannot block the foreground Plant SCADA task. You may have called a blocking function from one of the Page animation databases.
<b>283</b>	Trend has missed samples	The trend cannot run at the rate requested in the configuration. An attempt could have been made to trend the data too frequently, and the required data cannot be read from the I/O Device(s) in time for the next trend. Either increase the performance of the communication link to the PLCs or slow the rate of trend data acquisition.
<b>284</b>	Device is disabled	An attempt was made to access a device that is disabled. You can disable any devices (printers and other logging devices) with the DevDisable() function. When Plant SCADA (or your Cicode function)

Error No.	Error title	Description
		tries to access a disabled device, this message returns and all output is lost.
<b>285</b>	Foreground Cicode run is too long	The foreground Cicode task is taking too long to animate the display page. The Cicode is too complex and is taking too long to execute. Simplify the Cicode that is animating the page, or increase the [Code] TimeSlice parameter. If you cannot simplify the Cicode, you can create a separate task using TaskNew() to calculate your complex operation, and then use the Display functions to display the results. Cicode running from a TaskNew() call is in background mode and can run as long as required.
<b>286</b>	Out of Cicode threads	Plant SCADA has run out of Cicode tasks. Run fewer tasks (for example, reports, key commands, and Cicode tasks) in parallel, or increase the number of tasks with the [Code]Threads parameter. This error can be caused by a configuration error if you keep creating tasks that do not finish.
<b>287</b>	Floating point exception trap	An invalid floating-point number has been found. Check the floating-point data from the I/O Device.
<b>288</b>	Out of buffers	Plant SCADA is out of dynamic buffers. You have called a function that requests buffer space but no buffers are available. Check which function is causing the error and increase the associated buffers, or slow the rate of transfer to that function. If the error occurs on a server or LAN device, increase the number of buffers with the [Lan]ReadPool parameter.  This error can also be detected if

Error No.	Error title	Description
		<p>something is stopping the release of the buffers, for example, if network communication has stopped or a PLC has just come off-line. The error 'Out of buffers' can also be generated in the following ways:</p> <p>Calling QueWrite() when the queue functions have run out of buffers. You can increase the number of queue buffers with the [Code] Queue parameter.</p> <p>Calling WinFree() to free the last Cicode window. If WinFree() did free the last window, Plant SCADA would have no windows left.</p> <p>To verify which function is causing the hardware error, display the {ERRPAGE} and {ERRDESC} fields on the hardware alarm.</p>
<b>289</b>	Name does not exist	The specified name does not exist in this context. You are probably using the wrong name.
<b>290</b>	Not finished	A request has been made for trend data that has not yet finished trending.
<b>291</b>	File not Plant SCADA format	The specified file is not in Plant SCADA format. The file (trend, graphic, or any other file) is in an invalid format. Check that the name of the file is valid or that the file has not become corrupted.
<b>292</b>	Invalid function	The specified function name does not exist. You have tried to create a task, or called a remote procedure, or set an event function that does not exist.
<b>293</b>	File error	A general file error has been detected. Either a general hardware error has occurred, or the operating system is out of resources, or the file server is

Error No.	Error title	Description
		down.
<b>294</b>	File EOF	The end of the file was found. An attempt was made to read data off the end of a file or database.
<b>295</b>	Cicode stack overflow	<p>A Cicode evaluation stack overflow has occurred. There are too many local function variables or nested function calls. Reduce the number of local variables or increase the [Code] Stack parameter.</p> <p>The Cicode stack is used to store local function variables and function calls. If you have many nested functions and a large number of local function variables, the Cicode stack may overflow. When the Cicode stack overflows, the Cicode that caused the overflow is halted.</p> <p>You can estimate the size of the stack by counting the maximum number of local function variables in the deepest function calls. For example, if function A has 10 variables and calls function B with 30 variables, which calls function C with 40 variables, the stack needs to be <math>10 + 30 + 40 = 80</math> deep.</p>
<b>296</b>	Queue empty	An attempt has been made to read an element from an empty queue.
<b>297</b>	Semaphore owner died	The owner of a Cicode semaphore was halted, killed, or returned without releasing the semaphore. Reset the shared resource back to a known state (because the task that died may have left it in a mess), and then continue. For example, if you are sharing a printer, do a form feed.
<b>298</b>	Semaphore timeout	The requested semaphore was still in use after the specified timeout. Either try to get the semaphore

Error No.	Error title	Description
		again or abort the operation and tell the operator of the detected error.
<b>299</b>	Cancelled	The specified form or command was cancelled. This error is returned when a user presses the <b>Cancel</b> button on a form. The normal procedure is to abort the operation.
<b>300</b>	Trend not found	The trend does not exist at the specified AN and page. A trend function may have been called when the trend is not defined for that AN.
<b>301</b>	Trend pen not found	The required trend pen name does not exist in the Trends database or is not in the current user area. Check that the pen name exists and check the current user's privilege and area.
<b>302</b>	Trend data not valid	The requested trend data is not valid. Either the I/O Device data was bad, or the Plant SCADA trend server was shut down, or the trend data was disabled.
<b>303</b>	Invalid animation number	The AN specified in the function is not defined. You called one of the DspXXX animation functions, but you specified an animation number that was out of range or that had been deleted.
<b>304</b>	File server inoperative, stand-by active	Plant SCADA has detected that a file server has become inoperative, and will switch to the standby file server. The file server's inoperative condition is due to errors of the network or of the file server computer. This error is displayed only if you have enabled redundant file servers. If a redundant file server is not enabled, Plant SCADA and Windows become inoperative

Error No.	Error title	Description
		when the file server becomes inoperative. You should report this error to the operators to fix the file server.
<b>305</b>	Conflicting types of animation	The same AN is being used for two different types of animation. This error is detected if you try to display two (or more) incompatible types of animation on the same AN (for example, you try to display a symbol at a AN where a bar graph is already displayed). Check the configuration. If you need to display a new animation, you need to first delete the old animation with the DspDel() function.
<b>306</b>	SQL field value truncated	A maximum of 1000bytes (1Kb) can be returned from a single field call. If the field data is larger than this limit, it is truncated. You have tried to access a database where one of the fields is greater than 1000 bytes in size. Change the database field size to less than 1000 bytes so it can be accessed. In fact, you should change the field size to less than 256 bytes, the maximum allowable length of a Cicode string.
<b>307</b>	SQL database error	A general SQL error. Call the SQLErrMsg() function for details of the detected error.
<b>308</b>	SQL null field data returned	Data has been requested from a field that contained no data, or the SQL server does not support this type of field data. Plant SCADA will return an empty string. Call the SQLFieldInfo() function to list the fields in the database.
<b>309</b>	Trend data is gated	You have requested trend data that was gated (set to logging disabled) by the trigger expression (that is when it was acquired). The data is

Error No.	Error title	Description
		returned with the gated values set to 0.
<b>310</b>	Incompatible server version	Two servers are running incompatible versions of the Plant SCADA software. Install the latest version on each server. Contact Technical Support to arrange for an upgrade.
<b>311</b>	Alarm tag synchronize error	When the Alarm server shuts down it writes an alarm save file. If the alarm server is in tag mode (rather than record mode) this message will display. You can set the mode with the [Alarm]SaveStyle parameter. You can ignore this message as it is an alert only.
<b>312</b>	MAPI generic error	A generic MAPI error has been detected. Call the MailError() function to retrieve the MAPI error.
<b>313</b>	No MAPI	The MAPI mail system is not installed, or incorrectly installed on the computer.
<b>314</b>	MAPI offline	The computer is not logged on to the MAPI mail system. Call the MailLogon() function to log on to the MAPI mail system.
<b>315</b>	MAPI no mail	No mail was available. This message is returned from the MailRead() function if no mail is available.
<b>316</b>	dBASE record locked by another	The dBASE file is being accessed by another user. Check if the dBASE file has been opened in exclusive mode by the other user. This error can also be detected if another user is updating the dBASE file, and will usually occur if it is an indexed database, and the file is on a slow file server. You can adjust dBASE access with the [General]LockRetry and [General]LockDelay

Error No.	Error title	Description
		parameters.
<b>317</b>	Not in this version	The operation is not supported in this version of Plant SCADA or view-only client. You need to upgrade to a higher version of Plant SCADA or install a control client.
<b>318</b>	Invalid page function	You have called the PageGoto(), PageNext(), PagePrev(), PageDisplay(), or PageLast() as an exit command in the Pages database.
<b>319</b>	Low physical memory	Plant SCADA is low on physical memory. Increase available physical memory (not virtual memory). Reduce the size of SMARTDRV cache, close any other windows programs that are running, or add more RAM to your computer. You can set the minimum size of memory required by Plant SCADA with the [Memory]MinPhyK parameter. This parameter sets a value for the minimum physical memory before Plant SCADA will generate this error message.  This error may also be detected if your swap file is large (that is greater than 20 Mb). Reduce the size of your swap file. The swap file is configured with the Windows Control Panel (386 Enhanced icon).
<b>320</b>	Cannot free window	The WinFree() function has been called but Plant SCADA has no windows left. (Be aware that the last window and any child windows owned by the last window cannot be removed.)
<b>321</b>	Font cannot be found	The specified font cannot be found. Check the font name.
<b>322</b>	Cannot connect to LAN	Plant SCADA has detected an error on the network.

Error No.	Error title	Description
<b>323</b>	Super Genie not Associated	A Super Genie variable has not been associated correctly. This error can be detected if a variable passed to the Super Genie is the wrong data type or the variable does not exist. The error will also be detected if the Ass() function has not been called for the variable.
<b>325</b>	Project is not compiled	Changes have been made to the project while the system was running. Either restart the system or shutdown and re-compile.
<b>326</b>	Could not run the Plant SCADA compiler	The Plant SCADA compiler could not be found. Either the computer has run out of memory, or the compiler has been removed from its directory.
<b>327</b>	User type not found	An attempt was made to create a user of a type that has not been defined in the users database.
<b>328</b>	User already exists	An attempt was made to create a new user with the same name as an existing user.
<b>329</b>	Cannot have mixed trends	An attempt was made to display both a periodic trend and an event trend in the same trend window. Check the project configuration (Trend Tags and Page Trends databases) for mixed trends displayed in a trend window.
<b>336</b>	Event type trend is expected	One of the arguments passed to this trend function is only valid for event type trends.
<b>337</b>	Trend in file does not exist	The trend name inside the trend file does not exist in the trend database. It is likely that the trend file belongs to a trend which is deleted from the system configuration.

Error No.	Error title	Description
<b>338</b>	Plot Functions Sequence Mismatch	Plot functions are to be written in sequence since they depend on the data set up by other plot functions. Please refer to the description section for each Plot Function for the order of plot functions.
<b>339</b>	Plot Marker is not Defined	An undefined plot marker symbol has been used. Use the PlotSetMarker function <b>before</b> the PlotLine, PlotXYLine or PlotMarker functions.
<b>340</b>	Invalid subgroup size	The subgroup size specified for this SPC trend is not valid.
<b>341</b>	Trend Cursor Disabled	The trend cursor is currently disabled.
<b>342</b>	Debug break	The DebugBreak() Cicode function has been called. This indicates an invalid condition detected in user written Cicode. Enable the Cicode debugger to find the cause of the problem.
<b>343</b>	Foreground Cicode cannot break	A breakpoint has been hit in foreground Cicode. Foreground Cicode cannot be blocked. You can disable this error message in Debug Options, accessed through the Debug menu in the Cicode Editor.
<b>344</b>	Format overflow	This error occurs when the string being used to return an error message is not long enough for the information to go into it.
<b>345</b>	Data not ready	This error is returned when the requested data is not ready to be returned. Try again later.
<b>346</b>	Dynamic Out of licence points	The dynamic point count has exceeded the point limit. See Plant SCADA Licence Point Count.
<b>347</b>	Assertion did not succeed in user Cicode	An assertion in your Cicode has been proven FALSE, and the task

Error No.	Error title	Description
		terminated. Assertions are made using the Assert() function. If you set the [Code]DebugMessage parameter to 1, the assertion is logged and the operator prompted.
<b>348</b>	Property does not exist	The tag property does not exist.
<b>350</b>	RDB file not found	A RDB file is not found. Try a full compile and re-start.
<b>353</b>	Cannot connect to FTP	Problems connecting to the FTP server for file copy. Check that the service is running correctly by using a 3rd party tool outside Plant SCADA.
<b>354</b>	Unrecognized object class	The automation object to be used is not known or registered.
<b>355</b>	Object has no interface	The automation objects interface no longer exists. This is either a logic or code error.
<b>356</b>	Object automation exception	Generic automation exception code for logging issues.
<b>357</b>	Too many arguments	Formatting issues likely due to too many arguments (automation)
<b>358</b>	Too few arguments	Less arguments available than expected (automation)
<b>359</b>	Named object already exists	An object tried to be created when it was already existed and it was not expected to exist. Check your Cicode (automation)
<b>360</b>	Unrecognized named object	The name of an object did not match the internal records. Check your Cicode (automation).
<b>361</b>	Page CTG/RDB record mismatch	An animation object id was not found in the records. Try a full compile and re-start.
<b>362</b>	Object event queue flooded	Plant SCADA has run out of Cicode tasks while processing ActiveX events. Create fewer concurrent

Error No.	Error title	Description
		ActiveX events, or increase the number of tasks with the [Code]Threads parameter.
<b>363</b>	Incorrect number of arguments	Internal error in Cicode handling of objects. Try a full recompile and restart.
<b>364</b>	No 'this' argument	Internal error in Cicode handling of objects. Try a full recompile and restart.
<b>367</b>	File cannot be printed	Unable to print the specified file on the specified port.
<b>368</b>	Animation number invalid	Unable to display the animation at the given AN.
<b>369</b>	Wrong type for text display	Unable to display the given animation as text.
<b>370</b>	No FileFind instances to close	There are no FileFind instances to close.
<b>371</b>	No dialback response returned	The retry count for the dialback response has been exceeded.
<b>372</b>	Unrecognised ActiveX object	The ActiveX graphics data was unable to be loaded as it is an unrecognised object.
<b>374</b>	Date Time Conflict	A Date and/or time conflict has been detected. If you are attempting a TrnAddHistory(), verify that the file you are adding does not have a conflicting time or date with existing trend files.
<b>375</b>	Password Expired	The password has expired. Change password or adjust the [General]PasswordExpiry settings.
<b>376</b>	Error Writing to Files of Trend	An error has occurred while attempting to write to a trend file.
<b>377</b>	Error Reading Files of Trend	An error has occurred while attempting to read from a trend file.

Error No.	Error title	Description
<b>379</b>	Cannot modify field	The Users.DBF cannot be modified by direct user manipulation.
<b>380</b>	Name Exists	The user name specified already exists in the Users.DBF.
<b>381</b>	File Format Error	The file is not in the desired format. It may be corrupted.
<b>382</b>	Page data / variable tag data mismatch	Page data / variable tag data mismatch with tag-based driver.
<b>384</b>	The code this queue/semaphore was waiting for exited due to an error	Cicode task has been flagged as locked or inactive. This will not apply if a Cicode task was killed deliberately by the Cicode logic.
<b>385</b>	Cannot download websignature.xml	Unable to download the webclient signature file.
<b>388</b>	OID check disabled	Attempting to open a DBF file without checking to see if it has changed.
<b>391</b>	Unsupported Cicode function in Web Client	A number of Cicode functions can only be run in a non Web context. This error is flagging that such a function was tried from a Web client.
<b>392</b>	Timeout from Rnd Alarm Server	The timeout for receiving alarm data from the redundant alarm server has been exceeded.
<b>393</b>	RDB and associated page mismatch	The RDB page version does not match the one expected after compilation.
<b>394</b>	Remote license lost	The remote licence can not be found.
<b>400</b>	Project or file is Read-Only	The operation could not be completed because the project or file is read only.
<b>401</b>	Redundant server not found	The redundant server is current not available or was not found.
<b>402</b>	Cluster not specified	Cicode call is ambiguous because

Error No.	Error title	Description
		no cluster was specified.
<b>403</b>	Cluster not found	The provided cluster name was not seen as a valid cluster name.
<b>404</b>	Cluster name and tag mismatch	A cluster name and a tag with a cluster name prefix were both passed, but they do not match each other.
<b>405</b>	Cluster not connected	The operation cannot continue as the cluster is not connected.
<b>406</b>	Cluster already connected	The operation cannot continue as the cluster is already connected.
<b>407</b>	Databrowse pending	Databrowse session currently being connected and is not yet available for further commands.
<b>408</b>	Databrowse not supported	This normally means that data has been requested from the wrong client / server type, that is a client request on a server or vice versa.
<b>409</b>	Databrowse type not found	Browse type unknown. Check that the versions of citect32 are the same.
<b>410</b>	Databrowse session not found	iSession handle is invalid.
<b>411</b>	Databrowse session exists	Cannot open a session as it already exists.
<b>412</b>	Databrowse session EOF	Cannot browse beyond end of file. No more records left in the browse.
<b>413</b>	Databrowse field not found	The specified field is not present in the file.
<b>414</b>	Databrowse invalid field	The specified field is not valid for this browse function.
<b>415</b>	Databrowse no command	Browse command unknown. Check that the versions of citect32 are the same.
<b>416</b>	Databrowse no next cluster	Internal flag that all clusters have been processed. Users use this in

Error No.	Error title	Description
		Cicode to determine that there are no more clusters.
<b>417</b>	Cluster required for operation	The operation cannot continue as the cluster is required for operation of a server.
<b>418</b>	No server of type on cluster	There is no server of the required type configured on the server.
<b>419</b>	Client / Server ID mismatch	An ID given to a Cicode function is not of the correct type.
<b>420</b>	Not available outside process	This functionality is not available outside a process boundary for example, AlarmFirstCatRec is only able to be called from inside the Alarm Server process.
<b>421</b>	Invalid tran handle detected	Plant SCADA's internal communications subsystem detected an error while trying to send a message.
<b>422</b>	Dial call did not succeed	An attempt to make a call to a remote I/O device did not succeed. You can find more information on the cause of this problem (if it is reproducible) by setting [Dial]Debug=1 and [Dial]DebugLevel=3 and looking in the syslog.dat.
<b>423</b>	Waiting for initial data	Occurs when attempted to read the value before the initial value has been retrieved from the device.
<b>424</b>	Tag not found	Occurs when attempt to read or subscribe to a tag that does not exist.
<b>425</b>	No connector	Not used.
<b>426</b>	Write to named pipe did not succeed	An attempt to send a message via a name pipe has did not succeed.
<b>427</b>	Redirect from non-Cicode task	A function cannot be redirected to another component as a proxy RPC unless it is in the context of a

Error No.	Error title	Description
		Cicode call.
<b>428</b>	Data browse record is invalid	The specified record is not valid for this browse function.
<b>429</b>	Can't plot symbol to printer	The Cicode PlotMarker() function can only plot a symbol to the display. This error occurs when the PlotMarker() Cicode function is used with a user-defined marker and a printer was specified as the output when PlotOpen() was called.
<b>430</b>	Alarm sort parameters mismatch	The [Alarm]SortMode parameter value on client is different as compared to the value on the alarm server. This value should be same in order to display alarms in correct order on the client.
<b>431</b>	Summary sort parameters mismatch	Values for [Alarm]SummarySort and [Alarm]SummarySortMode parameters on client are different as compared to the values on the alarm server. These values should be same in order to display alarm summary in correct order on the client.
<b>432</b>	Property not ready	A specific tag property has been read in synchronous mode, before the value has been retrieved from the server. (similar to subscription value is pending)
<b>433</b>	Cicode type mismatch	Some server side changes are now allowed for Cicode, without restarting the client. This means that now it is possible to change a tag type to one that now cannot be converted as it was before. Eg val = TagA - TagB, If TagA has been converted from Integer to String, then the Cicode will raise a Cicode type mismatch as the '-' operation is not supported for strings

Error No.	Error title	Description
<b>434</b>	Invalid data conversion	Attempt to convert a value to a type that cannot store the value for example, ULong to Long, and the value is greater than the Long type can handle.
<b>436</b>	User password invalid	The password is incorrect. Check that the user name and password are typed in correctly.
<b>437</b>	Unable to load security provider	Cannot load or initialize the Windows security provider. This indicates that the installation of operating system may be corrupted or the Windows security provider was not included.
<b>439</b>	Authentication did not succeed	Cannot authenticate the given Windows user name and password. Check that the user name and password are typed in correctly
<b>440</b>	Authentication session does not exist	Cannot find the authentication session during the authentication process for the Windows user. This indicates that there was miscommunications in between client and server.
<b>441</b>	Role checking did not succeed	Cannot perform the role-check process for the Windows user. Check that the roles database in the project is not corrupted.
<b>442</b>	No linked role	The given Windows user is not linked to any role defined in the project.
<b>443</b>	Acquire user credentials did not succeed	Cannot obtain the user credential handle from Windows.
<b>444</b>	Acquire user access token did not succeed	Cannot obtain the user access token. This indicates that the security context of the Windows user was invalid.
<b>445</b>	Encrypt data did not succeed	Cannot encrypt the user information during the Windows

Error No.	Error title	Description
		user authentication process. This indicates that the security context of the Windows user was invalid.
<b>446</b>	Decrypt data did not succeed	Cannot decrypt the user information during the Windows user authentication process. This indicates that the security context of the Windows user was invalid.
<b>447</b>	The name of the table can not be found in the kernel.	Cicode UsrKernelTableInfo() has asked for a table that does not exist.
<b>449</b>	Invalid logout	The logout was called when there is no one logged in or the logged on user is system default user.
<b>450</b>	Invalid tag data	Invalid tag data has been detected.
<b>453</b>	Subscription value is pending	Occurs when a tag has been subscribed to, and attempted to read the value before the initial value has been retrieved from the server.
<b>457</b>	Old and new SQL functions mixed	An attempt has been made to terminate a SQL session by calling SQLDisconnect when it was opened by SQLCreate and SQLOpen, or by calling SQLClose and SQLDispose when it was opened by SQLConnect.
<b>512</b>	Time out error	Did not execute requested action on time.
<b>513</b>	Access denied error	Access denied to perform requested action.
<b>514</b>	Write unsuccessful	Can not write to view-only client. User needs to login with valid user credentials to get write access and to control alarms.
<b>517</b>	Can't swap pages from foreground	Cannot swap the page on foreground Cicode.
<b>518</b>	Too Many arguments for function	Too many arguments have been

Error No.	Error title	Description
		passed to a Cicode function. Check the number of arguments being passed to the function. If arguments are input directly from the operator, you should check that the correct number of arguments are being passed to the function.
<b>519</b>	Remote Cicode call Interrupted	Cicode call that triggers an RPC remote call is interrupted before the expected result is returned.
<b>520</b>	Alarm category out of range	A category number is out of its valid range (from 0 to 16375 inclusively).
<b>521</b>	Data browse record is deleted	A record is deleted during reload of ART server.
<b>522</b>	Trend archive property mismatch	A trend records archive properties are changed during start-up or reload.
<b>523</b>	Alarm priority out of range	A priority is out of its valid range on alarm category or alarm priority configuration. Its valid range is 0 – 255 (inclusively) on category configuration and 1 – 255 (inclusively) on priority configuration.
<b>524</b>	ComBreak is obsolete parameter	[Page]ComBreak and [Page]ComBreakText are obsolete parameters.
<b>525</b>	Tag in Last Usable Value	Tag quality is uncertain and its value is the last usable value.
<b>526</b>	Tran authentication unsuccessful	Login did not authenticate correctly.
<b>527</b>	High watermark reached on Tran	High watermark is reached on transaction, network is overloaded.
<b>528</b>	Reduced size of resized page	The specified resized page has been reduced in size to fit the constraints of the maximum height and width of 4096.

Error No.	Error title	Description
<b>529</b>	Too few arguments	Too few arguments have been passed to a Cicode function. Check the number of arguments being passed to the function. If arguments are input directly from the operator, you should check that the correct number of arguments are being passed to the function.
<b>530</b>	Null pointer as an argument	A string with invalid pointer has been passed to SCADA execution system.
<b>532</b>	Tag resolve timeout	Server responses to tag exists requests have timed out.
<b>533</b>	Server reload before initialize	The attempt to initiate a server side online change did not succeed because the server has not been fully initialized.
<b>534</b>	Server reload not prepared	The attempt to reload the main alarm server when the non-main alarm server is not reloaded first.
<b>535</b>	Alarm command partial success	An alarm command has succeeded on at least one cluster but has been unsuccessful on at least one other cluster in a multi cluster environment.
<b>536</b>	No record found for the operation	Indicates that for the given scope there is no record found for the requested operation.
<b>537</b>	Operation unsuccessful	Indicates that the requested operation was unsuccessful.
<b>540</b>	Database not connected	The connection to database, such as the alarm runtime database has not been established. This could be the hosting server such as the alarm server is not available, or the logon to the database was unsuccessful.
<b>541</b>	Data browse timeout	The data browse session timed out while opening the session.

Error No.	Error title	Description
<b>542</b>	Language changed remotely	The language for alarm display has changed remotely.
<b>543</b>	XML exception	Load or parse error in the XML.
<b>544</b>	Invalid Xpath expression	Invalid Xpath expression.
<b>545</b>	Invalid XML operation	Invalid operation on an XML document.
<b>546</b>	Exceeded XML Boundary	Exceeded the boundary, unable to add more to XmlDocument or XmlNode.
<b>547</b>	Exceeded Boundary	Cannot create a new connection or transaction, boundary exceeded.
<b>548</b>	Invalid Proxy dll	Specified dll is not a valid proxy dll.
<b>549</b>	Missing Proxy Method	Specified method cannot be found in proxy dll.
<b>550</b>	Invalid Proxy Method	Specified method cannot be invoked from proxy dll.
<b>551</b>	Ambiguous Proxy Method	More than one method in proxy dll matches the binding criteria.
<b>552</b>	Invalid OP Proxy Method	The method in proxy dll has one or more unspecified parameters.
<b>553</b>	Proxy Load Error	Cannot load proxy dll. For example, proxy dll may have a dependency that cannot be found.
<b>554</b>	Transaction Process Error	Other errors occurred when processing a transaction. The transaction cannot be processed.
<b>555</b>	Invalid URL	Web service address provided is not valid/well formed URL.
<b>556</b>	Transaction not ready	Not all parameters are loaded before processing the transaction.
<b>557</b>	Tag Name Exceed Length	Tag name exceeds the 254 character length limit.
<b>559</b>	CiVBA is not supported when an	A call to CiVBA code is not

Error No.	Error title	Description
	alarm server is in Ext Mem mode	supported when an alarm server is operating in Extended Memory mode.
<b>560</b>	ActiveX is not supported for x64 when an alarm server is in Ext Mem mode	A call to an ActiveX function is not supported when an alarm server is operating in Extended Memory mode.
<b>561</b>	AlarmSum index not current	The index passed to AlarmSum* functions must be current. I.E. either: - the index returned by most recent call to AlarmSumFirst/Last/Find/Next/Prev, OR - the index returned by the most recent call to AlarmSumAppend/Split.
<b>562</b>	AlarmSum not initialized	The alarm sum session needs to be initialized, by calling AlarmSumFirst/Last/Find, before AlarmSumNext/Prev is able to be called.
<b>563</b>	AlarmSum busy	AlarmSum* functions should not be called from multiple concurrent Cicode tasks. If busy this error will occur.
<b>564</b>	AlarmSum record deleted	The alarm summary record represented by the index is deleted.
<b>565</b>	AlarmSum invalid field name	The alarm summary Cicode function does not accept the specified field name.
<b>566</b>	Attempt to call x86 dll in x64 process	A 32 bit application has been called in a 64 bit process.
<b>567</b>	Attempt to call x64 dll in x86 process	A 64 bit application has been called in a 32 bit process.
<b>568</b>	Too many alarm record filters	The number of alarm record filters applied to a query needs to be less than 1000.

Error No.	Error title	Description
<b>569</b>	Attempted to index to an alarm that is currently out of range	The value specified for an alarm list offset needs to be within the range of alarm entries currently stored in the client cache.
<b>577</b>	View only client: Read Only	A View Only client cannot update tag values. To update tags via that machine, it needs to be reconfigured as a full remote client.
<b>578</b>	Cannot read the private key of a certificate when doing TLS exchange	The encrypted connection between a server and another computer is not working. This can occur if Runtime Manager is not running as a service. Check the settings described in Enable Encryption.
<b>579</b>	The TLS exchange to secure the connection failed	The encrypted connection between a server and another computer is not working. Check your <b>DNS Name</b> settings (see Add a Computer), and review the settings described in Enable Encryption.
<b>580</b>	Cannot find a certificate for the given thumbprint	The encrypted connection between a server and another computer is not working. Check the settings described in Enable Encryption.
<b>581</b>	A server has been configured to preference an IP address type that does not exist, or is not enabled, on this computer	Check the server's IP address configuration.
<b>582</b>	No action is required as alarm is already in the intended state	This error indicates that a redundant pair of alarm servers are temporarily out of synchronization. This means the standby alarm server has not yet carried out a property update that occurred on the main alarm server. You can limit the occurrence of this error with the parameter [Alarm]StandbyCommandDelay.
<b>583</b>	URL too long	The current URL for a Web Content control is longer than 255 characters. See

Error No.	Error title	Description
		DspWebContentGetURL.
<b>584</b>	Role passed to UserCreate() function is not defined	A role specified for a user via the UserCreate function has not been defined in Plant SCADA Studio's <b>Security</b> activity.

## Windows Socket Error Codes

The following Plant SCADA error codes are mapped to standard Windows socket errors. Generally these errors are beyond Plant SCADA's control, that is, there is an external reason why Plant SCADA cannot use the TCP/IP connection (assuming there are no Plant SCADA configuration issues).

Error No.	Error title	Description
<b>1330</b>	WSAENETDOWN	(10050) Network is down.
<b>1331</b>	Socket unreachable	(10051) Network is unreachable.
<b>1332</b>	Network reset	(10052) Network dropped connection on reset.
<b>1333</b>	Connection aborted	(10053) Software caused connection abort.
<b>1334</b>	Connection reset	(10054) Connection reset by peer.
<b>1335</b>	No buffers available	(10055) No buffer space available.
<b>1340</b>	Socket timeout	(10060) Connection timed out.
<b>1341</b>	Connection refused	(10061) Connection refused.
<b>1343</b>	Name too long	(10063) Item Name too long.
<b>1344</b>	Host down	(10064) Host is down.
<b>1345</b>	Host unreachable	(10065) No route to host.

## MAPI Errors

The table below describes the MAPI errors in Cicode.

Error number	Error Title	Description
0	SUCCESS_SUCCESS	The command completed successfully.
1	MAPI_USER_ABORT	The command was aborted by the user.
2	MAPI_E_FAILURE	General MAPI error.
3	MAPI_E_LOGIN_FAILURE	The login did not succeed, either the login user is unknown, misspelt, or the password is incorrect.
4	MAPI_E_DISK_FULL	The disk is full. The mail system will copy files into the temporary directory when mail is read. This can fill up the local hard disk.
5	MAPI_E_INSUFFICIENT_MEMORY	Insufficient memory to complete the command.
6	MAPI_E_ACCESS_DENIED	More privilege is required to complete the requested command.
8	MAPI_E_TOO_MANY_SESSIONS	You have tried to open too many sessions to the mail system.
9	MAPI_E_TOO_MANY_FILES	Too many attached files in a message.
10	MAPI_E_TOO_MANY_RECIPIENTS	Too many recipients for a mail message.
11	MAPI_E_ATTACHMENT_NOT_FOUND	Cannot find the attached file.
12	MAPI_E_ATTACHMENT_OPEN_FAILURE	Cannot open the specified attached file. Often because the file does not exist.
13	MAPI_E_ATTACHMENT_WRITE_FAILURE	Cannot write the attached file.
14	MAPI_E_UNKNOWN_RECIPIENT	Recipient of the mail message is unknown. Check if the user is configured in the mail system or check the spelling of the user's name.

Error number	Error Title	Description
15	MAPI_E_BAD_RECIPTYPE	Unknown recipient type.
16	MAPI_E_NO_MESSAGES	No new messages to read.
17	MAPI_E_INVALID_MESSAGE	The mail message is invalid.
18	MAPI_E_TEXT_TOO_LARGE	Text message is too large to be sent. If you want to send large text messages, write the text to a file and attach the file to the mail message.
19	MAPI_E_INVALID_SESSION	Mail session is invalid. You should not get this error message; contact Technical Support for this product.
20	MAPI_E_TYPE_NOT_SUPPORTED	You should not get this error message; contact Technical Support for this product.
21	MAPI_E_AMBIGUOUS_RECIPIENT	The recipient of the mail message is ambiguous. Specify the exact user name to which the mail is to be sent.
22	MAPI_E_MESSAGE_IN_USE	Mail message is in use. You should not get this error message: contact Technical Support for this product.
23	MAPI_E_NETWORK_FAILURE	Mail cannot be sent or received as the network has experienced an error.
24	MAPI_E_INVALID_EDITFIELDS	You should not get this error message; contact Technical Support for this product.
25	MAPI_E_INVALID_RECIPS	You should not get this error message; contact Technical Support for this product.
26	MAPI_E_NOT_SUPPORTED	Command is not supported by this implementation of MAPI.

## Using Cicode Programming Standards

Implementing programming practices results in Cicode that is more robust, manageable, and predictable in execution, regardless of the author. Using programming standards entails:

- Adopting modular programming techniques.
- Helping to ensure that programs are adequately described by suitable module headers.
- Formatting code to improve readability.

The following topics are presented as a suggested means of achieving good programming standards:

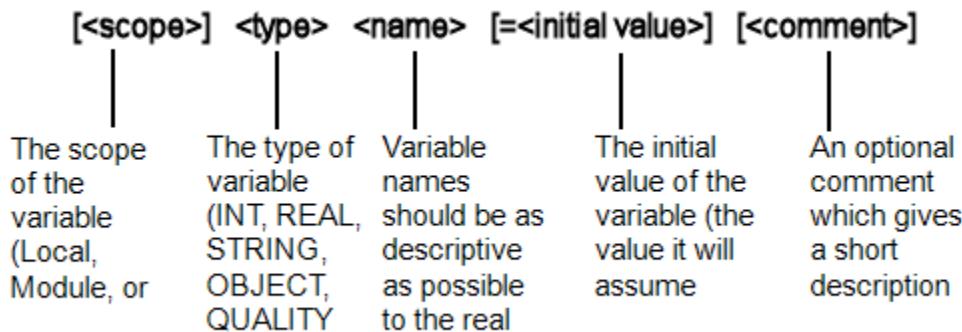
- Variable Declaration Standards
- Variable Scope Standards
- Variable Naming Standards
- Standards for Constants, Variable Tags, and Labels
- Formatting Simple Declarations
- Formatting Executable Statements
- Formatting Expressions
- Cicode Comments
- Formatting Functions
- Modular Programming
- Defensive Programming
- Function Error Handling
- Debug Error Trapping

## See Also

[Using Cicode Functions](#)

### Variable Declaration Standards

When declaring variables you should use consistent formatting. A variable declaration has up to five parts. Each part is separated by at least one tab stop:



**Note:** Parts contained within square brackets are optional. For example, you may omit the variable scope (it defaults to local). Parts contained within greater than (<) and less than (>) signs should be replaced with the relevant text/value. For example, you would replace <initial value> with an actual value. (You would not bracket your value with greater than and less than signs.)

When declaring your variables, the parts of each should align vertically (the scope part of each should be vertically aligned, the type part of each should be aligned, etc.). Each part of the declaration is allotted a set amount of space. If one part is missing, its space should be left blank. The missing part should not affect the positioning of the next part:

```
Module int miRecipeMax=100;  
int iRecipeMax;  
string sRecipeDefault ="Tasty";
```

## See Also

[Declaring the Variable Data Type](#)  
[Using Cicode Programming Standards](#)

## Variable Scope Standards

### Local variable standards

Local Variables should be initialized, for example:

```
INT iFile = 0;  
STRING sName = "";  
INT bSuccess = FALSE;
```

### Module variable standards

Module Variables should be initialized, for example:

```
MODULE INT mhForm = -1;  
MODULE STRING msPageName = "Loop";
```

### Global variable standards

Global variables should be initialized, for example:

```
GLOBAL INT ghTask = -1;  
GLOBAL STRING gsLastPage = "Menu";
```

## Variable Naming Standards

The following naming conventions should be applied to variables:

- Variable names should have a small case letter prefix as follows:

Type	Prefix	Used for
INT (32 bits)	i	index, loop counter
INT (32 bits) and OBJECT (32 bits)	h	handle
INT (32 bits)	b	boolean (TRUE/FALSE)
REAL (64 bits)	r	real type variables
STRING (255 bytes)	s	string type variables

- Variable names typically consist of up to three words. Each word in a variable name should start with a capital letter, for example:  
`iTrendType, rPeriod, sFileName`
- Module variable names should be prefixed with an "m", for example:  
`miTrendType, mrPeriod, msFileName`
- Global variable names should be prefixed with a "g", for example:  
`giTrendType, grPeriod, gsFileName`
- Local variable names should not be prefixed (when you declare a variable without specifying the scope, it is considered a Local variable by default):  
`iTrendType, rPeriod, sFileName`

## See Also

[Using Cicode Programming Standards](#)

## Standards for Constants, Variable Tags, and Labels

When coding constants, variable tags and labels in Cicode you should try to use the following standards:

- [Constants](#)
- [Variable Tags](#)
- [Labels.](#)

## See Also

[Using Cicode Programming Standards](#)

## Constants

In Cicode there is no equivalent of #defines of C language, or a type that will force variables to be constants (read-only variables). However, the variable naming convention makes constants easily identifiable so developers will treat those variables as read-only variables.

- Constants are recommended to have the prefix 'c'.
- Constants need to be declared and initialized at the beginning of the Cicode file and under no circumstances assigned a value again.

For example:

```
INT ciTrendTypePeriodic = 1;  
INT ciTrendTypeEvent = 2;  
STRING csPageName = "Mimic";
```

## See Also

[Using Cicode Programming Standards](#)

## Variable Tags

Variable tags that have been defined in the database (with the Variable Tags form) can be used in all functions in the Cicode files. Variable tags are identifiable because they will not have a prefix (also, they are generally in uppercase letters).

## See Also

[Using Cicode Programming Standards](#)

## Labels

Labels, like variable tags, can be used in all functions in the Cicode files. They can be either all upper case letters or mixed case. In order to differentiate them from the variable tags and other Cicode variables they should have an `\_` (underscore) in front of them. For example:

\_BILLING\_EVENT, \_UNIT\_OFFLINE, \_AfterHoursEvent

---

**Note:** There are a few labels without an underscore defined in the Labels form in the INCLUDE project. Although they do not follow the guidelines set in this document their wide usage makes changing those labels impractical. These labels are: TRUE, FALSE, BAD\_HANDLE, XFreak, XOutsideCL, XAboveUCL, XBelowLCL, XOutsideWL, XUpTrend, XDownTrend, XGradualUp, XGradualDown, XErratic, XStratification, XMixture, ROutsideCL, RAboveUCL, RBelowLCL

---

## See Also

[Using Cicode Programming Standards](#)

## Formatting Simple Declarations

The following conventions should be observed when formatting simple Cicode declarations:

- Only one item should be declared per declaration; there should be no comma separated list of variables.
- Tab stops should be used for declarations and indentation.

For example:

```
INT hFile,hForm; // WRONG
INT hFile; // RIGHT
INT hForm; // RIGHT
```

The reasons for this are:

- Only the first identifier in the WRONG case is obvious and the others are often missed in a quick glance;.
- It is harder to add a comment or initialization to an item in the WRONG case.
- Types, items, and initialization within a group of declarations should be vertically aligned.

For example:

```
STRING sFileName = "temp.dat"; // WRONG
INT iOffset = -1; // WRONG
```

```
INT iState = 3; // WRONG
STRING sFileName = "temp.dat"; // RIGHT
INT iOffset = -1; // RIGHT
INT iState = 3; // RIGHT
```

## See Also

[Using Cicode Programming Standards](#)

## Formatting Executable Statements

The following conventions should be observed when formatting executable statements:

- Statements are placed on new lines, indented one tab stop from the level of their surrounding block.

**Note:** Do not place more than one statement on a single line. While this practice is permitted in other programming languages, in Cicode the subsequent statements will not be interpreted and will effectively be lost, potentially affecting your program's runtime behavior.



### UNINTENDED EQUIPMENT OPERATION

Do not place more than one statement per line.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Although it may be argued that some statements are logically related, this is not sufficient justification. If they are logically related, place the statements on consecutive lines and separate the statements by a blank line before and after. For example:

```
hFile = -1; hForm = -1; // WRONG
hFile = -1; // RIGHT
hForm = -1; // RIGHT
```

**IF statements** can be used in one of the formats below. When indenting the IF statements, a tab stop should be used. For example:

- **Simple IF block**

```
IF <expression> THEN
...
END
```

- **IF-THEN-ELSE block**

```
IF <expression> THEN
...
ELSE
...
END
```

- To simulate **ELSEIF blocks**, use nested statements. For example:

```
IF <expression> THEN
...
```

```
ELSE
IF <expression> THEN
...
ELSE
IF <expression> THEN
...
ELSE
...
END
END
END
```

- For **WHILE** and **FOR** statements see [Working with Conditional Executors](#).

## See Also

[Working with Conditional Executors](#)

[Using Cicode Programming Standards](#)

## Formatting Expressions

The following conventions should be observed when formatting Cicode functions:

- When an expression forms a complete statement, it should, like any other statement, occupy one or more lines of its own and be indented to the current level. Operators should be surrounded by spaces. For example:

```
i= i*10+c-'0'; // WRONG
i = i * 10 + c - '0'; // RIGHT
```

- When a sub-expression is enclosed in brackets, the first symbol of the sub-expression should be placed hard against the opening bracket. The closing bracket should be placed immediately after the last character for the sub-expression. For example:

```
a = b * ( c - d ); // WRONG
a = b * (c - d); // RIGHT
```

- The round brackets which surround the arguments of a function attract no spaces, for example:

```
DisplayText( "hello" ); // WRONG
DisplayText("hello"); // RIGHT
```

- Commas, whether used as operators or separators, would be placed hard against the previous symbol and followed by a space. For example:

```
DevSeek(hDev ,0ffset); // WRONG
DevSeek(hDev, Offset); // RIGHT
```

## See Also

[Using Cicode Expressions](#)

[Using Cicode Commands](#)

[Using Cicode Programming Standards](#)

## Cicode Comments

Comments are designed to aid understanding and maintenance of code. You should place comments in the notes of the function header so as not to clutter up the code. Small one-line comments are acceptable to explain some small part of the code which may be hard to understand in the normal header comment.

## See Also

[Using Cicode Programming Standards](#)

## Formatting Functions

Cicode functions have up to seven parts: Scope, Type, Keyword, Name, Argument(s), Statement(s), Keyword.

- **[Scope]**

The scope of the function. If the scope is omitted, the function will be Public by default. That is, it will be available to all Cicode files, pages, and Plant SCADA databases (for example, Alarm.dbf). To make a function Private (that is only available within the file in which it is declared), you need to prefix it with the word PRIVATE.

- **[Type]**

The return type of the function. This should be on a separate line.

- **Keyword**

The keyword FUNCTION. This should be on a separate line.

- **Name**

The function name. Function names should follow the Function Naming Standards. This should be on a separate line.

- **Argument(s)**

The argument list. The arguments are separated by commas and they can have default values. The argument list is normally on the same line as the function name but multiple line argument list is also acceptable if it improves readability.

- **Statement(s)**

The statements. Each statement should be on a separate line.

- **Keyword**

The keyword END which marks the end of the function. This should be on a separate line.

---

**Note:** Parts contained within square brackets - [ ] - are optional. For example, the scope may be omitted and if so, it will default to Public. Parts contained within greater than & less than signs - < > - should be replaced with the relevant text/value. For example, you would replace <initial value> with an actual value. You would not bracket your value with greater than & less than signs.

---

## Example:

```
FUNCTION  
PlotInit()
```

```
<statements>
END
INT
FUNCTION
PlotOpen(STRING sName, INT nMode)
INT hPlot = _BAD_HANDLE;
...
hPlot = ....;
...
RETURN hPlot;
END
PRIVATE
STRING
FUNCTION
WasteInfoName(INT nType, INT nMode)
STRING sName = "Sydney";
...
sName = ....;
...
RETURN sName;
END
```

## See Also

[Writing Functions](#)

[Using Cicode Functions](#)

[Using Cicode Programming Standards](#)

## Format Templates

The format of a format template string is:

[text]{<name>[,width[,justification]]}[text]...

Rules for valid format template display

1. If the "width" value is not present then the width is set to the length of the number of characters inclusive between '{' and '}'. This means that the field value may be truncated or padded depending on the name value length.
2. If the "width" value is specified then that is the length of the field. This means that the name value length may be truncated or padded.
3. The justification is made up of a single character with the following behaviours as specified:
  - 'R' or 'r' will align the field on the right hand side. If the width is longer than the name value length then the left hand side of the name value is padded with spaces.
  - 'L' or 'l' will align the field on the left hand side. If the width is longer than the name value length then the right hand side of the name value is padded with spaces.
  - 'Z' or 'z' will align the field on the right hand side. If the width is longer than the name value length then the left hand side of the value is padded with zeros.
  - 'N' or 'n' will remove any extra padding that is used. Essentially any padding of the name value is trimmed.

4. If a justification is not specified then the name value is assumed to be left justified.
5. Any spaces appearing after the first comma onwards in the format template will be stripped out at no penalty to the user.

## Malformed format template display

There are two types of malformed templates and below are examples of each and the resulting output.

- **Internal malformation**

This is when there is a correct open and close brace '{' and '}' but inside the format template there is a malformation. For example there may be a space not a comma separating the name and the width. In this case the whole field is ignored which means nothing between and including '{' and '}' is displayed.

For example, take the following string:

```
< { LocalTimeDate , 20 , R } > TagLabel < { Tag , 20 L } > DescriptionLabel < {  
Desc , 20 , L } >
```

The output would as follows:

```
< 2009-07-17 11:13:17 > TagLabel < > DescriptionLabel < ValidAlarm1Desc >
```

---

**Note:** The "Tag" name value is not outputted as the field has no ',' between the width and justification.

- **Brace malformation**

This is when there is an open brace '{' but no closing brace '}'. In this case the malformation is printed as a string literal.

For example, take the following string:

```
< { LocalTimeDate , 20 , R } > TagLabel < { Tag , 20 , L > DescriptionLabel < {  
Desc , 20 , L } >
```

The output would be as follows:

```
< 2009-07-17 11:31:44 > TagLabel < { Tag , 20 , L > DescriptionLabel <  
ValidAlarm1Desc >
```

---

**Note:** The "Tag" name value is outputted as a literal as no closing brace '}' is detected.

## See Also

[Writing Functions](#)

[Using Cicode Functions](#)

[Using Cicode Programming Standards](#)

## Function Naming Standards

Function names should contain at least the following information:

- A three-to-five letter description of the function type (Trend, Plot, Win).
- A one or two word description of the data to be operated on (Info, ClientInfo, Mode).
- A one word action to be taken (Get, Set, Init, Read).

For example:

```
PlotInit();TrendClientOpen();TrendClientInfoRead();
```

## See Also

[Naming Functions](#)

## Source File Headers

Source files (the files that contain your Cicode) should have a header to provide a basic overview of the file. This header should be formatted as follows:

```
/** FILE: <name of file.CI>
/**
***** MODULE CONSTANTS*****
<module constants> //<comments (optional)>
***** MODULE VARIABLES *****
<module variables> //<comments (optional)>
*****
```

**Note:** Declare all module variables at the MODULE VARIABLES section at the beginning of the file and initialize the module variables.

For example:

```
/** FILE: Recipe.CI
/**
/** DESCRIPTION: Contains all functions for gathering recipe data.
/**
/** FUNCTIONS: PUBLIC
/** OpenRecipeDatabase
/** CloseRecipeDatabase
/** ReadRecipeData
/** WriteRecipeData
/** GatherRecipeData
/** RecipeForm
/** OpenRecipeDatabase
/**
/** PRIVATE
/** ButtonCallback
/**
***** MODULE CONSTANTS*****
module int cmiRecipeMax =100; //Maximum number of recipes
***** MODULE VARIABLES *****
module int miRecipeNumber =0; //Minimum number of recipes
*****
```

## Function Headers

Functions should have a descriptive header, formatted as follows:

```
/** FUNCTION : <name of function>
/**
/** DESCRIPTION : <suggests the operation, application source and
/** multi-tasking issues>
/** REVISION DATE BY COMMENTS
/** <revision number> <date> <author> <comments about the change>
```

```
/***
/** ARGUMENTS: <argument description>
/**
/** RETURNED VALUE: < description of possible return values>
/**
/** NOTES:
```

The order of functions in a file is important for efficient operation.

Initialization and shutdown functions should be placed at the top of the file. Command functions should be next. Local utility functions should be at the bottom of the file.

For example:

```
/** FUNCTION : OpenRecipeDatabase
/**
/** DESCRIPTION : Opens the specified database.
/**
/** REVISION DATE BY COMMENTS
/** 1 28/09/97 BS Original
/** 2 05/10/97 SFA AddedINI checking
/**
/** ARGUMENTS:
/**
/** STRING sName Name of the recipe database.
/**
/** INT dwMode Mode to open the recipe database.
/** 0 for read only, 1 for read/write.
/**
/** RETURNED VALUE: Handle if successful, otherwise -1.
/**
/** NOTES:
INT
FUNCTION
OpenRecipeDatabase(STRING sName, INT dwMode)
...
END
```

## Modular Programming

One of the more effective programming practices involves partitioning large, complex programming challenges into smaller and more manageable sub-tasks and reusable functions. A similar approach should be taken when using a programming language like Cicode to complete a task. Reducing the task to smaller tasks (or functions) has the following advantages:

- **Reduced Complexity** - Once the function is created and tested, the detailed operation about how it works need not be revisited. Users need only focus on the results produced by the function.
- **Avoids Duplicate Code** - Creating a generic function instead of copying similar code reduces the total amount of code in the system. It also means the function can be reused by separate code areas. This makes the code more maintainable because it is smaller in size, and only one instance needs to be modified.
- **Hides Information** - Information can be in the form of operations, data, or resources. Access to information can be controlled when functions are written that provide a limited set of actions to be performed on the information. For example, if a user wishes to log a message to a database, he or she should only send the message to a function, say `LogDBaseMessage("hello world")`, and the function should control the database resource. The function then becomes the single interface to the database resource. Resources that have

multiple interfaces to them are harder to control. This is because in a multitasking environment, the user cannot control or even know in advance the order of code execution, and hence a resource may be modified at the same time by different tasks. Information hiding can also smooth out any wrinkles in standard functions, minimizing possible misuse of resources such as semaphores, queues, devices, and files. Functions that do this are often called 'wrapper' functions as they add a protective shell to existing functions.

- **Improves Performance** - Optimizing code that resides in one place immediately increases the performance of code that calls this function. Scattered code will require multiple areas to be modified should any optimization be necessary.
- **Isolates Complex Code** - Code that requires complex operations such communications protocols, complex algorithms, boolean logic, or complex data manipulation is susceptible to errors. Placing this code in a separate function reduces the possibility of this code corrupting or halting other code.
- **Improves Readability** - A small function with meaningful parameter names assists readability as it is a step towards self-documenting code and reduces the need to scan multiple pages of code to establish what the operation is meant to achieve.

Modular programming has a few rules that define how functions should be structured - Cohesion - and how they are related to other functions - Coupling.

## See Also

[Defensive Programming](#)

[Using Cicode Programming Standards](#)

## Cohesion

A goal of modular programming is to create simple functions that perform a single task, but perform that task well. This can be described as how 'cohesive' a function is.

Two factors that affect the level of cohesion are:

- Number of tasks the function performs.
- Similarity of the tasks.

The following table illustrates the different levels of cohesion:

Number of tasks	Similarity	Cohesion level	Example
1	Not applicable	High	<code>Sin(x)</code>
More than 1	Similar	Moderate	<code>SinAndTan(x)</code>
More than 1	Dissimilar	Low	<code>SinAndLength(x)</code>
Many	Radically different	None	<code>SinAndDateAndTimeAndSQLNext(x)</code>

For example, the function `Sin(x)` will perform one task - return the trigonometric Sine value of `x`. This is an example of a highly cohesive function. The function `SinAndTan(x)` performs two tasks - calculate the trigonometric Sine and Tan of the value `X`. This function has lower cohesion than `Sin(x)` because it performs two

tasks.

Highly cohesive functions are more dependable, easier to modify, and easier to debug than functions that have lower levels of cohesion and are hence acceptable and encouraged.

Low cohesion functions are typically complex, prone to errors, and are more costly to fix. Low cohesion functions are regarded as poor programming practice and discouraged.

## Coupling

Another rule of modular programming is to reduce the number of relationships between functions. This is referred to as function coupling. Functions that have few, or no, relationships between them are loosely coupled. Loosely coupled functions provide simple, visible interfaces to the function. This makes the functions easier to use and modify. For example, the Cicode function **TimeCurrent()** is a loosely coupled function. To use this function, a user need only call its name, and the function will return with the desired result. The user does not need to be aware of any relationships because there are no parameters passed to the function, and it does not read from, or write to, any global data. There is very little likelihood of error with this function; it only returns a time/date variable and does not support error codes. In the unlikely event that the function did not return the time/date variable, it would be through no error of the calling function because it has no relationship to it.

Functions that have many relationships between them are tightly coupled. For example, a user written function like **AddCustomerRecord(hDatabase, sFirstName, sSurname, sAddress, sAge, sPhone)** has a higher level of coupling than the function **TimeCurrent()**. Tightly coupled functions are inflexible in their use, less robust, and harder to maintain. The **AddCustomerRecord()** function is less robust because it could experience an error of its own accord, or if the function calling it passes bad data to it. Tightly coupled functions are harder to maintain because modifying a function with many relationships in it may result in modifications to other functions to accept the data.

The different types of function relationships are listed below:

- **Passed parameters.** A simple, visible form of loose coupling that is encouraged. Once the number of parameters passed to a function exceeds seven, you should consider partitioning the function into two smaller functions. These types of relationships are acceptable.
- **Control information.** Control information causes the called function to behave in a different way. For example, the function **ChangeData(iMode)**, behaves differently depending on the value of the variable **iMode** that is passed into it. It may be responsible for deleting, inserting, or updating data. In addition to being a tightly coupled function, it has low cohesion because it performs multiple tasks. This function could be divided into three separate functions to perform the separate tasks. These types of relationships are moderately acceptable.
- **Shared common data.** This is often referred to as global variable data. This is an invisible form of tight coupling that, particularly in pre-emptive, multi-tasking environments, can result in an unpredictable, hard-to-maintain program. When functions write to the global variable data there is no monitoring of or restriction on who is writing to the variable. Hence the value can be indeterminate. Global variables are acceptable when they are used for read-only purposes, otherwise their use is discouraged. Similarly, module variable data in Plant SCADA should be treated the same way. The use of local function variables is encouraged to decrease function coupling.

## Defensive Programming

Defensive programming is an approach to improve software and source code. It aims to improve general quality by reducing the number of software bugs. It promotes making the source code readable and understandable. It

aims to make your code behave in a predictable manner despite unexpected input or user actions.

You should try to:

- Verify that your code does not produce unexplained hardware alarms.
- Check that denominators in division are not zero.
- Check that array indexes cannot go out of range.
- Check that arguments from external sources are valid.
- Check that loop terminations are obvious and achievable.
- Only write code once. If you find that two sections of code look identical or almost identical it is worth spending the time to re-write or re-design it. This will generally reduce the size of the code in question by a third or more, which reduces complexity and therefore maintenance and debugging time. An effective method to achieve this is to convert the identical code to a new function.
- Do not access the module data in any function other than the member functions.
- Write the member functions whenever an array is defined. Do not try to pass arrays between functions, make the arrays the centre piece of the object.
- Cicode is a multitasking language. Several tasks (commands, expressions and tasks created by TaskNew function) can be executed concurrently. This powerful feature of Cicode should be used with care as some of the functions may be modifying module data. It is essential that the number of tasks running at any point in time be minimized. This may require the use of semaphores to help protect the module data from interference and corruption. (For the use of semaphores, refer to SemOpen, SemClose, SemSignal and SemWait functions in on-line help or the Cicode Reference manual).

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

- Write your Cicode programs with the minimum number of concurrent instructions suitable to your application.
- Use semaphores or some related means to coordinate program flow if your program will execute a high number of concurrent instructions.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## **See Also**

[Using Cicode Programming Standards](#)

[Modular Programming](#)

[Function Error Handling](#)

## **Function Error Handling**

Errors are handled by examining the return values of the functions. The Cicode functions can be classified as regards their return value as follows:

Functions returning	Calling functions should check for
Error code only	0 no error (success) > 0 error code
Handles	-1 bad handle ≥ 0 valid handle
Random values	The return of IsError()

The following Cicode functions can halt the current task:

DevOpen, DevHistory, DevNext, DevPrev, DevSeek, DevFind, DevFlush, DevRecNo, DevRead, DevReadLn, DevAppend, DevDelete, DevZap, DevControl, DevPrint, DevModify, ErrTrap; FileOpen, FileClose, FileReadBlock, FileWriteBlock, FileSeek, FileDelete, FileRename, FileSize, FileReadLn, FileCopy; FormNew; SQLConnect, SQLTraceOn, SQLTraceOff, SQLErrMsg.

If an error is detected in one of these functions, your Cicode task will generate a hardware error and be halted. You may stop your Cicode task from being halted by using the [ErrSet](#) function and checking for errors using [IsError](#).

The parameter [\[Code\]HaltOnError](#) allows you to stop any errors detected in these functions from halting your Cicode. If you set the following:

```
[code]
HaltOnError=0
```

then your Cicode will continue to run after a hardware error is detected in these functions.

**For example:**

- **Example of error code only**

```
INT
FUNCTION
ExampleInit()
INT nError = 0;
nError = ExampleOpen("MyForm");
IF nError = 0 THEN
...
END
END
INT
FUNCTION
ExampleOpen(STRING sName)
INT nError = 0;
...
IF <an error has been detected> THEN
nError = 299;
END
RETURN nError;
END
```

- **Example of handles**

```
INT
FUNCTION
ExampleInit()
INT hFile = BAD_HANDLE;
```

```
...
hFile = ExampleFileOpen("MyFile.txt");
IF hFile <> BAD_HANDLE THEN
...
END
END
FUNCTION
ExampleFileOpen(STRING sName)
INT hFile = BAD_HANDLE;
hFile = FileOpen(sName, "r+");
IF hFile = BAD_HANDLE THEN
hFile = FileOpen(sName, "r");
END
RETURN hFile;
END
```

- **Example of random values**

```
INT
FUNCTION
ExampleInit()
INT nSamples = 0;
...
ErrSet(1);
nSamples = ExampleSamples();
IF IsError() = 0 THEN
...
END
ErrSet(0);
END
INT
FUNCTION
ExampleSamples()
INT nSamples = 0;
INT nError = 0;
...
ErrTrap(nError);
RETURN nSamples;
END
```

## See Also

[Debugging Cicode](#)

## Debug Error Trapping

The functions listed below can also be used during normal project execution to trap run-time problems:

- [DebugMsg Function](#)
- [Assert Function](#)

## See Also

[Debugging Cicode](#)

## DebugMsg Function

DebugMsg internally calls the TraceMsg function if the debug switch is on. The implementation of this function can be found in DEBUG.CI in the INCLUDE project. You can turn the debug switch on or off by doing any of the following:

- Calling DebugMsgSet(INT bDebugMsg) on the Kernel Cicode window. (Or, this function can be called from a keyboard command or something similar.)
- Changing the [\[Code\]DebugMessage](#) parameter in the INI file.

For example:

```
INT
FUNCTION
FilePrint(STRING sDeviceName, STRING sFileName)

INT hFile;
INT hDev;
STRING Str1;

hDev = DevOpen(sDeviceName, 0);
IF (hDev = -1) THEN
DebugMsg("Invalid arg to FilePrint - 'DeviceName'");
RETURN 261; /* File does not exist */
END
hFile = FileOpen(sFileName, "r");
IF (hFile = -1) THEN
DebugMsg("Invalid arg to FilePrint - 'FileName'");
DevClose(hDev);
RETURN 261; /* File does not exist */
END
WHILE NOT FileEof(hFile) DO
Str1 = FileReadLn(hFile);
DevWriteLn(hDev, Str1);
END
FileClose(hFile);
DevClose(hDev);
RETURN 0;
END
```

## Assert Function

Assert reports an error if the test passed by the argument does not return the expected value. The implementation of this function can be found in DEBUG.CI in the INCLUDE project.

For example:

```
INT
FUNCTION
FileDisplayEx(STRING sFileName)

INT hFile;

hFile = FileOpen(sFileName, "r");
ASSERT(hFile <> -1);
```

```
...
FileClose(hFile);
RETURN 0;
END
```

## See Also

[Debugging Cicode](#)

## Cicode Function Categories

Cicode includes the following function categories:

<a href="#">Accumulator Functions</a>	<a href="#">Menu Functions</a>
<a href="#">ActiveX Functions</a>	<a href="#">Miscellaneous Functions</a>
<a href="#">Alarm Functions</a>	<a href="#">.Net Functions</a>
<a href="#">Alarm Filter Functions</a>	<a href="#">Page Functions</a>
<a href="#">Array Functions</a>	<a href="#">Plot Functions</a>
<a href="#">Clipboard Functions</a>	<a href="#">Process Analyst Functions</a>
<a href="#">Cluster Functions</a>	<a href="#">Quality Functions</a>
<a href="#">Color Functions</a>	<a href="#">Report Functions</a>
<a href="#">Communication Functions</a>	<a href="#">Scheduler Functions</a>
<a href="#">Dynamic Data Exchange Functions</a>	<a href="#">Screen Profile Functions</a>
<a href="#">Device Functions</a>	<a href="#">Security Functions</a>
<a href="#">Display Functions</a>	<a href="#">Sequence of Events (SOE) Functions</a>
<a href="#">DLL Functions</a>	<a href="#">Server Functions</a>
<a href="#">Equipment Database Functions</a>	<a href="#">SPC Functions</a>
<a href="#">Error Functions</a>	<a href="#">SQL Functions</a>
<a href="#">Event Functions</a>	<a href="#">String Functions</a>
<a href="#">File Functions</a>	<a href="#">Super Genie Functions</a>
<a href="#">Form Functions</a>	<a href="#">Table (Array) Functions</a>
<a href="#">Format Functions</a>	<a href="#">Tag Functions</a>
<a href="#">Fuzzy Logic Functions</a>	<a href="#">Task Functions</a>
<a href="#">Group Functions</a>	<a href="#">Time and Date Functions</a>
<a href="#">I/O Device Functions</a>	<a href="#">Timestamp Functions</a>
<a href="#">Keyboard Functions</a>	<a href="#">Trend Functions</a>
<a href="#">Mail Functions</a>	<a href="#">Window Functions</a>
<a href="#">Map Functions</a>	<a href="#">XML Functions</a>
<a href="#">Math and Trigonometry Functions</a>	

## See Also

[Using Cicode Functions](#)

## Accumulator Functions

Following are functions relating to accumulators.

<a href="#">AccControl</a>	Controls accumulators for example, motor run hours.
<a href="#">AccumBrowseClose</a>	Closes an accumulator browse session.
<a href="#">AccumBrowseFirst</a>	Gets the oldest accumulator entry.
<a href="#">AccumBrowseGetField</a>	Gets the field indicated by the cursor position in the browse session.
<a href="#">AccumBrowseNext</a>	Gets the next accumulator entry in the browse session.
<a href="#">AccumBrowseNumRecords</a>	Returns the number of records in the current browse session.
<a href="#">AccumBrowseOpen</a>	Opens an accumulator browse session.
<a href="#">AccumBrowsePrev</a>	Gets the previous accumulator entry in the browse session.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

### AccControl

Controls accumulators, for example, motor run hours. You can reset the values of Run Time, Totalizer Inc, and No. of Starts (defined in the accumulator database), re-read these values from the I/O device, or flush pending writes of these values to the I/O device.

### Syntax

LONG **AccControl**(*sName*, *nMode* [, *sClusterName*] )

*sName*:

The name of the accumulator or a mask for the names of accumulators. You can use the following wildcards:

- \* matches all following characters, for example, "Motor\*" matches all accumulators starting with the word "Motor"

- ? matches any single character, for example, "Motor?10" matches "MotorA10" and "MotorB10"

This argument can be prefixed by the name of the cluster for example ClusterName.AccumulatorName.

*nMode:*

The mode of the control:

- 1 - Reset Run Time and Totalizer value
- 2 - Reset No. of Starts
- 3 - Reset Run Time, Totalizer value, and No. of Starts
- 4 - Flush pending writes to the I/O device
- 5 - Re-read Run Time, Totalizer value, and No. of Starts from the I/O device

*sClusterName:*

Name of the cluster in which the accumulator resides. This is optional if you have one cluster or are resolving the reports server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Example

```
! Reset all accumulator variables for accumulator "MCC123".  
AccControl("MCC123", 3, "ClusterXYZ");
```

## See Also

[Accumulator Functions](#)

## AccumBrowseClose

The AccumBrowseClose function terminates an active data browse session and cleans up all resources associated with the session.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

INT **AccumBrowseClose**(*iSession*)

*iSession*

The handle to a browse session previously returned by an [AccumBrowseOpen](#) call.

## Return Value

0 (zero) if the accumulator browse session exists, otherwise an error code is returned.

## Related Functions

[AccumBrowseFirst](#), [AccumBrowseGetField](#), [AccumBrowseNext](#), [AccumBrowseNumRecords](#), [AccumBrowseOpen](#), [AccumBrowsePrev](#)

## See Also

[Accumulator Functions](#)

### AccumBrowseFirst

The AccumBrowseFirst function places the data browse cursor at the first record.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **AccumBrowseFirst**(*iSession*)

*iSession*

The handle to a browse session previously returned by a [AccumBrowseOpen](#) call.

## Return Value

0 (zero) if the accumulator browse session exists, otherwise an error code is returned.

## Related Functions

[AccumBrowseClose](#), [AccumBrowseGetField](#), [AccumBrowseNext](#), [AccumBrowseNumRecords](#), [AccumBrowseOpen](#), [AccumBrowsePrev](#)

## See Also

[Accumulator Functions](#)

### AccumBrowseGetField

The AccumBrowseGetField function retrieves the value of the specified field from the record the data browse cursor is currently referencing.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

STRING **AccumBrowseGetField**(LONG *Session*, STRING *FieldName*)

*Session*:

The handle to a browse session previously returned by a [AccumBrowseOpen](#) call.

**FieldName:**

The name of the field that references the value to be returned. Supported fields are:

AREA, CLUSTER, EQUIPMENT, NAME, PRIV, RUNNING, STARTS, TOTALISER, TRIGGER, VALUE

See [Browse Function Field Reference](#) for information about fields.

## Return Value

The value of the specified field as a string. An empty string may or may not be an indication that an error has been detected. The last error should be checked in this instance to determine if an error has actually occurred.

## Related Functions

[AccumBrowseClose](#), [AccumBrowseFirst](#), [AccumBrowseNext](#), [AccumBrowseNumRecords](#), [AccumBrowseOpen](#), [AccumBrowsePrev](#)

## Example

```
STRING fieldValue = "";
STRING fieldName = "TYPE";
INT errorCode = 0;
...
fieldValue = AccumBrowseGetField(iSession, sFieldName);
IF fieldValue <> "" THEN
    // Successful case
ELSE
    // Function returned an error
END
...
```

## See Also

[Accumulator Functions](#)

## AccumBrowseNext

The AccumBrowseNext function moves the data browse cursor forward one record. If you call this function after you have reached the end of the records, error 412 is returned (Databrowse session EOF).

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT AccumBrowseNext(*iSession*)**

*iSession*

The handle to a browse session previously returned by a [AccumBrowseOpen](#) call.

## Return Value

0 (zero) if the accumulator browse session exists, otherwise an error code is returned.

## Related Functions

[AccumBrowseClose](#), [AccumBrowseFirst](#), [AccumBrowseGetField](#), [AccumBrowseNumRecords](#), [AccumBrowseOpen](#), [AccumBrowsePrev](#)

## See Also

[Accumulator Functions](#)

## AccumBrowseNumRecords

The AccumBrowseNumRecords function returns the number of records that match the filter criteria.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

LONG **AccumBrowseNumRecords**(*iSession*)

*iSession*

The handle to a browse session previously returned by a [AccumBrowseOpen](#) call.

## Return Value

The number of records that have matched the filter criteria. A value of 0 denotes that no records have matched. A value of -1 denotes that the browse session is unable to provide a fixed number. This may be the case if the data being browsed changed during the browse session.

## Related Functions

[AccumBrowseClose](#), [AccumBrowseFirst](#), [AccumBrowseGetField](#), [AccumBrowseNext](#), [AccumBrowseOpen](#), [AccumBrowsePrev](#)

## Example

```
INT numRecords = 0;  
...  
numRecords = AccumBrowseNumRecords(iSession);  
IF numRecords <> 0 THEN  
    // Have records  
ELSE  
    // No records  
END  
...
```

## See Also

[Accumulator Functions](#)

### AccumBrowseOpen

The AccumBrowseOpen function initiates a new browse session and returns a handle to the new session that can be used in subsequent data browse function calls.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT AccumBrowseOpen( STRING *Filter*, STRING *Fields* [, STRING *Clusters*] )**

### *Filter:*

A filter expression specifying the records to return during the browse. An empty string indicates that all records will be returned. Where a fieldname is not specified in the filter, it is assumed to be tagname. For example, the filter "AAA" is equivalent to "name=AAA".

The following regular expressions are supported: \*expr, expr\*, and \*expr\*. To specify an exclusion filtering condition, use the NOT keyword after the = operator.

---

**Note:** Use the following fields with care in filters since they return the actual value of the variable tag which they refer to.

RUNNING, STARTS, TOTALISER, TRIGGER, VALUE.

---

### *Fields:*

Specifies via a comma delimited string the columns to be returned during the browse. An empty string indicates that the server will return all available columns. Supported fields are:

AREA, CLUSTER, EQUIPMENT, ITEM, NAME, PRIV, RUNNING, STARTS, TOTALISER, TRIGGER, VALUE.

See [Browse Function Field Reference](#) for information about fields.

### *Clusters:*

An optional parameter that specifies via a comma delimited string the subset of the clusters to browse. An empty string indicates that the connected clusters will be browsed.

## Return Value

Returns an integer handle to the browse session. Returns -1 when an error is detected.

The returned entries will be ordered alphabetically by name. After a reload of the Report Server, any new records may be added at the end.

## Related Functions

[AccumBrowseClose](#), [AccumBrowseFirst](#), [AccumBrowseGetField](#), [AccumBrowseNext](#), [AccumBrowseNumRecords](#),  
[AccumBrowsePrev](#)

## Example

```
INT iSession;
...
iSession = AccumBrowseOpen("NAME=ABC*", "NAME,AREA",
"ClusterA,ClusterB");
IF iSession <> -1 THEN
    // Successful case
ELSE
    // Function returned an error
END
...
```

## See Also

[Accumulator Functions](#)

## AccumBrowsePrev

The AccumBrowsePrev function moves the data browse cursor back one record. If you call this function after you have reached the beginning of the records, error 412 is returned (Databrowse session EOF).

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **AccumBrowsePrev**(*iSession*)

*iSession*

The handle to a browse session previously returned by a [AccumBrowseOpen](#) call.

## Return Value

0 (zero) if the accumulator browse session exists, otherwise an error code is returned.

## Related Functions

[AccumBrowseClose](#), [AccumBrowseFirst](#), [AccumBrowseGetField](#), [AccumBrowseNext](#), [AccumBrowseNumRecords](#), [AccumBrowseOpen](#)

## See Also

[Accumulator Functions](#)

## ActiveX Functions

Following are functions relating to ActiveX objects:

---

**Note:** ActiveX objects are not supported on a 64-bit process, such as an alarm server operating in Extended Memory mode. If a call to an ActiveX function occurs from a 64-bit process, an error code will be returned, a hardware alarm will be raised and the Cicode thread will stop.

---

For information regarding methods you can use to extend Plant SCADA that do not require ActiveX, see the topic [Extensibility](#) in the Plant SCADA documentation.

<a href="#">_ObjectCallMethod</a>	Calls a specific method for an ActiveX object.
<a href="#">_ObjectGetProperty</a>	Retrieves a specific property of an ActiveX object.
<a href="#">_ObjectSetProperty</a>	Sets a specific property of an ActiveX object.
<a href="#">AnByName</a>	Retrieves the animation point number of an ActiveX object.
<a href="#">CreateControlObject</a>	Creates a new instance of an ActiveX object.
<a href="#">CreateObject</a>	Creates the automation component of an ActiveX object.
<a href="#">ObjectAssociateEvents</a>	Allows you to change the ActiveX object's event class.
<a href="#">ObjectAssociatePropertyWithTag</a>	Establishes an association between a variable tag and an ActiveX object property.
<a href="#">ObjectByName</a>	Retrieves an ActiveX object.
<a href="#">ObjectHasInterface</a>	Queries the ActiveX component to determine if its specific interface is supported.
<a href="#">ObjectIsValid</a>	Determines if the given handle for an object is valid.
<a href="#">ObjectToStr</a>	Converts an object handle to a string.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

### [\\_ObjectCallMethod](#)

Calls a specific method for an ActiveX object. (See the documentation for your ActiveX object for details on methods and properties.)

**Note:** The parameter list passed to the control can only have Cicode variables or variable tags; it cannot use values returned directly from a function because an ActiveX control may modify parameters passed to it.

For example:

```
//Calculate a value and pass to ActiveX control
_ObjectCallMethod(hControl, "DoSomething", CalcValue());
```

This is not allowed because the return value of a function cannot be modified. The following should be used instead:

```
INT nMyValue;
//Calculate Value
```

```
nMyValue = CalcValue();
//Pass Value to ActiveX control
_ObjectCallMethod(hControl, "DoSomething", nMyValue);
```

## Syntax

VARIANT **\_ObjectCallMethod**(*hObject*, *sMethod*, *vParameters*)

*hObject*:

The handle for the object (as returned by the [ObjectByName](#) function).

*sMethod*:

The name of the method.

*vParameters*:

A variable length parameter list of method arguments. The variables will be passed however you enter them, and will then be coerced into appropriate automation types. Likewise, any values modified by the automation call will be written back - with appropriate coercion - into the passed Cicode variable.

## Return Value

The return value from the method - if successful, otherwise an error code is returned.

## Related Functions

[ObjectByName](#), [DspAnCreateControlObject](#), [CreateObject](#), [CreateControlObject](#)

## Example

See [CreateControlObject](#).

## See Also

[ActiveX Functions](#)

## **\_ObjectGetProperty**

Gets a specific property of an ActiveX object.

## Syntax

VARIANT **\_ObjectGetProperty**(*hObject*, *sProperty*)

*hObject*:

The handle for the object (as returned by the [ObjectByName](#) function).

*sProperty*:

The name of the property you want to get.

## Return Value

The value of the property - if successful, otherwise an error code is returned.

## Related Functions

[ObjectName](#), [DspAnCreateControlObject](#), [CreateObject](#), [CreateControlObject](#)

## Example

See [CreateControlObject](#)

## See Also

[ActiveX Functions](#)

## \_ObjectSetProperty

Sets a specific property of an ActiveX object.

## Syntax

**INT \_ObjectSetProperty(*hObject*, *sProperty*, *vValue*)**

*hObject*:

The handle for the object (as returned by the [ObjectName](#) function).

*sProperty*:

The name of the property you want to set.

*vValue*:

The value to which the property will be set. This value can be of any data type. Appropriate coercion will take place when creating the equivalent automation parameter.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[ObjectName](#), [DspAnCreateControlObject](#), [CreateObject](#), [CreateControlObject](#)

## Example

See [CreateControlObject](#)

## See Also

[ActiveX Functions](#)

### AnByName

Retrieves the animation point number of an ActiveX object.

## Syntax

LONG **AnByName**(*sName*)

*sName*:

The name given to the ActiveX object. This name is visible in the "Identification" tab of the ActiveX control in the Graphics Builder and is used to access the object.

If the animation number for an object is 35 and you renamed the object to "Fred", use AnByName("Fred"); which will return 35.

If you left the name of the object as the default use AnByName("AN35"); which will return 35.

## Return Value

The animation point number of the object - if successful, otherwise an error code is returned.

## Related Functions

[CreateControlObject](#)

## See Also

[ActiveX Functions](#)

### CreateControlObject

Creates a new instance of an ActiveX object.

An object created using this function remains in existence until the page is closed or the associated Cicode Object is deleted. This function does not require an existing animation point. When the object is created, an animation point is created internally. This animation point is freed when the object is destroyed.

**Note:** ActiveX objects are not supported on a 64-bit process, such as an alarm server operating in Extended Memory mode. If a call to this function occurs from a 64-bit process, an error code will be returned, a hardware alarm will be raised and the Cicode thread will stop.

For information regarding methods you can use to extend Plant SCADA that do not require ActiveX, see the topic [Extensibility](#) in the Plant SCADA documentation

## Syntax

OBJECT **CreateControlObject**(*sClass*, *sName*, *x1*, *x2*, *y1*, *y2*, *sEventClass*)

*sClass*:

The class of the object. You can use the object's human readable name, its program ID, or its GUID. If the class does not exist, the function will return an error message.

For example:

- "Calendar Control 8.0" - human readable name
- "MSCAL.Calendar.7" - Program ID
- "{8E27C92B-1264-101C-8A2F-040224009C02}" - GUID

*sName*:

The name for the object in the form of "AN" followed by its AN number, for example, "AN35". This name is used to access the object.

*x1*:

The x coordinate of the object's top left hand corner as it will appear in your Plant SCADA window.

*y1*:

The y coordinate of the object's top left hand corner as it will appear in your Plant SCADA window.

*x2*:

The x coordinate of the object's bottom right hand corner as it will appear in your Plant SCADA window.

*y2*:

The y coordinate of the object's bottom right hand corner as it will appear in your Plant SCADA window.

*sEventClass*:

The string you would like to use as the event class for the object.

## Return Value

The newly created object, if successful, otherwise an error is generated.

## Related Functions

[DspAnCreateControlObject](#), [CreateObject](#), [AnByName](#)

## Example

```
// This function creates a single instance of the calendar control
at the designated location with an object name of "CalendarEvent"
and an event class of "CalendarEvent"//
FUNCTION
CreateCalendar()
    OBJECT Calendar;
    STRING sCalendarClass;
    STRING sEventClass;
    STRING sObjectName;
    sCalendarClass = "MSCal.Calendar.7";
```

```
sEventClass = "CalendarEvent";
sObjectName = "MyCalendar";
Calendar = CreateControlObject(sCalendarClass, sObjectName, 16,
100, 300, 340, sEventClass);
END
// This function shows how to change the title font of the
calendar//
FUNCTION
CalendarSetFont(STRING sFont)
    OBJECT Font;
    OBJECT Calendar;
    Calendar = ObjectByName("MyCalendar");
    Font = _ObjectGetProperty(Calendar, "TitleFont");
    _ObjectSetProperty(Font, "Name", sFont);
END
// This function shows how to change the background color of the
calendar//
FUNCTION
CalendarSetColor(INT nRed, INT nGreen, INT nBlue)
    OBJECT Calendar;
    Calendar = ObjectByName("MyCalendar");
    _ObjectSetProperty(Calendar, "BackColor",
PackedRGB(nRed,nGreen,nBlue));
END
// This function shows how to call the NextDay method of the
calendar//
FUNCTION
CalendarNextDay()
    OBJECT Calendar;
    Calendar = ObjectByName("MyCalendar");
    _ObjectCallMethod(Calendar, "NextDay");
END
// This function shows you how to write a mouse click event
handler for the calendar//
FUNCTION
CalendarEvent_Click(OBJECT This)
    INT nDay;
    INT nMonth;
    INT nYear;
    nDay = _ObjectGetProperty(This, "Day");
    nMonth = _ObjectGetProperty(This, "Month");
    nYear = _ObjectGetProperty(This, "Year");
    ...
    Your code goes here...
    ...
END
```

## See Also

[ActiveX Functions](#)

### CreateObject

Creates a new instance of an ActiveX object. If you use this function to create an ActiveX object, it will have no

visual component (only the automation component will be created).

If you assign an object created with the CreateObject function to a local variable, that object will remain in existence until the variable it is assigned to goes out of scope. This means that such an object will only be released when the Cicode function that created it ends.

If you assign an object created with the CreateObject function to a module or global scope variable, then that object will remain in existence until the variable either has another object assigned or is set to NullObject, *provided the CreateObject call is not made within a loop*.

Objects created by calls to CreateObject within WHILE or FOR loops are only released on termination of the Cicode function in which they are created, regardless of the scope of the variable to which the object is assigned. The use of CreateObject within a loop may therefore result in the exhaustion of system resources, and is not generally recommended unless performed as shown in the examples below.

**Note:** ActiveX objects are not supported on a 64-bit process, such as an alarm server operating in Extended Memory mode. If a call to this function occurs from a 64-bit process, an error code will be returned, a hardware alarm will be raised and the Cicode thread will stop.

For information regarding methods you can use to extend Plant SCADA that do not require ActiveX, see the topic [Extensibility](#) in the Plant SCADA documentation.

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

Do not use the CreateObject() function within a loop except in strict accordance with the following instructions.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## **Syntax**

OBJECT **CreateObject(sClass)**

*sClass:*

The class of the object. You can use the object's human readable name, its program ID, or its GUID. If the class does not exist, the function will return an error.

For example:

- "Calendar Control 8.0" - human readable name
- "MSCAL.Calendar.7" - Program ID
- "{8E27C92B-1264-101C-8A2F-040224009C02}" - GUID

## **Return Value**

The newly created object, if successful, otherwise an error is generated.

## **Related Functions**

[DspAnCreateControlObject](#), [CreateControlObject](#)

## Example

The following examples show correct techniques for calling CreateObject() within a loop.

```
/* In the example below, the variable objTest is local. Resources
associated with calls to ProcessObject() will be released each
time that function ends. */
FUNCTION Forever()
    WHILE 1 DO
        ProcessObject();
        Sleep(1);
    END
END
FUNCTION ProcessObject()
    .OBJECT objTest;
    objTest=CreateObject("MyObject");
    - do something
END
/* In the example below, the variable objTest is global. Resources
associated with calls to ProcessObject() will be released when
objTest is set to NullObject. */
FUNCTION Forever()
    WHILE 1 DO
        ProcessObject();
        Sleep(1);
    END
END
FUNCTION ProcessObject()
    objTest=CreateObject("MyObject");
    - do something
    objTest=NullObject;
END
```

## See Also

[ActiveX Functions](#)

### ObjectAssociateEvents

Allows you to change the ActiveX object's event class. If you have inserted an object on a graphics page using Graphics Builder, it allows you to change the event class to something other than the default of "PageName\_ObjectName".

## Syntax

**INT ObjectAssociateEvents(*sEventClass*, *hSource*)**

*sClass*:

The class of the object. You can use the object's human readable name, its program ID, or its GUID. If the class does not exist, the function will report an error.

*hSource*:

The source object firing the events which are to be handled by the event handler.

For example:

- "Calendar Control 8.0" - human readable name
- "MSCAL.Calendar.7" - Program ID
- "{8E27C92B-1264-101C-8A2F-040224009C02}" - GUID

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspAnCreateControlObject](#), [CreateControlObject](#)

## Example

### Inserting ActiveX objects using Cicode

If you have created an ActiveX object using Cicode (for example, by calling the function [CreateControlObject](#)), the parameter 'sEventClass' would have required you to define an event class for the object to enable event handling. If you want to change the class you used, you can call **ObjectAssociateEvents**.

### Inserting ActiveX objects via Graphics Builder

If you have inserted an ActiveX object in Graphics Builder, runtime will automatically create an event class for the object in the form **PageName\_ObjectName**. If this is the case, you may want to change the object's event class.

Using the example of an ActiveX sludge tank controller, the default event class for the object could be "PageName\_AN35". This means any events handlers for the object would take the form "PageName\_AN35\_Click" (presuming this example relates to a click event). You may want to define this more clearly, in which case you can call the following:

```
// This function redefines the event class for the ActiveX sludge
tank controller at AN35 to "SludgeTank". //
ObjectAssociateEvents ("SludgeTank", ObjectByName("AN35"));
..
```

With the event class for the object now defined as "SludgeTank", the event handlers can take the form "SludgeTank\_Click".

This would be useful if you define event handlers in relation to an object that will eventually be copied to other graphics pages, as it will reduce the need to redefine the event handlers to identify the default event class associated with each new placement of the object.

## See Also

[ActiveX Functions](#)

### ObjectAssociatePropertyWithTag

Establishes an association between an ActiveX property and a variable tag. This means that any changes made to an ActiveX object property will be mirrored in the variable tag.

Generally, ActiveX objects issue "property change notifications" to Plant SCADA whenever a change occurs to a specific property value. This notification tells Plant SCADA when to read the property value.

**Note:** An association will not succeed if property change notifications are not supported and the *OnChangeEvent* argument is left blank. Verify that the scaling and units of the associated tag are compatible with the ActiveX property values. However, some property changes do not trigger property change notifications. If this is the case, you need to choose an appropriate "on change" event instead - an event fired by the ActiveX object in response to a change. An "appropriate" event is one that happens to be fired whenever the property value changes. For example, the MS Calendar Control fires an AfterUpdate event whenever a day button is pressed.

## Syntax

**INT ObjectAssociatePropertyWithTag(*sObject*, *sPropertyName*, *sTagName* [, *sOnChangeEvent*] )**

*sObject*:

The object instance that associates a property with a tag.

*sPropertyName*:

The name of the ActiveX property to associate with the tag.

*sTagName*:

The name of the Plant SCADA variable tag to associate with the property.

*sOnChangeEvent*:

The name of the "on change" event that informs Plant SCADA of a change to the ActiveX object. This is required where the ActiveX object does not automatically generate a property change notification. Choose an event that happens to be fired whenever the ActiveX object property changes, for example, the MS Calendar Control fires an AfterUpdate event whenever a day button is pressed.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspAnCreateControlObject](#), [CreateObject](#), [CreateControlObject](#)

## See Also

[ActiveX Functions](#)

## ObjectName

Retrieves an ActiveX object. This is useful when you know the object by name only (this will often be the case for objects created during configuration, rather than those created at runtime using a Cicode function).

## Syntax

**OBJECT ObjectByName(STRING *Name*)**

**Name:**

The name used to access the object, as specified when creating it in Cicode. For objects created in the Graphics Builder, the object name is set in the Access (Identification) tab, and defaults to "AN" followed by its AN number, for example, "AN35". The *Name* argument should be enclosed in quotes:

"<Name>".

**Return Value**

The requested object, if successful, otherwise an error is generated.

**Related Functions**

[DspAnCreateControlObject](#), [CreateObject](#), [CreateControlObject](#)

**Example**

See [CreateControlObject](#).

**See Also**

[ActiveX Functions](#)

**ObjectHasInterface**

Queries the ActiveX component to determine if its specific interface is supported. (Refer to the ActiveX object's documentation for details of its interfaces.)

**Syntax**

OBJECT **ObjectHasInterface**(*hObject*, *sInterface*)

***hObject*:**

The handle for the object (as returned by the [ObjectByName](#) function).

***sInterface*:**

The name of the interface (case sensitive).

**Return Value**

0 if the interface is not supported, otherwise 1.

**Related Functions**

[ObjectByName](#), [CreateObject](#), [CreateControlObject](#)

**Example**

```
hPen = _ObjectGetProperty(hControl, "Pen");
IF ObjectHasInterface(hPen, "IDigitalPen") THEN
    //Fill is only supported on digital pen
    _ObjectSetProperty(hPen, "Fill", 0)
END
```

## See Also

[ActiveX Functions](#)

### ObjectIsValid

Determines if the given handle for an object is a valid handle. This function is useful for programmatically checking that an object was returned for a call.

## Syntax

```
INT ObjectIsValid(hObject)
```

*hObject*:

The handle for the object (as returned by the [ObjectByName](#) function).

## Return Value

0 if the handle is not valid, otherwise 1.

## Related Functions

[ObjectByName](#), [CreateObject](#), [CreateControlObject](#)

## Example

```
hFont = _ObjectGetProperty(hControl, "Font");
IF ObjectIsValid(hFont) THEN
    _ObjectSetProperty(hFont, "Size", 22)
END
```

## See Also

[ActiveX Functions](#)

### ObjectToStr

Converts an object handle to a string. This means you can print the object handle in a trace message, which allows you confirm if the handle is valid.

## Syntax

STRING **ObjectToStr(hObject)**

*hObject*:

The handle for the object (as returned by the [ObjectName](#) function).

## Return Value

A string containing the converted object handle

## Related Functions

[ObjectName](#), [CreateObject](#), [CreateControlObject](#)

## See Also

[ActiveX Functions](#)

## Alarm Functions

Alarm functions display alarms and their related alarm help pages, and acknowledge, disable, and enable alarms. They provide information about alarms and allow your operators to add comments to alarm records. You can also access alarms at the record level (on the alarms server) for more complex operations.

See also [Alarm Filter Functions](#).

<a href="#">AlarmAck</a>	Acknowledges alarms.
<a href="#">AlarmAckRec</a>	Acknowledges alarms by record number.
<a href="#">AlarmAckTag</a>	Acknowledges a specified alarm
<a href="#">AlarmActive</a>	Determines if any alarms are active in the user's area.
<a href="#">AlarmCatGetFormat</a>	Returns the display format string of the specified alarm category.
<a href="#">AlarmClear</a>	Clears acknowledged, inactive alarms from the active alarm list.
<a href="#">AlarmClearRec</a>	Clear an alarm by its record number.
<a href="#">AlarmClearTag</a>	Clear an alarm by its tag name.
<a href="#">AlarmComment</a>	Allows users to add comments to alarm summary entries during runtime.
<a href="#">AlarmCommentRecID</a>	Allows an operator to add a comment to a selected

	alarm summary or SOE entry during runtime.
AlarmCount	Counts the available alarms for the selected filter criteria.
AlarmCountEquipment	Counts the available alarms for specified equipment in conjunction with the selected filter criteria.
AlarmCountList	Counts the available alarms for the selected alarm list.
AlarmDelete	Deletes alarm summary entries that are currently displayed.
AlarmDisable	Disables alarms.
AlarmDisableRec	Disables alarms by record number.
AlarmDisableTag	Disables alarms by tag name.
AlarmDsp	Displays alarms.
AlarmDspClusterAdd	Adds a cluster to a client's alarm list.
AlarmDspClusterInUse	Determines if a cluster is included in a client's alarm list.
AlarmDspClusterRemove	Removes a cluster from a client's alarm list.
AlarmDspLast	Displays the latest, unacknowledged alarms.
AlarmDspNext	Displays the next page of alarms.
AlarmDspPrev	Displays the previous page of alarms.
AlarmEnable	Enables alarms.
AlarmEnableRec	Enables alarms by record number.
AlarmEnableTag	Enables alarms by tag name.
AlarmEventQue	Opens the alarm event queue.
AlarmFirstCatRec	Searches for the first occurrence of an alarm category and type.
AlarmFirstPriRec	Searches for the first occurrence of an alarm priority and type.
AlarmFirstTagRec	Searches for the first occurrence of an alarm tag, name, and description.
AlarmGetDelay	Gets the delay setting for an alarm.

<a href="#">AlarmGetDelayRec</a>	Gets the delay setting for an alarm via the alarm record number.
<a href="#">AlarmGetDsp</a>	Gets field data from the alarm record that is displayed at the specified AN.
<a href="#">AlarmGetFieldRec</a>	Gets alarm field data from the alarm record number.
<a href="#">AlarmGetInfo</a>	Gets information about an alarm list displayed at an AN.
<a href="#">AlarmGetOrderbyKey</a>	Retrieves the list of key(s) used to determine the order of the alarm list.
<a href="#">AlarmGetThreshold</a>	Gets the thresholds of analog alarms.
<a href="#">AlarmGetThresholdRec</a>	Gets the thresholds of analog alarms by the alarm record number.
<a href="#">AlarmHelp</a>	Displays the help page for the alarm where the cursor is positioned.
<a href="#">AlarmHighestPriority</a>	Returns the priority and alarm state of the current highest priority alarm for the given equipment.
<a href="#">AlarmListCreate</a>	Creates an alarms list at a specified AN.
<a href="#">AlarmListDestroy</a>	Destroys an alarms list at a specified AN.
<a href="#">AlarmListDisplay</a>	Displays an alarms list at a specified AN.
<a href="#">AlarmListFill</a>	Fills an alarm list at a specified AN to be used by another routine.
<a href="#">AlarmNextCatRec</a>	Searches for the next occurrence of an alarm category and type.
<a href="#">AlarmNextPriRec</a>	Searches for the next occurrence of an alarm priority and type.
<a href="#">AlarmNextTagRec</a>	Searches for the next occurrence of an alarm tag, name, and description.
<a href="#">AlarmNotifyVarChange</a>	Activates a time-stamped digital or time-stamped analog alarm
<a href="#">AlarmQueryFirstRec</a>	Searches for the first occurrence of an alarm category (or priority) and type.
<a href="#">AlarmQueryNextRec</a>	Searches for the next occurrence of an alarm category (or priority) and type.

<a href="#">AlarmSetDelay</a>	Changes the delay setting for an alarm.
<a href="#">AlarmSetDelayRec</a>	Changes the delay set for an alarm via the alarm record number.
<a href="#">AlarmSetInfo</a>	Changes the display parameters for the alarm list displayed at an AN.
<a href="#">AlarmSetQuery</a>	This function is now obsolete. Use the Alarm Filter Functions.
<a href="#">AlarmSetThreshold</a>	Changes the thresholds of analog alarms.
<a href="#">AlarmSetThresholdRec</a>	Changes the thresholds of analog alarms by the alarm record number.
<a href="#">AlarmSplit</a>	Splits an alarm summary entry which has no Off time.
<a href="#">AlarmSumAppend</a>	Appends a new blank record to the alarm summary.
<a href="#">AlarmSumCommit</a>	Commits the alarm summary record to the alarm summary device.
<a href="#">AlarmSumDelete</a>	Deletes alarm summary entries.
<a href="#">AlarmSumFind</a>	Finds an alarm summary index for an alarm record and alarm on time.
<a href="#">AlarmSumFirst</a>	Gets the oldest alarm summary entry.
<a href="#">AlarmSumGet</a>	Gets field information from an alarm summary entry.
<a href="#">AlarmSumLast</a>	Gets the latest alarm summary entry.
<a href="#">AlarmSumNext</a>	Gets the next alarm summary entry.
<a href="#">AlarmSumPrev</a>	Gets the previous alarm summary entry.
<a href="#">AlarmSumSet</a>	Sets field information in an alarm summary entry.
<a href="#">AlarmSumSplit</a>	Duplicates an alarm summary entry.
<a href="#">AlarmSumType</a>	Retrieves a value that indicates a specified alarm's type.
<a href="#">AlarmTagFromEquipment</a>	Returns the first tag associated with a piece of equipment.
<a href="#">AlmBrowseAck</a>	Acknowledges the alarm tag at the current cursor position in an active data browse session.
<a href="#">AlmBrowseClose</a>	This function is now obsolete.

AlmBrowseClose	Closes an alarm tag's browse session.
AlmBrowseDisable	Disables the alarm tag at the current cursor position in an active data browse session.
AlmBrowseEnable	Enables the alarm tag at the current cursor position in an active data browse session.
AlmBrowseFirst	Gets the oldest alarm tags entry.
AlmBrowseGetField	Gets the field indicated by the cursor position in the browse session.
AlmBrowseNext	Gets the next alarm tags entry in the browse session.
AlmBrowseNumRecords	Returns the number of records in the current browse session.
AlmBrowseOpen	Opens an alarm tags browse session.
AlmBrowsePrev	Gets the previous alarm tags entry in the browse session.
AlmSummaryAck	Acknowledges the alarm at the current cursor position in an active data browse session.
AlmSummaryClear	Clears the alarm at the current cursor position in an active data browse session.
AlmSummaryClose	Closes an alarm summary browse session.
AlmSummaryCommit	Commits the value changes made by AlmSummarySetFieldValue().
AlmSummaryDelete	Deletes alarm summary entries from the browse session.
AlmSummaryDeleteAll	Deletes all alarm summary entries from the browse session.
AlmSummaryDisable	Disables the alarm at the current cursor position in an active data browse session.
AlmSummaryEnable	Enables the alarm at the current cursor position in an active data browse session.
AlmSummaryFirst	Gets the oldest alarm summary entry.
AlmSummaryGetField	Gets the field indicated by the cursor position in the browse session.
AlmSummaryLast	Places the data browse cursor at the latest summary

	record from the last cluster of the available browsing cluster list.
AlmSummaryNext	Gets the next alarm summary entry in the browse session.
AlmSummaryNumRecords	Retrieves the number of records in an alarm summary browse session.
AlmSummaryOpen	Opens an alarm summary browse session.
AlmSummaryPrev	Gets the previous alarm summary entry in the browse session.
AlmSummarySetFieldValue	Sets the value of the field indicated by the cursor position in the browse session. AlmSummaryCommit() is required to commit the value changes. The value changes are discarded if the cursor moves away.
AlmTagsAck	This command is now deprecated. Use the AlmBrowseAck function instead.
AlmTagsClear	This command is now deprecated.
AlmTagsClose	This command is now deprecated. Use the AlmBrowseClose function instead.
AlmTagsDisable	This command is now deprecated. Use the AlmBrowseDisable function instead.
AlmTagsEnable	This command is now deprecated. Use the AlmBrowseEnable function instead.
AlmTagsFirst	This command is now deprecated. Use the AlmBrowseFirst function instead.
AlmTagsGetField	This command is now deprecated. Use the AlmBrowseGetField function instead.
AlmTagsNext	This command is now deprecated. Use the AlmBrowseNext function instead.
AlmTagsNumRecords	This command is now deprecated. Use the AlmBrowseNumRecords function instead.
AlmTagsOpen	This command is now deprecated. Use the AlmBrowseOpen function instead.
AlmTagsPrev	This command is now deprecated. Use the AlmBrowsePrev function instead.

HwAlarmQue	Returns the handle of the hardware alarm queue.
------------	---

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## AlarmAck

Acknowledges alarms. You can acknowledge the alarm where the cursor is positioned, one or more alarm lists on the active page, a whole category of alarms, or alarms of a particular priority.

---

**Note:** This function cannot be used with StruxureWare templates.

This command takes the currently logged in user into account. In other words, only the alarms that the user can see are acknowledged.

---

**Note:** This function will only work while the record is writable which is controlled by the ArchiveAfter parameter. Refer to the topic [Configure the Archiving Parameters](#) for more information.

You would normally call this function from a keyboard command. No action is taken if the specified alarms have already been acknowledged.

---

**Note:** Alarm commands on single clusters will return either 0 if successful or an error code if unsuccessful. Alarm commands across multiple clusters may also return a partial success result, whereby a command has been successful on 'at least' one cluster but unsuccessful on another cluster.

If this function is not called from a foreground task, it becomes a blocking function.

## Syntax

**INT AlarmAck(INT Mode, INT Value [, STRING ClusterName])**

*Mode:*

The type of acknowledgment:

0 - Acknowledge a single alarm.

- Set *Value* to the AN where the alarm is displayed.
- If *Value* is set to 0, the current cursor position will be used.

1 - Acknowledge a page of alarms. An alarm page can contain more than one alarm list:

- Set *Value* to the AN where the alarm list is displayed.
- Set *Value* to 0 to acknowledge the (displayed) alarm list (on the active page) where the cursor is positioned.
- Set *Value* to -1 to acknowledge all (displayed) alarm lists on the active page. This only applies to alarm lists created using AlarmDsp (and not those created using AlarmDspLast).

2 - Acknowledge a category of alarms:

- Set *Value* to the alarm category (0 to 16375) of the alarms to be acknowledged. Please be aware that Alarm category 0 indicates all categories; alarm category 255 indicates hardware alarms.
- Set *Value* to the group number to acknowledge a group of categories.

3 - Acknowledge alarms of a specific priority.

Set *Value* to the alarm priority (0-255) of the alarms to be acknowledged. Alarm priority 0 indicates all priorities. Hardware alarms are not affected by priority.

Set *Value* to the group handle to acknowledge a group of alarms of different priorities.

---

**Note:** Alarm Summary page no longer supports Modes 1, 2, and 3.

*Value*:

Used with Mode 1 and 2 to specify which alarms to acknowledge.

*sClusterName*:

Used with Mode 2 or 3 to specify the name of the cluster in which the alarms being acknowledged reside. This argument is optional if the client is connected to only one cluster containing an alarm server or are resolving the alarm server via the current cluster context.

This argument is not required where the *mode* is 2 and the value is 255 (hardware alarm category).

This argument is enclosed in quotation marks "".

## Return Value

0 (zero) if successful, otherwise an error code will return

---

**Note:** In some cases this function is non-blocking. Under these circumstances, any error that is detected when the alarm server processes the command will not be returned.

## Related Functions

[GrpOpen](#)

## Example

System keyboard	
Key Sequence	LeftButton
Command	AlarmAck(0, 0)
Comment	Acknowledge the alarm where the cursor is positioned
System keyboard	
Key Sequence	ShiftLeftButton
Command	AlarmAck(1, -1)
Comment	Acknowledge a page of alarms
Comment	Acknowledge the alarm where the cursor is positioned

System keyboard	
Key Sequence	AlarmAck ### Enter
Command	AlarmAck(2, Arg1, "clusterXYZ")
Comment	Acknowledge alarms of a specified category in cluster XYZ

System keyboard	
Key Sequence	AckPri ##### Enter
Command	AlarmAck(3,Arg1, "clusterXYZ")
Comment	Acknowledge alarms of a specific priority in cluster XYZ

! Acknowledge alarms of the specified group of categories.

FUNCTION

```
AckGrp(STRING CategoryGroup)
    INT hGrp;
    hGrp=GrpOpen("CatGroup",1);
    StrToGrp(hGrp,CategoryGroup);
    AlarmAck(2,hGrp, "clusterXYZ");
    GrpClose(hGrp);
END
```

## See Also

[Alarm Functions](#)

### AlarmAckRec

Acknowledges alarms by record number on both the primary and standby alarm servers. This function can be called from alarm server or client and should not be used with a MsgRPC() call to the alarm server.

This is a blocking function. If the function is called from a foreground task that is unable to block, an error will be returned.

## Syntax

INT **AlarmAckRec**(LONG *Record* [, STRING *ClusterName*] )

*Record*:

The alarm record number, returned from any of the following alarm functions:

- AlarmFirstCatRec or AlarmNextCatRec: used to search for a record by alarm category, area, and type (acknowledged, disabled, etc.).
- AlarmFirstPriRec or AlarmNextPriRec: used to search for a record by alarm priority, area, and type (acknowledged, disabled, etc.).

- **AlarmFirstTagRec** or **AlarmNextTagRec**: used to search for a record by alarm tag, name, and description.
- **AlarmGetDsp**: used to find the record that is displayed at a specified AN, for either an alarm list or alarm summary entry. Set the sField argument in **AlarmGetDsp** to "RecNo".

To store this value, use data type Int in Cicode or Long for variable tags (Long needs 4 bytes).

#### *ClusterName:*

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[AlarmFirstCatRec](#), [AlarmFirstTagRec](#), [AlarmNextTagRec](#), [AlarmGetDelayRec](#)

## Example

```
/* Acknowledge all unacknowledged (Type 1) alarms of the specified
alarm category. */
FUNCTION
AutoAccept(INT Category)
    INT Current;
    INT Next;
    Current=AlarmFirstCatRec(Category,1);
    WHILE Current<>-1 DO
        Next=AlarmNextCatRec(Current,Category,1);
        AlarmAckRec(Current);
        Current=Next;
    END
END
```

## See Also

[Alarm Functions](#)

## AlarmAckTag

Acknowledge a specified alarm.

If this function is not called from a foreground task, it becomes a blocking function.

## Syntax

INT **AlarmAckTag**(STRING *Tag*, [, STRING *ClusterName*])

*Tag*:

A string that identifies the alarm to acknowledge. It can be one of the following:

- An alarm tag — for example, "Fire1"
- An alarm equipment item — for example, "Motor1.AlarmFire"

If you enter an empty string (" "), the function will return an error code at runtime.

*ClusterName*:

The cluster where the tag resides

## Return Value

0 (zero) if successful, otherwise an error code will return.

## See Also

[Alarm Functions](#)

## AlarmActive

Determines if any alarms are active in the user's area. Call this function from the Page Strings database, to display an alarm message at a specified AN on a graphics page. You can specify the type of alarms, for example, active hardware alarms or disabled non-hardware alarms.

## Syntax

INT **AlarmActive**(*nType* [, *sClusterName*])

*nType*:

The type of alarms to check:

Non-hardware alarms

0 - Active alarms

1 - Unacknowledged alarms, ON and OFF

2 - Highest priority unacknowledged alarm

3 - Disabled alarms

Hardware alarms

5 - Active alarms

6 - Unacknowledged alarms, ON and OFF

*sClusterName*:

The name of the cluster to check for active alarms. If this argument is blank or empty, the function will check the connected clusters.

## Return Value

- 1 or 0 for Non-hardware alarms (modes 0, 1, and 3).
- The priority of the highest priority unacknowledged alarm (mode 2).
- The number of active alarms for Hardware alarms (modes 5 and 6).

## Example

Strings	
AN	9
Expression	AlarmActive(5)
True Text	"Hardware Alarms Active"
False Text	"No Active Hardware Alarms"
Comment	Display the alarm status at AN 9

## See Also

[Alarm Functions](#)

### AlarmCatGetFormat

Returns the display format string of the specified alarm category.

## Syntax

STRING **AlarmCatGetFormat(INT Category [, INT Type] )**

*Category:*

The alarm category.

*Type:*

The type of display format string:

0 - Alarm format. Default value.

1 - Summary format.

2 - SOE format

## Return Value

The display format string of the specified category. If the alarm category is not specifically defined or it has no format string specified in your project, the format string of category 0 will be returned.

## Example

```
sFormat = AlarmCatGetFormat(0, 0);
! sFormat is assigned to the format string as defined in the Alarm Format field of the
Alarm Categories form for category 0 in your project.
```

## See Also

[Alarm Functions](#)

### AlarmClear

Clears an acknowledged (and off) alarm from the active alarm list. You can clear the alarm where the cursor is positioned, one or more alarm lists on the active page, a whole category of alarms, or alarms of a particular priority.

If you set the [\[Alarm\]AckHold](#) parameter to 1, alarms that go off and have been acknowledged are not removed from the active list until this function is called.

## Syntax

`INT AlarmClear(Mode, Value [, ClusterName] )`

*Mode*:

The type of clear:

0 - Clear a single alarm where the cursor is positioned:

- Set *Value* to 0 (zero) - it is not used.

1 - Clear a page of alarms. An alarm page can contain more than one alarm list:

- Set *Value* to the AN where the alarm list is displayed.
- Set *Value* to 0 to clear the (displayed) alarm list (on the active page) where the cursor is positioned.
- Set *Value* to -1 to clear every (displayed) alarm list on the active page.

2 - Clear a category of alarms:

- Set *Value* to the alarm category (0 to 16375) of the alarms to clear. Please be aware that alarm category 0 indicates all categories; alarm category 255 indicates hardware alarms.
- Set *Value* to the group number to clear a group of categories.

3 - Clear alarms of a specific priority.

- Set *Value* to the alarm priority (0-255) of the alarms to be cleared.

Alarm priority 0 indicates all priorities. Hardware alarms are not affected by priority. Set *Value* to the group handle to clear a group of alarms of different priorities.

*Value*:

Used with Mode 1 or 2 to specify which alarms to clear.

*ClusterName*:

Used with Mode 2 or 3 to specify the name of the cluster in which the alarms being cleared reside. This argument is optional if the client is connected to only one cluster containing an Alarm Server or you are resolving the alarm server via the current cluster context.

This argument is not required where the *mode* is 2 and the value is 255 (hardware alarm category).

This argument is enclosed in quotation marks "".

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[AlarmAck](#)

## Example

System keyboard	
Key Sequence	Clear
Command	AlarmClear(0, 0)
Comment	Clear the alarm where the cursor is positioned
System keyboard	
Key Sequence	ClearAll
Command	AlarmClear(1, -1)
Comment	Clear a page of alarms
System keyboard	
Key Sequence	AlarmClear ### Enter
Command	AlarmClear(2, Arg1, "clusterXYZ")
Comment	Clear alarms of a specified category in cluster XYZ
System keyboard	
Key Sequence	ClearPri ##### Enter
Command	AlarmClear(3,Arg1, "clusterXYZ")
Comment	Clear alarms of a specific priority in cluster XYZ

## See Also

[Alarm Functions](#)

## AlarmClearRec

Clears an alarm by its record number on both the Primary and Standby Alarms Servers. This function can be called from Alarm Server or Client.

This function should not be used with a MsgRPC() call to the Alarm Server.

## Syntax

INT **AlarmClearRec**(*Record* [, *ClusterName*] )

*Record*:

The alarm record number, returned from any of the following alarm functions:

- **AlarmFirstCatRec** or **AlarmNextCatRec** - used to search for a record by alarm category, area, and type (acknowledged, disabled, etc.).
- **AlarmFirstPriRec** or **AlarmNextPriRec** - used to search for a record by alarm priority, area, and type (acknowledged, disabled, etc.).
- **AlarmFirstTagRec** or **AlarmNextTagRec** - used to search for a record by alarm tag, name, and description.
- **AlarmGetDsp** - used to find the record that is displayed at a specified AN, for either an alarm list or alarm summary entry. Set the *sField* argument in **AlarmGetDsp** to "RecNo".

To store this value, use data type Int in Cicode or Long for variable tags (Long needs 4 bytes).

*ClusterName*:

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[AlarmFirstCatRec](#), [AlarmAck](#), [AlarmFirstTagRec](#), [AlarmNextTagRec](#), [AlarmGetDelayRec](#), [AlarmGetDsp](#)

## See Also

[Alarm Functions](#)

## AlarmClearTag

Clears alarms by Tag on both the Primary and Standby Alarms Servers.

If this function is not called from a foreground task, it becomes a blocking function.

## Syntax

INT **AlarmClearTag**(*STRING Tag*, [, *STRING ClusterName*])

**Tag:**

A string that identifies the alarm to acknowledge. It can be one of the following:

- An alarm tag — for example, "Fire1"
- An alarm equipment item — for example, "Motor1.AlarmFire"

Specify an empty string (" ") to match all alarm tags.

**ClusterName:**

The cluster where the tag resides.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[AlarmAckTag](#), [AlarmEnableTag](#), [AlarmDisableTag](#)

## See Also

[Alarm Functions](#)

## AlarmComment

Allows an operator to add a comment to a selected alarm summary or SOE entry during runtime. You would normally call this function from a keyboard command.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT AlarmComment(STRING Comment[,INT AN])**

**Comment:**

The comment to add to the alarm summary entry or SOE entry. Currently for the Alarm summary page the maximum length of a comment is 128 characters. The maximum length for a comment on the SOE page is 244 characters. If you exceed the maximum length it will be truncated and an ellipsis appended.

**AN:**

An animation identifier. Enter the value of the AN where the alarm is displayed. If AN is not specified, the default value will be 0, which means the current cursor position will be used.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[AlarmDsp](#)

## Example

System keyboard	
Key Sequence	Com ##### Enter
Command	AlarmComment(Arg1)
Comment	Add an alarm comment to the alarm where the cursor is positioned

**Note:** If you have installed Plant SCADA on an English operating system and want to add comments at runtime using a Unicode language (such as Korean, Russian or Chinese), you will need to change the Windows™ **Region** setting for the runtime computer. To do this, go to **Control Panel** and open the **Region** dialog box. On the **Administrative** tab, use the **Change system locale** button to select the required system locale. Be aware that you will have to restart your computer. When you launch runtime, select the matching **Language** on the login form. Your runtime comments will be recorded using the specified language characters.

## See Also

[Alarm Functions](#)

### AlarmCommentRecID

Allows an operator to add a comment to a selected alarm summary or SOE entry during runtime. You would normally call this function from a keyboard command.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT AlarmCommentRecID(*sComment*, *sRecID*, *nListType*, *sCluster*)**

*Comment:*

The comment to add to the alarm summary entry or SOE entry. Currently for the Alarm summary page the maximum length of a comment is 128 characters. The maximum length for a comment on the SOE page is 244 characters. If you exceed the maximum length it will be truncated and an ellipsis appended.

*RecID:*

Can be the value:

- from the SOE record's "RECORDID" field. OR
- from the Summary record's "SUMMARYID" field.

**Note:** The Summary Alarm List is not provided in the Situational Awareness Starter Project.

*nListType:*

Following are the nListType values for SOE or Alarm Summary lists.

10 - All summary alarms

15 - Sequence of events with configuration events filtered out

16 - Sequence of events with configuration events

*sCluster:*

Specifies the cluster context for the alarms displayed in the list.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[AlarmDsp](#)

## Example

System keyboard	
Key Sequence	Com ##### Enter
Command	AlarmCommentReID(Arg1)
Comment	Add an alarm comment to the alarm where the cursor is positioned

## See Also

[Alarm Functions](#)

## AlarmCount

Counts the available alarms for the selected filter criteria.

Alarm counts will display and update at different rates depending on how extensively they are used in your system. Consider the number of counts being used on pages, combined with the number of clients using those pages at any one time. The number of configured alarms will also contribute to the alarm count update rate. For example on a system with 50,000 configured alarms and 10 clients displaying a page with 100 alarm counts, the alarm count update rate will be degraded.

This function is a blocking function if *CachedMode* is set to zero (0).

## Syntax

LONG **AlarmCount**(INT *Type* [, STRING *FilterCriteria* [, LONG *KeepAliveSeconds* [, INT *CachedMode*]])

*nType:*

The type of alarms to display:

#### **Non-hardware alarms**

- 0 - All active alarms, that is Types 1 and 2
- 1 - All unacknowledged alarms, ON and OFF
- 2 - All acknowledged ON alarms
- 3 - All disabled alarms
- 4 - All configured (non-hardware) alarms, that is Types 0 to 3, plus acknowledged OFF alarms.

#### **Hardware alarms**

- 5 - All active alarms, that is Types 6 and 7
- 6 - All unacknowledged alarms, ON and OFF
- 7 - All acknowledged ON alarms
- 8 - All disabled alarms
- 9 - All configured alarms, that is Types 5 to 8

#### **Alarm General**

- 11 - All ON alarms
- 12 - All OFF alarms
- 13 - All ON hardware alarms
- 14 - All OFF hardware alarms
- 17 - All unacknowledged ON alarms
- 18 - All unacknowledged OFF alarms

If you omit the *Type*, the default is 1.

#### *FilterCriteria:*

A filter name OR filter text.

See the topic [Implementing Alarm Filters Using Cicode](#) for more information about filter syntax.

#### *KeepAliveSeconds:*

Optional length of time (in seconds) that the count will remain in memory. Default is 30 seconds.

#### *CachedMode:*

Optional flag that causes the current cached value to be supplied even when the value is being refreshed. This makes the function non-blocking. If the property has not yet been cached, an error is set.

0 - Do not force cached read. Cicode is blocking

1 - Force cached read. Cicode is non-blocking

Default value is 1 (true).

A count in memory will be accessed when its filter criteria matches a subsequent filter criteria and the count's *KeepAliveSeconds* period will be extended.

A count will stay in the cache for 'at least' the duration specified by *KeepAliveSeconds*, and may stay in the cache for an unspecified period of time before being discarded.

The period set for an existing count will be overridden in the event a longer duration is set using *KeepAliveSeconds*.

A count that is added to the cache is not immediately available for reading, so a foreground call to `AlarmCount` that causes a new count to be added will return a value of -1 and an error of 345 (Data not ready).

## Return Value

Returns counted alarms for the selected filter criteria. Returns -1 when an error is detected.

## Example

```
INT iRet, iErr; // return values
INT iCountWhile=0; // loop counter

// counts all unacknowledged alarms, ON and OFF, default non-blocking cicode
iRet = AlarmCount(1);
iErr = IsError(); //check error code

// repeat the above non-blocking function
INT iCountWhile=0; // loop counter
WHILE (iRet = 0) AND (iErr = 0) AND (iCountWhile < 10) DO
    SleepMS(100);
    iRet = AlarmCount(1);
    iErr = IsError();
    iCountWhile = iCountWhile + 1;
END

// counts all disabled alarms, default non-blocking cicode
iRet = AlarmCount(3);
iErr = IsError();

// repeat this non-blocking function as shown above (see while loop)
// counts all unacknowledged alarms with category 10 - non-blocking cicode
iRet = AlarmCount(1,"Category=10",1,1);
iErr = IsError();

// repeat this non-blocking function as shown above (see while loop)
// counts all unacknowledged alarms with category 10 - blocking cicode
iRet = AlarmCount(1,"Category=10",1,0);
IF iRet = -1 THEN
    iErr = IsError(); // get error code)
END

// counts all unacknowledged tag alarms of equipA without it's children equipment
// and keeps the count in memory for 3 seconds - blocking cicode
// (where is equipment structure is equipA.equipB.equipC.equipD)
iRet = AlarmCount(1,"Equipment=equipA",3,0);
IF iRet = -1 THEN
    iErr = IsError(); // get error code)
END

// counts all acknowledged ClusterA tag ON alarms of equipA and it's children equipment
// and keeps the count in memory for 3 seconds - blocking cicode
// (where is equipment structure is equipA.equipB.equipC.equipD)
iRet = AlarmCount(2,"Equipment=equipA*;cluster=ClusterA",3,0);
IF iRet = -1 THEN
    iErr = IsError(); // get error code)
END
```

```
// counts all active tag alarms of equipB and it's children equipment
// and keeps the count in memory for 3 seconds - non-blocking cicode
// (where is equipment structure is equipA.equipB.equipC.equipD)
iRet = AlarmCount(0,"Equipment=equipA.equipB*",3,1);
iErr = IsError()

// counts all ON alarms of equipB and it's children equipment
// and keeps the count in memory for 3 seconds - non-blocking cicode
// (where is equipment structure is equipA.equipB.equipC.equipD)
iRet = AlarmCount(11,"Equipment=equipA.equipB*",3,1);
iErr = IsError()

// This example shows how to count the alarms using a named filter
// This example requires that the named filter "Myfilter" exists.
INT nActiveAlarmType;
INT nCount;
INT nError;
nCount = AlarmCount(nActiveAlarmType, "MyFilter");
IF nCount greater 0 THEN
    nError = IsError();
ELSE
    nError = 0;
END
```

## Related Functions

[AlarmCountList](#)

## See Also

[Alarm Functions](#)

## AlarmCountEquipment

Counts the available alarms for specified equipment and referenced equipment in conjunction with the selected filter criteria.

## Syntax

**LONG AlarmCountEquipment(INT Type [,STRING EquipmentFilter, [STRING FilterCriteria [, LONG KeepAliveSeconds [, INT CachedMode, [INT IncludingReference]]]]])**

*nType:*

The type of alarms to display:

**Non-hardware alarms**

0 - All active alarms, that is Types 1 and 2

1 - All unacknowledged alarms, ON and OFF

2 - All acknowledged ON alarms

3 - All disabled alarms

4 - All configured (non-hardware) alarms, that is Types 0 to 3, plus acknowledged OFF alarms.

#### Hardware alarms

5 - All active alarms, that is Types 6 and 7

6 - All unacknowledged alarms, ON and OFF

7 - All acknowledged ON alarms

8 - All disabled alarms

9 - All configured alarms, that is Types 5 to 8

#### Alarm General

11 - All ON alarms

12 - All OFF alarms

13 - All ON hardware alarms

14 - All OFF hardware alarms

17 - All unacknowledged ON alarms

18 - All unacknowledged OFF alarms

If you omit the Type, the default is 1.

#### *EquipmentFilter:*

A comma-separated list of equipment names or categories (RefCat) to filter alarms prior to applying the other filter specified on the FilterCriteria argument. This field has been extended to support wildcards, and hence accept partial filter strings.

Filtering using an equipment name or category (RefCat):

Expression has been extended to allow:

- "EQUIPMENT=XYZ" (this may have an asterisk as the last character, which is interpreted as a wildcard). This restricts the result to only equipment that match this filter. Multiple "EQUIPMENT=" elements are treated as having an "OR" condition. This filter is evaluated on the client.
- "REFCAT=XYZ" (this may have an asterisk as the first and/or last character, these are interpreted as wildcards). This restricts the result to only equipment that match this filter, however, this filter is applied after any equipment filters and reference expansions have taken place. Multiple "REFCAT=" elements are treated as having an "OR" condition, and these in turn are treated as having an "AND" condition with the "EQUIPMENT=" elements. This filter is evaluated on the client.
- "PAGE=XYZ" (this is an exact literal match only). This restricts the result to only those equipment that match this filter. This filter cannot be combined with other equipment filter elements. This filter is evaluated on the server.

What this means is:

- "XYZ" is equivalent to "EQUIPMENT=XYZ\*"
- "PAGE=PageXYZ" includes all the equipment that have "PageXYZ" in the PAGE field in the Equipment table.

---

**Note:** Filters that take the form "XYZ" should no longer be used. It is recommended you use the "EQUIPMENT=XYZ" form, further "XYZ" can now be expressed as "EQUIPMENT=XYZ,EQUIPMENT=XYZ.\*" as this will include all items on the "XYZ" equipment and all items on sub-equipment of the "XYZ" equipment.

---

#### *FilterCriteria:*

A filter name OR filter text.

See the topic [Implementing Alarm Filters Using Cicode](#) for more information about filter syntax.

*KeepAliveSeconds:*

Optional length of time (in seconds) that the count will remain in memory. Default is 30 seconds.

*CachedMode:*

Optional flag that causes the current cached value to be supplied even when the value is being refreshed. This makes the function non-blocking. If the property has not yet been cached, an error is set.

0 - Do not force cached read. Cicode is blocking

1 - Force cached read. Cicode is non-blocking

Default value is 1 (true).

A count in memory will be accessed when its filter criteria matches a subsequent filter criteria and the count's *KeepAliveSeconds* period will be extended.

A count will stay in the cache for 'at least' the duration specified by *KeepAliveSeconds* and may stay in the cache for an unspecified period of time before being discarded.

The period set for an existing count will be overridden in the event a longer duration is set using *KeepAliveSeconds*.

A count that is added to the cache is not immediately available for reading, so a foreground call to *AlarmCount* that causes a new count to be added will return a value of -1 and an error of 345 (Data not ready).

*IncludingReference:*

0 - Do not include alarms belonging to referenced equipment in Alarm Count for current equipment

1 - Include alarms belonging to referenced equipment in Alarm Count for current equipment. This is set by default.

## Return Value

Returns counted alarms for the selected filter criteria. Returns -1 when an error is detected.

## Example

```
// counts all unacknowledged alarms, ON and OFF (default non-blocking Cicode) in
Plant.Area.Room* equipment and any lower-level equipment.
iRet = AlarmCountEquipment(1, "Plant.Area.Room");

// counts all disabled alarms (default non-blocking Cicode) in any lower-level equipment
from Plant.Area.
iRet = AlarmCountEquipment(3, "Plant.Area.");

// counts all unacknowledged alarms in Plant.Area.Room*, Plant.Area.Flat* equipment and any
lower-level equipment with category 10 (non-blocking Cicode)
iRet = AlarmCountEquipment(1, "Plant.Area.Room, Plant.Area.Flat", "Category=10", 1, 1);

// counts all unacknowledged alarms in Plant.Area.Room.* , Plant.Area.Flat.* equipment and
any lower-level equipment with category 10 (non-blocking Cicode).
// Note that Plant.Area.Room and Plat.Area.Flat equipment themselves do not count as a
period is given at the end of each expression.
iRet = AlarmCountEquipment(1, "Plant.Area.Room., Plant.Area.Flat.", "Category=10", 1, 1);
```

## Related Functions

[AlarmCount](#), [AlarmCountList](#)

## See Also

[Alarm Functions](#)

### AlarmCountList

Counts the available alarms for the selected filter criteria.

## Syntax

LONG **AlarmCountList**(INT *AN*)

*AN*:

An animation identifier of an alarm list.

## Return Value

Returns counted alarms for the selected filter criteria. Returns -1 when an error is detected.

## Example

```
// counts all listed alarms on the selected alarm page
INT iRet, iErr; // return values
iRet = AlarmCountList(21);
IF iRet = -1 THEN
    iErr = IsError(); // get error code)
END
```

## Related Functions

[AlarmCount](#)

## See Also

[Alarm Functions](#)

### AlarmDelete

Deletes alarm summary entries that are currently displayed. You can delete the alarm where the cursor is positioned, one or more alarm lists on the active page, a whole category of alarms, or alarms of a particular priority.

You would normally call this function from a keyboard command.

If this function is not called from a foreground task, it becomes a blocking function.

**Note:** This function will only work while the record is writable which is controlled by the ArchiveAfter Parameter. Refer to the topic *Configure the Archiving Parameters* in the main help for more information.

## Syntax

```
INT AlarmDelete(Mode, Value [, ClusterName] )
```

*Mode*:

The type of deletion:

0 - Delete a single alarm.

- Set *Value* to the AN where the alarm is displayed.
- If *Value* is set to 0, the current cursor position will be used.

1 - Delete a page of alarms. AN alarm page can contain more than one alarm list:

- Set *Value* to the AN where the alarm list is displayed.
- Set *Value* to 0 to delete the (displayed) alarm list (on the active page) where the cursor is positioned.
- Set *Value* to -1 to delete every (displayed) alarm list on the active page.

2 - Delete a category of alarms.

- Set *Value* to the alarm category (0-16375) of the alarms to delete. Please be aware that alarm category 0 indicates all categories; alarm category 255 indicates hardware alarms.

- Set *Value* to the group number to delete a group of categories.

3 - Delete alarms of a specific priority.

- Set *Value* to the alarm priority (0-255) of the alarms to be deleted.

Alarm priority 0 indicates all priorities. Hardware alarms are not affected by priority. Set *Value* to the group handle to delete a group of alarms of different priorities.

*Value*:

Used with Mode 1 or 2 to specify which alarms to delete.

*ClusterName*:

Used with Mode 2 or 3 to specify the name of the cluster in which the alarms being deleted reside. This argument is optional if the client is connected to only one cluster containing an Alarm Server or you are resolving the alarm server via the current cluster context. This argument is enclosed in quotation marks "".

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[GrpOpen](#), [AlarmSumDelete](#), [AlmSummaryDelete](#)

## Example

System keyboard	
Key Sequence	DelSum
Command	AlarmDelete(0, 0)
Comment	Delete the alarm summary entry where the cursor is positioned
System keyboard	
Key Sequence	ShiftDelSum
Command	AlarmDelete(1, -1)
Comment	Delete a page of alarm summary entries
System keyboard	
Key Sequence	SumDelete ### Enter
Command	AlarmDelete(2, Arg1, "clusterXYZ")
Comment	Delete alarm summary entries of a specified category in cluster XYZ
System keyboard	
Key Sequence	DelSum ##### Enter
Command	AlarmDelete(3,Arg1, "clusterXYZ")
Comment	Delete alarm summary entries of a specified priority in cluster XYZ

## See Also

[Alarm Functions](#)

### AlarmDisable

Disables alarms. You can disable the alarm where the cursor is positioned, one or more alarm lists on the active page, a whole category of alarms, or alarms of a particular priority.

You would normally call this function from a keyboard command. No action is taken if the alarms are already disabled. Use the [AlarmEnable](#) function to re-enable an alarm.

After you disable an alarm, the behavior is determined by the [\[Alarm\]DisplayDisable](#) parameter for your system to choose between taking the alarm off scan or suppressing it from the active alarm display.

If this function is not called from a foreground task, it becomes a blocking function.

## Syntax

**INT AlarmDisable(INT Mode, INT Value [, STRING ClusterName [, INT EndTime [, STRING Comment]]])**

*Mode:*

The type of disable:

0 - Disable a single alarm.

- Set *Value* to the AN where the alarm is displayed.

- If *Value* is set to 0, the current cursor position will be used.

1 - Disable a page of alarms. An alarm page can contain more than one alarm list:

- Set *Value* to the AN where the alarm list is displayed.

- Set *Value* to 0 to disable the (displayed) alarm list (on the active page) where the cursor is positioned.

- Set *Value* to -1 to disable all (displayed) alarm lists on the active page. This only applies to alarm lists created using [AlarmDsp](#) (and not those created using [AlarmDspLast](#)).

2 - Disable a category of alarms.

- Set *Value* to the alarm category (0-16375) of the alarms to be disabled. Please be aware that alarm category 0 indicates all categories; alarm category 255 indicates hardware alarms.

- Set *Value* to the group number to disable a group of categories.

3 - Disable alarms of a specific priority.

- Set *Value* to the alarm priority (0-255) of the alarms to be disabled.

Alarm priority 0 indicates all priorities. Hardware alarms are not affected by priority. Set *Value* to the group handle to disable a group of alarms of different priorities.

*Value:*

Used with Mode 1 and 2 to specify which alarms to disable.

*ClusterName:*

Used with Mode 2 or 3 to specify the name of the cluster where the alarms being disabled reside in. This argument is optional if the client is connected to only one cluster containing an alarm server, or if the alarm server is resolved via the current cluster context.

This argument is not required where the *mode* is 2 and the value is 255 (hardware alarm category).

This argument is enclosed in quotation marks "".

*EndTime:*

A date/time variable that indicates when the alarm will no longer be disabled. If this parameter is omitted or set to 0, the alarm will be disabled indefinitely.

*Comment:*

An optional comment limited to 200 characters explaining why the alarm is disabled. If the comment exceeds 200 characters, hardware error 274 ("Invalid argument passed") will be displayed.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[GrpOpen](#), [AlarmEnable](#), [AlarmDisableRec](#)

## Examples

```
// Disable the alarm where the cursor is positioned
AlarmDisable(0, 0);
// Disable a page of alarms
AlarmDisable(1, -1);
// Disable category 3 alarms in cluster XYZ
AlarmDisable(2, 3, "XYZ");
// Disable the alarm where the cursor is positioned for the next 60 minutes
nEndTime = DateAdd(TimeCurrent(), 3600);
AlarmDisable(0, 0, "Cluster1", nEndTime, "Shelve alarm for 60 minutes.");
// Disable a page of alarms until 20 Dec 2016 6:30am local time
nEndTime = DateAdd(StrToDate("20/12/2016"), StrToTime("6:30"));
AlarmDisable(1, -1, "Cluster1", nEndTime, "Shelve alarms until 20 Dec 2016 6:30am");
```

## See Also

[Alarm Functions](#)

### AlarmDisableRec

Disables alarms by record number on both the Primary and Standby Alarms Servers. This function can be called from alarm server or client and should not be used with a MsgRPC call to the alarm server.

This is a blocking function. If the function is called from a foreground task that is unable to block, an error will be returned.

## Syntax

INT **AlarmDisableRec**(LONG *Record* [, STRING *ClusterName* [, INT *EndTime* [, STRING *Comment*]]) )

*Record*:

The alarm record number, returned from any of the following alarm functions:

- *AlarmFirstCatRec()* or *AlarmNextCatRec()* - used to search for a record by alarm category, area, and type (acknowledged, disabled, etc.).
- *AlarmFirstPriRec()* or *AlarmNextPriRec()* - used to search for a record by alarm priority, area, and type (acknowledged, disabled, etc.).
- *AlarmFirstTagRec()* or *AlarmNextTagRec()* - used to search for a record by alarm tag, name, and description.
- *AlarmGetDsp()* - used to find the record that is displayed at a specified AN, for either an alarm list or alarm summary entry. Set the *sField* argument in *AlarmGetDsp()* to "RecNo".

To store this value, use data type Int in Cicode or Long for variable tags (Long needs 4 bytes).

*ClusterName*:

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

**EndTime:**

A date/time variable that indicates when the alarm will no longer be disabled. If this parameter is omitted or set to 0, the alarm will be disabled indefinitely.

**Comment:**

An optional comment limited to 200 characters explaining why the alarm is disabled. If the comment exceeds 200 characters, hardware error 274 ("Invalid argument passed") will be displayed.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[AlarmFirstTagRec](#), [AlarmNextTagRec](#), [AlarmDisable](#)

## Examples

```
/* Disable/enable the specified "Pump" alarm. Flag determines
whether the alarm is disabled (Flag=0) or enabled (Flag=1). */
FUNCTION
DisablePumps(STRING sTag, INT Flag)
    INT Current;
    INT Next;
    Current=AlarmFirstTagRec(sTag, "Pump", "");
    WHILE Current<>-1 DO
        Next=AlarmNextTagRec(Current,sTag, "Pump", "");
        IF Flag=0 THEN
            AlarmDisableRec(Current);
        ELSE
            AlarmEnableRec(Current);
        END
        Current=Next;
    END
END

// Disable alarm for the next 60 minutes
Current=AlarmFirstTagRec(sTag, "Pump", "");
nEndTime = DateAdd(TimeCurrent(), 3600);
AlarmDisableRec(Current, "Cluster1", nEndTime, "Shelve alarm for 60 minutes.");

// Disable alarm until 20 Dec 2016 6:30am local time
Current=AlarmFirstTagRec(sTag, "Pump", "");
nEndTime = DateAdd(StrToDate("20/12/2016"), StrToTime("6:30"));
AlarmDisableRec(Current, "Cluster1", nEndTime, "Shelve alarms until 20 Dec 2016 6:30am");
```

## See Also

[Alarm Functions](#)

## AlarmDisableTag

Disables alarms by Tag on both the Primary and Standby Alarms Servers.

If this function is not called from a foreground task, it becomes a blocking function.

## Syntax

```
INT AlarmDisableTag(STRING Tag, [, STRING ClusterName] [, INT EndTime [, STRING Comment]] )
```

*Tag*:

A string that identifies the alarm to acknowledge. It can be one of the following:

- An alarm tag — for example, "Fire1"
- An alarm equipment item — for example, "Motor1.AlarmFire"

Specify an empty string (" ") to match all alarm tags.

*ClusterName*:

The cluster where the tag resides.

*EndTime*:

A date/time variable that indicates when the alarm will no longer be disabled. If this parameter is omitted or set to 0, the alarm will be disabled indefinitely.

*Comment*:

An optional comment limited to 200 characters explaining why the alarm is disabled. If the comment exceeds 200 characters, hardware error 274 ("Invalid argument passed") will be displayed.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[AlarmAckTag](#), [AlarmEnableTag](#), [AlarmClearTag](#)

## Example

```
// Disable alarm for the next 60 minutes
nEndTime = DateAdd(TimeCurrent(), 3600);
AlarmDisableTag("ALARM_1", "Cluster1", nEndTime, "Shelve alarm for 60 minutes.");
// Disable alarm until 20 Dec 2016 6:30am local time
nEndTime = DateAdd(StrToDate("20/12/2016"), StrToTime("6:30"));
AlarmDisableTag("ALARM_1", "Cluster1", nEndTime, "Shelve alarms until 20 Dec 2016 6:30am");
```

## See Also

[Alarm Functions](#)

## AlarmDsp

Displays an alarm list, starting at a specified AN and then on subsequent ANs. You specify the number of alarms to display, the type of alarms and the name of the cluster the alarms belong to, for example, active hardware alarms or disabled non-hardware alarms in cluster XYZ. Before you call this function, you need to first add animation points to the graphics page for each alarm to be displayed.

If you only need to display the standard alarm page, use the PageAlarm function - it uses this AlarmDsp() function to display alarms. If you need more control over the display of alarms you can use this function, but only to display alarms on the alarm page. Use the AlarmDspLast function to display alarms on another graphics page (it uses less memory).

## Syntax

**INT AlarmDsp(INT AN, INT Count [, INT Type] [, STRING ClusterName] [, INT NoDraw] [, STRING CallbackFunc] )**

**AN:**

The AN where the first alarm is to display.

---

**Note:** The [\[Animator\]MaxAn](#) parameter sets the maximum AN which AlarmDsp will work with.

---

**Count:**

The number of alarms to display.

**nType:**

The type of alarms to display:

### Non-hardware alarms

0 - All active alarms, that is Types 1 and 2

1 - All unacknowledged alarms, ON and OFF

2 - All acknowledged ON alarms

3 - All disabled alarms

4 - All configured (non-hardware) alarms, that is Types 0 to 3, plus acknowledged OFF alarms.

### Hardware alarms

5 - All active alarms, that is Types 6 and 7

6 - All unacknowledged alarms, ON and OFF

7 - All acknowledged ON alarms

8 - All disabled alarms

9 - All configured alarms, that is Types 5 to 8

### Alarm Summary

10 - All summary alarms

15 – Sequence of events with configuration events filtered out

16 - Sequence of events

### Alarm General

11 - All ON alarms

12 - All OFF alarms

13 - All ON hardware alarms

14 - All OFF hardware alarms

17 - All unacknowledged ON alarms

18 - All unacknowledged OFF alarms

If you omit the Type, the default is 1.

*ClusterName:*

The cluster name to which the alarms belong. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

If the client is connected to only one cluster containing an alarm server then this argument is optional, the list returned will be limited to alarms within this cluster.

If the client is connected to clusters containing more than one alarm server then the Cluster Name needs to be specified. If a cluster name is not specified, alarms are returned for all clusters.

*NoDraw:*

Makes call to alarm server to update the ALMCB but does not automatically perform the animation of the data when the result is returned.

*CallbackFunc:*

Callback function to associate with the return of the ALMCB data from the Alarm Server.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[AlarmDspNext](#), [AlarmDspPrev](#), [AlarmDspLast](#), [AlarmGetInfo](#), [AlarmDspClusterAdd](#), [AlarmDspClusterRemove](#), [AlarmDspClusterInUse](#)

## Example

Advanced Animation	
Command	AlarmDsp(20,15,3)
Comment	Display 15 disabled alarms at AN 20

## See Also

[Alarm Functions](#)

## AlarmDspClusterAdd

Adds a cluster to a client's alarm list. Alarms in the specified cluster (that correspond to the mode set in [AlarmDsp](#)) will be added to the alarm list at the AN number.

## Syntax

**INT AlarmDspClusterAdd(*nAN*, *sClusterName*)**

*nAN*:

The AN used in the original AlarmDsp call.

*sClusterName*:

The name of the cluster to be used for this alarm list. The argument is enclosed in quotation marks ("").

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[AlarmDspClusterRemove](#), [AlarmDspClusterInUse](#)

## See Also

[Alarm Functions](#)

## AlarmDspClusterInUse

Determines if a cluster is included in a client's alarm list.

## Syntax

**INT AlarmDspClusterInUse(*nAN*, *sClusterName*)**

*nAN*:

The AN used in the original [AlarmDsp](#) call.

*sClusterName*:

The name of the cluster to query an alarm list for to determine if it's included. The argument is enclosed in quotation marks ("").

## Return Value

Returns a Boolean value: True (1) if successful, otherwise False (0) is returned.

## Related Functions

[AlarmDspClusterAdd](#) [AlarmDspClusterRemove](#) [AlarmDsp](#)

## See Also

[Alarm Functions](#)

### AlarmDspClusterRemove

Removes a cluster from a client's alarms list. Alarms for the specified cluster will be removed from the alarms list at the AN number.

If the cluster to be removed is the last cluster, the call will be unsuccessful.

## Syntax

**INT AlarmDspClusterRemove(*nAN*, *sClusterName*)**

*nAN*:

The AN used in the original [AlarmDsp](#) call.

*sClusterName*:

The name of the cluster to remove from this alarm list. The argument is enclosed in quotation marks ("").

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[AlarmDspClusterAdd](#), [AlarmDspClusterInUse](#), [AlarmDsp](#)

## See Also

[Alarm Functions](#)

### AlarmDspLast

Displays the latest alarms, at a specified AN with the cluster name. Use this function to display the last alarms. You can specify the number of alarms to display of a specified type, for example, active hardware alarms or disabled non-hardware alarms.

## Syntax

**INT AlarmDspLast(*nAN* [, *nCount*] [, *nType*] [, *sClusterName*] [, *iNoDraw*] [, *sCallbackFunc*] )**

*nAN*:

The AN where the last alarms are to be displayed.

---

**Note:** The [\[Animator\]MaxAn](#) parameter sets the maximum AN which **AlarmDspLast** will work with.

---

*Count*:

The number of alarms to display. If you omit the Count, the default is 1.

*nType:*

The type of alarms to display:

#### **Non-hardware alarms**

0 - All active alarms, that is Types 1 and 2

1 - All unacknowledged alarms, ON and OFF

2 - All acknowledged ON alarms

3 - All disabled alarms

4 - All configured (non-hardware) alarms, that is Types 0 to 3, plus acknowledged OFF alarms.

#### **Hardware alarms**

5 - All active alarms, that is Types 6 and 7

6 - All unacknowledged alarms, ON and OFF

7 - All acknowledged ON alarms

8 - All disabled alarms

9 - All configured alarms, that is Types 5 to 8

#### **Alarm Summary**

10 - All summary alarms

15 – Sequence of events with configuration events filtered out

16 - Sequence of events

#### **Alarm General**

11 - All ON alarms

12 - All OFF alarms

13 - All ON hardware alarms

14 - All OFF hardware alarms

17 - All unacknowledged ON alarms

18 - All unacknowledged OFF alarms

If you omit the Type, the default is 1.

*sClusterName:*

The cluster name to which the alarms belong. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks " ".

If a cluster name is not specified, alarms are returned for all clusters.

*iNoDraw:*

Makes call to alarm server to update the ALMCB but does not automatically perform the animation of the data when the result is returned.

*sCallbackFunc:*

Callback function to associate with the return of the ALMCB data from the Alarm Server.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[AlarmDsp](#)

## Example

Advanced Animation	
Command	AlarmDspLast(11, 'ClusterXYZ')
Comment	Display the last alarm at AN 11
Advanced Animation	
Command	AlarmDspLast(21,3, 'ClusterXYZ')
Comment	Display the last 3 alarms at AN 21

## See Also

[Alarm Functions](#)

## AlarmDspNext

Displays the next page of alarms. This function pages down (scrolls) the alarms displayed by the [AlarmDsp](#) function. You would normally call this function from a keyboard command.

## Syntax

**INT AlarmDspNext(INT AN)**

**AN:**

The AN where the alarm list is displayed, or:

-1 - Scroll every alarm list displayed on the page.

0 - Scroll the alarm list where the cursor is positioned.

---

**Note:** An alarm page can contain more than one alarm list.

---

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[AlarmDsp](#), [AlarmDspPrev](#)

## Example

System keyboard	
Key Sequence	NextAlarm
Command	AlarmDspNext(20)
Comment	Display the next page of alarms (from the alarm list) at AN20

## See Also

[Alarm Functions](#)

## AlarmDspPrev

Displays the previous page of alarms. This function pages up (scrolls) the alarms displayed by the [AlarmDsp](#) function. You would normally call this function from a keyboard command.

## Syntax

**INT AlarmDspPrev(INT AN)**

**AN:**

The AN where the alarm list is displayed, or:

-1 - Scroll every alarm list displayed on the page.

0 - Scroll the alarm list where the cursor is positioned.

---

**Note:** An alarm page can contain more than one alarm list.

---

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[AlarmDsp](#), [AlarmDspNext](#)

## Example

System keyboard	
Key Sequence	PrevAlarm
Command	AlarmDspPrev(20)
Comment	Display the previous page of alarms (from the alarm list) at AN20

## See Also

[Alarm Functions](#)

### AlarmEnable

Enables an alarm on the active alarm list. You can enable the alarm where the cursor is positioned, one or more alarm lists on the active page, a whole category of alarms, or alarms of a particular priority.

No action is taken if the alarms are already enabled. You would normally call this function from a keyboard command.

If this function is not called from a foreground task, it becomes a blocking function.

## Syntax

**INT AlarmEnable(INT Mode, INT Value [, STRING ClusterName [, INT bAcknowledge]] )**

*Mode:*

The type of enable:

0 - Enable a single alarm where the cursor is positioned.

- Set *Value* to the AN where the alarm list is displayed.

• If *Value* is set to 0, the current cursor position will be used.

1 - Enable a page of alarms. An alarm page can contain more than one alarm list:

- Set *Value* to the AN where the alarm list is displayed.

• Set *Value* to 0 to enable the (displayed) alarm list (on the active page) where the cursor is positioned.

• Set *Value* to -1 to enable all (displayed) alarm lists on the active page. This only applies to alarm lists created using AlarmDsp (and not those created using AlarmDspLast).

2 - Enable a category of alarms.

• Set *Value* to the alarm category (0-16375) of the alarms to be enabled. Please be aware that alarm category 0 indicates all categories; alarm category 255 indicates hardware alarms.

- Set *Value* to the group number to enable a group of categories.

3 - Enable alarms of a specific priority.

- Set *Value* to the alarm priority (0-255) of the alarms to be enabled.

Alarm priority 0 indicates all priorities. Hardware alarms are not affected by priority. 3) Set *Value* to the group handle to enable a group of alarms of different priorities.

**Value:**

Used with Mode 0, 1 and 2 to specify which alarms to enable.

**ClusterName:**

Used with Mode 2 or 3 to specify the name of the cluster where the alarms being enabled reside in. This argument is optional if the client is connected to only one cluster containing an Alarm Server or are resolving the alarm server via the current cluster context.

This argument is not required where the *mode* is 2 and the value is 255 (hardware alarm category).

This argument is enclosed in quotation marks "".

**bAcknowledge:**

Forces acknowledgment of an alarm after it is enabled. The accepted values are:

0 — (default) enforced acknowledgment will not be applied.

1 — alarm will be acknowledged when enabled.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[GrpOpen](#), [AlarmDisable](#), [AlarmEnableRec](#)

## Examples

```
// Enable the alarm where the cursor is positioned
AlarmEnable(0, 0);

// Enable a page of alarms
AlarmEnable(1, -1);

// Enable category 3 alarms in cluster XYZ
AlarmEnable(2, 3, "XYZ");

// Enable the alarm where the cursor is positioned and enforce acknowledgment
AlarmEnable(0, 0, "Cluster1", 1);
```

## See Also

[Alarm Functions](#)

### AlarmEnableRec

Enables alarms by record number on both the primary and standby alarms servers. This function can be called from an alarm server or client and should not be used with a MsgRPC call to the alarm server.

This is a blocking function. If the function is called from a foreground task that is unable to block, an error will be returned.

## Syntax

```
INT AlarmEnableRec(INT Record [, STRING ClusterName [, INT bAcknowledge]])
```

### Record:

The alarm record number, returned from any of the following alarm functions:

- `AlarmFirstCatRec()` or `AlarmNextCatRec()` — used to search for a record by alarm category, area, and type (acknowledged, disabled, etc.).
- `AlarmFirstPriRec()` or `AlarmNextPriRec()` — used to search for a record by alarm priority, area, and type (acknowledged, disabled, etc.).
- `AlarmFirstTagRec()` or `AlarmNextTagRec()` — used to search for a record by alarm tag, name, and description.
- `AlarmGetDsp()` — used to find the record that is displayed at a specified AN, for either an alarm list or alarm summary entry. Set the `sField` argument in `AlarmGetDsp()` to "RecNo".

To store this value, use data type Int in Cicode or Long for variable tags (Long needs 4 bytes).

### ClusterName:

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks " ".

### bAcknowledge:

Forces acknowledgment of an alarm after it is enabled. The accepted values are:

- 0 — (default) enforced acknowledgment will not be applied.  
1 — alarm will be acknowledged when enabled.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[AlarmFirstTagRec](#), [AlarmNextTagRec](#), [AlarmEnable](#), [AlarmDisableRec](#), [AlarmGetDsp](#)

## Examples

```
/* Disable/enable the specified "Pump" alarm. Flag determines
whether the alarm is disabled (Flag=0) or enabled and acknowledged (Flag=1). */
FUNCTION
DisablePumps(STRING sTag, INT Flag)
    INT Current;
    INT Next;
    Current=AlarmFirstTagRec(sTag,"Pump","");
    WHILE Current<>-1 DO
        Next=AlarmNextTagRec(Current,sTag,"Pump","");
        IF Flag=0 THEN
            AlarmDisableRec(Current);
        ELSE
            AlarmEnableRec(Current,"",1);
        END
    
```

```
    Current=Next;  
    END  
END
```

## See Also

[Alarm Functions](#)

### AlarmEnableTag

Enables alarms by Tag on both the Primary and Standby Alarms Servers.

If this function is not called from a foreground task, it becomes a blocking function.

## Syntax

```
INT AlarmEnableTag(STRING Tag [, STRING ClusterName [, INT bAcknowledge]])
```

*Tag*:

A string that identifies the alarm to acknowledge. It can be one of the following:

- An alarm tag — for example, "Fire1"
- An alarm equipment item — for example, "Motor1.AlarmFire"

Specify an empty string (" ") to match all alarm tags.

*ClusterName*:

The cluster where the tag resides.

*bAcknowledge*:

Forces acknowledgment of an alarm after it is enabled. The accepted values are:

0 — (default) enforced acknowledgment will not be applied.

1 — alarm will be acknowledged when enabled.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[AlarmAckTag](#), [AlarmDisableTag](#), [AlarmClearTag](#)

## Example

```
// Enable alarm and enforce acknowledgment  
AlarmEnableTag("ALARM_1", "Cluster1", 1);
```

## See Also

[Alarm Functions](#)

### AlarmEventQue

Opens the alarm event queue. The alarms server writes events into this queue as they are processed. These events include activated, reset, acknowledged, enabled and disabled alarms.

**Note:** This function can only be called on the alarm server.

To read events from this queue, use the [QueRead\(\)](#) or [QuePeek\(\)](#) functions. The data put into the queue is the alarm record identifier (into the **Type** field) and the alarm event format (into the **Str** field). The function puts every state change into the queue and Plant SCADA does not use this queue for anything.

To use this function, you need to enable the alarm event queue with the [\[Alarm\]EventQue](#) parameter. This parameter will tell the Alarms Server to start placing events into the queue. The [\[Alarm\]EventFmt](#) parameter defines the format of the data placed into the string field. You can enable the EventQue parameter without setting the event format so that the Alarms Server does not place a formatted string into the queue.

Enabling this formatting feature can increase CPU loading and reduce performance of the Alarms Server as every alarm is formatted and placed in the queue. You should reconsider using this feature if a decrease in performance is noticeable.

The maximum length of each queue is controlled by the [\[Code\]Queue](#) parameter. You may need to adjust this parameter so as not to miss alarm events. When the queue is full, the alarms server will discard events.

#### **WARNING**

##### **UNINTENDED EQUIPMENT OPERATION**

You may need to adjust the [\[Code\]Queue](#) parameter so as not to miss alarm events. When the queue is full, the Alarms Server will discard events.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Syntax

`AlarmEventQue()`

## Return Value

The handle of the alarm event queue, or -1 if the queue cannot be opened.

## Related Functions

[QueRead](#), [QuePeek](#), [TagWriteEventQue](#)

## Example

```
hQue = AlarmEventQue()
WHILE TRUE DO
    QueRead(hQue, nRecord, sAlarmFmt, 1);
    /* do whatever with the alarm event */
    ...
    Sleep(0);
END
```

## See Also

[Alarm Functions](#)

### AlarmFirstCatRec

Searches for the first occurrence of an alarm category and type. You can search all areas, the current area only, or specify an area to limit the search.

This function returns an alarm record identifier that you can use in other alarm functions, for example, to acknowledge, disable, or enable the alarm, or to get field data on that alarm.

---

**Note:** Record numbers obtained from [AlarmGetDsp](#) are not valid for this function.

This is a blocking function. If the function is called from a foreground task that is unable to block, the return value will be -1 and a hardware alarm set. Use [IsError\(\)](#) to retrieve the error code.

## Syntax

LONG **AlarmFirstCatRec**(INT *Category*, INT *Type* [, INT *Area*] [, STRING *ClusterName*] )

### *Category*:

The alarm category or group number to match. Set Category to 0 (zero) to match all alarm categories.

### *nType*:

The type of alarms to display:

#### **Non-hardware alarms**

0 - All active alarms, that is Types 1 and 2

1 - All unacknowledged alarms, ON and OFF

2 - All acknowledged ON alarms

3 - All disabled alarms

4 - All configured (non-hardware) alarms, that is Types 0 to 3, plus acknowledged OFF alarms.

#### **Alarm General**

11 - All ON alarms

12 - All OFF alarms

17 - All unacknowledged ON alarms

18 - All unacknowledged OFF alarms

If you omit the Type, the default is 1.

---

**Note:** If you call this function with the *Type* argument set to an unsupported value, it may return unexpected results.

**Area:**

The area in which to search for alarms. If you do not specify an area, or if you set Area to -1, only the current area will be searched.

**ClusterName:**

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

The alarm record identifier or -1 if no match is found.

## Related Functions

[GrpOpen](#), [AlarmNextCatRec](#), [AlarmFirstPriRec](#), [AlarmNextPriRec](#), [AlarmGetFieldRec](#), [AlarmAckRec](#),  
[AlarmDisableRec](#), [AlarmEnableRec](#), [AlarmGetThresholdRec](#), [AlarmSetThresholdRec](#)

## Example

See [AlarmAckRec](#)

## See Also

[Alarm Functions](#)

### AlarmFirstPriRec

Searches for the first occurrence of an alarm priority and type. You can search all areas, the current area only, or specify an area to limit the search.

This function returns an alarm record identifier that you can use in other alarm functions, for example, to acknowledge, disable, or enable the alarm, or to get field data on that alarm.

---

**Note:** Record numbers obtained from [AlarmGetDsp](#) are not valid for this function.

---

This is a blocking function. If the function is called from a foreground task that is unable to block, the return value will be -1 and a hardware alarm set. Use [IsError\(\)](#) to retrieve the error code.

---

**Note:** This function will return a match for an Acknowledge Off alarm with [\[Alarm\]AckHold=1](#) even after it has been cleared using [AlarmClear](#) or [AlarmClearRec](#).

## Syntax

LONG **AlarmFirstPriRec**(INT *Priority*, INT *Type* [, INT *Area*] [, STRING *ClusterName*] )

***Priority:***

The alarm Priority or group handle of a group of alarm priorities. Set Priority to 0 (zero) to match all alarm priorities.

***nType:***

The type of alarms to display:

#### Non-hardware alarms

- 0 - All active alarms, that is Types 1 and 2
- 1 - All unacknowledged alarms, ON and OFF
- 2 - All acknowledged ON alarms
- 3 - All disabled alarms
- 4 - All configured (non-hardware) alarms, that is Types 0 to 3, plus acknowledged OFF alarms.

#### Alarm General

- 11 - All ON alarms
- 12 - All OFF alarms
- 17 - All unacknowledged ON alarms
- 18 - All unacknowledged OFF alarms

If you omit the Type, the default is 1.

---

**Note:** If you call this function with the *Type* argument set to an unsupported value, it may return unexpected results.

---

#### Area:

The area in which to search for alarms. If you do not specify an area, or if you set Area to -1, only the current area will be searched.

#### ClusterName:

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

The alarm record identifier or -1 if no match is found. If you do not specify an area, only alarms in the current area on the alarms server are searched.

## Related Functions

[GrpOpen](#), [AlarmNextCatRec](#), [AlarmNextPriRec](#), [AlarmGetFieldRec](#), [AlarmAckRec](#), [AlarmDisableRec](#),  
[AlarmEnableRec](#), [AlarmGetThresholdRec](#), [AlarmSetThresholdRec](#)

## Example

```
/* Acknowledge all unacknowledged (Type 1) alarms of the specified
alarm priority. */
FUNCTION
AutoAccept(INT iPriority)
    INT iCurrent;
    INT iNext;
    iCurrent=AlarmFirstPriRec(iPriority,1,-1);
    WHILE iCurrent <>-1 DO
        iNext=AlarmNextPriRec(iCurrent,iPriority,1,-1);
```

```
    AlarmAckRec(iCurrent);
    iCurrent=iNext;
END
END
```

## See Also

[Alarm Functions](#)

### AlarmFirstTagRec

Searches for the first occurrence of an alarm tag, name, and description.

This is a blocking function. If the function is called from a foreground task that is unable to block, the return value will be -1 and a hardware alarm set. Use IsError() to retrieve the error code.

---

**Note:** Record numbers obtained from [AlarmGetDsp](#) are not valid for this function.

This function returns an alarm record identifier that you can use in other alarm functions, for example, to acknowledge, disable, or enable the alarm, or to get field data on that alarm.

---

**Note:** This function will return a match for an Acknowledge Off alarm with [Alarm]AckHold=1 even after it has been cleared using [AlarmClear](#) or [AlarmClearRec](#).

For complex filtering operations it is more efficient to use the alarm tag browse functions [AlmBrowseOpen](#) and [AlmBrowseNext](#)

## Syntax

LONG **AlarmFirstTagRec**(STRING Tag, STRING Name, STRING Description [, STRING ClusterName] )

*Tag:*

A string that identifies the tag to be matched. It can be one of the following:

- An alarm tag — for example, "Fire1"
- An alarm equipment item — for example, "Motor1.AlarmFire"

Specify an empty string (" ") to match all alarm tags.

*Name:*

The alarm name to be matched. Specify an empty string (" ") to match all alarm names.

*Description:*

The alarm description to be matched. Specify an empty string (" ") to match all alarm descriptions.

*ClusterName:*

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

The alarm record identifier or -1 if no match is found.

---

**Note:** The order in which alarms are returned has changed from version 7.20. In later versions, unacknowledged alarms are no longer returned first.

## Related Functions

[AlarmNextTagRec](#), [AlarmGetFieldRec](#), [AlarmAckRec](#), [AlarmDisableRec](#), [AlarmEnableRec](#), [AlarmGetThresholdRec](#), [AlarmSetThresholdRec](#), [AlmBrowseOpen](#), [AlmBrowseNext](#)

## Example

See [AlarmDisableRec](#)

## See Also

[Alarm Functions](#)

## AlarmGetDelay

Gets the delay setting for the alarm the cursor is currently positioned over.

## Syntax

LONG **AlarmGetDelay**(*nType*)

*nType*:

The type of delay:

- 0 - Delay (digital alarm/advanced alarm)
- 1 - High high delay (analog alarm)
- 2 - High delay (analog alarm)
- 3 - Low delay (analog alarm)
- 4 - Low low delay (analog alarm)
- 5 - Deviation delay (analog alarm)

## Return Value

The alarm delay if successful, otherwise -1 is returned. Use [IsError\(\)](#) to retrieve extended error information.

## Related Functions

[AlarmNotifyVarChange](#), [AlarmSetDelayRec](#), [AlarmGetDelayRec](#)

## See Also

[Alarm Functions](#)

## AlarmGetDelayRec

Gets the delay setting for an alarm via the alarm record number.

This is a blocking function. If the function is called from a foreground task that is unable to block, the return value will be -1 and a hardware alarm set.

## Syntax

```
LONG AlarmGetDelayRec(LONG Record, INT Type [, STRING ClusterName] )
```

### Record:

The alarm record number, returned from any of the following alarm functions:

- AlarmFirstCatRec() or AlarmNextCatRec() - used to search for a record by alarm category, area, and type (acknowledged, disabled, etc.).
- AlarmFirstPriRec() or AlarmNextPriRec() - used to search for a record by alarm priority, area, and type (acknowledged, disabled, etc.).
- AlarmFirstTagRec() or AlarmNextTagRec() - used to search for a record by alarm tag, name, and description.
- AlarmGetDsp() - used to find the record that is displayed at a specified AN, for either an alarm list or alarm summary entry. Set the *sField* argument in AlarmGetDsp() to "RecNo".

### Type:

The type of delay:

- 0 - Delay (digital alarm/advanced alarm)
- 1 - High high delay (analog alarm)
- 2 - High delay (analog alarm)
- 3 - Low delay (analog alarm)
- 4 - Low low delay (analog alarm)
- 5 - Deviation delay (analog alarm)

### ClusterName:

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

The alarm delay if successful, otherwise -1 is returned. Use IsError() to retrieve extended error information.

## Related Functions

[AlarmFirstPriRec](#), [AlarmNextPriRec](#), [AlarmFirstTagRec](#), [AlarmNextTagRec](#), [AlarmNotifyVarChange](#),  
[AlarmSetDelayRec](#), [AlarmGetDelay](#), [AlarmGetDsp](#)

## See Also

[Alarm Functions](#)

## AlarmGetDsp

Gets field data from the alarm record that is displayed at the specified AN. You can use this function for Alarm Pages, Sequence of Events pages, and Alarm Summary pages (an Alarm Page, SOE Page, and Alarm Summary needs to be displayed before this function can be used).

You can call this function on an Alarms Server or a client to get the contents of any field in the alarm record at that AN.

You can return the record number of the alarm record for use in other alarm functions, for example, to acknowledge, disable, or enable an alarm (on an Alarms Server).

The `AlarmGetDsp()` function does not support hardware alarms.

If this function is not called from a foreground task, it becomes a blocking function.

## Syntax

STRING `AlarmGetDsp(INT AN, STRING Field)`

*AN*:

AN number of an ALMCB Alarm record. Equal to AN where actual ALMCF resides + Offset into the list of ALMCF records.

*Field*:

The name of the field from which the data is retrieved.

The contents of the following fields can be retrieved when the alarm page is displayed.

ALMCOMMENT, AREA, CATEGORY, CAUSE1..8, CLUSTER, COMMENT, CONSEQUENCE1..8, CUSTOM1..8, DATE, DATEEXT, DEADBAND, DELAY, DEVDELAY, DEVIATION, DESC, DISABLECOMMENT, DISABLEENDDATE, DISABLEENDDATEEXT, DISABLEENDTIME, EQUIPMENT, FONT, FORMAT, HDELAY, HELP, HHDELAY, HIGH, HIGHHIGH, LDELAY, LLDELAY, LOGSTATE, LOW, LOWLOW, NAME, PAGING, PAGINGGROUP, PRIORITY, RATE, RECN0, RESPONSE1..8, RESPONSEN0, STATE, STATE\_DESC, TAG, TIME, TYPE, USERLOCATION, VALUE.

The contents of any of the above fields (except for State) and the following fields can be retrieved when the Alarm Summary is displayed:

ACKDATE, ACKDATEEXT, ACKTIME, DELTATIME, FULLNAME, NATIVE\_SUMDESC, NATIVE\_COMMENT, OFFDATE, OFFDATEEXT, OFFMILLI, OFFTIME, ONDATE, ONDATEEXT, ONMILLI, ONTIME, SUMSTATE, SUMDESC, USERNAME.

The following fields can be retrieved when the Alarm SOE is displayed:

ALMCOMMENT, AREA, CATEGORY, CLUSTER, COMMENT, CUSTOM1..8, DATE, DATEEXT, DEADBAND, DEVIATION, EQUIPMENT, FULLNAME, HIGH, HIGHHIGH, LOGSTATE, LOW, LOWLOW, MESSAGE, NAME, NATIVE\_COMMENT, NATIVE\_SUMDESC, PAGING, PAGINGGROUP, PRIORITY, RATE, RECORDID, STATE, STATE\_DESC, SUMDESC, SUMSTATE, TAG, TIME, TYPE, USERLOCATION, USERNAME.

See [Browse Function Field Reference](#) for more information about these fields.

## Return Value

The alarm field data (as a string) or empty string "".

## Related Functions

[AlarmDsp](#)

## Example

```
! Display the tag and category for the alarm at the specified AN.
FUNCTION
AlarmData(INT AN)
    STRING Category;
    STRING Tag;
    Category=AlarmGetDsp(AN,"Category");
    Tag=AlarmGetDsp(AN,"Tag");
    Prompt("Alarm "+Tag+" is Category "+Category);
END
```

## See Also

[Alarm Functions](#)

### AlarmGetInfo

Gets data on the alarm list displayed at a specified AN. Use this function to display the current alarm list information on an alarm page. If only one alarm list has been configured on an alarm page, modes 2 and 3 of this function return the current alarm page information.

---

**Note:** You cannot retrieve the order by key setting for an alarm list using this function, as it can only returns numeric values. To retrieve this information, use the function [AlarmGetOrderbyKey](#).

---

If this function is not called from a foreground task, it becomes a blocking function.

## Syntax

LONG **AlarmGetInfo**(INT *AN*, INT *Type* [, STRING *ClusterName*])

*AN*:

The AN where the alarm list (with the required information) is displayed. Set the AN to 0 (zero) to get information on the alarm list where the cursor is positioned.

*Type*:

The type of data:

0 - Alarm page number. The vertical offset (in pages) from the AN where the alarm list commenced. The alarm list need to have scrolled off the first page for this type to return a non-zero value.

1 - Alarm list offset. The vertical offset (in lines) from the AN where the alarm list commenced. You need to have scrolled off the first page of alarms for this type to return a non zero value.

2 - Category of alarms displayed on the alarm list. You can use a group number to display a group of categories.

3 - Type of alarms displayed on the alarm list. See [AlarmDsp\(\)](#) for a list of these types.

7 - Priority of alarms displayed on the alarm list. The return value may be a group number if the alarm list

contains alarms of more than one priority.

8 - Display mode of the alarm list.

9 - Sorting mode of the alarm list.

10 – Reading this field is invalid, use the function [AlarmGetOrderbyKey](#).

11 – Retrieves the error code for the last alarm summary request that was not able to be processed due to a buffer overflow. The last request error value will be reset on the next successful response from the servers.

12 – Returns values as follows:

- 0 = no named filter, and no custom filter.
- 1 = named filter set, no custom filter (this means that the content of the named filter is empty).
- 2 = no named filter, but there is custom filtering applied (this is possible if the filter is edited via the AN using `AlarmFilterEdit` functions or some other method).
- 3 = named filter set, custom filtering is applied (it is possible that this is due to the named filter being edited or any other method through the AN).

13 – Returns values as follows;

- 0 = data from none of the clusters is ready
- 1 = only data from some clusters is ready
- 2 = data from all clusters is ready

14 - Identifies if the data is available for a particular cluster. Returns 1 if data is ready, or 0.

15 – Auto-refresh mode, where mode is:

- -1 = always auto refresh
- 0 = auto refresh disabled
- 1 = auto refresh page 1, disable on all other pages.

**Note:** Type 15 is only in use if displaying the alarm summary (AlarmDsp Type 10).

16 – Timeout occurred. This value is set to 0 when a data fetch is initiated for the supplied alarm list (indicated by AN). This value is set to 1 if a timeout occurs within the fetch. The timeout period is specified by the INI parameter [Alarm]AlarmListRequestTimeout.

- 0 = no timeout occurred
- 1 = timeout occurred.

17 – Alarm page number for specified AN is displayed. When the alarm list is scrolled by vertical offset (in lines), the list may span across two pages of alarms. Calling this function returns the page number of the alarm list that the specified AN is displaying. This differs to Type 0 which returns the page number based on the first line of the alarm display.

*ClusterName:*

Specifies the name of the cluster in which the Alarm Server resides. This is only required for type 14. The argument is enclosed in quotation marks " ".

## Return Value

Alarm list data as a numeric value.

## Related Functions

[AlarmDsp](#), [AlarmSetInfo](#), [AlarmGetOrderbyKey](#).

## Example

```
/* In the following examples, data is returned on the alarm
list where the cursor is positioned. */
page = AlarmGetInfo(0,0);
! returns the alarm page number.
offset = AlarmGetInfo(0,1);
! returns the alarm list offset.
cat = AlarmGetInfo(0,2);
! returns the alarm category displayed.
type = AlarmGetInfo(0,3);
! returns the type of alarms displayed.
```

## See Also

[Alarm Functions](#)

## AlarmGetFieldRec

Gets the contents of the specified field in the specified alarm record.

This is a blocking function. If the function is called from a foreground task that is unable to block, the return value will be -1 and a hardware alarm set. Use IsError() to retrieve the error code.

## Syntax

STRING **AlarmGetFieldRec**(LONG *Record*, STRING *Field* [, INT *Ver* [, STRING *ClusterName*]])

### *Record:*

The alarm record number, returned from any of the following alarm functions:

- `AlarmFirstCatRec()` or `AlarmNextCatRec()` - used to search for a record by alarm category, area, and type (acknowledged, disabled, etc.).
- `AlarmFirstPriRec()` or `AlarmNextPriRec()` - used to search for a record by alarm priority, area, and type (acknowledged, disabled, etc.).
- `AlarmFirstTagRec()` or `AlarmNextTagRec()` - used to search for a record by alarm tag, name, and description.

To store this value, use data type Int in Cicode or Long for variable tags (Long needs 4 bytes).

### *Field:*

The name of the field from which the data is retrieved.

ACKDATE, ACKDATEEXT, ACKMILLI, ACKTIME, ACKUTC, ACQDESC, ACQERROR, ALARMTYPE, ALMCOMMENT, AREA, ARR\_SIZE, CATEGORY, CAUSE1...8, CLASSIFICATION, CLUSTER, COMMENT, CONSEQUENCE1...8, CUSTOM1...8, DATE, DATEEXT, DEADBAND, DELAY, DELTAMILLI, DELTATIME, DESC, DEVDELAY, DEVIATION, DISABLECOMMENT, DISABLEDDATE, DISABLEDTIME, DISABLEENDDATE, DISABLEENDDATEEXT, DISABLEENDTIME, ENG\_ZERO, ERRDESC, ERPPAGE, EQUIPMENT, FORMAT, FULLNAME, GROUP, HDELAY, HELP, HHDELAY, HIGH,

HIGHHIGH, HISTORIAN, ITEM, LDELAY, LLDELAY, LOCALTIMEDATE, LOGSTATE, LOW, LOWLOW, MESSAGE, MILLISEC, NAME, NATIVE\_COMMENT, NATIVE\_DESC, NATIVE\_NAME, NATIVE\_SUMDESC, OFFDATE, OFFDATEEXT, OFFMILLI, OFFTIME, OFFTIMEDATE, OFFUTC, OLD\_DESC, ONDATE, ONDATEEXT, ONMILLI, ONTIME, ONTIMEDATE, ONUTC, PAGING, PAGINGGROUP, PRIORITY, PRIV, PSI\_TYPE, RATE, RECEIPTLOCALTIMEDATE, RECEIPTDATE, RECEIPTDATEEXT, RECEIPTMILLISEC, RECEIPTTIME, RECEIPTTIMEINT, RECEIPTTIMETICKS, RESPONSE1...8, RESPONENUM, SETPOINT, STATE, STATE\_DESC, STATE\_DESC0...7, SUMDESC, SUMSTATE, SUMTYPE, TAG, TAGEX, TAGGENLINK, TIME, TIMEDATE, TIMEINT, TIMETICKS, TSQUALITY, TYPE, TYPENUM, USERDESC, USERNAME, USERLOCATION, VALUE.

See [Browse Function Field Reference](#) for more information about these fields.

*nVer:*

The version of an alarm.

If an alarm has been triggered more than once in a given period, the version lets you distinguish between different instances of the alarm's activity.

The version is used in filtering alarms for display. A query function passes a value to this parameter in order to get field information for a particular alarm.

This parameter is not needed when you use AlarmGetFieldRec() for purposes other than filtering. It will default to 0 if omitted.

*ClusterName:*

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

The alarm field data (as a string) or empty string "".

## Related Functions

[AlarmFirstTagRec](#), [AlarmNextTagRec](#),

## Example

```
FUNCTION
GetNameFromTag(STRING sTag)
    INT record;
    STRING sName
    record = AlarmFirstTagRec(sTag, "", "");
    IF record <> -1 THEN
        sName = AlarmGetFieldRec(record,"NAME");
    ELSE
        sName = "";
    END
    RETURN sName;
END
```

## See Also

[Alarm Functions](#)

### AlarmGetOrderbyKey

Retrieves the list of key(s) that are used to determine the order of the alarm list. These keys can be set by the [AlarmSetInfo](#) function.

## Syntax

STRING **AlarmGetOrderbyKey**(*nAN*)

*nAN*:

The AN where the alarm list (with the required information) is displayed.

## Return Value

Order-by key (as a string).

## Example

```
page = AlarmGetOrderbyKey(21);
! returns the order-by key string of the alarm list at AN '21'.
```

## See Also

[Alarm Functions](#)

### AlarmGetThreshold

Gets the threshold of the analog alarm where the cursor is positioned.

## Syntax

REAL **AlarmGetThreshold**(*nType*)

*nType*:

The type of threshold:

0 - High high

1 - High

2 - Low

3 - Low low

4 - Deadband

5 - Deviation

6 - Rate of change

## Return Value

The alarm threshold.

## Related Functions

[AlarmGetThresholdRec](#), [AlarmSetThreshold](#), [AlarmSetThresholdRec](#)

## See Also

[Alarm Functions](#)

### AlarmGetThresholdRec

Gets the threshold of analog alarms by the alarm record number.

This is a blocking function. If the function is called from a foreground task that is unable to block, the return value will be -1 and a hardware alarm set. Use IsError() to retrieve the error code.

## Syntax

REAL **AlarmGetThresholdRec**(LONG *Record*, INT *Type* [, STRING *ClusterName*])

*Record*:

The alarm record number, returned from any of the following alarm functions:

- `AlarmFirstCatRec()` or `AlarmNextCatRec()` - used to search for a record by alarm category, area, and type (acknowledged, disabled, etc.).
- `AlarmFirstPriRec()` or `AlarmNextPriRec()` - used to search for a record by alarm priority, area, and type (acknowledged, disabled, etc.).
- `AlarmFirstTagRec()` or `AlarmNextTagRec()` - used to search for a record by alarm tag, name, and description.

To store this value, use data type Int in Cicode or Long for variable tags (Long needs 4 bytes).

*Type*:

The type of threshold:

0 - High high

1 - High

2 - Low

3 - Low low

4 - Deadband

5 - Deviation

6 - Rate of change

*ClusterName*:

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are

resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

The alarm threshold or 0.0.

## Related Functions

[AlarmFirstCatRec](#), [AlarmNextCatRec](#), [AlarmFirstPriRec](#), [AlarmNextPriRec](#), [AlarmFirstTagRec](#), [AlarmNextTagRec](#),  
[AlarmGetThreshold](#), [AlarmSetThreshold](#), [AlarmSetThresholdRec](#)

## See Also

[Alarm Functions](#)

## AlarmHelp

Displays the alarm help page (associated with the alarm) where the cursor is positioned. You can assign a help page to each alarm when you define it (using the Digital Alarms or the Analog Alarms database, depending on the type of alarm). You need to also define the help page in the pages database.

## Syntax

INT **AlarmHelp()**

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[PageAlarm](#)

## Example

System keyboard	
Key Sequence	AlmHelp
Command	AlarmHelp()
Comment	Display the alarm help page

## See Also

[Alarm Functions](#)

### AlarmHighestPriority

Returns the alarm priority and/or state of the current highest priority alarm for the given equipment in conjunction with the selected filter criteria.

## Syntax

```
LONG AlarmHighestPriority(INT Mode [, STRING EquipmentFilter] [, STRING FilterCriteria] [, LONG KeepAliveSeconds] [, INT CachedMode] [, INT IncludingReference])
```

#### Mode

0 - return the priority of the highest priority alarm.

1 - return the alarm state of the highest priority alarm.

2 - return the results of mode 0 and 1 combined into a two-byte integer result. Use LowByte() to extract the priority of the highest priority alarm and use HighByte() to extract the alarm state of the highest priority alarm.

3 - return the current setting for the alarm server's **Highest Priority Order** field ("Priority, Alarm State" or "Alarm State, Priority").

4- Same as mode 0. Alarm Priorities where **Show on Indicator** is set to False are excepted.

5- Same as mode 1. Alarm Priorities where **Show on Indicator** is set to False are excepted.

6- Same as mode 2. Alarm Priorities where **Show on Indicator** is set to False are excepted.

For more information about the field **Show on Indicator** refer to the topic [Configure Display Properties for an Alarm Priority](#).

#### EquipmentFilter:

A comma-separated list of equipment names to filter alarms prior to applying the filter specified in the FilterCriteria argument. Each equipment name may be a part of full equipment name from the root. For example, "Region.Plant" will count alarms with equipment names literally starting with "Region.Plant" that includes "Region.PlantA" and "Region.Plant.AreaB". If it is needed to count children of a particular equipment tree only, the equipment name should end with a period('.'). A comma in between equipment expressions is regarded as an OR operator. If an empty string is specified, return Highest Priority for all Alarms.

#### FilterCriteria:

A filter name OR filter text. See the topic [Implementing Alarm Filters Using Cicode](#) for more information about filter syntax.

#### KeepAliveSeconds:

Optional length of time (in seconds) for which the count will remain in memory. Default is 30 seconds.

The *CachedMode* parameter affects the behavior of this parameter. Refer to the section below for more information.

#### CachedMode:

Optional flag that causes the current cached value to be supplied even when the value is being refreshed. This makes the function non-blocking. If the property has not yet been cached, an error is set.

0 - Do not force cached read. Cicode is blocking

1 - Force cached read. Cicode is non-blocking

Default value is 1 (true).

A priority value in the cache is accessed when its filter criteria matches the filter criteria in a subsequent call, and the priority value's *KeepAliveSeconds* duration is extended. A priority will stay in the cache for 'at least' the duration specified by *KeepAliveSeconds*, and may stay in the cache for an unspecified period of time before being discarded.

The period set for an existing priority will be overridden in the event a longer duration is set using *KeepAliveSeconds*.

A priority that is added to the cache is not immediately available for reading, so a foreground call to *AlarmHighestPriority* that causes a new count to be added will return a value of -1 and an error of 345 (Data not ready).

*IncludingReference:*

0 - Do not include alarms belonging to referenced equipment in *AlarmHighestPriority* for current equipment

1 - Include alarms belonging to referenced equipment in *AlarmHighestPriority* for current equipment.

Set to 1 by default.

*Type:*

The rule used to determine the relative priority of alarms (currently fixed to 0).

## Return Value

For Mode 0, return the priority ID of the current highest priority alarm.

For Mode 1, return the alarm state of the current highest priority alarm:

- 1 - On and Unacknowledged
- 2 - Off and Unacknowledged
- 3 - On and Acknowledged
- 4 - Disabled

For Mode 3, return the current setting for the alarm server's **Highest Priority Order** field:

- 0 - Priority, Alarm State
- 1 - Alarm State, Priority

## Example

```
INT nPriority;
INT nState;
INT nCombined;
INT nSetting;

// Priority of highest priority alarm in Plant.Area.Room* equipment and all referenced
// equipment (default non-blocking Cicode).
nPriority = AlarmHighestPriority(0, "Plant.Area.Room");

// Alarm state of highest priority alarm in Plant.Area.Room* equipment and all referenced
// equipment (default non-blocking Cicode).
```

```
nState = AlarmHighestPriority(1, "Plant.Area.Room");

// State and Priority of highest priority alarms in Plant.Area.Room*, Plant.Area.Flat*
equipment and all referenced equipment with category 10 as a two-byte integer (non-blocking
Cicode).
nCombined = AlarmHighestPriority(2, "Plant.Area.Room", Plant.Area.Flat", "Category=10", 1,
1);
nPriority = LowByte(nCombined);
nState = HighByte(nCombined);

// Current setting for the alarm server's Highest Priority Order field
nSetting = AlarmHighestPriority(3);
```

## Related Functions

[AlarmCount](#), [AlarmCountEquipment](#), [AlarmCountList](#)

## See Also

[Alarm Functions](#)

### AlarmListCreate

Creates an alarms list at a specified AN.

## Syntax

```
INT AlarmListCreate(INT nAN, INT alarm_type, INT width, INT height, INT rowheight, INT drawheader [, STRING
sCluster [, STRING sFormat [, INT hFontRow [, INT hFontHeader]]]])
```

*nAN*:

The AN number to associate with the alarms list.

*alarm\_type*:

The type of alarms to display in the alarms list:

Non-hardware alarms

0 - All active alarms, that is Types 1 and 2

1 - All unacknowledged alarms, ON and OFF

2 - All acknowledged ON alarms

3 - All disabled alarms

4 - All configured (non-hardware) alarms, that is Types 0 to 3, plus acknowledged OFF alarms.

Alarm Summary

10 - All summary alarms

15 - Sequence of events with configuration events filtered out

16 - Sequence of events

Alarm General

*11* - All ON alarms

*12* - All OFF alarms

*17* - All unacknowledged ON alarms

*18* - All unacknowledged OFF alarms

*width*:

The width of the area in which the alarms list will display (in pixels).

*height*:

The height of the area in which the alarms list will display (in pixels).

*rowheight*:

The height of each row in the alarms list (in pixels).

*drawheader*:

Determines if the alarms list includes a header row. You need to enter a value; there is no default.

*0* = no header row

*1* = include a header row

*sCluster*:

Specifies the cluster context for the alarms displayed in the list. This value is optional. If not specified, it defaults to all clusters.

*sFormat*:

The name of the display format applied to the alarms list. This determines the arrangement of columns that are presented. A customized name can be defined for a display format using the parameter [\[Format\]FormatName](#).

This value is optional. If not specified, the default format for the type of alarm specified in the *alarm\_type* parameter is used.

*hFontRow*:

The handle for the font you would like to use in the alarm list rows. This value is optional. If not specified, it defaults to *-1* (the default font).

Font handles are generated by the Cicode function [DspFont](#).

*hFontHeader*:

The handle for the font you would like to use in the alarm list header row. This value is optional. If not specified, it defaults to *-1* (the default font).

Font handles are generated by the Cicode function [DspFont](#).

## Return Value

0 (zero) if successful, otherwise an error code is returned. The error code can be obtained by calling the *IsError* Cicode function.

## Related Functions

[AlarmListDisplay](#), [AlarmListDestroy](#)

## See Also

[Alarm Functions](#)

### AlarmListDestroy

Destroys an alarms list at a specified AN.

## Syntax

**INT AlarmListDestroy(INT *nAN*)**

*nAN*:

The AN number associated with the alarms list you want to destroy.

## Return Value

0 (zero) if successful, otherwise an error code is returned. The error code can be obtained by calling the IsError Cicode function.

## Related Functions

[AlarmListDisplay](#), [AlarmListCreate](#)

## See Also

[Alarm Functions](#)

### AlarmListDisplay

Displays an alarms list at a specified AN.

## Syntax

**INT AlarmListDisplay(INT *nAN*)**

*nAN*:

The AN number associated with the alarms list you want to display.

## Return Value

0 (zero) if successful, otherwise an error code is returned. The error code can be obtained by calling the IsError Cicode function.

## Related Functions

[AlarmListCreate](#), [AlarmListDestroy](#)

## See Also

[Alarm Functions](#)

### AlarmListFill

Fills an alarms list at a specified AN to be used by another routine (for example, the interlocks processing routine).

## Syntax

`INT AlarmListFill(INT nAN)`

*nAN*:

The AN number associated with the alarms list you want to use.

## Return Value

0 (zero) if successful, otherwise an error code is returned. The error code can be obtained by calling the IsError Cicode function.

## Related Functions

[AlarmListCreate](#), [AlarmListDestroy](#), [AlarmListDisplay](#)

## See Also

[Alarm Functions](#)

### AlarmNextCatRec

Searches for the next occurrence of an alarm category and type, commencing with the specified alarm record identifier (returned from the previous search through the [AlarmFirstCatRec](#) function). You can search all areas, the current area only, or specify an area to limit the search.

This function returns an alarm record identifier that you can use in other alarm functions, for example, to acknowledge, disable, or enable the alarm, or to get field data on that alarm.

This is a blocking function. If the function is called from a foreground task that is unable to block, the return value will be -1 and a hardware alarm set. Use IsError() to retrieve the error code.

## Syntax

LONG **AlarmNextCatRec**(LONG *Record*, INT *Category*, INT *Type* [, INT *Area*] [, STRING *ClusterName*] )

*Record*:

The alarm record number, returned from any of the following alarm functions:

- **AlarmFirstCatRec()** or **AlarmNextCatRec()** - used to search for a record by alarm category, area, and type (acknowledged, disabled, etc.).
- **AlarmFirstPriRec()** or **AlarmNextPriRec()** - used to search for a record by alarm priority, area, and type (acknowledged, disabled, etc.).
- **AlarmFirstTagRec()** or **AlarmNextTagRec()** - used to search for a record by alarm tag, name, and description.

*Category*:

The alarm category or group number to match. Set Category to 0 (zero) to match all alarm categories.

*nType*:

The type of alarms to display:

**Non-hardware alarms**

- 0 - All active alarms, that is Types 1 and 2
- 1 - All unacknowledged alarms, ON and OFF
- 2 - All acknowledged ON alarms
- 3 - All disabled alarms
- 4 - All configured (non-hardware) alarms, that is Types 0 to 3, plus acknowledged OFF alarms.

**Alarm General**

- 11 - All ON alarms
- 12 - All OFF alarms
- 17 - All unacknowledged ON alarms
- 18 - All unacknowledged OFF alarms

If you omit the *Type*, the default is 1.

---

**Note:** If you call this function with the *Type* argument set to an unsupported value, it may return unexpected results.

---

*Area*:

The area in which to search for alarms. If you choose to omit the area, or if you set Area to -1, only the current area will be searched.

*ClusterName*:

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

The alarm record identifier or -1 if no match is found.

## Related Functions

[GrpOpen](#), [AlarmFirstCatRec](#), [AlarmFirstPriRec](#), [AlarmNextPriRec](#), [AlarmGetFieldRec](#), [AlarmAckRec](#),  
[AlarmDisableRec](#), [AlarmEnableRec](#), [AlarmGetThresholdRec](#), [AlarmSetThresholdRec](#)

## Example

See [AlarmAckRec](#).

## See Also

[Alarm Functions](#)

### AlarmNextPriRec

Searches for the next occurrence of an alarm of a specified priority and type, commencing with the specified alarm record identifier (returned from the previous search through the [AlarmFirstPriRec\(\)](#) function). You can search all areas, the current area only, or specify an area to limit the search.

This function returns an alarm record identifier that you can use in other alarm functions, for example, to acknowledge, disable, or enable the alarm, or to get field data on that alarm.

---

**Note:** Record numbers obtained from [AlarmGetDsp](#) are not valid for this function.

This is a blocking function. If the function is called from a foreground task that is unable to block, the return value will be -1 and a hardware alarm set. Use [IsError\(\)](#) to retrieve the error code.

## Syntax

**INT AlarmNextPriRec(LONG Record, INT Priority, INT Type [, INT Area] [, STRING ClusterName] )**

*Record:*

The alarm record number, returned from any of the following alarm functions:

- [AlarmFirstCatRec\(\)](#) or [AlarmNextCatRec\(\)](#) - used to search for a record by alarm category, area, and type (acknowledged, disabled, etc.).
- [AlarmFirstPriRec\(\)](#) or [AlarmNextPriRec\(\)](#) - used to search for a record by alarm priority, area, and type (acknowledged, disabled, etc.).
- [AlarmFirstTagRec\(\)](#) or [AlarmNextTagRec\(\)](#) - used to search for a record by alarm tag, name, and description.
- [AlarmGetDsp\(\)](#) - used to find the record that is displayed at a specified AN, for either an alarm list or alarm summary entry. Set the sField argument in [AlarmGetDsp\(\)](#) to "RecNo".

To store this value, use data type Int in Cicode or Long for variable tags (Long needs 4 bytes).

*Priority:*

The alarm Priority or group handle of a group of alarm priorities. Set Priority to 0 (zero) to match all alarm priorities.

*nType:*

The type of alarms to display:

**Non-hardware alarms**

- 0 - All active alarms, that is Types 1 and 2
- 1 - All unacknowledged alarms, ON and OFF
- 2 - All acknowledged ON alarms
- 3 - All disabled alarms
- 4 - All configured (non-hardware) alarms, that is Types 0 to 3, plus acknowledged OFF alarms.

#### Alarm General

- 11 - All ON alarms
- 12 - All OFF alarms
- 17 - All unacknowledged ON alarms
- 18 - All unacknowledged OFF alarms

If you omit the Type, the default is 1.

---

**Note:** If you call this function with the *Type* argument set to an unsupported value, it may return unexpected results.

---

#### Area:

The area in which to search for alarms. Set Area to -1 to search all areas. If you do not specify an area, only alarms in the current area on the Alarms Server are searched.

#### ClusterName:

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

The alarm record identifier or -1 if no match is found.

## Related Functions

[GrpOpen](#), [AlarmFirstCatRec](#), [AlarmFirstPriRec](#), [AlarmNextCatRec](#), [AlarmGetFieldRec](#), [AlarmAckRec](#),  
[AlarmDisableRec](#), [AlarmEnableRec](#), [AlarmGetThresholdRec](#), [AlarmSetThresholdRec](#), [AlarmSetInfo](#)

## See Also

[Alarm Functions](#)

## AlarmNextTagRec

Searches for the next occurrence of an alarm tag, name, and description, starting with the alarm record identifier (returned from the previous search through the [AlarmFirstTagRec](#) function).

This function returns an alarm record identifier that you can use in other alarm functions, for example, to acknowledge, disable, or enable the alarm, or to get field data on that alarm.

---

**Note:** Record numbers obtained from [AlarmGetDsp](#) are not valid for this function.

This is a blocking function. If the function is called from a foreground task that is unable to block, the return value will be -1 and a hardware alarm set. Use [IsError\(\)](#) to retrieve the error code.

For complex filtering operations it is more efficient to use the alarm tag browse functions [AlmBrowseOpen](#) and [AlmBrowseNext](#).

## Syntax

**LONG AlarmNextTagRec(LONG Record, STRING Tag, STRING Name, Description [,STRING ClusterName] )**

*Record:*

The alarm record number, returned from any of the following alarm functions:

- `AlarmFirstCatRec()` or `AlarmNextCatRec()` - used to search for a record by alarm category, area, and type (acknowledged, disabled, etc.).
- `AlarmFirstPriRec()` or `AlarmNextPriRec()` - used to search for a record by alarm priority, area, and type (acknowledged, disabled, etc.).
- `AlarmFirstTagRec()` or `AlarmNextTagRec()` - used to search for a record by alarm tag, name, and description.
- `AlarmGetDsp()` - used to find the record that is displayed at a specified AN, for either an alarm list or alarm summary entry. Set the `sField` argument in `AlarmGetDsp()` to "RecNo".

To store this value, use data type Int in Cicode or Long for variable tags (Long needs 4 bytes).

*Tag:*

A string that identifies the tag to be matched. It can be one of the following:

- An alarm tag — for example, "Fire1"
- An alarm equipment item — for example, "Motor1.AlarmFire"

Specify an empty string (" ") to match all alarm tags.

*Name:*

The alarm name to be matched. Specify an empty string (" ") to match all alarm names.

*Description:*

The alarm description to be matched. Specify an empty string (" ") to match all alarm descriptions.

*ClusterName:*

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks " ".

## Return Value

The alarm record identifier or -1 if no match is found.

## Related Functions

[AlarmFirstTagRec](#), [AlarmGetFieldRec](#), [AlarmAckRec](#), [AlarmDisableRec](#), [AlarmEnableRec](#), [AlarmGetDelayRec](#),  
[AlarmGetThresholdRec](#), [AlarmSetThresholdRec](#), [AlmBrowseOpen](#), [AlmBrowseNext](#)

## Example

See [AlarmDisableRec](#).

## See Also

[Alarm Functions](#)

### AlarmNotifyVarChange

This function is used to provide time-stamped digital and time-stamped analog alarms with data. When called, it notifies the alarm server that the specified variable tag has changed.

The alarm server will then check all time-stamped digital and time-stamped analog alarms that use the variable tag to see if their alarm states need to be updated as a result of the change. Any alarm state changes that result from this check will be given the timestamp passed into this function as their time of occurrence.

**Note:** Although you can hardcode a value into the setpoint when using analog alarms, you cannot use hardcoded values with time-stamped analog alarms. If the setpoint is hardcoded, this function cannot be used to notify the alarm when the variable changes.

## Syntax

INT **AlarmNotifyVarChange**(*Tag*, *Value*, *Timestamp* [, *TimestampMS*] [, *sClusterName*] [, *bSync*] )

*Tag*:

Name of the variable tag that has changed as a string. This name may include the name of the tag's cluster in the form *cluster.tagname*. This cluster name may be different from the cluster of the alarm server indicated by *sClusterName* below.

The Tag parameter is resolved on the alarm server, so the alarm server should be configured to connect to the tag's cluster.

*Value*:

Value of the variable tag at the time of the change as a floating-point number

*Timestamp*:

Time/date at which the variable tag changed in the standard Plant SCADA time/date variable format (Seconds since 1970).

*TimestampMS*:

Millisecond portion of the time at which the variable tag changed.

*sClusterName*:

Name of the cluster of the alarm server. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

*bSync*:

An optional boolean argument that specifies whether the command is synchronous (blocking) or asynchronous (non-blocking). If it is specified as synchronous (true) the function will wait until the notification has been recorded into the alarm database before further code execution. If it is specified as asynchronous (false) the function will only return an error if no alarm server is currently available.

## Return Value

For synchronous mode, the return value will be the error that was detected when the function was called. For

asynchronous mode, the return value will be 0, unless there was no server available.

## Example

```
AlarmNotifyVarChange("LOOP_1_SP", 50.0, TimeCurrent() - 10, 550,  
"ClusterXYZ");
```

This will tell the alarm server in cluster XYZ that the value of variable tag LOOP\_1\_SP changed to 50.0 at 9.450 seconds ago.

See also [Add a Time Stamped Digital Alarm](#) and [Add a Time Stamped Analog Alarm](#).

## See Also

[Alarm Functions](#)

### AlarmQueryFirstRec

Searches for the first occurrence of an alarm category (or priority) and type. This is a wrapper function of [AlarmFirstCatRec](#) and [AlarmFirstPriRec](#).

This is a blocking function. If the function is called from a foreground task that is unable to block, the return value will be -1 and a hardware alarm set. Use [IsError\(\)](#) to retrieve the error code.

## Syntax

```
LONG AlarmQueryFirstRec(Group, nType, Area, QueryType [, sClusterName] )
```

***Group:***

Alarm category if *QueryType* is 0 or alarm priority if *QueryType* is 1.

***nType:***

The type of alarms to display:

#### Non-hardware alarms

0 - All active alarms, that is Types 1 and 2

1 - All unacknowledged alarms, ON and OFF

2 - All acknowledged ON alarms

3 - All disabled alarms

4 - All configured (non-hardware) alarms, that is Types 0 to 3, plus acknowledged OFF alarms.

#### Hardware alarms

5 - All active alarms, that is Types 6 and 7

6 - All unacknowledged alarms, ON and OFF

7 - All acknowledged ON alarms

8 - All disabled alarms

9 - All configured alarms, that is Types 5 to 8

#### Alarm Summary

*10* - All summary alarms  
*15* – Sequence of events with configuration events filtered out  
*16* - Sequence of events

#### **Alarm General**

*11* - All ON alarms  
*12* - All OFF alarms  
*13* - All ON hardware alarms  
*14* - All OFF hardware alarms  
*17* - All unacknowledged ON alarms  
*18* - All unacknowledged OFF alarms

If you omit the Type, the default is 1.

#### *Area:*

Area in which to search for alarms. Set Area to -1 to search all areas.

#### *QueryType:*

Query type.

0 - Search by category.

1 - Search by priority.

#### *sClusterName:*

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks " ".

## **Return Value**

The alarm record identifier or -1 if no match is found.

## **Related Functions**

[AlarmQueryNextRec](#)

## **See Also**

[Alarm Functions](#)

## **AlarmQueryNextRec**

Searches for the next occurrence of an alarm category (or priority) and type, commencing with the specified alarm record identifier (returned from the previous search through the alarm query functions).

This is a blocking function. If the function is called from a foreground task that is unable to block, the return value will be -1 and a hardware alarm set. Use IsError() to retrieve the error code.

This is wrapper function of [AlarmNextCatRec](#) and [AlarmNextPriRec](#).

## Syntax

LONG **AlarmQueryNextRec**(*Record*, *Group*, *nType*, *Area*, *QueryType* [, *sClusterName*] )

*Record*:

Alarm record number.

*Group*:

Alarm Category if *QueryType* is 0 or alarm priority if *QueryType* is 1.

*nType*:

The type of alarms to display:

### Non-hardware alarms

0 - All active alarms, that is Types 1 and 2

1 - All unacknowledged alarms, ON and OFF

2 - All acknowledged ON alarms

3 - All disabled alarms

4 - All configured (non-hardware) alarms, that is Types 0 to 3, plus acknowledged OFF alarms.

### Hardware alarms

5 - All active alarms, that is Types 6 and 7

6 - All unacknowledged alarms, ON and OFF

7 - All acknowledged ON alarms

8 - All disabled alarms

9 - All configured alarms, that is Types 5 to 8

### Alarm Summary

10 - All summary alarms

15 – Sequence of events with configuration events filtered out

16 - Sequence of events

### Alarm General

11 - All ON alarms

12 - All OFF alarms

13 - All ON hardware alarms

14 - All OFF hardware alarms

17 - All unacknowledged ON alarms

18 - All unacknowledged OFF alarms

If you omit the Type, the default is 1.

*Area*:

Area in which to search for alarms. Set Area to -1 to search all areas.

*QueryType*:

Query type.

0 - Search by category.

1 - Search by priority.

*sClusterName*:

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

The alarm record identifier or -1 if no match is found.

## Related Functions

[AlarmQueryFirstRec](#)

## See Also

[Alarm Functions](#)

## AlarmSetDelay

Changes the delay setting for an alarm (that is Delay, High High Delay, Deviation Delay, etc.). This function acts on the alarm that the cursor is positioned over. Use this function during runtime to change the delay values that were specified in the alarms database. Delay changes made using this process are persistent (that is they are saved to the project).

## Syntax

INT **AlarmSetDelay**(*nType*, *Value*)

*nType*:

The type of delay:

- 0 - Delay (digital alarm/advanced alarm)
- 1 - High high delay (analog alarm)
- 2 - High delay (analog alarm)
- 3 - Low delay (analog alarm)
- 4 - Low low delay (analog alarm)
- 5 - Deviation delay (analog alarm)

*Value*:

The new value for the delay. Enter a blank value " " to remove the delay setting.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[AlarmGetDelay](#), [AlarmSetDelayRec](#), [AlarmGetDelayRec](#)

## See Also

[Alarm Functions](#)

### AlarmSetDelayRec

Changes the delay setting for an alarm (that is Delay, High High Delay, Deviation Delay, etc.) by the alarm record number. You can only call this function on an alarms server for local alarms, or on a redundant server if one has been configured.

This is a blocking function. If the function is called from a foreground task that is unable to block, an error will be returned.

## Syntax

**INT AlarmSetDelayRec(LONG Record, INT Type, INT Value, STRING ClusterName)**

*Record:*

The alarm record number, returned from any of the following alarm functions:

- `AlarmFirstCatRec()` or `AlarmNextCatRec()` - used to search for a record by alarm category, area, and type (acknowledged, disabled, etc.).
- `AlarmFirstPriRec()` or `AlarmNextPriRec()` - used to search for a record by alarm priority, area, and type (acknowledged, disabled, etc.).
- `AlarmFirstTagRec()` or `AlarmNextTagRec()` - used to search for a record by alarm tag, name, and description.
- `AlarmGetDsp()` - used to find the record that is displayed at a specified AN, for either an alarm list or alarm summary entry. Set the `sField` argument in `AlarmGetDsp()` to "RecNo".

*nType:*

The type of delay:

0 - Delay (digital alarm/advanced alarm/double point status alarm)

1 - High high delay (analog alarm)

2 - High delay (analog alarm)

3 - Low delay (analog alarm)

4 - Low low delay (analog alarm)

5 - Deviation delay (analog alarm)

*Value:*

The new value for the delay. Enter a blank value " " to remove the delay setting.

*sClusterName:*

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Related Functions

[AlarmGetDelay](#), [AlarmNotifyVarChange](#), [AlarmGetDelayRec](#)

## See Also

[Alarm Functions](#)

### AlarmSetInfo

Controls different aspects of the alarm list displayed at a specified AN. Currently applies only to non-hardware alarm lists.

## Syntax

**INT AlarmSetInfo(INT AN, INT Type, STRING Value)**

**AN:**

The AN where the alarm list originally commenced. (AN alarm page can contain more than one alarm list). You can also specify:

-1 - Change the display parameters of all alarm lists displayed on the page.

0 - Change the display parameters of the alarm list where the cursor is positioned.

**Type:**

The type of data. The aspects and related types are listed below:

Display aspect	Types
Change display line and page offset	0, 1
Formatting of alarms in the alarm list	4, 5, 6
Filtering of alarms	2, 3, 7, 8 <b>Note:</b> If a requested alarm filter is too complex (for example, it contains too many conditions or too many nested brackets), the filter is cleared (no filter is used). The hardware alarm "Too many alarms in filters" is generated on the client components, and a tracelog error message is logged.
Sorting of alarms - to control the sorting aspect of the alarm list, type 9 and 10 should be used together.	9, 10
Linking or unlinking to a named filter.	12

0 - Alarm page number. The vertical offset (in pages) from the AN where the alarm list commenced.

1 - Alarm list offset. The vertical offset (in lines) from the AN where the alarm list commenced.

2 - Category of alarms displayed on the alarm list. To specify all categories, use a value of 0.

You can use a group handle to display a group of categories. (A group can be defined via the Security activity, or by using the GrpOpen() function.) Before you can display a group of categories, you need to first open the group using the GrpOpen() function. You would usually do this by entering the GrpOpen() function as the Page entry command for your alarm page (set using Page Properties). Be aware, however, that you should not close the group until you close the display page. If you do, the group will be lost and the group handle will become invalid. The page would then be unable to continue displaying the desired group. The handle may be reused for another group, which means the page may display a different category, or display all alarms.

You would normally close the group by entering the GrpClose() function as the Page exit command.

3 - Type of alarms displayed on the alarm list. See AlarmDsp() for a list of these types.

4 - Display all alarms according to the format and fonts specified for one category (specified in Value).

5 - The display format for all alarms specified by a format handle. All of the alarm categories will display in the same format.

6 - The display font for all user alarms specified by a font handle. All of the user alarms will appear in the same font and color.

7 - The priority of the alarms to be displayed in the alarm list. You can use a group number to display a group of priorities.

You can use a group handle to display a group of priorities. (A group can be defined using Groups - from the Project Editor System menu - or by using the GrpOpen() function.) Before you can display a group of priorities, you need to first open the group using the GrpOpen() function. You would usually do this by entering the GrpOpen() function as the Page entry command for your alarm page (set using Page Properties). Be aware, however, that you should not close the group until you close the display page. If you do, the group will be lost and the group handle will become invalid. The page would then be unable to continue displaying the desired group. You would normally close the group by entering the GrpClose() function as the Page exit command.

8 - Use the Value argument of the AlarmSetInfo() function to specify whether the display mode of the alarm list is based on Alarm Category or Priority:

- Set the Value argument to 0 (zero) to display by Category.
- Set the Value argument to 1 to display by Priority.

9 - Use the Value argument of the AlarmSetInfo() function to specify the sorting mode of the alarm list:

- Set the Value argument to 0 (zero) to display alarms sorted by ON time within their groups.
- Set the Value argument to 1 to display alarms sorted by the order-by keys. Please be aware that this option will only be meaningful if you have already called the AlarmSetInfo() function with a Type of 10 to set the order-by keys.

10 - Use the Alarm Order-by key specified in the Value argument of the AlarmSetInfo() function to determine the order in which the alarm list will be displayed.

The AlarmSetInfo() function should then be called again using a Type of 9 and a Value of 1 for Plant SCADA to sort the alarms in the order specified.

---

**Note:** It is recommended that you sort an alarm summary or SOE list using the following order-by keys (as specified in the *Value* argument):

- Alarm summary – OnDate, OnTime, OnMilliseconds
- SOE – Date, Time, Milliseconds.

If you use any other fields as an order-by key, you will have to apply a time range to your query using the Alarm Filter Functions. Otherwise, your query will not return any results and the hardware error "Invalid Argument Passed" will be generated.

---

11 – Invalid to set this field.

12 – Associate or disassociate a named filter. By setting this field to text, you associate the specified AN to a named filter which is then applied to an alarm display list. Setting this type to empty text, will unlink from any named filter, but the disassociated alarm list will retain its value, hence the filter will still be in place until a new filter is applied. If setting the *Value* to text that does not correspond to a named filter, the value read back (using *AlarmGetFilterName*) will be empty.

15 – Auto-refresh mode, where mode is:

-1 : always auto refresh

0 : auto refresh disabled

1 : auto refresh page 1, disable on all other pages.

---

**Note:** Type 15 is only in use if displaying the alarm summary (*AlarmDsp* Type 10).

---

16 – Invalid to set this field.

*Value*:

The meaning of the *Value* argument depends on the data type specified in the *Type* argument.

If you set *Type* = 0 (offset in pages) or *Type* = 1 (offset is lines), the *Value* argument must be within the range of alarm entries currently stored in the client cache, otherwise error code 569 will be returned.

If you set *Type* = 8, the *Value* argument determines whether alarms are displayed by category or priority:

0 - Alarm list displayed by Category.

1 - Alarm list displayed by Priority.

If you set *Type* = 10, the *Value* argument specifies the order-by keys to be used in sorting. Up to sixteen keys may be specified:

{KeyName [,SortDirection]}[ {KeyName [,SortDirection]}]

The Keyname argument specifies the name of the pre-defined order-by key to be used. The valid options are a subset of the alarm display fields: Tag, Name, Category, Priority, Area, Priv, Time, State, and Type.

The SortDirection argument is optional, and indicates whether the sort will be ascending or descending. Valid options are: 0 Descending (default), 1 Ascending.

For example:

{Time,0} : sorts by <Time> (descending)

{Tag,1} : sorts by <Tag> (ascending)

{Tag,1}{Time} : sorts by <Tag> (ascending), then <Time> (descending)

---

**Note:** The fields Disabled, Acknowledged and Active are hidden from the alarm list display and can only be used for filtering and sorting. If you would like to sort by Unacknowledged alarms first, followed by Acknowledged Active alarm and Date time, then you can use the following syntax:

{Acknowledged,1}{Active,0}{Time,0}

i.e. Acknowledged, ascending; Active, descending; time, descending

---

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[GrpOpen](#), [AlarmDsp](#), [AlarmGetInfo](#)

## Examples

In the following example, the alarm list is set to display in the order of the order-by key. Please be aware that this is a two-step process requiring two calls to the AlarmSetInfo() function, and that it applies only to non-hardware alarm lists.

```
! Set the order-by key.  
AlarmSetInfo(21,10,"{Time}");  
! Set the sorting mode.  
AlarmSetInfo(21,9,1);
```

Type 8 of the function is used to set the display mode to either category or priority. This is helpful when filtering based on either of these fields. In order to filter on category 2 we should use:

```
AlarmSetInfo(21, 8, 0);  
AlarmSetInfo(21, 2, 2);
```

Once we do this the alarms with category 2 will be displayed in the alarm list and remaining although active will not be displayed.

Similarly if we want to filter on priority we set the mode to priority and then use type 7. For example, to filter on priority 4 we should use:

```
AlarmSetInfo(21, 8, 1); ! priority mode  
AlarmSetInfo(21, 7, 4); ! apply filter
```

In the following examples, the display parameters of the alarm list where the cursor is positioned are changed.

```
! Change the vertical offset (pages) to 2.  
AlarmSetInfo(0,0,2);  
! Change the vertical offset (lines) to 15.  
AlarmSetInfo(0,1,15);
```

Change the alarm category to 10.

```
AlarmSetInfo(0,2,10);
```

Change the type of alarms displayed to type 5 (hardware alarms).

```
AlarmSetInfo(0,3,5);
```

In the following examples, the display parameters of the alarm list at AN 20 are changed.

```
! Display alarms with category 120 format and fonts  
AlarmSetInfo(20, 4, 120);  
! Display alarms with a new format  
hFmt=FmtOpen("MyFormat","{Name}{Desc,20}",0);  
AlarmSetInfo(20, 5, hFmt);  
! Display alarms with a new font  
hFont = DspFont("Times",-60,black,gray);  
AlarmSetInfo(20, 6, hFont);
```

The following example displays alarms with categories 1-10, 20, or 25. Before AlarmSetInfo() is run, the page entry command for the alarm display page is configured as follows:

- On page entry command: hGrp=GrpOpen("MyGrp",1); StrToGrp(hGrp,"1..10,20,25");

The page exit command for the alarm display page is configured as follows:

- On page exit command: GrpClose(hGrp); AlarmSetInfo(20, 2, hGrp);

---

**Note:** **hGrp** is defined in the variables database.

## See Also

[Alarm Functions](#)

### AlarmSetThreshold

Changes the thresholds (that is High High, Low etc.) of analog alarms. This function acts on the analog alarm where the cursor is positioned. Use this function to change (at run time) the threshold values that were specified in the Analog Alarms database. Threshold changes made using this function are permanent (that is they are saved to the project). The display format currently specified for the record (in the Analog Alarms settings) will be applied to these values.

## Syntax

**INT AlarmSetThreshold(*nType*, *Value*)**

*nType*:

The type of threshold:

0 - High high

1 - High

2 - Low

3 - Low low

4 - Deadband

5 - Deviation

6 - Rate of change

*Value*:

The new value of the threshold. Enter a blank value " " to remove the threshold.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[AlarmSetThresholdRec](#)

## Example

System keyboard	
Key Sequence	SetHighHigh ### Enter
Command	AlarmSetThreshold(0, Arg1)
Comment	Change the threshold of a high high alarm

System keyboard	
Key Sequence	SetHigh ### Enter
Command	AlarmSetThreshold(1, Arg1)
Comment	Change the threshold of a high alarm

System keyboard	
Key Sequence	SetLow ### Enter
Command	AlarmSetThreshold(2, Arg1)
Comment	Change the threshold of a low alarm

System keyboard	
Key Sequence	SetlowLow ### Enter
Command	AlarmSetThreshold(3, Arg1)
Comment	Change the threshold of a low low alarm

## See Also

[Alarm Functions](#)

### AlarmSetThresholdRec

Changes the threshold (that is High High, Low etc.) of analog alarms by the alarm record number. You can call this function only on an Alarms Server for alarms on that server, or on the redundant server (if a redundant server is configured).

Threshold changes made using this function are permanent (that is they are saved to the project). The display format currently specified for the record (in the Analog Alarms form) will be applied to these values.

**Note:** Record numbers obtained from AlarmGetDsp are not valid for this function.

This is a blocking function. If the function is called from a foreground task that is unable to block, an error will be returned.

To permanently update alarm threshold limits using AlarmSetThresholdRec, set the parameter [\[Alarm\]UseConfigLimits](#) to 1.

## Syntax

**INT AlarmSetThresholdRec (LONG Record, INT Type, STRING Value [, STRING ClusterName])**

*Record:*

The alarm record number, returned from any of the following alarm functions:

- `AlarmFirstCatRec()` or `AlarmNextCatRec()` - used to search for a record by alarm category, area, and type (acknowledged, disabled, etc.).

- `AlarmFirstPriRec()` or `AlarmNextPriRec()` - used to search for a record by alarm priority, area, and type (acknowledged, disabled, etc.).

- `AlarmFirstTagRec()` or `AlarmNextTagRec()` - used to search for a record by alarm tag, name, and description.

- `AlarmGetDsp()` - used to find the record that is displayed at a specified AN, for either an alarm list or alarm summary entry. Set the `sField` argument in `AlarmGetDsp()` to "RecNo".

To store this value, use data type Int in Cicode or Long for variable tags (Long needs 4 bytes).

*Type:*

The type of threshold:

0 - High high

1 - High

2 - Low

3 - Low low

4 - Deadband

5 - Deviation

6 - Rate of change

*Value:*

The new value of the threshold. Enter a blank value "" to remove the threshold.

*ClusterName:*

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[AlarmSetThreshold](#)

## See Also

[Alarm Functions](#)

## AlarmSplit

Splits an alarm summary entry which has no Off time. The current entry will be given an off time equal to the time when AlarmSplit is called, and the new entry will have the same On time, with empty Off time.

If the alarm summary record has a valid Off time, the function does nothing.

You would normally call this function from a keyboard or mouse command.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT AlarmSplit([INT AN])**

**AN:**

Animation Number where alarm is displayed. You can:

- Set AN to where alarm is displayed which equals to AN of alarm list + Offset into the list
- Do not specify AN or set it to 0 or negative value to use current cursor value.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[AlarmSumSplit](#)

## Example

System keyboard	
Key Sequence	Split
Command	AlarmSplit()
Comment	Duplicates an alarm summary entry

## See Also

[Alarm Functions](#)

## AlarmSumAppend

Appends a new blank record to the alarm summary. Use this function to add new alarm summary entries, either for actual alarms or as special user summary entries.

If you specify a valid alarm tag in the *sTag* field, the summary entry is linked to the actual alarm. If you specify an asterisk '\*' as the first letter of the tag, the summary entry becomes a user event.

User events are not attached to alarm records, so their status will not change. Manually change the status of the user event, by calling the AlarmSumSet() function with the index returned by AlarmSumAppend(). As user events are not attached to alarms, they don't have the alarm fields - so the AlarmSumGet() function will not return any field data.

The latest entry in the Alarm summary will reflect the events of the alarm whether the alarm summary entry was appended created by an actual alarm event.

You can use user events to keep a record of logins, or control operations that you need to display in the alarm summary etc. The fields of UserEvents needs to be set immediately after creation using the AlarmSumSet() function.

To give an appended alarm summary entry the appearance of having been Acknowledged set the Acknowledge time of the summary entry. When the summary entry is linked to an actual alarm the function AlmSummaryAck() can be used to acknowledge the actual alarm linked to the summary entry. However, the AlmSummaryAck function will not directly affect the alarm summary entry.

AlarmSumAppend() can only be used if the Alarm Server is on the current machine. When the Alarm Server is not in the calling process, this function will become blocking and cannot be called from a foreground task. In this case, the return value will be undefined and a Cicode hardware alarm will be raised.

This function can only be used if the Alarm Server is on the current machine.

---

**Note:** The index passed to AlarmSum\* functions must be current,i.e. either:

- the index returned by most recent call to AlarmSumFirst/Last/Find/Next/Prev,  
OR
  - the index returned by the most recent call to AlarmSumAppend OR AlarmSumSplit
- 

The AlarmSumNext and AlarmSumPrev Cicode functions are unable to use the index returned by AlarmSumAppend.

---

**Note:** This function will only work while the record is writable which is controlled by the ArchiveAfter Parameter. Refer to the topic *Configure the Archiving Parameters* in the main help for more information.

---

**Note:** In a redundant pair scenario, this function will return an error condition and not do anything if the current server is not main. You can call this function on the primary server and the standby server so that function can succeed on one of the two servers in the specified cluster. The change will be synchronised to the other server after a short period of time (typically in 5 seconds).

---

## Syntax

**INT AlarmSumAppend(*sTag* [, *ClusterName*, *OnTime*, *OnMilli*, *bRedundant*] )**

*sTag*:

The alarm tag to append. Use an asterisk '\*' as the first letter to append a user event to the alarm summary. Please be aware that the AlarmSumAppend function returns the alarm summary index - not the error code.

*ClusterName*:

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

*OnTime*:

The alarm OnTime (if omitted or is equal to 0, will default to current time, i.e. the time the function was invoked).

*OnMilli*:

Milliseconds part of the alarm's ON time

**bRedundant:(optional)**

New alarm record is created on both redundant server instances.

**Note:** This parameter has been deprecated and is always set to true.

## Return Value

The index of the alarm summary entry, or -1 if the record could not be appended.

**Note:** The index is a 32-bit integer. Storing it in a variable that is smaller than 32 bit integer (such as a 16-bit integer) may cause overflow which may result in an incorrect value.

## Related Functions

[AlarmSumCommit](#), [AlarmSumDelete](#), [AlarmSumSet](#), [AlarmSumType](#), [AlarmDelete](#), [MsgRPC](#), [AlarmSumGet](#),  
[AlarmSumFirst](#), [AlarmSumNext](#), [AlarmSumLast](#), [AlarmSumPrev](#), [AlarmSumFind](#), [AlarmSplit](#),

## Example

```
! Append alarm to summary display
AlarmSumAppend("CV101");
! Append user event
iIndex = AlarmSumAppend("*MyEvent");
AlarmSumSet(iIndex, "Comment", "My event comment");
```

## See Also

[Alarm Functions](#)

### AlarmSumCommit

Commits the alarm summary entry to the Summary Device specified in the alarm category of the entry. The same alarm summary entry is unable to be committed twice. The Cicode function returns an error code if the entry has been committed by timeout (referring to [Alarm]SummaryTimeout parameter), or if AlarmSumCommit() is called twice. An entry is not committed by timeout if AlarmSumCommit() is called on this entry.

If the Alarm Server is not in the calling process, the calling process needs to run from the same machine as the alarm server.

This function can only be used if the Alarm Server is on the current machine. When the Alarm Server is not in the calling process, this function will become blocking and cannot be called from a foreground task. In this case, the return value will be undefined and a Cicode hardware alarm will be raised.

In a redundant pair scenario, this function will return an error condition and not do anything if the current server is not main. You can call this function on the primary server and the standby server so that function can succeed on one of the two servers in the specified cluster. The change will be synchronized to the other server after a short period of time (typically in 5 seconds).

**Note:** This function will only work while the record is writable which is controlled by the ArchiveAfter Parameter. Refer to the topic [Configure the Archiving Parameters](#) for more information.

## Syntax

```
INT AlarmSumCommit(Index [, ClusterName] )
```

*Index*:

The alarm summary index (returned from the `AlarmSumFirst()`, `AlarmSumNext()`, `AlarmSumLast()`, `AlarmSumPrev()`, `AlarmSumAppend()`, or `AlarmSumFind()` function).

*ClusterName*:

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[AlarmSumDelete](#), [AlarmSumSet](#), [AlarmSumType](#), [AlarmSumAppend](#), [AlarmDelete](#), [MsgRPC](#), [AlarmSumGet](#), [AlarmSumFirst](#), [AlarmSumNext](#), [AlarmSumLast](#), [AlarmSumPrev](#), [AlarmSumFind](#), [AlarmSplit](#),

## Example

```
/* This function commits alarm summary entries that match the
specified tag. */
FUNCTION
SumCommitTag(STRING sTag)
    INT Next;
    INT Index;
    STRING Name;
    Index=AlarmSumFirst();
    WHILE Index<>-1 DO
        Name=AlarmSumGet(Index, "Tag");
        Next=AlarmSumNext(Index);
        IF Name=sTag THEN
            AlarmSumCommit(Index);
        END
        Index=Next;
    END
END
```

## See Also

[Alarm Functions](#)

### AlarmSumDelete

Deletes an alarm summary entry. You identify the alarm summary entry by the *Index*, returned by one of the alarm summary search functions.

By embedding this function in a loop, you can delete a series of alarm summary entries. To start deleting from the oldest entry, call the AlarmSumFirst() function to get the index, and then call AlarmSumNext() in a loop. To delete back from the most recent entry, call AlarmSumLast() and then AlarmSumPrev() in a loop.

You can also get the *Index* from the AlarmSumFind() function, which finds an alarm summary entry by its alarm record identifier and time of activation.

This function can only be used if the Alarm Server is on the current machine. When the Alarm Server is not in the calling process, this function will become blocking and cannot be called from a foreground task. In this case, the return value will be undefined and a Cicode hardware alarm will be raised.

**Note:**

- In a redundant pair scenario, this function will return an error condition and not do anything if the current server is not main. You can call this function on the primary server and the standby server so that function can succeed on one of the two servers in the specified cluster. The change will be synchronised to the other server after a short period of time (typically in 5 seconds).
- This function will only work while the record is writable which is controlled by the ArchiveAfter parameter. Refer to the topic [Configure the Archiving Parameters](#) for more information.

## Syntax

```
INT AlarmSumDelete(Index [, ClusterName] )
```

**Index:**

The alarm summary index (returned from the AlarmSumFirst(), AlarmSumNext(), AlarmSumLast(), AlarmSumPrev(), AlarmSumAppend(), or AlarmSumFind() function).

**ClusterName:**

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

0 (zero) if the specified alarm entry exists, otherwise an error is returned.

## Related Functions

[AlarmSumCommit](#), [AlarmSumSet](#), [AlarmSumType](#), [AlarmSumAppend](#), [AlarmDelete](#), [MsgRPC](#), [AlarmSumGet](#), [AlarmSumFirst](#), [AlarmSumNext](#), [AlarmSumLast](#), [AlarmSumPrev](#), [AlarmSumFind](#), [AlarmSplit](#), [AlarmSumSplit](#), [AlmSummaryDelete](#)

## Example

```
/* This function deletes all alarm summary entries that match the
specified tag. */
FUNCTION
SumDelTag(STRING sTag)
    INT Next;
    INT Index;
    STRING Name;
    Index=AlarmSumFirst();
    WHILE Index<>-1 DO
```

```
Name=AlarmSumGet(Index, "Tag");
Next=AlarmSumNext(Index);
IF Name=sTag THEN
    AlarmSumDelete(Index);
END
Index=Next;
END
END
```

## See Also

[Alarm Functions](#)

### AlarmSumFind

Finds the alarm summary index for an alarm that you specify by the alarm record identifier and alarm activation time (OnTime). You can use this index in the AlarmSumGet() function to get field data from an alarm record, in the AlarmSumSet() function to change the existing data in that record, or in the AlarmSumDelete() function to delete the record. If calling this function from a remote client, use the MsgRPC() function.

To work with a series of alarm summary records, call this function to get the index, and then call either AlarmSumNext() to move forwards in the summary, or AlarmSumPrev() to move backwards in the summary.

---

**Note:** Record numbers obtained from AlarmGetDsp are not valid for this function. Instead use AlarmFirstTagRec() to get the record. This should be called from the server side using MsgRPC.

This function can only be used if the Alarm Server is on the current machine. When the Alarm Server is not in the calling process, this function will become blocking and cannot be called from a foreground task. In this case, the return value will be undefined and a Cicode hardware alarm will be raised.

---

**Note:** The index passed to AlarmSum\* functions must be current. ie either:

- the index returned by most recent call to AlarmSumFirst/Last/Find/Next/Prev
  - OR
  - the index returned by the most recent call to AlarmSumAppend OR AlarmSumSplit
- 

## Syntax

INT **AlarmSumFind**(*Record*, *OnTime* [, *ClusterName*] )

*Record*:

The alarm record number, returned from any of the following alarm functions:

- AlarmFirstCatRec() or AlarmNextCatRec() - used to search for a record by alarm category, area, and type (acknowledged, disabled, etc.).
- AlarmFirstPriRec() or AlarmNextPriRec() - used to search for a record by alarm priority, area, and type (acknowledged, disabled, etc.).
- AlarmFirstTagRec() or AlarmNextTagRec() - used to search for a record by alarm tag, name, and description.

To store this value, use data type Int in Cicode or Long for variable tags (Long needs 4 bytes).

*OnTime*:

The ON time of the alarm associated with the Record, that is, the time that the alarm was activated.

AlarmSumFind() requires that the OnTime argument contains the number of seconds from Midnight, so the

formulation:

```
iOnTime = StrToTime(AlarmSumGet(iIndex, "OnTime"));
```

will NOT yield the correct result. The correct formulation for this calculation is:

```
OnTime = StrToTime(AlarmSumGet(iIndex, "OnTime")) + TimeMidnight(TimeCurrent());
```

*ClusterName*:

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

The index of the alarm summary entry, or -1 if no alarm summary entry is found.

---

**Note:** The index is a 32-bit integer. Storing it in a variable that is smaller than 32 bit integer (such as a 16-bit integer) may cause overflow which may result in an incorrect value.

---

## Related Functions

[AlarmSumCommit](#), [AlarmSumDelete](#), [AlarmSumSet](#), [AlarmSumType](#), [AlarmSumAppend](#), [AlarmDelete](#), [MsgRPC](#), [AlarmSumGet](#), [AlarmSumFirst](#), [AlarmSumNext](#), [AlarmSumLast](#), [AlarmSumPrev](#), [AlarmSplit](#), [AlarmSumSplit](#)

## Example

```
/* This function sets the summary comment from the alarm record
number and the ontime of the summary event. */
FUNCTION
SumSetComment(INT AN, STRING sComment)
    INT nRecord;
    INT iOnTime;
    INT hAlarm1;
    STRING AlmTag;
    AlmTag = AlarmGetDsp(AN, "Tag");
    iOnTime = StrToDate(AlarmGetDsp(AN,"OnDate"))+StrToTime(AlarmGetDsp(AN,"OnTime"));
    hAlarm1 = MsgOpen("Alarm", 0, 0);
    MsgRPC(hAlarm1, "AlmSvrSumSetComment", "^^" + AlmTag + "^," + IntToStr(iOnTime) + ",^"
    + sComment + "^^", 1);
    MsgClose("Alarm", hAlarm1);
END
FUNCTION
AlmSvrSumSetComment(STRING AlmTag, INT iOnTime, STRING sComment)
    INT nRecord = AlarmFirstTagRec(AlmTag, "", "");
    INT Index = AlarmSumFind(nRecord, iOnTime);
    IF Index <> -1 THEN
        AlarmSumSet(Index, "Comment", sComment);
    END
END
```

## See Also

[Alarm Functions](#)

## AlarmSumFirst

Gets the index of the oldest alarm summary entry. You can use this index in the AlarmSumGet() function to get field data from an alarm record, in the AlarmSumSet() function to change the existing data in that record, or in the AlarmSumDelete() function to delete the record.

To work with a series of alarm summary records, call this function to get the index, and then call AlarmSumNext() within a loop, to move forwards in the alarm summary.

This function can only be used if the Alarm Server is on the current machine. When the Alarm Server is not in the calling process, this function will become blocking and cannot be called from a foreground task. In this case, the return value will be undefined and a Cicode hardware alarm will be raised.

---

**Note:** The index passed to AlarmSum\* functions must be current, i.e. either:

- the index returned by most recent call to AlarmSumFirst/Last/Find/Next/Prev  
OR
  - the index returned by the most recent call to AlarmSumAppend OR AlarmSumSplit.
- 

## Syntax

```
INT AlarmSumFirst( [ClusterName] )
```

*ClusterName*:

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

The index of the oldest alarm summary entry, or -1 if no alarm summary entry is found.

---

**Note:** The index is a 32-bit integer. Storing it in a variable that is smaller than 32 bit integer (such as a 16-bit integer) may cause overflow which may result in an incorrect value.

---

## Related Functions

[AlarmSumCommit](#), [AlarmSumDelete](#), [AlarmSumSet](#), [AlarmSumType](#), [AlarmSumAppend](#), [AlarmDelete](#), [MsgRPC](#), [AlarmSumGet](#), [AlarmSumNext](#), [AlarmSumLast](#), [AlarmSumPrev](#), [AlarmSumFind](#), [AlarmSplit](#), [AlarmSumSplit](#)

## Example

```
/* This function finds all alarm summary entries that match the
specified tag and sets the "OffTime" to the time specified. The
alarm entry is not acknowledged or set to the off state, the alarm
summary "OffTime" field is all that is affected. */
FUNCTION
SumSetTime(STRING sTag, INT Time)
    INT Index;
    STRING Name;
    Index=AlarmSumFirst();
    WHILE Index<>-1 DO
        Name=AlarmSumGet(Index,"Tag");
        IF Name=sTag THEN
```

```
    AlarmSumSet(Index, "OffTime", Time);
END
Index=AlarmSumNext(Index);
END
END
```

## See Also

[Alarm Functions](#)

### AlarmSumGet

Gets field data from an alarm summary entry. The data is returned as a string. You identify the alarm summary entry by the Index, returned by one of the alarm summary search functions. If calling this function from a remote client, use the MsgRPC function.

By embedding this function in a loop, you can get data from a series of alarm summary entries. To start from the oldest entry, call the AlarmSumFirst() function to get the index, and then call AlarmSumNext() in a loop. To work back from the most recent entry, call AlarmSumLast() and then AlarmSumPrev() in a loop.

You can also get the Index from the AlarmSumFind() function, which finds an alarm summary entry by its alarm record identifier and time of activation.

---

**Note:** Record numbers obtained from AlarmGetDsp are not valid for this function.

This function can only be used if the Alarm Server is on the current machine. When the Alarm Server is not in the calling process, this function will become blocking and cannot be called from a foreground task. In this case, the return value will be undefined and a Cicode hardware alarm will be raised.

## Syntax

INT **AlarmSumGet**(*Index*, *sField* [, *ClusterName*] )

*Index*:

The alarm summary index (returned from the AlarmSumFirst(), AlarmSumNext(), AlarmSumLast(), AlarmSumPrev(), AlarmSumAppend(), or AlarmSumFind() function).

*sField*:

The name of the field from which to extract the data:

Tag	Alarm tag
AckDate	Alarm acknowledged date
AckTime	Alarm acknowledged time
Category	Alarm category
Comment	Alarm comment
DeltaTime	Alarm active time
Desc	Alarm description

Help	Help page
Name	Alarm name
OffDate	Alarm OFF date
OffTime	Alarm OFF time
OnDate	Alarm ON date
OnTime	Alarm ON time
State	Alarm state

**ClusterName:**

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

**Return Value**

Field data from the alarm summary entry (as a string).

**Related Functions**

[AlarmSumCommit](#), [AlarmSumDelete](#), [AlarmSumSet](#), [AlarmSumType](#), [AlarmSumAppend](#), [AlarmDelete](#), [MsgRPC](#), [AlarmSumFirst](#), [AlarmSumNext](#), [AlarmSumLast](#), [AlarmSumPrev](#), [AlarmSumFind](#), [AlarmSplit](#), [AlarmSumSplit](#)

**Example**

See [AlarmSumFirst](#).

**See Also**

[Alarm Functions](#)

**AlarmSumLast**

Gets the index of the most recent alarm summary entry. You can use this index in the `AlarmSumGet()` function to get field data from an alarm record, in the `AlarmSumSet()` function to change the existing data in that record, or in the `AlarmSumDelete()` function to delete the record.

To work with a series of alarm summary records, call this function to get the index, and then call `AlarmSumPrev()` within a loop, to move backwards in the alarm summary.

This function can only be used if the Alarm Server is on the current machine.

This function can only be used if the Alarm Server is on the current machine. When the Alarm Server is not in the calling process, this function will become blocking and cannot be called from a foreground task. In this case, the return value will be undefined and a Cicode hardware alarm will be raised.

---

**Note:** The index passed to AlarmSum\* functions must be current. ie either:

- the index returned by most recent call to AlarmSumFirst/Last/Find/Next/Prev
  - OR
  - the index returned by the most recent call to AlarmSumAppend OR AlarmSumSplit.
- 

## Syntax

INT AlarmSumLast( [ClusterName] )

*ClusterName:*

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

The index of the most recent alarm summary entry, or -1 if no alarm summary entry is found.

---

**Note:** The index is a 32-bit integer. Storing it in a variable that is smaller than 32 bit integer (such as a 16-bit integer) may cause overflow which may result in an incorrect value.

---

## Related Functions

[AlarmSumCommit](#), [AlarmSumDelete](#), [AlarmSumSet](#), [AlarmSumType](#), [AlarmSumAppend](#), [AlarmDelete](#),  
[AlarmSumGet](#), [AlarmSumFirst](#), [AlarmSumNext](#), [AlarmSumPrev](#), [AlarmSumFind](#), [AlarmSplit](#), [AlarmSumSplit](#)

## Example

```
/* This function finds all alarm summary entries that match the
specified tag and sets the "OffTime" to the time specified. The
alarm entry is not acknowledged or set to the off state, the alarm
summary "OffTime" field is all that is affected. */

FUNCTION
SumSetTime(STRING sTag, INT Time)
    INT Index;
    STRING Name;
    Index=AlarmSumLast();
    WHILE Index<>-1 DO
        Name=AlarmSumGet(Index,"Tag");
        IF Name=sTag THEN
            AlarmSumSet(Index,"OffTime",Time);
        END
        Index=AlarmSumPrev(Index);
    END
END
```

## AlarmSumNext

Gets the index of the next alarm summary entry, that is, the entry that occurred later than the entry specified by Index. You can use this index in the AlarmSumGet() function to get field data from an alarm record, in the

AlarmSumSet() function to change the existing data in that record, or in the AlarmSumDelete() function to delete the record.

You can use this function to work with a series of alarm summary records. Call the AlarmSumFirst() or AlarmSumFind() function to get the index, and then call AlarmSumNext() within a loop, to move forwards in the alarm summary.

You can also get the index of an entry as soon as it displays on the alarm summary. Alarm summary entries are recorded with the most recent entry at the end of the list. Call AlarmSumLast() to get the index for the most recent entry, and then call AlarmSumNext() to get the index for the next entry that occurs.

This function can only be used if the Alarm Server is on the current machine. When the Alarm Server is not in the calling process, this function will become blocking and cannot be called from a foreground task. In this case, the return value will be undefined and a Cicode hardware alarm will be raised.

---

**Note:** The index passed to AlarmSum\* functions must be current. ie either:

- the index returned by most recent call to AlarmSumFirst/Last/Find/Next/Prev
  - OR
  - the index returned by the most recent call to AlarmSumAppend OR AlarmSumSplit.
- 

## Syntax

**INT AlarmSumNext(*Index* [, *ClusterName*] )**

*Index*:

The alarm summary index (returned from the AlarmSumFirst(), AlarmSumNext(), AlarmSumLast(), AlarmSumPrev(), or AlarmSumFind() function).

---

**Note:** If the supplied index was returned by AlarmSumAppend, AlarmSum Next and AlarmSumPrev will return -1, and an error (561, invalid index) is raised. To retrieve extended error information, call IsError().

---

*ClusterName*:

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

The index of the next alarm summary entry or -1 if no more alarm summary entries are found.

---

**Note:** The index is a 32-bit integer. Storing it in a variable that is smaller than 32 bit integer (such as a 16-bit integer) may cause overflow which may result in an incorrect value.

---

## Related Functions

[AlarmSumCommit](#), [AlarmSumDelete](#), [AlarmSumSet](#), [AlarmSumType](#), [AlarmSumAppend](#), [AlarmDelete](#), [MsgRPC](#), [AlarmSumGet](#), [AlarmSumFirst](#), [AlarmSumLast](#), [AlarmSumPrev](#), [AlarmSumFind](#), [AlarmSplit](#), [AlarmSumSplit](#)

## Example

See [AlarmSumFirst](#)

## See Also

[Alarm Functions](#)

### AlarmSumPrev

Gets the index of the previous alarm summary entry, that is, the entry that occurred before the entry specified by *Index*. You can use this index in the AlarmSumGet function to get field data from an alarm record, in the AlarmSumSet function to change the existing data in that record, or in the AlarmSumDelete function to delete the record.

You can use this function to work with a series of alarm summary records. Call the AlarmSumLast() or AlarmSumFind() function to get the index, and then call AlarmSumPrev() within a loop, to move backwards in the alarm summary.

This function can only be used if the Alarm Server is on the current machine. When the Alarm Server is not in the calling process, this function will become blocking and cannot be called from a foreground task. In this case, the return value will be undefined and a Cicode hardware alarm will be raised.

---

**Note:** The index passed to AlarmSum\* functions must be current. ie either:

- the index returned by most recent call to AlarmSumFirst/Last/Find/Next/Prev
  - OR
  - the index returned by the most recent call to AlarmSumAppend OR AlarmSumSplit.
- 

## Syntax

**INT AlarmSumPrev(*Index* [, *ClusterName*] )**

*Index*:

The alarm summary index (returned from the AlarmSumFirst(), AlarmSumNext(), AlarmSumLast(), AlarmSumPrev(), or AlarmSumFind() function).

---

**Note:** If the supplied index was returned by AlarmSumAppend, AlarmSum Next and AlarmSumPrev will return -1, and an error (561, invalid index) is raised. To get extended error information, call IsError().

---

*ClusterName*:

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

The index of the previous alarm summary entry or -1 if no more alarm summary entries are found.

---

**Note:** The index is a 32-bit integer. Storing it in a variable that is smaller than 32 bit integer (such as a 16-bit integer) may cause overflow which may result in an incorrect value.

---

## Related Functions

[AlarmSumCommit](#), [AlarmSumSet](#), [AlarmSumType](#), [AlarmSumAppend](#), [AlarmDelete](#), [MsgRPC](#), [AlarmSumGet](#), [AlarmSumFirst](#), [AlarmSumNext](#), [AlarmSumLast](#), [AlarmSumFind](#), [AlarmSplit](#), [AlarmSumSplit](#)

## Example

See [AlarmSumLast](#).

## See Also

[Alarm Functions](#)

## AlarmSumSet

Sets field information in an alarm summary entry. You identify the alarm summary entry by the *Index*, returned by one of the alarm summary search functions.

By embedding this function in a loop, you can change field data in a series of alarm summary entries. To start from the oldest entry, call the AlarmSumFirst function to get the index, and then call AlarmSumNext in a loop. To work back from the latest entry, call AlarmSumLast and then AlarmSumPrev in a loop.

You can also get the *Index* from the AlarmSumFind function, which finds an alarm summary entry by its alarm record identifier and time of activation.

This function can only be used if the Alarm Server is on the current machine. When the Alarm Server is not in the calling process, this function will become blocking and cannot be called from a foreground task. In this case, the return value will be undefined and a Cicode hardware alarm will be raised.

Fields of an appended alarm can only be set using this function or the replacement function AlmSummarySetField.

You can use user events to keep a record of logins, or control operations that you need to display in the alarm summary etc. The fields of UserEvents need to be set immediately after creation using this function. These entries in the Alarm Summary cannot be filtered from the summary in an AlmSummaryOpen browse session.

---

**Note:**

- In a redundant pair scenario, this function will return an error condition and not do anything if the current server is not main. You can call this function on the primary server and the standby server so that function can succeed on one of the two servers in the specified cluster. The change will be synchronised to the other server after a short period of time (typically in 5 seconds).
  - This function will only work while the record is writable which is controlled by the ArchiveAfter Parameter. Refer to the topic *Configure the Archiving Parameters* in the main help for more information.
- 

## Syntax

INT **AlarmSumSet**(*Index*, *sField*, *sData* [, *ClusterName*] )

*Index*:

The alarm summary index (returned from the AlarmSumFirst(), AlarmSumNext(), AlarmSumLast(), AlarmSumPrev(), AlarmSumAppend(), or AlarmSumFind() function).

*sField*:

The name of the field in which data is to be set:

AckTime	Alarm acknowledged time
Comment	Alarm comment

OffMilli (for time stamped alarms only)	Alarm millisecond off time
OffTime	Alarm OFF time

*sData:*

The new value of the field.

*ClusterName:*

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

0 (zero) if the alarm summary entry exists, otherwise an error is returned.

## Related Functions

[AlarmSumCommit](#), [AlarmSumDelete](#), [AlarmSumType](#), [AlarmSumAppend](#), [AlarmDelete](#), [MsgRPC](#), [AlarmSumGet](#), [AlarmSumFirst](#), [AlarmSumNext](#), [AlarmSumLast](#), [AlarmSumPrev](#), [AlarmSumFind](#), [AlarmSplit](#), [AlarmSumSplit](#), [AlmSummarySetFieldValue](#)

## Example

See [AlarmSumFirst](#).

## See Also

[Alarm Functions](#)

## AlarmSumSplit

Splits the alarm summary entry identified by index.

On operation success, the OffTime of the current alarm summary entry is filled with specified OnTime. Meanwhile, a new alarm summary entry is raised with the specified OnTime and no OffTime.

---

**Note:** This function will only work while the record is writable which is controlled by the ArchiveAfter Parameter. Refer to the topic [Configure the Archiving Parameters](#) in the main help for more information.

The operation represents a logical split at the specified OnTime to an active alarm instance. You can use this function to add a comment related to the specified OnTime.

The alarm summary entry must not have OffTime, or the function would return -1 and do nothing.

The function is blocking and cannot be called from a foreground task. In this case, the return value will be undefined and a Cicode hardware alarm will be raised.

This function can only be used if the alarm server is on the current machine. If the Alarm Server is not in the calling process, the calling process needs to run from the same machine as the alarm server.

In a redundant pair scenario, the function would return error condition and do nothing if the current server is not main. You can call this function on both the primary server and the standby server so that function can

succeed on one of the two servers in the specified cluster. The change will be synchronized to the other server after a short period of time (typically in 5 seconds).

---

**Note:** The index passed to AlarmSum\* functions must be current. ie either:

- the index returned by most recent call to AlarmSumFirst/Last/Find/Next/Prev OR
  - the index returned by the most recent call to AlarmSumAppend OR AlarmSumSplit.
- 

## Syntax

INT **AlarmSumSplit**(*Index* [, *ClusterName*, *OnTime*, *OnMilli*, *bRedundant*] )

*Index*:

The alarm summary index (returned from the AlarmSumFirst(), AlarmSumNext(), AlarmSumLast(), AlarmSumPrev(), AlarmSumAppend(), or AlarmSumFind() function).

*ClusterName*:

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

*OnTime*:

The alarm OnTime (if omitted or is equal to 0, will default to current time, i.e. the time the function was invoked).

*OnMilli*:

Milliseconds part of the alarm's ON time

*bRedundant:(optional)*

New alarm record is created on both redundant server instances. This parameter has been deprecated and is always set to true.

## Return Value

The Index of the new entry, or -1 on error.

---

**Note:** The index is a 32-bit integer. Storing it in a variable that is smaller than 32 bit integer (such as a 16-bit integer) may cause overflow which may result in an incorrect value.

---

## Related Functions

[AlarmSumCommit](#), [AlarmSumDelete](#), [AlarmSumSet](#), [AlarmSumType](#), [AlarmSumAppend](#), [AlarmDelete](#), [MsgRPC](#), [AlarmSumGet](#), [AlarmSumFirst](#), [AlarmSumNext](#), [AlarmSumLast](#), [AlarmSumPrev](#), [AlarmSumFind](#), [AlarmSplit](#)

## Example

```
/* This function finds the first alarm summary entry that matches
the specified tag, splits that entry and then adds the specified
comment to the new entry. */
FUNCTION
AlarmSplitAdd(STRING Tag, STRING Comment)
    INT Index;
    STRING Name;
```

```
Index=AlarmSumFirst();
WHILE Index<>-1 DO
    Name=AlarmSumGet(Index, "Tag");
    IF Name=sTag THEN
        AlarmSumSplit(Index);
        Index=AlarmSumFirst();
        AlarmSumSet(Index, "Comment", Comment);
        Index=-1;
    ELSE
        Index=AlarmSumNext(Index);
    END
END
END
```

## See Also

[Alarm Functions](#)

### AlarmSumType

Retrieves a value that indicates a specified alarm's type, that is whether it's a digital alarm, an analog alarm, hardware alarm, etc.

This function can only be used if the Alarm Server is on the current machine. When the Alarm Server is not in the calling process, this function will become blocking and cannot be called from a foreground task. In this case, the return value will be undefined and a Cicode hardware alarm will be raised.

## Syntax

**INT AlarmSumType(*Index* [, *ClusterName*] )**

*Index*:

The alarm summary index (returned from the AlarmSumFirst(), AlarmSumNext(), AlarmSumLast(), AlarmSumPrev(), AlarmSumAppend(), or AlarmSumFind() function).

*ClusterName*:

Specifies the name of the cluster in which the Alarm Server resides. This is optional if you have one cluster or are resolving the alarm server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

A number that represents one of the following alarm types:

- 0 = digital alarm
- 1 = analog alarm
- 2 = advanced alarm
- 3 = Multi-Digital alarm
- 4 = Argyle analog alarm
- 5 = user-generated event

- 6 = high resolution alarm
- 8 = time-stamped digital alarm
- 9 = time-stamped analog alarm
- -1 indicates an invalid response to the request.

## Related Functions

[AlarmSumCommit](#), [AlarmSumDelete](#), [AlarmSumSet](#), [AlarmSumType](#), [AlarmSumAppend](#), [AlarmDelete](#), [MsgRPC](#),  
[AlarmSumGet](#), [AlarmSumFirst](#), [AlarmSumNext](#), [AlarmSumLast](#), [AlarmSumPrev](#), [AlarmSumFind](#), [AlarmSplit](#)

## See Also

[Alarm Functions](#)

### AlarmTagFromEquipment

Returns the first tag associated with a piece of equipment.

## Syntax

**INT AlarmTagFromEquipment(STRING Tag, [, STRING ClusterName])**

*Tag:*

A string that identifies the alarm to acknowledge. It can be one of the following:

- An alarm tag — for example, "Fire1"
- An alarm equipment item — for example, "Motor1.AlarmFire"

Specify an empty string (" ") to match all alarm tags.

*ClusterName:*

The cluster in which the tag resides

## Return Value

0 (zero) if successful, otherwise an error code will be returned

## See Also

[Alarm Functions](#)

### AlmBrowseAck

The AlmBrowseAck function acknowledges the alarm tag at the current cursor position in an active data browse session.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

INT AlmBrowseAck(LONG *iSession*)

*iSession*:

The handle to a browse session previously returned by a [AlmBrowseOpen](#) call.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmBrowseDisable](#), [AlmBrowseEnable](#), [AlmBrowseClose](#), [AlmBrowseFirst](#), [AlmBrowseGetField](#), [AlmBrowseNext](#),  
[AlmBrowseNumRecords](#), [AlmBrowseOpen](#), [AlmBrowsePrev](#)

## See Also

[Alarm Functions](#)

## AlmBrowseClose

The AlmBrowseClose function terminates an active data browse session and cleans up resources associated with the session.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

INT AlmBrowseClose(LONG *iSession*)

*iSession*:

The handle to a browse session previously returned by a [AlmBrowseOpen](#) call.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmBrowseAck](#), [AlmBrowseDisable](#), [AlmBrowseEnable](#), [AlmBrowseFirst](#), [AlmBrowseGetField](#), [AlmBrowseNext](#),  
[AlmBrowseNumRecords](#), [AlmBrowseOpen](#), [AlmBrowsePrev](#)

## See Also

[Alarm Functions](#)

## AlmBrowseDisable

The AlmBrowseDisable function disables the alarm tag at the current cursor position in an active data browse session.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

```
INT AlmBrowseDisable(INT iSession [, INT EndTime [, STRING Comment]] )
```

*iSession*:

The handle to a browse session previously returned by an [AlmBrowseOpen](#) call.

*Comment*

An optional comment limited to 200 characters explaining why the alarm is disabled. If the comment exceeds 200 characters, hardware error 274 ("Invalid argument passed") will be displayed.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmBrowseAck](#), [AlmBrowseEnable](#), [AlmBrowseClose](#), [AlmBrowseFirst](#), [AlmBrowseGetField](#), [AlmBrowseNext](#),  
[AlmBrowseNumRecords](#), [AlmBrowseOpen](#), [AlmBrowsePrev](#)

## Examples

```
// Retrieve the handle to a browse session returned by an AlmBrowseOpen call.  
// ...  
// Disable the associated alarm for the next 60 minutes  
nEndTime = DateAdd(TimeCurrent(), 3600);  
AlmBrowseDisable(iSession, nEndTime, "Shelve alarm for 60 minutes.");  
// Retrieve the handle to a browse session returned by an AlmBrowseOpen call.  
// ...  
// Disable the associated alarm until 20 Dec 2016 6:30am local time  
nEndTime = DateAdd(StrToDate("20/12/2016"), StrToTime("6:30"));  
AlmBrowseDisable(iSession, nEndTime, "Shelve alarms until 20 Dec 2016 6:30am");
```

## See Also

[Alarm Functions](#)

## AlmBrowseEnable

The AlmBrowseEnable function enables the alarm tag at the current cursor position in an active data browse session.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

```
INT AlmBrowseEnable(INT iSession [, INT bAcknowledge])
```

*iSession*:

The handle to a browse session previously returned by an [AlmBrowseOpen](#) call.

*bAcknowledge*:

Forces acknowledgment of an alarm after it is enabled. The accepted values are:

0 — (default) enforced acknowledgment will not be applied.

1 — alarm will be acknowledged when enabled.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmBrowseAck](#) [AlmBrowseDisable](#), [AlmBrowseClose](#), [AlmBrowseFirst](#), [AlmBrowseGetField](#), [AlmBrowseNext](#),  
[AlmBrowseOpen](#), [AlmBrowsePrev](#)

## Example

```
// Retrieve the handle to a browse session returned by an AlmBrowseOpen call.  
// ...  
//Enable the associated alarm and enforce acknowledgment  
AlmBrowseEnable(iSession, 1);
```

## See Also

[Alarm Functions](#)

## AlmBrowseFirst

The [AlmBrowseFirst](#) function places the data browse cursor at the first record.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

```
INT AlmBrowseFirst(LONG iSession)
```

*iSession*:

The handle to a browse session previously returned by a [AlmBrowseOpen](#) call.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmBrowseAck](#), [AlmBrowseDisable](#), [AlmBrowseEnable](#), [AlmBrowseClose](#), [AlmBrowseGetField](#), [AlmBrowseNext](#), [AlmBrowseNumRecords](#), [AlmBrowseOpen](#), [AlmBrowsePrev](#)

## See Also

[Alarm Functions](#)

### AlmBrowseGetField

The AlmBrowseGetField function retrieves the value of the specified field from the record the data browse cursor is currently referencing.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

STRING **AlmBrowseGetField**(LONG *iSession*, STRING *FieldName*)

*iSession*:

The handle to a browse session previously returned by a [AlmBrowseOpen](#) call.

*sFieldName*:

The name of the field that references the value to be returned. Supported fields are:

ACKDATE, ACKDATEEXT, ACKTIME, ACQERROR, ALARMTYPE, ALMCOMMENT, AREA, CATEGORY, CAUSE1, CAUSE2, CAUSE3, CAUSE4, CAUSE5, CAUSE6, CAUSE7, CAUSE8, CLUSTER, COMMENT, CONSEQUENCE1, CONSEQUENCE2, CONSEQUENCE3, CONSEQUENCE4, CONSEQUENCE5, CONSEQUENCE6, CONSEQUENCE7, CONSEQUENCE8, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8, DATE, DATEEXT, DEADBAND, DELTATIME, DESC, DEVDELAY, DEVIATION, DISABLEENDDATE, DISABLEENDDATEEXT, DISABLEENDTIME, DISABLECOMMENT, ERRDESC, ERRPAGE, EQUIPMENT, FORMAT, GROUP, HDELAY, HELP, HHDELAY, HIGH, HIGHHIGH, LDELAY, LEVEL, LLDELAY, LOCALTIMEDATE, LOGSTATE, LOW, LOWLOW, MILLISEC, NAME, NATIVE\_COMMENT, NATIVE\_DESC, NATIVE\_NAME, NATIVE\_SUMDESC, OFFDATE, OFFDATEEXT, OFFMILLI, OFFTIME, OFFTIMEDATE, OLD\_DESC, ONDATE, ONDATEEXT, ONMILLI, ONTIME, ONTIMEDATE, PAGING, PAGINGGROUP, PRIORITY, PRIV, QUALITY, RATE, RECEIPTLOCALTIMEDATE, RECEIPTDATE, RESPONSE1, RESPONSE2, RESPONSE3, RESPONSE4, RESPONSE5, RESPONSE6, RESPONSE7, RESPONSE8, RESPONSENUM, RESPTIME1, RESPTIME2, RESPTIME3, RESPTIME4, RESPTIME5, RESPTIME6, RESPTIME7, RESPTIME8, STATE, STATE\_DESC, STATE\_DESC0, STATE\_DESC1, STATE\_DESC2, STATE\_DESC3, STATE\_DESC4, STATE\_DESC5, STATE\_DESC6, STATE\_DESC7, SUMDESC, SUMSTATE, SUMTYPE, TAG, TAGEX, TIME, TIMEDATE, TYPE, TYPENUM, USERNAME, VALUE.

See [Browse Function Field Reference](#) for information about fields.

## Return Value

The value of the specified field as a string. An empty string may or may not be an indication that an error has been detected. The last error should be checked in this instance to determine if an error has actually occurred.

## Related Functions

[AlmBrowseAck](#), [AlmBrowseDisable](#), [AlmBrowseEnable](#), [AlmBrowseClose](#), [AlmBrowseFirst](#), [AlmBrowseNext](#),  
[AlmBrowseNumRecords](#), [AlmBrowseOpen](#), [AlmBrowsePrev](#)

## Example

```
STRING fieldValue = "";
STRING fieldName = "TYPE";
INT errorCode = 0;
...
fieldValue = AlmBrowseGetField(iSession, sFieldName);
IF fieldValue <> "" THEN
    // Successful case
ELSE
    // Function returned an error
END
...
```

## AlmBrowseNext

The AlmBrowseNext function moves the data browse cursor forward one record. If you call this function after you have reached the end of the records, error 412 is returned (Databrowse session EOF).

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **AlmBrowseNext**(LONG *iSession*)

*iSession*:

The handle to a browse session previously returned by a [AlmBrowseOpen](#) call.

## Return Value

0 (zero) if the browse has successfully been moved to the next record, otherwise an error code is returned.

## Related Functions

[AlmBrowseAck](#), [AlmBrowseDisable](#), [AlmBrowseEnable](#), [AlmBrowseClose](#), [AlmBrowseFirst](#), [AlmBrowseGetField](#),  
[AlmBrowseNumRecords](#), [AlmBrowseOpen](#), [AlmBrowsePrev](#)

## See Also

[Alarm Functions](#)

## AlmBrowseNumRecords

The AlmBrowseNumRecords function returns the number of records that match the filter criteria.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

```
LONG AlmBrowseNumRecords(LONG iSession)
```

*iSession*:

The handle to a browse session previously returned by a AlmBrowseOpen call.

## Return Value

The number of records that have matched the filter criteria. A value of 0 denotes that no records have matched. A value of -1 denotes that the browse session is unable to provide a fixed number. This may be the case if the data being browsed changed during the browse session.

## Related Functions

[AlmBrowseAck](#), [AlmBrowseDisable](#), [AlmBrowseEnable](#), [AlmBrowseClose](#), [AlmBrowseFirst](#), [AlmBrowseGetField](#),  
[AlmBrowseNext](#), [AlmBrowseOpen](#), [AlmBrowsePrev](#)

## Example

```
INT numRecords = 0;  
...  
numRecords = AlmBrowseNumRecords(iSession);  
IF numRecords <> 0 THEN  
    // Have records  
ELSE  
    // No records  
END  
...
```

## See Also

[Alarm Functions](#)

## AlmBrowseOpen

The AlmBrowseOpen function initiates a new browse session and returns a handle to the new session that can be used in subsequent data browse function calls. Use this function to browse all configured alarms.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

**Note:** After calling AlmBrowseOpen() it is necessary to call AlmBrowseFirst() in order to place the cursor at the beginning of the browse session, otherwise a hardware alarm is invoked.

## Syntax

```
LONG AlmBrowseOpen( Filter, STRING Fields [, STRING Clusters [,INT AutoCloseMode]] )
```

### *Filter:*

A filter expression specifying the records to return during the browse. An empty string indicates that all records will be returned. Where a fieldname is not specified in the filter, it is assumed to be tagname. For example, the filter "AAA" is equivalent to "Tag=AAA". Multiple filters separated by semicolons are supported.

---

**Note:** When using Date/Time fields specify in the number of seconds since 1970. For example,  
LOCALTIMEDATE>=1348723732.

See the topic [Implementing Alarm Filters Using Cicode](#) for more information about filter syntax.

### *Fields:*

Specifies via a comma delimited string the columns to be returned during the browse. An empty string indicates that the server will return all available columns. Supported fields are:

ACKDATE, ACKDATEEXT, ACKMILLI, ACKTIME, ACKUTC, ACQDESC, ACQERROR, ALARMTYPE, ALMCOMMENT, AREA, ARR\_SIZE, CATEGORY, CAUSE1...8, CLASSIFICATION, CLUSTER, COMMENT, CONSEQUENCE1...8, CUSTOM1...8, DATE, DATEEXT, DEADBAND, DELAY, DELTAMILLI, DELTATIME, DESC, DEVDELAY, DEVIATION, DISABLECOMMENT, DISABLEDDATE, DISABLEDTIME, DISABLEENDDATE, DISABLEENDDATEEXT, DISABLEENDTIME, ENG\_ZERO, ERRDESC,ERRPAGE, EQUIPMENT, FORMAT, FULLNAME, GROUP, HDELAY, HELP, HHDELAY, HIGH, HIGHHIGH, HISTORIAN, ITEM, LDELAY, LLDELAY, LOCALTIMEDATE, LOGSTATE, LOW, LOWLOW, MESSAGE, MILLSEC, NAME, NATIVE\_COMMENT, NATIVE\_DESC, NATIVE\_NAME, NATIVE\_SUMDESC, OFFDATE, OFFDATEEXT, OFFMILLI, OFFTIME, OFFTIMEDATE, OFFUTC, OLD\_DESC, ONDATE, ONDATEEXT, ONMILLI, ONTIME, ONTIMEDATE, ONUTC, PAGING, PAGINGGROUP, PRIORITY, PRIV, PSI\_TYPE, RATE, RECEIPTLOCALTIMEDATE, RECEIPTDATE, RECEIPTDATEEXT, RECEIPTMILLISEC, RECEIPTTIME, RECEIPTTIMEINT, RECEIPTTIMETICKS, RESPONSE1...8, RESPONSENUM, SETPOINT, STATE, STATE\_DESC, STATE\_DESCO...7, SUMDESC, SUMSTATE, SUMTYPE, TAG, TAGEX, TAGGENLINK, TIME, TIMEDATE, TIMEINT, TIMETICKS, TSQUALITY, TYPE, TYPENUM, USERDESC, USERNAME, USERLOCATION, VALUE.

See [Browse Function Field Reference](#) for information about fields.

### *Clusters:*

An optional parameter that specifies via a comma delimited string the subset of the clusters to browse. An empty string indicates that the connected clusters will be browsed.

### *AutoCloseMode*

An optional parameter to automatically close the browsing session at page navigation.

0 - (Default) Will not automatically close the browsing session. Use AlmBrowseClose to close the session manually.

1 - The browsing session will be closed when the page is changed or otherwise closed.

---

**Note:** All other modes are reserved.

## Return Value

Returns an integer handle to the browse session. Returns -1 when an error is detected.

The returned entries will be ordered alphabetically by name.

## Related Functions

[AlmBrowseAck](#), [AlmBrowseDisable](#), [AlmBrowseEnable](#), [AlmBrowseClose](#), [AlmBrowseFirst](#), [AlmBrowseGetField](#),  
[AlmBrowseNext](#), [AlmBrowseNumRecords](#), [AlmBrowsePrev](#)

## Example

```
INT iSession;
...
iSession = AlmBrowseOpen("NAME=ABC*", "NAME,TYPE",
"ClusterA,ClusterB");
IF iSession <> -1 THEN
    // Successful case
ELSE
    // Function returned an error
END
...
```

## See Also

[Alarm Functions](#)

### AlmBrowsePrev

The AlmBrowsePrev function moves the data browse cursor back one record. If you call this function after you have reached the beginning of the records, error 412 is returned (Databrowse session EOF).

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **AlmBrowsePrev**(LONG *iSession*)

*iSession*:

The handle to a browse session previously returned by a AlmBrowseOpen call.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmBrowseAck](#), [AlmBrowseDisable](#), [AlmBrowseEnable](#), [AlmBrowseClose](#), [AlmBrowseFirst](#), [AlmBrowseGetField](#),  
[AlmBrowseNext](#), [AlmBrowseNumRecords](#), [AlmBrowseOpen](#)

## See Also

[Alarm Functions](#)

## AlmSummaryAck

The AlmSummaryAck function acknowledges the alarm in the active alarm list which is linked to the current entry of the alarm summary browse session.

If the current alarm summary browse session entry is a user event the function will have no effect.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

**INT AlmSummaryAck(LONG Session)**

*Session:*

The handle to a browse session previously returned by a AlmSummaryOpen call.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmSummaryClear](#), [AlmSummaryClose](#), [AlmSummaryDelete](#), [AlmSummaryDeleteAll](#), [AlmSummaryDisable](#),  
[AlmSummaryEnable](#), [AlmSummaryFirst](#), [AlmSummaryGetField](#), [AlmSummaryLast](#), [AlmSummaryNext](#),  
[AlmSummaryOpen](#), [AlmSummaryPrev](#), [AlmSummaryNumRecords](#)

## See Also

[Alarm Functions](#)

## AlmSummaryCommit

The AlmSummaryCommit function triggers the actual write of the value for the field previously specified by AlmSummarySetFieldValue.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

---

**Note:** This function will only work while the record is writable which is controlled by the [Alarm]ArchiveAfter parameter. Refer to the topic [Configure the Archiving Parameters](#) for more information.

---

## Syntax

**AlmSummaryCommit**(*iSession*)

*iSession:*

The handle to a browse session previously returned by a AlmSummaryOpen call.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error is returned.

## Related Functions

[AlmSummaryAck](#) [AlmSummaryClear](#), [AlmSummaryClose](#), [AlmSummaryDelete](#), [AlmSummaryDeleteAll](#),  
[AlmSummaryDisable](#), [AlmSummaryEnable](#), [AlmSummaryFirst](#), [AlmSummaryGetField](#), [AlmSummaryLast](#),  
[AlmSummaryNext](#), [AlmSummaryOpen](#), [AlmSummaryPrev](#), [AlmSummarySetFieldValue](#)

## Example

```
INT errorCode = 0;  
...  
errorCode = AlmSummaryCommit(iSession);  
IF errorCode = 0 THEN  
    // Successful case  
ELSE  
    // Function returned an error  
END  
...
```

## See Also

[Alarm Functions](#)

## AlmSummaryClear

The AlmSummaryClear function clears the latched alarm at the current cursor position in an active data browse session. A latched alarm is an alarm which is OFF and acknowledged when [Alarm]AckHold is set to 1. The function does nothing if the alarm summary entry at the cursor is not linked to a latched alarm.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

INT **AlmSummaryClear**(LONG Session)

*Session:*

The handle to a browse session previously returned by a AlmSummaryOpen call.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmSummaryAck](#), [AlmSummaryClose](#), [AlmSummaryDelete](#), [AlmSummaryDeleteAll](#), [AlmSummaryDisable](#),

[AlmSummaryEnable](#), [AlmSummaryGetField](#), [AlmSummaryLast](#), [AlmSummaryNext](#), [AlmSummaryOpen](#),  
[AlmSummaryPrev](#), [AlmSummaryNumRecords](#)

## See Also

[Alarm Functions](#)

### AlmSummaryClose

The AlmSummaryClose function terminates an active data browse session and cleans up all resources associated with the session.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

INT **AlmSummaryClose**(LONG *iSession*)

*iSession*

The handle to a browse session previously returned by a AlmSummaryOpen call.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmSummaryAck](#), [AlmSummaryClear](#), [AlmSummaryDelete](#), [AlmSummaryDeleteAll](#), [AlmSummaryDisable](#),  
[AlmSummaryEnable](#), [AlmSummaryFirst](#), [AlmSummaryGetField](#), [AlmSummaryLast](#), [AlmSummaryNext](#),  
[AlmSummaryOpen](#), [AlmSummaryPrev](#), [AlmSummaryNumRecords](#)

## See Also

[Alarm Functions](#)

### AlmSummaryDelete

The AlmSummaryDelete function deletes the record in the filtered list that the cursor is currently referencing. The cursor moves to the next available entry in the data browse session after the current alarm summary is deleted.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

---

**Note:** This function will only work while the record is writable which is controlled by the ArchiveAfter parameter. Refer to the topic [Configure the Archiving Parameters](#) for more information.

## Syntax

```
INT AlmSummaryDelete(LONG iSession)
```

*iSession:*

The handle to a filtered list previously returned by a AlmSummaryOpen call.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmSummaryAck](#), [AlmSummaryClear](#), [AlmSummaryClose](#), [AlmSummaryDeleteAll](#) [AlmSummaryDisable](#),  
[AlmSummaryEnable](#), [AlmSummaryFirst](#), [AlmSummaryGetField](#), [AlmSummaryLast](#), [AlmSummaryNext](#),  
[AlmSummaryOpen](#), [AlmSummaryPrev](#), [AlmSummaryNumRecords](#)

## Example

```
INT errorCode = 0;  
...  
errorCode = AlmSummaryDelete(iSession);  
IF errorCode = 0 THEN  
// Successful case  
ELSE  
// Function returned an error  
END  
...
```

## See Also

[Alarm Functions](#)

### AlmSummaryDeleteAll

The AlmSummaryDeleteAll function deletes every record from the filtered list source.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

**Note:** This function will only work while the record is writable which is controlled by the ArchiveAfter parameter. Refer to the topic [Configure the Archiving Parameters](#) for more information.

## Syntax

```
INT AlmSummaryDeleteAll(LONG iSession)
```

*iSession:*

The handle to a filtered list previously returned by a AlmSummaryOpen call.

## Return Value

0 (zero) if the alarm filtered list session exists, otherwise an error code is returned.

## Related Functions

[AlmSummaryAck](#), [AlmSummaryClear](#), [AlmSummaryClose](#), [AlmSummaryDelete](#), [AlmSummaryDisable](#),  
[AlmSummaryEnable](#), [AlmSummaryFirst](#), [AlmSummaryGetField](#), [AlmSummaryLast](#), [AlmSummaryNext](#),  
[AlmSummaryOpen](#), [AlmSummaryPrev](#), [AlmSummaryNumRecords](#)

## Example

```
INT errorCode = 0;
...
errorCode = AlmSummaryDeleteAll(iSession);
IF errorCode = 0 THEN
    // Successful case
ELSE
    // Function returned an error
END
```

## See Also

[Alarm Functions](#)

## AlmSummaryDisable

The AlmSummaryDisable function disables the alarm at the current cursor position in an active data browse session.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

INT **AlmSummaryDisable**(INT *iSession* [, INT *EndTime* [, STRING *Comment*]] )

*iSession*:

The handle to a browse session previously returned by an [AlmSummaryOpen](#) call.

*EndTime*:

A date/time variable that indicates when the alarm will no longer be disabled. If this parameter is omitted or set to 0, the alarm will be disabled indefinitely.

*Comment*:

An optional comment limited to 200 characters explaining why the alarm is disabled. If the comment exceeds 200 characters, hardware error 274 ("Invalid argument passed") will be displayed.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmSummaryAck](#), [AlmSummaryClear](#), [AlmSummaryClose](#), [AlmSummaryDelete](#), [AlmSummaryDeleteAll](#),  
[AlmSummaryEnable](#), [AlmSummaryFirst](#), [AlmSummaryGetField](#), [AlmSummaryLast](#), [AlmSummaryNext](#),  
[AlmSummaryOpen](#), [AlmSummaryPrev](#), [AlmSummaryNumRecords](#)

## Examples

```
// Retrieve the handle to a browse session returned by an AlmSummaryOpen call.  
// ...  
// Disable the associated alarm for the next 60 minutes  
nEndTime = DateAdd(TimeCurrent(), 3600);  
AlmSummaryDisable(iSession, nEndTime, "Shelve alarm for 60 minutes.");  
// Retrieve the handle to a browse session returned by an AlmSummaryOpen call.  
// ...  
// Disable the associated alarm until 20 Dec 2016 6:30am local time  
nEndTime = DateAdd(StrToDate("20/12/2016"), StrToTime("6:30"));  
AlmSummaryDisable(iSession, nEndTime, "Shelve alarms until 20 Dec 2016 6:30am");
```

## AlmSummaryEnable

The AlmSummaryEnable function enables the alarm at the current cursor position in an active data browse session.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

INT **AlmSummaryEnable**(INT *iSession* [, INT *bAcknowledge*])

*iSession*:

The handle to a browse session previously returned by an [AlmSummaryOpen](#) call.

*bAcknowledge*:

Forces acknowledgment of an alarm after it is enabled. The accepted values are:

0 — (default) enforced acknowledgment will not be applied.

1 — alarm will be acknowledged when enabled.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmSummaryAck](#), [AlmSummaryClear](#), [AlmSummaryClose](#), [AlmSummaryDelete](#), [AlmSummaryDeleteAll](#),  
[AlmSummaryDisable](#), [AlmSummaryFirst](#), [AlmSummaryGetField](#), [AlmSummaryLast](#), [AlmSummaryNext](#),  
[AlmSummaryOpen](#), [AlmSummaryPrev](#), [AlmSummaryNumRecords](#)

## Example

```
// Retrieve the handle to a browse session returned by an AlmSummaryOpen call.  
// ...  
//Enable the associated alarm and enforce acknowledgment  
AlmSummaryEnable(iSession, 1);
```

## See Also

[Alarm Functions](#)

### AlmSummaryFirst

The AlmSummaryFirst function places the data browse cursor at the first record.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **AlmSummaryFirst**(LONG *iSession*)

*iSession*:

The handle to a browse session previously returned by a AlmSummaryOpen call.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmSummaryAck](#), [AlmSummaryClear](#), [AlmSummaryDelete](#), [AlmSummaryDeleteAll](#), [AlmSummaryDisable](#),  
[AlmSummaryEnable](#), [AlmSummaryGetField](#), [AlmSummaryLast](#), [AlmSummaryNext](#), [AlmSummaryOpen](#),  
[AlmSummaryPrev](#), [AlmSummaryNumRecords](#)

## See Also

[Alarm Functions](#)

### AlmSummaryGetField

The AlmSummaryGetField function retrieves the value of the specified field from the record the data browse

cursor is currently referencing.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

```
STRING AlmSummaryGetField(LONG iSession, STRING sFieldName)
```

*iSession*:

The handle to a browse session previously returned by a AlmSummaryOpen call.

*sFieldName*:

The name of the field that references the value to be returned. Supported fields are:

ACKDATE, ACKDATEEXT, ACKTIME, ACQERROR, ALARMTYPE, ALMCOMMENT, AREA, CATEGORY, CLUSTER, COMMENT, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8, DATE, DATEEXT, DEADBAND, DELTATIME, DESC, DEVIATION, ERRDESC, ERPPAGE, EQUIPMENT, FORMAT, FULLNAME, GROUP, HELP, HIGH, HIGHHIGH, LOCALTIMEDATE, LOGSTATE, LOW, LOWLOW, MILLISEC, NAME, NATIVE\_COMMENT, NATIVE\_DESC, NATIVE\_NAME, NATIVE\_SUMDESC, NODE, OFFDATE, OFFDATEEXT, OFFMILLI, OFFTIME, OFFTIMEDATE, OLD\_DESC, ONDATE, ONDATEEXT, ONMILLI, ONTIME, ONTIMEDATE, PAGING, PAGINGGROUP, PRIORITY, PRIV, QUALITY, RATE, STATE, STATE\_DESC, STATE\_DESC0, STATE\_DESC1, STATE\_DESC2, STATE\_DESC3, STATE\_DESC4, STATE\_DESC5, STATE\_DESC6, STATE\_DESC7, SUMDESC, SUMSTATE, SUMTYPE, TAG, TAGEX, TIME, TIMEDATE, TYPE, TYPENUM, USERNAME, VALUE.

See [Browse Function Field Reference](#) for information about fields.

## Return Value

The value of the specified field as a string. An empty string may or may not be an indication that an error has been detected. The last error should be checked in this instance to determine if an error has actually occurred.

## Related Functions

[AlmSummaryAck](#), [AlmSummaryClear](#), [AlmSummaryClose](#), [AlmSummaryDelete](#), [AlmSummaryDeleteAll](#),  
[AlmSummaryDisable](#), [AlmSummaryEnable](#), [AlmSummaryFirst](#), [AlmSummaryLast](#), [AlmSummaryNext](#),  
[AlmSummaryOpen](#), [AlmSummaryPrev](#), [AlmSummaryNumRecords](#)

## Example

```
STRING fieldValue = "";
STRING fieldName = "TYPE";
INT errorCode = 0;
...
fieldValue = AlmSummaryGetField(iSession, sFieldName);
IF fieldValue <> "" THEN
    // Successful case
ELSE
    // Function returned an error
END
...
```

## See Also

[Alarm Functions](#)

### AlmSummaryLast

The AlmSummaryLast function places the data browse cursor at the most recent summary record from the last cluster of the available browsing cluster list.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT AlmSummaryLast(LONG *iSession*)**

*iSession*:

The handle to a browse session previously returned by a [AlmSummaryOpen](#) call.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmSummaryAck](#), [AlmSummaryClear](#), [AlmSummaryClose](#), [AlmSummaryDelete](#), [AlmSummaryDeleteAll](#),  
[AlmSummaryDisable](#), [AlmSummaryEnable](#), [AlmSummaryFirst](#), [AlmSummaryGetField](#), [AlmSummaryNext](#),  
[AlmSummaryOpen](#), [AlmSummaryPrev](#), [AlmSummaryNumRecords](#)

## See Also

[Alarm Functions](#)

### AlmSummaryNext

The AlmSummaryNext function moves the data browse cursor forward one record. If you call this function after you have reached the end of a summary, error 412 is returned (Databrowse session EOF).

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT AlmSummaryNext(LONG *iSession*)**

*iSession*:

The handle to a browse session previously returned by a [AlmSummaryOpen](#) call.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmSummaryAck](#), [AlmSummaryClose](#), [AlmSummaryDelete](#), [AlmSummaryDeleteAll](#), [AlmSummaryDisable](#),  
[AlmSummaryEnable](#), [AlmSummaryFirst](#), [AlmSummaryGetField](#), [AlmSummaryLast](#), [AlmSummaryOpen](#),  
[AlmSummaryPrev](#), [AlmSummaryNumRecords](#)

## See Also

[Alarm Functions](#)

## AlmSummaryNumRecords

The AlmSummaryNumRecords function retrieves the number of records in an alarm summary browse session. This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

LONG **AlmSummaryNumRecords**(LONG *iSession*)

*iSession*:

The handle to a browse session previously returned by a [AlmSummaryOpen](#) call.

## Return Value

The number of records that have matched the filter criteria. A value of 0 denotes that no records have matched. A value of -1 denotes that the browse session is unable to provide a fixed number. This may be the case if the data being browsed changed during the browse session.

## Related Functions

[AlmSummaryAck](#), [AlmSummaryClear](#), [AlmSummaryClose](#), [AlmSummaryDelete](#), [AlmSummaryDeleteAll](#),  
[AlmSummaryDisable](#), [AlmSummaryEnable](#), [AlmSummaryFirst](#), [AlmSummaryGetField](#), [AlmSummaryLast](#),  
[AlmSummaryNext](#), [AlmSummaryPrev](#)

## Example

```
INT numRecords = 0;  
...  
numRecords = AlmSummaryNumRecords(iSession);  
IF numRecords <> 0 THEN  
    // Have records  
ELSE  
    // No records
```

```
END  
...
```

## See Also

[Alarm Functions](#)

### AlmSummaryOpen

The AlmSummaryOpen function initiates a new browse session and returns a handle to the new session that can be used in subsequent data browse function calls.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

The AlarmSummaryOpen function allows direct, successful subsequent execution of [AlmSummaryNext](#) and [AlmSummaryPrev](#) functions.

The AlmSummaryNext function executed directly after AlmSummaryOpen will place the data browse cursor at the earliest summary record.

The AlmSummaryPrev function executed directly after AlmSummaryOpen will place the cursor at the most recent summary record.

---

**Note:** Performance may be affected when using the AlmSummaryOpen browse functions in a system which has accumulated large volumes of alarm summary entries.

---

To improve performance:

- Specify only the required fields in the Fields parameter of the AlmSummaryOpen Cicode function. By default the parameter is empty and the browse will retrieve all available fields. Refer to Example 1.
- Specify a small OnTime filter for the alarm when calling the Cicode function. Confirm that the AlmSummaryClose is called for each browse session that has been opened using AlmSummaryOpen. Refer to Example 2.
- Specify a small row limit in the RowLimit parameter of the AlmSummaryOpen function. The default value is -1 which will use the default row limit specified in the Citect.ini parameter [Alarm]QueryRowLimit. Refer to Example 3.

## Syntax

LONG **AlmSummaryOpen**(STRING *Filter*, STRING *Fields* [, STRING *Clusters* [, INT *RowLimit*, [,INT *AutoCloseMode*]]]])

*Filter:*

A filter expression specifying the records to return during the browse. An empty string indicates that all records will be returned. Where a fieldname is not specified in the filter, it is assumed to be tagname. For example, the filter "AAA" is equivalent to "TAG=AAA".

*Fields:*

Specifies via a comma delimited string the columns to be returned during the browse. An empty string indicates that the server will return all available columns. Supported fields are:

ACKDATE, ACKDATEEXT, ACKMILLI, ACKTIME, ACKUTC, ACQDESC, ACQERROR, ALARMTYPE, ALMCOMMENT, AREA, ARR\_SIZE, CATEGORY, CAUSE1...8, CLASSIFICATION, CLUSTER, COMMENT, CONSEQUENCE1...8,

CUSTOM1...8, DATE, DATEEXT, DEADBAND, DELAY, DELTAMILLI, DELTATIME, DESC, DEVDELAY, DEVIATION, DISABLECOMMENT, DISABLEDDATE, DISABLEDTIME, DISABLEENDDATE, DISABLEENDDATEEXT, DISABLEENDTIME, ENG\_ZERO, ERRDESC,ERRPAGE, EQUIPMENT, FORMAT, FULLNAME, GROUP, HDELAY, HELP, HHDELAY, HIGH, HIGHHIGH, HISTORIAN, ITEM, LDELAY, LLDELAY, LOCALTIMEDATE, LOGSTATE, LOW, LOWLOW, MESSAGE, MILLISEC, NAME, NATIVE\_COMMENT, NATIVE\_DESC, NATIVE\_NAME, NATIVE\_SUMDESC, OFFDATE, OFFDATEEXT, OFFMILLI, OFFTIME, OFFTIMEDATE, OFFUTC, OLD\_DESC, ONDATE, ONDATEEXT, ONMILLI, ONTIME, ONTIMEDATE, ONUTC, PAGING, PAGINGGROUP, PRIORITY, PRIV, PSI\_TYPE, RATE, RECEIPTLOCALTIMEDATE, RECEIPTDATE, RECEIPTDATEEXT, RECEIPTMILLISEC, RECEIPTTIME, RECEIPTTIMEINT, RECEIPTTIMETICKS, RESPONSE1...8, RESPONENUM, SETPOINT, STATE, STATE\_DESC, STATE\_DESCO...7, SUMDESC, SUMSTATE, SUMTYPE, TAG, TAGEX, TAGGENLINK, TIME, TIMEDATE, TIMEINT, TIMETICKS, TSQUALITY, TYPE, TYPENUM, USERDESC, USERNAME, USERLOCATION, VALUE.

See [Browse Function Field Reference](#) for information about fields.

#### *Clusters:*

An optional parameter that specifies via a comma delimited string the subset of the clusters to browse. An empty string indicates that the connected clusters will be browsed.

#### *RowLimit:*

The default value of iRowLimit is -1. This will cause the browse session to use the default global maximum row limit (configured by the INI parameter [\[Alarm\]BrowseRowLimit](#)), or the default value of the INI parameter if not specified.

Specifying a small iRowLimit can reduce the round-trip time and the memory usage of AlmSummaryOpen Cicode function in a system with large alarm summary history.

#### *AutoCloseMode*

Optional parameter to close session at page navigation.

0 - Will not automatically close the browsing session. Use AlmSummaryClose to close the session manually.

1 - The browsing session will be closed when the page is changed or otherwise closed.

## Return Value

Returns an integer handle to the browse session. Returns -1 on error.

## Related Functions

[AlmSummaryAck](#), [AlmSummaryClear](#), [AlmSummaryClose](#), [AlmSummaryDelete](#), [AlmSummaryDeleteAll](#), [AlmSummaryDisable](#), [AlmSummaryEnable](#), [AlmSummaryFirst](#), [AlmSummaryGetField](#), [AlmSummaryLast](#), [AlmSummaryNext](#), [AlmSummaryPrev](#), [AlmSummaryNumRecords](#)

## Example

```
Example 1 - Specifying the fields
INT iSession;
...
iSession = AlmSummaryOpen("NAME=ABC*", "NAME,TYPE",
"ClusterA,ClusterB");
IF iSession <> -1 THEN
// Successful case
```

```
ELSE
// Function returned an error
END
...
Example 2 - Specifying the time range for multiple incremental browse sessions
INT session;
STRING t0 ;
STRING t1;
// Below is an example of browsing alarm summary records incrementally in small time ranges
// of 20 minutes interval over an hour from 10:00am TO 11:00am on 9th March 2015 local
time.
// 10:00 ~ 10:20
t0 = IntToStr(TimestampToInt(TimestampCreate(2015,03,09,10,00,00,000)));
t1 = IntToStr(TimestampToInt(TimestampCreate(2015,03,09,10,20,00,000)));
session = AlmSummaryOpen("OnTime >= " + t0 + " AND OnTime < " + t1, "");
IF (session >= 0) THEN
    AlmSummaryFirst(session);
    // Do something with the browse session
    // ...
    AlmSummaryClose(session);
END

// 10:20 ~ 10:40
t0 = IntToStr(TimestampToInt(TimestampCreate(2015,03,09,10,20,00,000)));
t1 = IntToStr(TimestampToInt(TimestampCreate(2015,03,09,10,40,00,000)));
session = AlmSummaryOpen("OnTime >= " + t0 + " AND OnTime < " + t1, "");
IF (session >= 0) THEN
    AlmSummaryFirst(session);
    // Do something with the browse session
    // ...
    AlmSummaryClose(session);
END

// 10:40 ~ 11:00
t0 = IntToStr(TimestampToInt(TimestampCreate(2015,03,09,10,40,00,000)));
t1 = IntToStr(TimestampToInt(TimestampCreate(2015,03,09,11,00,00,000)));
session = AlmSummaryOpen("OnTime >= " + t0 + " AND OnTime < " + t1, "");
IF (session >= 0) THEN
    AlmSummaryFirst(session);
    // Do something with the browse session
    // ...
    AlmSummaryClose(session);
END
...

Example 3 - Specify RowLimit for multiple incremental browse sessions
INT session;
INT rowLimit = 500;
STRING continuationFilter;
REAL ts;
STRING lastOnTime;

// Below is an example of browsing alarm summary records incrementally in small row limit
// of 50
// The OnTime and Tag of the last record is used as the continuation for the next
incremental session.
```

```
// First session
session = AlmSummaryOpen("", "", "Cluster1", rowLimit);
IF (session < 0) THEN
    RETURN;
END

AlmSummaryFirst(session);
// Do something with the browse session, and call AlmSummaryNext(session) to iterate
// ...

// Capture the timestamp of the last record for browse continuation with a next session.
AlmSummaryLast(session);
ts = (StrToReal(AlmSummaryGetField(session, "OnMilli")) / 1000) +
    StrToDate(AlmSummaryGetField(session, "OnDate")) +
    StrToTime(AlmSummaryGetField(session, "OnTime"));
lastOnTime = RealToStr(ts, 0, 3);
AlmSummaryClose(session);

// Continue browsing using small row limit.
continuationFilter = "OnTime > " + lastOnTime;
session = AlmSummaryOpen(continuationFilter, "", "Cluster1", rowLimit);
IF (session < 0) THEN
    RETURN;
END

IF AlmSummaryFirst(session) <> 0 THEN
    AlmSummaryClose(session);
    RETURN;
END

// Do something with the browse session, and call AlmSummaryNext(session) to iterate
// ...

// Capture the timestamp of the last record for browse continuation with a next session.
AlmSummaryLast(session);
ts = (StrToReal(AlmSummaryGetField(session, "OnMilli")) / 1000) +
    StrToDate(AlmSummaryGetField(session, "OnDate")) +
    StrToTime(AlmSummaryGetField(session, "OnTime"));
lastOnTime = RealToStr(ts, 0, 3);
AlmSummaryClose(session);
...
```

## See Also

[Alarm Functions](#)

### AlmSummaryPrev

The AlmSummaryPrev function moves the data browse cursor back one record. If you call this function after you have reached the beginning of a summary, error 412 is returned (Databrowse session EOF).

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT AlmSummaryPrev(LONG Session)**

*iSession:*

The handle to a browse session previously returned by a AlmSummaryOpen call.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmSummaryAck](#), [AlmSummaryClear](#), [AlmSummaryClose](#), [AlmSummaryDelete](#), [AlmSummaryDeleteAll](#),  
[AlmSummaryDisable](#), [AlmSummaryEnable](#), [AlmSummaryFirst](#), [AlmSummaryGetField](#), [AlmSummaryLast](#),  
[AlmSummaryNext](#), [AlmSummaryOpen](#), [AlmSummaryNumRecords](#)

## See Also

[Alarm Functions](#)

## AlmSummarySetFieldValue

The AlmSummarySetFieldValue function sets a new value for the specified field for the record the data browse cursor is currently referencing. The value is not committed until a call to AlmSummaryCommit is made.

The new values are discarded if the cursor is moved away from the current alarm summary entry before AlmSummaryCommit is made in the data browse session

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

---

**Note:** This function will only work while the record is writable which is controlled by the [\[Alarm\]ArchiveAfter](#) parameter. Refer to the topic [Configure the Archiving Parameters](#) for more information.

---

## Syntax

**LONG AlmSummarySetFieldValue(*iSession*, *sFieldname*, *sFieldValue*)**

*iSession:*

The handle to a browse session previously returned by a AlmSummaryOpen call.

*sFieldName:*

The name of the field whose value is to be updated. Supported fields are:

AckTime	Alarm acknowledged time
Comment	Alarm Comment
OffMilli (for time stamped alarms only)	Alarm millisecond off time

OffTime	Alarm OFF time
---------	----------------

See [Browse Function Field Reference](#) for additional information about fields.

*sFieldValue:*

The field value to update.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmSummaryAck](#), [AlmSummaryClear](#), [AlmSummaryClose](#), [AlmSummaryCommit](#), [AlmSummaryDelete](#), [AlmSummaryDeleteAll](#), [AlmSummaryDisable](#), [AlmSummaryEnable](#), [AlmSummaryFirst](#), [AlmSummaryGetField](#), [AlmSummaryLast](#), [AlmSummaryNext](#), [AlmSummaryOpen](#), [AlmSummaryNumRecords](#)

## Example

```
STRING sFieldValue = "NEW_COMMENT";
STRING sFieldName = "COMMENT";
INT errorCode = 0;
...
errorCode = AlmSummarySetFieldValue(iSession, sFieldName,
    sFieldValue);
IF errorCode = 0 THEN
    // Successful case
ELSE
    // Function returned an error
END
...
```

## See Also

[Alarm Functions](#)

## AlmTagsAck

---

**Note:** This command is now deprecated. Use [AlmBrowseAck](#) function instead.

The AlmTagsAck function acknowledges the alarm tag at the current cursor position in an active data browse session.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

INT **AlmTagsAck**(*iSession*)

*iSession:*

The handle to a browse session previously returned by a AlmTagsOpen call.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmTagsClear](#), [AlmTagsDisable](#), [AlmTagsEnable](#), [AlmTagsClose](#), [AlmTagsFirst](#), [AlmTagsGetField](#), [AlmTagsNext](#), [AlmTagsNumRecords](#), [AlmTagsOpen](#), [AlmTagsPrev](#)

## AlmTagsClear

---

**Note:** This command is now deprecated.

The AlmTagsClear function clears the alarm tag at the current cursor position in an active data browse session. This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

INT **AlmTagsClear**(*iSession*)

*iSession*:

The handle to a browse session previously returned by a AlmTagsOpen call.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmTagsAck](#), [AlmTagsDisable](#), [AlmTagsEnable](#), [AlmTagsClose](#), [AlmTagsFirst](#), [AlmTagsGetField](#), [AlmTagsNext](#), [AlmTagsNumRecords](#), [AlmTagsOpen](#), [AlmTagsPrev](#)

## See Also

[Alarm Functions](#)

## AlmTagsClose

---

**Note:** This command is now deprecated. Use [AlmBrowseClose](#) function instead.

The AlmTagsClose function terminates an active data browse session and cleans up all resources associated with the session.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

INT AlmTagsClose(*iSession*)

*iSession*:

The handle to a browse session previously returned by a AlmTagsOpen call.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmTagsAck](#), [AlmTagsClear](#), [AlmTagsDisable](#), [AlmTagsEnable](#), [AlmTagsFirst](#), [AlmTagsGetField](#),  
[AlmTagsNumRecords](#), [AlmTagsOpen](#), [AlmTagsPrev](#)

## See Also

[Alarm Functions](#)

## AlmTagsDisable

---

**Note:** This command is now deprecated. Use [AlmBrowseDisable](#) function instead.

The AlmTagsDisable function disables the alarm tag at the current cursor position in an active data browse session.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

INT AlmTagsDisable([INT *iSession* [, INT *EndTime* [, STRING *Comment*] ] )

*iSession*:

The handle to a browse session previously returned by a AlmTagsOpen call.

*EndTime*:

A date/time variable that indicates when the alarm will no longer be disabled. If this parameter is omitted or set to 0, the alarm will be disabled indefinitely.

*Comment*:

An optional comment limited to 200 characters explaining why the alarm is disabled. If the comment exceeds 200 characters, hardware error 274 ("Invalid argument passed") will be displayed.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmTagsAck](#), [AlmTagsClear](#), [AlmTagsDisable](#), [AlmTagsClose](#), [AlmTagsFirst](#), [AlmTagsGetField](#), [AlmTagsNext](#),  
[AlmTagsNumRecords](#), [AlmTagsOpen](#), [AlmTagsPrev](#)

## See Also

[Alarm Functions](#)

### AlmTagsEnable

---

**Note:** This command is now deprecated. Use [AlmBrowseEnable](#) function instead.

---

The AlmTagsEnable function enables the alarm tag at the current cursor position in an active data browse session.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

INT **AlmTagsEnable**(INT *iSession* [, INT *bAcknowledge*])

*iSession*:

The handle to a browse session previously returned by a AlmTagsOpen call.

*bAcknowledge*:

Forces acknowledgment of an alarm after it is enabled. The accepted values are:

0 — (default) enforced acknowledgment will not be applied.

1 — alarm will be acknowledged when enabled.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmTagsAck](#), [AlmTagsClear](#), [AlmTagsDisable](#), [AlmTagsClose](#), [AlmTagsFirst](#), [AlmTagsGetField](#), [AlmTagsNext](#),  
[AlmTagsNumRecords](#), [AlmTagsOpen](#), [AlmTagsPrev](#)

## See Also

[Alarm Functions](#)

### AlmTagsFirst

---

**Note:** This command is now deprecated. Use [AlmBrowseFirst](#) function instead.

---

The AlmTagsFirst function places the data browse cursor at the first record.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT AlmTagsFirst(*iSession*)**

*iSession*:

The handle to a browse session previously returned by a AlmTagsOpen call.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmTagsAck](#), [AlmTagsClear](#), [AlmTagsDisable](#), [AlmTagsClose](#), [AlmTagsEnable](#), [AlmTagsGetField](#),  
[AlmTagsNumRecords](#), [AlmTagsOpen](#), [AlmTagsPrev](#)

## See Also

[Alarm Functions](#)

## AlmTagsGetField

---

**Note:** This command is now deprecated. Use [AlmBrowseGetField](#) function instead.

---

The AlmTagsGetField function retrieves the value of the specified field from the record the data browse cursor is currently referencing.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

**STRING AlmTagsGetField(*iSession*, *sFieldName*)**

*iSession*:

The handle to a browse session previously returned by a AlmTagsOpen call.

*sFieldName*:

The name of the field that references the value to be returned. Supported fields are:

ACKDATE, ACKDATEEXT, ACKTIME, ACQERROR, ALARMTYPE, ALMCOMMENT, AREA, CATEGORY, CAUSE1, CAUSE2, CAUSE3, CAUSE4, CAUSE5, CAUSE6, CAUSE7, CAUSE8, CLUSTER, COMMENT, CONSEQUENCE1, CONSEQUENCE2, CONSEQUENCE3, CONSEQUENCE4, CONSEQUENCE5, CONSEQUENCE6, CONSEQUENCE7, CONSEQUENCE8, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8, DATE, DATEEXT, DEADBAND, DELTATIME, DESC, DEVIATION, DISABLEENDDATE, DISABLEENDDATEEXT, DISABLEENDTIME, DISABLECOMMENT, ERRDESC, ERRPAGE, EQUIPMENT, FORMAT, GROUP, HELP, HIGH, HIGHHIGH, LEVEL, LOCALTIMEDATE, LOGSTATE, LOW, LOWLOW, MILLISEC, NAME, NATIVE\_COMMENT, NATIVE\_DESC, NATIVE\_NAME, NATIVE\_SUMDESC, OFFDATE, OFFDATEEXT, OFFMILLI, OFFTIME, OFFTIMEDATE, OLD\_DESC, ONDATE, ONDATEEXT, ONMILLI, ONTIME, ONTIMEDATE, PAGING, PAGINGGROUP, PRIORITY, PRIV, QUALITY, RATE,

RECEIPTLOCALTIMEDATE, RECEIPTDATE, RESPONSE1, RESPONSE2, RESPONSE3, RESPONSE4, RESPONSE5, RESPONSE6, RESPONSE7, RESPONSE8, RESPONSENUM, RESPTIME1, RESPTIME2, RESPTIME3, RESPTIME4, RESPTIME5, RESPTIME6, RESPTIME7, RESPTIME8, STATE, STATE\_DESC, STATE\_DESC0, STATE\_DESC1, STATE\_DESC2, STATE\_DESC3, STATE\_DESC4, STATE\_DESC5, STATE\_DESC6, STATE\_DESC7, SUMDESC, SUMSTATE, SUMTYPE, TAG, TAGEX, TIME, TIMEDATE, TYPE, TYPENUM, USERNAME, VALUE.

See [Browse Function Field Reference](#) for information about fields.

## Return Value

The value of the specified field as a string. An empty string may or may not be an indication that an error has been detected. The last error should be checked in this instance to determine if an error has actually occurred.

## Related Functions

[AlmTagsAck](#), [AlmTagsClear](#), [AlmTagsDisable](#), [AlmTagsEnable](#), [AlmTagsClose](#), [AlmTagsFirst](#), [AlmTagsNext](#), [AlmTagsNumRecords](#), [AlmTagsOpen](#),

## Example

```
STRING fieldValue = "";
STRING fieldName = "TYPE";
INT errorCode = 0;
...
fieldValue = AlmTagsGetField(iSession, sFieldName);
IF fieldValue <> "" THEN
    // Successful case
ELSE
    // Function returned an error
END
...
```

## See Also

[Alarm Functions](#)

## AlmTagsNext

---

**Note:** This command is now deprecated. Use [AlmBrowseNext](#) function instead.

The AlmTagsNext function moves the data browse cursor forward one record. If you call this function after you have reached the end of the records, error 412 is returned (Databrowse session EOF).

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **AlmTagsNext**(*iSession*)

*iSession*:

The handle to a browse session previously returned by a AlmTagsOpen call.

## Return Value

0 (zero) if the browse has successfully been moved to the next record, otherwise an error code is returned.

## Related Functions

[AlmTagsAck](#), [AlmTagsClear](#), [AlmTagsEnable](#), [AlmTagsClose](#), [AlmTagsFirst](#), [AlmTagsGetField](#), [AlmTagsNumRecords](#),  
[AlmTagsOpen](#), [AlmTagsPrev](#)

## See Also

[Alarm Functions](#)

## AlmTagsNumRecords

---

**Note:** This command is now deprecated. Use [AlmBrowseNumRecords](#) function instead.

---

The AlmTagsNumRecords function returns the number of records that match the filter criteria.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

LONG **AlmTagsNumRecords**(*iSession*)

*iSession*:

The handle to a browse session previously returned by a AlmTagsOpen call.

## Return Value

The number of records that have matched the filter criteria. A value of 0 denotes that no records have matched. A value of -1 denotes that the browse session is unable to provide a fixed number. This may be the case if the data being browsed changed during the browse session.

## Related Functions

[AlmTagsAck](#), [AlmTagsClear](#), [AlmTagsDisable](#), [AlmTagsEnable](#), [AlmTagsClose](#), [AlmTagsFirst](#), [AlmTagsGetField](#),  
[AlmTagsNext](#), [AlmTagsOpen](#), [AlmTagsPrev](#)

## Example

```
INT numRecords = 0;  
...  
numRecords = AlmTagsNumRecords(iSession);  
IF numRecords <> 0 THEN  
    // Have records
```

```
ELSE
    // No records
END
...
```

## AlmTagsOpen

**Note:** This command is now deprecated. Use [AlmBrowseOpen](#) function instead.

The AlmTagsOpen function initiates a new browse session and returns a handle to the new session that can be used in subsequent data browse function calls.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

**Note:** After calling AlmTagsOpen() it is necessary to call AlmTagsFirst() in order to place the cursor at the beginning of the browse session, otherwise a hardware alarm is invoked.

## Syntax

```
LONG AlmTagsOpen(sFilter, sFields [, sClusters] )
```

*sFilter*:

A filter expression specifying the records to return during the browse. An empty string indicates that all records will be returned. Where a fieldname is not specified in the filter, it is assumed to be tagname. For example, the filter "AAA" is equivalent to "Tag=AAA".

*sFields*:

Specifies via a comma delimited string the columns to be returned during the browse. An empty string indicates that the server will return all available columns. Supported fields are:

ACKDATE, ACKDATEEXT, ACKTIME, ACQERROR, ALARMTYPE, ALMCOMMENT, AREA, CATEGORY, CLUSTER, COMMENT, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8, DATE, DATEEXT, DEADBAND, DELTATIME, DESC, DEVIATION, DISABLEENDTIME, ERRDESC, ERPPAGE, EQUIPMENT, FORMAT, GROUP, HELP, HIGH, HIGHHIGH, ITEM, LEVEL, LOCALTIMEDATE, LOGSTATE, LOW, LOWLOW, MILLISEC, NAME, NATIVE\_COMMENT, NATIVE\_DESC, NATIVE\_NAME, NATIVE\_SUMDESC, OFFDATE, OFFDATEEXT, OFFMILLI, OFFTIME, OFFTIMEDATE, OLD\_DESC, ONDATE, ONDATEEXT, ONMILLI, ONTIME, ONTIMEDATE, PAGING, PAGINGGROUP, PRIORITY, PRIV, QUALITY, RATE, RECEIPTLOCALTIMEDATE, RECEIPTDATE, STATE, STATE\_DESC, STATE\_DESC0, STATE\_DESC1, STATE\_DESC2, STATE\_DESC3, STATE\_DESC4, STATE\_DESC5, STATE\_DESC6, STATE\_DESC7, SUMDESC, SUMSTATE, SUMTYPE, TAG, TAGEX, TIME, TIMEDATE, TYPE, TYPENUM, USERNAME, VALUE.

See [Browse Function Field Reference](#) for information about fields.

*sClusters*:

An optional parameter that specifies via a comma delimited string the subset of the clusters to browse. An empty string indicates that the connected clusters will be browsed.

## Return Value

Returns an integer handle to the browse session. Returns -1 when an error is detected.

The returned entries will be ordered alphabetically by name.

## Related Functions

[AlmTagsAck](#), [AlmTagsClear](#), [AlmTagsDisable](#), [AlmTagsEnable](#), [AlmTagsClose](#), [AlmTagsFirst](#), [AlmTagsGetField](#), [AlmTagsNext](#), [AlmTagsNumRecords](#), [AlmTagsPrev](#)

## Example

```
INT iSession;  
...  
iSession = AlmTagsOpen("NAME=ABC*", "NAME,TYPE",  
"ClusterA,ClusterB");  
IF iSession <> -1 THEN  
// Successful case  
ELSE  
// Function returned an error  
END  
...
```

## See Also

[Alarm Functions](#)

## AlmTagsPrev

---

**Note:** This command is now deprecated. Use [AlmBrowsePrev](#) function instead.

The AlmTagsPrev function moves the data browse cursor back one record. If you call this function after you have reached the beginning of the records, error 412 is returned (Databrowse session EOF).

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **AlmTagsPrev**(*iSession*)

*iSession*:

The handle to a browse session previously returned by a AlmTagsOpen call.

## Return Value

0 (zero) if the alarm browse session exists, otherwise an error code is returned.

## Related Functions

[AlmTagsAck](#), [AlmTagsClear](#), [AlmTagsDisable](#), [AlmTagsEnable](#), [AlmTagsClose](#), [AlmTagsFirst](#), [AlmTagsGetField](#), [AlmTagsNext](#), [AlmTagsNumRecords](#), [AlmTagsOpen](#)

## See Also

[Alarm Functions](#)

### HwAlarmQue

Returns the handle of the hardware alarm queue. The Alarms Server writes hardware alarm information into this queue as each hardware alarm occurs. To read events from this queue, use the QueRead or QuePeek functions. The data written into the queue is the hardware alarm format, and is stored in the Str field.

To use this function, you need to enable the hardware alarm queue by specifying the [\[Alarm\]HwAlarmQueMax](#) parameter. This parameter specifies the maximum length that the queue can grow to. The [\[Alarm\]HwAlarmFmt](#) parameter defines the format of the data placed into the string field. If HwAlarmFmt is not specified then the format defaults to "Time: {Time,12} Date:{Date,11} Desc:{Desc,40}" .

The following format fields are relevant to hardware alarms:

- {Tag,n}
- {TagEx,n}
- {Cluster,n}
- {Name,n}
- {State,n}
- {Time,n}
- {Date,n}
- {Desc,n}
- {ErrDesc,n}
- {ErrPage,n}

For a description of the fields see the "Alarm Display Fields" help page.

The number of buffers available for user queues is controlled by the [\[Code\]Queue](#) parameter. Each entry in any user queue consumes one buffer. When all buffers have been used the alarms server will not be able to add new hardware alarms to the queue, and the error message "Out Of Buffers Usr.Que" will be written to syslog.dat.

---

**Note:** When similar hardware alarms are triggered, for example "Tag not found" alarms, the hardware alarm page and hardware alarm queue show the last invalid entry. The previous entry of the same description is overwritten.

---

## Syntax

**HwAlarmQue( )**

## Return Value

The handle of the hardware alarm queue, or -1 if the queue cannot be opened.

## Related Functions

[QueRead](#), [QuePeek](#)

## Example

```
hQue = HwAlarmQue()
WHILE TRUE DO
    QueRead(hQue, nAlarmType, sHwAlarmString, 1);
    /* do whatever with the alarm information */
    ....
    Sleep(0);
END
```

## See Also

[Alarm Functions](#)

## Alarm Filter Functions

The following Cicode functions relate to alarm filters.

<a href="#">AlarmFilterClose</a>	Removes the session from memory.
<a href="#">AlarmFilterEditAppend</a>	Appends the provided expression to the current filter session content without any validation.
<a href="#">AlarmFilterEditAppendEquipment</a>	Appends the provided expression that includes equipment names or category to the current filter session content without any validation.
<a href="#">AlarmFilterEditClose</a>	Removes the session from the memory.
<a href="#">AlarmFilterEditCommit</a>	Validates the filter built in this session and, if valid, applies the filter to the list associated with the session.
<a href="#">AlarmFilterEditFirst</a>	Retrieves the first part of the filter.
<a href="#">AlarmFilterEditLast</a>	Retrieves the last part of the filter.
<a href="#">AlarmFilterEditNext</a>	Retrieves the next part of the filter.
<a href="#">AlarmFilterEditOpen</a>	Creates a session for the historical list associated with the provided animation number (aN) or FilterName.
<a href="#">AlarmFilterEditPrev</a>	Retrieves the previous part of the filter.
<a href="#">AlarmFilterEditSet</a>	It replaces the current filter session content by the

	provided expression without any validation.
<a href="#">AlarmFilterForm</a>	Available when using the Legacy Filter Form. Use to display an alarm form to specify filter criteria for an alarm list or a named filter.
<a href="#">AlarmFilterOpen</a>	Creates a new named filter.
<a href="#">AlarmGetFilterName</a>	Retrieves the name of the filter for the AN.
<a href="#">AlarmResetQuery</a>	Use when using the AlarmFilterForm(). Clears the filter of the specified filter source.
<a href="#">LibAlarmFilterForm</a>	Displays a generic alarm filter popup for specifying filter criteria for either an alarm list or named filter. Only available if 'Lib_controls' project is included in main project.

**Note:** If a requested filter is too complex (for example, it contains too many conditions or too many nested brackets), the filter is cleared (no filter is used). The hardware alarm "Too many alarms in filters" is generated on the client components, and a tracelog error message is logged.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## AlarmFilterClose

This function removes the named filter from memory.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT AlarmFilterClose(STRING *FilterName*)**

*FilterName*

Name of the filter

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Example

```
// This example shows how to open and close the named filter "Myfilter"
// nothing of interest is done with the filter in this example.
INT nOpenModeOld = 0;
```

```
INT nOpenModeNew = 1;
INT nOpenModeAny = 2;
INT nCloseModeManual = 0;
INT nCloseModePageChanged = 1;
INT nError;
nError = AlarmFilterOpen("MyFilter", nOpenModeNew, nCloseModeManual);
IF nError = 0 THEN
    nError = AlarmFilterClose("MyFilter");
END
```

## Related Functions

[AlarmFilterOpen](#)

## See Also

[Alarm Filter Functions](#)

### AlarmFilterEditAppend

The **AlarmFilterEditAppend** function takes a session handle and a filter expression as parameters. It appends the provided expression to the current filter session content without any validation. This does not apply to all filters on the list (see [AlarmFilterEditCommit](#)).

## Syntax

```
INT AlarmFilterEditAppend(INT hSession, STRING FilterCriteria)
```

*hSession*:

Session handle for the historical list previously returned by the function [AlarmFilterEditOpen](#).

*FilterCriteria*:

Filter expression as a string. For example: "(Tag=A) OR (TAG=B)"

See the topic [Implementing Alarm Filters Using Cicode](#) for more information about filter syntax.

---

**Note:** If a requested filter is too complex (for example, it contains too many conditions or too many nested brackets), the filter is cleared (no filter is used). The hardware alarm "Too many alarms in filters" is generated on the client components, and a tracelog error message is logged.

---

## Return Value

0 (zero) if the alarm filter session exists, otherwise an error code is returned.

## Example

```
// This example shows how to update an edit session.
// This example requires that the edit session hEdit exists.
// This example shows how you would split the parts of the filter
// to avoid an overflow error when handling strings.
INT nError;
```

```
nError = AlarmFilterEditSet(hEdit, "Tag");
nError = AlarmFilterEditAppend(hEdit, "=");
nError = AlarmFilterEditAppend(hEdit, "Dig*");
nError = AlarmFilterEditCommit(hEdit);
sRet = AlarmFilterEditFirst(iHndl); // Tag=Dig*;
```

## Related Functions

[AlarmFilterEditSet](#)

## See Also

[Alarm Filter Functions](#)

### AlarmFilterEditAppendEquipment

Appends the provided expression that can include equipment names to the current filter session content without validation.

## Syntax

**INT AlarmFilterEditAppendEquipment(INT *hSession*, STRING *EquipmentFilter*, STRING *Ref*)**

*hSession*:

Session handle for the historical list previously returned by the function [AlarmFilterEditOpen](#).

*EquipmentFilter*:

A comma-separated list of equipment names to filter alarms prior to applying the other filter specified on the FilterCriteria argument. This field has been extended to support wildcards, and hence accept partial filter strings.

Expression has been extended to allow:

"EQUIPMENT=XYZ" (this may have an asterisk as the last character, this is interpreted as a wildcard)

What this means is:

- "XYZ" becomes "EQUIPMENT=XYZ\*"
- "EQUIPMENT=XYZ" stays as "EQUIPMENT=XYZ"

---

**Note:** Filters that take the form "XYZ" should no longer be used. It is recommended you use the "EQUIPMENT=XYZ" form, further "XYZ" can now be expressed as "EQUIPMENT=XYZ,EQUIPMENT=XYZ.\*" as this will include all items on the "XYZ" equipment and all items on sub-equipment of the "XYZ" equipment.

*Ref*:

0 - Do not include alarms belonging to referenced equipment in Filter

1 - Include alarms belonging to referenced equipment in Filter

## Return Value

0 (zero) if the alarm filter session exists, otherwise an error code is returned.

## Example

```
// This example shows how to update an edit session.  
// This example requires that the edit session hEdit exists.  
// This example assumes that the ExampleEquipment has references to  
// RefEquipment.RefItem1 and RefEquipment.RefItem2.  
INT nError;  
nError = AlarmFilterEditSet(hEdit, "");  
nError = AlarmFilterEditAppendEquipment(hEdit, "EQUIPMENT=ExampleEquipment", 1);  
nError = AlarmFilterEditCommit(hEdit);  
sRet = AlarmFilterEditFirst(hEdit); // "((Cluster=Cluster1 AND Equipment=ExampleEquipment)  
OR (Cluster=Cluster1 AND Equipment=RefEquipment AND  
(ITEM=RefItem1 OR ITEM=RefItem2))"
```

## Related Functions

[AlarmFilterEditSet](#)

## See Also

[Alarm Filter Functions](#)

## AlarmFilterEditClose

The AlmFilterEditClose function removes the session from the memory. The filter is not reset and is valid until a new filter is created and applied.

## Syntax

INT **AlarmFilterEditClose**(INT *hSession*)

*Session*

Session handle for the historical list previously returned by the function [AlarmFilterEditOpen](#).

## Return Value

0 (zero) if the alarm filter session exists, otherwise an error code is returned.

## Example

```
iHndl = AlarmFilterEditOpen(iAN);  
iRet = AlarmFilterEditSet(iHndl, "Tag=Dig*;Category=1;Area=1;");  
iRet = AlarmFilterEditAppend(iHndl, "Priority<20");  
sRet = AlarmFilterEditFirst(iHndl); // Tag=Dig*;  
sRet = AlarmFilterEditNext(iHndl); // Category=1;  
sRet = AlarmFilterEditLast(iHndl); // Priority<20;  
sRet = AlarmFilterEditPrev(iHndl); // Area=1;  
iRet = AlarmFilterEditClose(iHndl);
```

## Related Functions

[AlarmFilterEditOpen](#)

## See Also

[Alarm Filter Functions](#)

### AlarmFilterEditCommit

The AlarmFilterEditCommit function takes a session handle as parameter. It validate the filter created in this session and, if valid, applies this filter to the list associated with the session.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT AlarmFilterEditCommit(INT *hSession*)**

*hSession*:

Session handle for the historical list previously returned by the function [AlarmFilterEditOpen](#).

## Return Value

0 (zero) if the alarm filter session exists, otherwise an error code is returned.

## Example

```
// This example requires that the edit session hEdit exists.  
INT nError;  
nError = AlarmFilterEditSet(hEdit,"tag=Dig*;Category=1;");  
nError = AlarmFilterEditCommit(hEdit);
```

## Related Functions

[AlarmFilterEditSet](#), [AlarmFilterEditAppend](#)

## See Also

[Alarm Filter Functions](#)

### AlarmFilterEditFirst

This function takes a session handle parameter. It gets the first part of the filter. Each part is either:

- A filter expression delimited with ";"
- A partial filter expression truncated at 254 character (if no ";" found before)

## Syntax

```
STRING AlarmFilterEditFirst(INT hSession)
```

*hSession*

Session handle for the historical list previously returned by the function [AlarmFilterEditOpen](#).

## Return Value

First part of filter or if does not exist an empty string "".

## Example

```
iHndl = AlarmFilterEditOpen(iAN);
iRet = AlarmFilterEditSet(iHndl,"Tag=Dig*;Category=1;Area=1;");
iRet = AlarmFilterEditAppend(iHndl, "Priority<20");
sRet = AlarmFilterEditFirst(iHndl); // Tag=Dig*;
sRet = AlarmFilterEditNext(iHndl); // Category=1;
sRet = AlarmFilterEditLast(iHndl); // Priority<20;
sRet = AlarmFilterEditPrev(iHndl); // Area=1;
iRet = AlarmFilterEditClose(iHndl);
```

## Related Functions

[AlarmFilterEditNext](#), [AlarmFilterEditPrev](#)

## See Also

[Alarm Filter Functions](#)

## AlarmFilterEditLast

This function takes a session handle parameter. It gets the last part of the filter. Each part is either:

- A filter expression delimited with ";"
- A partial filter expression truncated at 254 character (if no ";" found before).

## Syntax

```
STRING AlarmFilterEditLast(hSession)
```

*hSession*

Session handle for the historical list previously returned by the function [AlarmFilterEditOpen](#).

## Return Value

First part of filter or if does not exist an empty string "".

## Example

```
iHndl = AlarmFilterEditOpen(iAN);
iRet = AlarmFilterEditSet(iHndl,"Tag=Dig*;Category=1;Area=1;");
iRet = AlarmFilterEditAppend(iHndl, "Priority<20");
sRet = AlarmFilterEditFirst(iHndl); // Tag=Dig*;
sRet = AlarmFilterEditNext(iHndl); // Category=1;
sRet = AlarmFilterEditLast(iHndl); // Priority<20;
sRet = AlarmFilterEditPrev(iHndl); // Area=1;
iRet = AlarmFilterEditClose(iHndl);
```

## Related Functions

[AlarmFilterEditPrev](#)

## See Also

[Alarm Filter Functions](#)

### AlarmFilterEditNext

This function takes a session handle parameter. It gets the next part of the filter. Each part is either:

- A filter expression delimited with ";"
- A partial filter expression truncated at 254 character (if no ";" found before)

## Syntax

**INT AlarmFilterEditNext(INT *hSession*)**

*hSession*:

Session handle for the historical list previously returned by the function [AlarmFilterEditOpen](#).

## Return Value

Next part of filter or if does not exist an empty string "".

## Example

```
iHndl = AlarmFilterEditOpen(iAN);
iRet = AlarmFilterEditSet(iHndl,"Tag=Dig*;Category=1;Area=1;");
iRet = AlarmFilterEditAppend(iHndl, "Priority<20");
sRet = AlarmFilterEditFirst(iHndl); // Tag=Dig*;
sRet = AlarmFilterEditNext(iHndl); // Category=1;
sRet = AlarmFilterEditLast(iHndl); // Priority<20;
sRet = AlarmFilterEditPrev(iHndl); // Area=1;
iRet = AlarmFilterEditClose(iHndl);
```

## Related Functions

[AlarmFilterEditFirst](#), [AlarmFilterEditPrev](#)

## See Also

[Alarm Filter Functions](#)

### AlarmFilterEditOpen

The AlmFilterEditOpen function creates a session for the historical list (or lists) associated with the provided animation number (AN) or FilterName or all alarm lists displayed on the page via (-1) option. This session is initialised with the current filter applied on the lists.

It returns a session handle which will be used as parameter in all other functions to reference the session or -1 if the parameter is not a valid animation number or if this animation number is not linked to an historical list.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **AlarmFilterEditOpen**(STRING *FilterName* or INT *AN* or INT -1 [, INT *AutoCloseMode*])

*FilterName*

Name of Filter

*AN*

Animation Number, for example 21 or 11

-1

Change the display parameters of all alarm lists displayed on the page

*AutoCloseMode*

Values for AutoCloseMode are bit flags.

The values for the bits are:

- 0 bit - Will not automatically close filter. Use [AlarmFilterEditClose](#).
- 1 bit - When set, the named filter will be closed when the page is changed or otherwise closed.

---

**Note:** All other modes are reserved.

---

## Return Value

Returns a session handle to the filter browse session. Returns -1 when an error is detected.

## Example

```
iHndl = AlarmFilterEditOpen(iAN);
iRet = AlarmFilterEditSet(iHndl,"Tag=Dig*;Category=1;Area=1;");
iRet = AlarmFilterEditAppend(iHndl, "Priority<20");
sRet = AlarmFilterEditFirst(iHndl); // Tag=Dig*;
```

```
sRet = AlarmFilterEditNext(iHndl); // Category=1;
sRet = AlarmFilterEditLast(iHndl); // Priority<20;
sRet = AlarmFilterEditPrev(iHndl); // Area=1;
iRet = AlarmFilterEditClose(iHndl);
```

## Related Functions

[AlarmFilterEditClose](#)

## See Also

[Alarm Filter Functions](#)

### AlarmFilterEditPrev

This function takes a session handle parameter. It gets the previous part of the filter. Each part is either:

- A filter expression delimited with ";"
- A partial filter expression truncated at 254 character (if no ";" found before)

## Syntax

```
STRING AlarmFilterEditPrev(INT hSession)
```

*hSession:*

Session handle for the historical list previously returned by the function [AlarmFilterEditOpen](#).

## Return Value

Previous part of filter or if does not exist an empty string "".

## Example

```
iHndl = AlarmFilterEditOpen(iAN);
iRet = AlarmFilterEditSet(iHndl, "Tag=Dig*;Category=1;Area=1;");
iRet = AlarmFilterEditAppend(iHndl, "Priority<20");
sRet = AlarmFilterEditFirst(iHndl); // Tag=Dig*;
sRet = AlarmFilterEditNext(iHndl); // Category=1;
sRet = AlarmFilterEditLast(iHndl); // Priority<20;
sRet = AlarmFilterEditPrev(iHndl); // Area=1;
iRet = AlarmFilterEditClose(iHndl);
```

## Related Functions

[AlarmFilterEditFirst](#), [AlarmFilterEditNext](#), [AlarmFilterEditLast](#)

## See Also

[Alarm Filter Functions](#)

### AlarmFilterEditSet

The AlarmFilterEditSet function takes a session handle and a filter expression as parameters. It replaces the current filter session content by the provided expression without any validation. This does not apply to all filters on the list (see [AlarmFilterEditCommit](#)).

## Syntax

**INT AlarmFilterEditSet(INT *hSession*, STRING *FilterCriteria*)**

*hSession*:

Session handle for the historical list previously returned by the function [AlarmFilterEditOpen](#).

*FilterCriteria*:

Filter expression as a string. For example:"(Tag=A) OR (TAG=B)"

See the topic [Implementing Alarm Filters Using Cicode](#) for more information about filter syntax.

Expression has been extended to allow:

- "REFCAT=XYZ" (this may have an asterisk as the first and/or last character, these are interpreted as wildcards)
- "PAGE=XYZ" (this is an exact literal match only)

What this means is:

- "PAGE=PageXYZ" becomes "EQUIPMENT=..." that includes all the equipment that have "PageXYZ" in the PAGE field in the Equipment table
- "REFCAT=\*XYZ\*" becomes "EQUIPMENT=..." that includes all the equipment defined in the Equipment References table that matches the supplied pattern and the CATEGORY field in that table. This value MAY contain an asterisk as the first, and or the last character, but the content between any bounding asterisks is treated as literal including an asterisk.

---

**Note:** If a requested filter is too complex (for example, it contains too many conditions or too many nested brackets), the filter is cleared (no filter is used). The hardware alarm "Too many alarms in filters" is generated on the client components, and a tracelog error message is logged.

---

## Return Value

0 (zero) if the alarm filter session exists, otherwise an error code is returned.

## Example

```
iHndl = AlarmFilterEditOpen(iAN);
iRet = AlarmFilterEditSet(iHndl,"Tag=Dig*;Category=1;Area=1;");
iRet = AlarmFilterEditAppend(iHndl, "Priority<20");
sRet = AlarmFilterEditFirst(iHndl); // Tag=Dig*;
sRet = AlarmFilterEditNext(iHndl); // Category=1;
sRet = AlarmFilterEditLast(iHndl); // Priority<20;
sRet = AlarmFilterEditPrev(iHndl); // Area=1;
```

```
iRet = AlarmFilterEditClose(iHndl);
```

## Related Functions

[AlarmFilterEditOpen](#), [AlarmFilterEditClose](#), [AlarmFilterEditAppend](#)

## See Also

[Alarm Filter Functions](#)

### AlarmFilterEditHasField

The `AlarmFilterEditHasField` function checks whether an alarm field name or any field in a set of field names is used in the filter specified in an alarm filter edit session.

## Syntax

```
INT AlarmFilterEditHasField(INT hSession, STRING sFieldName)
```

*Session:*

Session handle for the historical list previously returned by the function [AlarmFilterEditOpen](#).

*sFieldName*

The alarm field name or names separated by comma (no space) to be checked if it is used in filter criteria.

## Return Value

1 (TRUE) if the filter criteria is valid and alarm field name is used, otherwise 0 (FALSE) is returned.

## Example

```
iHndl = AlarmFilterEditOpen(iAN);
iRet = AlarmFilterEditSet(iHndl, "Tag=Dig*;Category=1;Area=1");
iRet = AlarmFilterEditHasField(iHndl, "Tag"); // 1 as Tag field is used
iRet = AlarmFilterEditHasField(iHndl, "Desc"); // 0 as Desc field is not used
iRet = AlarmFilterEditSet(iHndl, "NewCar=Fantastic;");
iRet = AlarmFilterEditHasField(iHndl, "NewCar"); // 0 as filter criteria is invalid
```

## Related Functions

[AlarmFilterEditSet](#)

## See Also

[Alarm Filter Functions](#)

## AlarmFilterForm

Displays a form for specifying filtering criteria for either an alarm list or a named filter. This function uses the AlarmFilterEdit family of Cicode functions to set the filter.

**Note:** If the user passes in the Animation number of an alarm list as its FilterSrc argument, the filter set by this function will override the filter set by function [AlarmSetInfo](#) as both functions modify the internal filter associated with the alarm list.

## Syntax

```
INT AlarmFilterForm(INT nIndex, INT nMode = 0, INT nDebug = 0, STRING nFilterSrc = "-1")
```

*nIndex*

0 - Active Alarm form

1- Summary Alarm form

2 – Disabled alarms

3 – Sequence of events (SOE)

-1 – auto-detection, only works if argument sFilterSrc is AN of an alarm list

*nMode*

0 is for form having datetime as drop list and no shift interface;

1 is for form having datetime as droplets and shift interface;

2 is for form having datetime as field and no shift interface;

3 is for form having datetime as field and shift interface

*nDebug*

For debugging filter values; 1 will write filter parameters to file - filter.txt in the [run] folder

*nFilterSrc*

The source where the filter will be applied to. It can either be the Animation Number (AN) of an alarm list or the name of a filter created via the [AlarmFilterOpen](#) Cicode function. If it is not specified, it is defaulted to -1 which refers to all alarm lists on the current page.

## Return Value

1 if filter has been applied, 0 if filter has been cleared, or -1 if no change is made.

## Related Functions

[AlarmResetQuery](#), [AlarmFilterOpen](#)

**Note:** The Alarm Filter Help is available when using the Legacy Alarm Filter Form at Runtime.

## See Also

[Alarm Filter Functions](#)

## AlarmFilterOpen

This function creates a named filter. The filter is initialized with empty content (matches all alarms). If unable to open the named filter an error code is returned.

See the topic *About Named Filters* in the main Plant SCADA help for more information about filter syntax.

---

**Note:** If a requested filter is too complex (for example, it contains too many conditions or too many nested brackets), the filter is cleared (no filter is used). The hardware alarm "Too many alarms in filters" is generated on the client components, and a tracelog error message is logged.

---

## Syntax

```
INT AlarmFilterOpen(STRING FilterName, INT OpenMode [, INT AutoCloseMode])
```

*FilterName*

Name of the filter.

*OpenMode*

The values for *OpenMode* are:

0 - Open an existing named filter.

1- Create a new named filter.

2- Attempts to open an existing named filter. If the named filter does not exist, a new named filter is created.

*AutoCloseMode*

Values for *AutoCloseMode* are bit flags.

The values for the bits are:

- 0 bit - Will not automatically close filter. Use [AlarmFilterClose](#).

- 1 bit - When set, the named filter will be closed when the page is changed or otherwise closed.

---

**Note:** All other modes are reserved.

---

## Return Value

0 (zero) if the filter was opened or created or an error if unsuccessful.

## Example

```
// This example shows how to open and close the named filter "Myfilter"
// nothing of interest is done with the filter in this example.
INT nOpenModeOld = 0;
INT nOpenModeNew = 1;
INT nOpenModeAny = 2;
INT nCloseModeManual = 0;
INT nCloseModePageChanged = 1;
INT nError;
nError = AlarmFilterOpen("MyFilter", nOpenModeNew, nCloseModeManual);
IF nError = 0 THEN
nError = AlarmFilterClose("MyFilter");
END
```

## Related Functions

[AlarmFilterClose](#), [AlarmSetInfo](#)

## See Also

[Alarm Filter Functions](#)

### AlarmGetFilterName

Retrieves the name of the linked named filter for the supplied An. If empty text, there is currently no linked named filter.

## Syntax

STRING **AlarmGetFilterName**(INT *An*)

*An*

Animation number

## Return Value

Name of linked filter or " ".

## Example

```
// This example shows how to link, unlink and check the linking of a
// named filter to an alarm list (by its animation number)
// This example requires that the named filter "Myfilter" exists.
STRING sName;
INT nError;
INT nAnimationNumber=21;
INT nSetInfoFilterName=12;
nError = AlarmSetInfo(nAnimationNumber, nSetInfoFilterName, "MyFilter");
IF nError = 0 THEN
    sName = AlarmGetFilterName(nAnimationNumber); //MyFilter"
END
nError = AlarmSetInfo(nAnimationNumber, nSetInfoFilterName, "");
IF nError = 0 THEN
    sName = AlarmGetFilterName(nAnimationNumber); //"
END
```

## Related Functions

[AlarmFilterOpen](#)

## See Also

[Alarm Filter Functions](#)

### AlarmResetQuery

Clears the filter of the specified filter source. Used to reset the filter set up by the Cicode function [AlarmFilterForm](#). This function uses the AlarmFilterEdit family of Cicode functions, to reset the filter.

**Note:** If the user passes in the Animation number of an alarm list as its FitlerSrc argument, the filter set by this function will override the filter set by function [AlarmSetInfo](#) as both functions modify the internal filter associated with the alarm list.

## Syntax

**AlarmResetQuery(STRING FilterSrc)**

*FilterSrc*

The source where the filter will be applied to. It can either be the Animation Number (AN) of an alarm list or the name of a filter created via the [AlarmFilterOpen](#) Cicode function. If it is not specified, it is defaulted to -1 which refers to all alarm lists on the current page.

## Return Value

None

## Related Functions

[AlarmFilterForm](#)

**Note:** The Alarm Filter Help is available when using the Legacy Alarm Filter Form at Runtime.

## See Also

[Alarm Filter Functions](#)

### LibAlarmFilterForm

**Note:** This function is only available if the Lib\_Controls project is included in the user project.

Displays a generic alarm filter pop-up for specifying filtering criteria for either an alarm list or a named filter. This function uses the AlarmFilterEdit family of Cicode functions to set the filter.

**Note:** If the user passes in the Animation number of an alarm list as its FitlerSrc argument, the filter set by this function will override the filter set by function [AlarmSetInfo\(\)](#) as both functions modify the internal filter associated with the alarm list.

## Syntax

`INT LibAlarmFilterForm([INT FormType [, STRING FilterSrc [, STRING Title [, INT Mode [, STRING AppliedCallbackFn]]]]])`

### *FormType*

Type of alarm filter form:

-1 – (Default) auto detect if the FilterSrc is set to a valid alarm list AN

0 – active alarm

1 – alarm summary

2 – disabled alarms

3 – sequence of events (SOE)

### *sFilterSrc*

The source where the filter will be applied to. It can either be the Animation Number (AN) of an alarm list or the name of a filter created via the AlarmFilterOpen() Cicode function. If it is not specified, it is defaulted to -1 which refers to all alarm lists on the current page. If the FilterSrc cannot be opened, it will prompt the user to create the filter source.

### *sTitle*

The title for the filter form. If it is not specified or left blank, the title is determined by the form type:

0 – Active Alarm Filter

1 – Summary Alarm Filter

2 – Disabled Alarm Filter

3 – SOE Filter

Any other value – (<FilterSrc> - Alarm Filter)

### *nMode*

The mode of the form:

0 – (Default) Displays form in synchronous mode. The function waits for the user to close the form before returning. Under this mode, the "Apply" button is not shown on the form.

1 – Display form in asynchronous mode. The function does not wait for the user to close the form and returns immediately. Under this mode, the "Apply" button is shown on the form. Using this button, the user can apply filter to the alarm list(s) multiple times without closing the form.

### *sAppliedCallbackFn*

The callback function statement to be called when the criteria (rules) specified in the form is applied either via the OK or Apply buttons. Like other callback functions available to other library controls genie. The callback function supports keyword substitutions during callback:

#Filter - The FilterSrc specified when showing the form

#Result – The result of the filter application. See return value for details.

## Return Value

1 if filter has been applied, 0 if filter has been cleared, or -1 if no change is made.

## Related Functions

[AlarmResetQuery](#)

**Note:** The Alarm Filter Help is available when using the Legacy Alarm Filter Form at Runtime.

## See Also

[Alarm Filter Functions](#)

## Array Functions

Plant SCADA supports the following array functions:

<a href="#">ArrayCopy</a>	Makes a copy of an array.
<a href="#">ArrayCreate</a>	Creates an array.
<a href="#">ArrayCreateByAn</a>	Creates an array at a specified AN.
<a href="#">ArrayDestroy</a>	Destroys an array.
<a href="#">ArrayDestroyByAn</a>	Destroys an array associated with the specified AN.
<a href="#">ArrayExists</a>	Determines if an array with a specified handle exists.
<a href="#">ArrayExistsByAn</a>	Determines if an array associated with a specified AN exists.
<a href="#">ArrayFillFromAlarmDataByAn</a>	This function is used to fill an array with information from internal alarm record caches.
<a href="#">ArrayGetArrayByAn</a>	Retrieves an array associated with a specified AN.
<a href="#">ArrayGetInfo</a>	Retrieves the size of the x-, y-, or z-dimension for an array.
<a href="#">ArrayGetInt</a>	Retrieves an integer value from an array.
<a href="#">ArrayGetInt</a>	Retrieves an integer value from an array associated with a specified AN.
<a href="#">ArrayGetMapName</a>	Retrieves the name of the map associated with an array.
<a href="#">ArrayGetMapNameByAn</a>	Retrieves the name of the map for an array associated with a specified AN.
<a href="#">ArrayGetString</a>	Retrieves a string from an array.
<a href="#">ArrayGetStringByAn</a>	Retrieves a string from an array associated with a

	specified AN.
<a href="#">ArrayIsDirty</a>	Determines if an array is 'dirty' (information in the array has changed).
<a href="#">ArraySetInt</a>	Sets an integer value within an array.
<a href="#">ArraySetIntByAn</a>	Sets an integer value in an array associated with a specified AN.
<a href="#">ArraySetIsDirty</a>	Allows you to indicate that an array is dirty (meaning the information in the array has changed).
<a href="#">ArraySetString</a>	Sets a string value in an array.
<a href="#">ArraySetStringByAn</a>	Sets a string value in an array associated with a specified AN.
<a href="#">ArraySwap</a>	Swaps the contents of two arrays.
<a href="#">DspArrayByAn</a>	Displays an alarm list associated with a specified AN at the location of that AN.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## ArrayCopy

Makes a copy of an array.

## Syntax

**INT ArrayCopy(INT hArray)**

*hArray:*

The handle of the array you want to copy.

## Return Value

The handle of the created copy. If unsuccessful, -1 is returned. The error code can be obtained by calling the IsError Cicode function.

## Related Functions

[ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExists](#), [ArrayExistsByAn](#), [ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetIntByAn](#),

[ArrayGetMapName](#), [ArrayGetMapNameByAn](#), [ArrayGetString](#), [ArrayGetStringByAn](#), [ArrayIsDirty](#), [ArraySetInt](#),  
[ArraySetIntByAn](#), [ArraySetIsDirty](#), [ArraySetString](#), [ArraySetStringByAn](#), [ArraySwap](#), [DspArrayByAn](#)

## Example

```
...
hArray2 = ArrayCopy(hArray1);
...
```

## See Also

[Array Functions](#)

## ArrayCreate

Creates an array. An array can include up to 2,097,153 elements, but each dimension needs to have less than 32765.

A Plant SCADA project will only support up to 32765 arrays.

**Note:** Very large arrays may require a lot of memory. You need to calculate how much memory will be used by an array. For example, an array of 100 x 100 x 100 storing a 100 character string value in each cell will use approximately 100,000,000 bytes of memory.

## Syntax

**INT ArrayCreate(STRING sArrayName, INT x [, INT y [, INT z]])**

*sArrayName*:

The name of the array.

*x*:

The size of the array's x–dimension (from 1 to 32765).

*y*:

The size of the array's y–dimension (from 1 to 32765). This value is optional. If not specified, it defaults to 1.

*z*:

The size of the array's z–dimension (from 1 to 32765). This value is optional. If not specified, it defaults to 1.

## Return Value

The handle of the array. If unsuccessful, –1 is returned. The error code can be obtained by calling the IsError Cicode function.

**Note:** Any arrays that are created in custom Cicode should also be destroyed in custom Cicode, otherwise the program may eventually consume all available memory (see [ArrayDestroy](#)).

## Related Functions

[ArrayCopy](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExists](#), [ArrayExistsByAn](#),

[ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetIntByAn](#),  
[ArrayGetMapName](#), [ArrayGetMapNameByAn](#), [ArrayGetString](#), [ArrayGetStringByAn](#), [ArrayIsDirty](#), [ArraySetInt](#),  
[ArraySetIntByAn](#), [ArraySetIsDirty](#), [ArraySetString](#), [ArraySetStringByAn](#), [ArraySwap](#), [DspArrayByAn](#)

## Example

```
...  
  
hArray = ArrayCreate("TestArray", 128, 128, 128);  
err0 = IsError(); // error = 0 - number of elements 2,097,152  
  
hArray = ArrayCreate("TestArray", 129, 128, 128);  
err1 = IsError(); // error = 272 out of memory  
  
hArray = ArrayCreate("TestArray", 128, 128, 129);  
err2 = IsError(); // error = 272 out of memory  
  
hArray = ArrayCreate("TestArray", 32765, 1, 1);  
err3 = IsError(); // error = 0  
  
hArray = ArrayCreate("TestArray", 32765, 10, 5);  
err4 = IsError(); // error = 0 - number of elements 1,638,250  
  
hArray = ArrayCreate("TestArray", 32766, 1, 1);  
err5 = IsError(); // error = 257 - value is out of range  
  
hArray = ArrayCreate("TestArray", 1, 32766, 1);  
err6 = IsError(); // error = 257 - value is out of range  
  
hArray = ArrayCreate("TestArray", 1, 1, 32766);  
err7 = IsError(); // error = 257 - value is out of range  
  
hArray = ArrayCreate("TestArray", 32765);  
err8 = IsError(); // error = 0  
  
...
```

## See Also

[Array Functions](#)

### ArrayCreateByAn

Creates an array at a specified AN. An array can include up to 2,097,153 elements, but each dimension needs to have less than 32765.

A Plant SCADA project will only support up to 32765 arrays.

**Note:** Very large arrays may require a lot of memory. You need to calculate how much memory will be used by an array. For example, an array of 100 x 100 x 100 storing a 100 character string value in each cell will use approximately 100,000,000 bytes of memory.

## Syntax

```
INTArrayCreateByAn(INT nAN, INT x [, INT y [, INT z]])
```

*nAN*:

The AN number to associate with the array.

Elements in a Genie can access array cell values associated with a particular AN number without having the array handle. The Genie creates the array and associates it with the AN of one of the items in the Genie when it is initialized or the page is created. Other items in the Genie can then get or set values in that array. The List View Genie uses this 'ByAn' array feature.

*x*:

The size of the array's x–dimension (from 1 to 32765).

*y*:

The size of the array's y–dimension (from 1 to 32765). This value is optional. If not specified, it defaults to 1.

*z*:

The size of the array's z–dimension (from 1 to 32765). This value is optional. If not specified, it defaults to 1.

## Return Value

The handle of the array. If unsuccessful, –1 is returned. The error code can be obtained by calling the IsError Cicode function.

**Note:** Any arrays that are created in custom Cicode should also be destroyed in custom Cicode, otherwise the program may eventually consume all available memory (see [ArrayDestroyByAn](#)).

## Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExists](#), [ArrayExistsByAn](#),  
[ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetIntByAn](#),  
[ArrayGetMapName](#), [ArrayGetMapNameByAn](#), [ArrayGetString](#), [ArrayGetStringByAn](#), [ArrayIsDirty](#), [ArraySetInt](#),  
[ArraySetIntByAn](#), [ArraySetIsDirty](#), [ArraySetString](#), [ArraySetStringByAn](#), [ArraySwap](#), [DspArrayByAn](#)

## Example

```
...  
  
hArray = ArrayCreateByAn(hAn, 128, 128, 128);  
err0 = IsError(); // error = 0 - number of elements 2,097,152  
  
hArray = ArrayCreateByAn(hAn, 129, 128, 128);  
err1 = IsError(); // error = 272 out of memory  
  
hArray = ArrayCreateByAn(hAn, 128, 128, 129);  
err2 = IsError(); // error = 272 out of memory  
  
hArray = ArrayCreateByAn(hAn, 32765, 1, 1);  
err3 = IsError(); // error = 0  
  
hArray = ArrayCreateByAn(hAn, 32765, 10, 5);
```

```
err4 = IsError(); // error = 0 - number of elements 1,638,250

hArray = ArrayCreateByAn(hAn, 32766, 1, 1);
err5 = IsError(); // error = 257 - value is out of range

hArray = ArrayCreateByAn(hAn, 1, 32766, 1);
err6 = IsError(); // error = 257 - value is out of range

hArray = ArrayCreateByAn(hAn, 1, 1, 32766);
err7 = IsError(); // error = 257 - value is out of range

hArray = ArrayCreateByAn(hAn, 3, 3, 3);
err8 = IsError(); // error = 0

...
```

## See Also

[Array Functions](#)

### ArrayDestroy

Destroys an array.

## Syntax

**INTArrayDestroy(INT hArray)**

*hArray:*

The handle of the array that will be destroyed.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroyByAn](#), [ArrayExists](#), [ArrayExistsByAn](#),  
[ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetIntByAn](#),  
[ArrayGetMapName](#), [ArrayGetMapNameByAn](#), [ArrayGetString](#), [ArrayGetStringByAn](#), [ArrayIsDirty](#), [ArraySetInt](#),  
[ArraySetIntByAn](#), [ArraySetIsDirty](#), [ArraySetString](#), [ArraySetStringByAn](#), [ArraySwap](#), [DspArrayByAn](#)

## Example

```
Error1 = ArrayDestroy(hArray);
```

## See Also

[Array Functions](#)

## ArrayDestroyByAn

Destroys an array associated with the specified AN.

### Syntax

**INTArrayDestroyByAn(INT nAN)**

*nAN:*

The AN number associated with the array that will be destroyed.

### Return Value

0 (zero) if successful, otherwise an error is returned.

### Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayExists](#), [ArrayExistsByAn](#),  
[ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetIntByAn](#),  
[ArrayGetMapName](#), [ArrayGetMapNameByAn](#), [ArrayGetString](#), [ArrayGetStringByAn](#), [ArrayIsDirty](#), [ArraySetInt](#),  
[ArraySetIntByAn](#), [ArraySetIsDirty](#), [ArraySetString](#), [ArraySetStringByAn](#), [ArraySwap](#), [DspArrayByAn](#)

### Example

```
error1 = ArrayDestroyByAn(nAn);
```

### See Also

[Array Functions](#)

## ArrayExists

Determines if an array with a specified handle exists.

### Syntax

**INT ArrayExists(INT hArray)**

*hArray:*

The handle of the array.

### Return Value

TRUE if the array exists. FALSE if the array does not exist.

## Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExistsByAn](#),  
[ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetIntByAn](#),  
[ArrayGetMapName](#), [ArrayGetMapNameByAn](#), [ArrayGetString](#), [ArrayGetStringByAn](#), [ArrayIsDirty](#), [ArraySetInt](#),  
[ArraySetIntByAn](#), [ArraySetIsDirty](#), [ArraySetString](#), [ArraySetStringByAn](#), [ArraySwap](#), [DspArrayByAn](#)

## Example

```
...
IF (ArrayExists(hArray1)) THEN
    MapName = ArrayGetMapName(hArray1);
END
...
```

## See Also

[Array Functions](#)

## ArrayExistsByAn

Determines if an array associated with a specified AN exists.

## Syntax

INT **ArrayExistsByAn(INT nAN)**

*nAN:*

The AN associated with an array.

## Return Value

TRUE if the array exists. FALSE if the array does not exist.

## Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExists](#),  
[ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetIntByAn](#),  
[ArrayGetMapName](#), [ArrayGetMapNameByAn](#), [ArrayGetString](#), [ArrayGetStringByAn](#), [ArrayIsDirty](#), [ArraySetInt](#),  
[ArraySetIntByAn](#), [ArraySetIsDirty](#), [ArraySetString](#), [ArraySetStringByAn](#), [ArraySwap](#), [DspArrayByAn](#)

## Example

```
...
IF (ArrayExistsByAn(hAN)) THEN
    MapName = ArrayGetMapNameByAn(hAn);
END
...
```

## See Also

[Array Functions](#)

### ArrayFillFromAlarmDataByAn

This function is used to fill an array with information from internal alarm record caches. It is used by alarm page Genies to obtain the alarm record information to be displayed on the page.

## Syntax

**INT ArrayFillFromAlarmDataByAn(INT *hAN*)**

*hAN*:

The AN number for the alarm list you want to use to fill the array.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExists](#), [ArrayExistsByAn](#),  
[ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetIntByAn](#), [ArrayGetMapName](#), [ArrayGetMapNameByAn](#),  
[ArrayGetString](#), [ArrayGetStringByAn](#), [ArrayIsDirty](#), [ArraySetInt](#), [ArraySetIntByAn](#), [ArraySetsDirty](#), [ArraySetString](#),  
[ArraySetStringByAn](#), [ArraySwap](#), [DspArrayByAn](#)

## Example

```
error1 = ArrayFillFromAlarmDataByAn(600);
```

## See Also

[Array Functions](#)

### ArrayGetArrayByAn

Retrieves an array associated with a specified AN.

## Syntax

**INT ArrayGetArrayByAn(INT *nAN*)**

*nAN*:

The AN number associated with the array that will be retrieved.

## Return Value

The handle of the array. If unsuccessful, -1 is returned. The error code can be obtained by calling the [IsError](#) Cicode function.

## Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExists](#), [ArrayExistsByAn](#), [ArrayFillFromAlarmDataByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetIntByAn](#), [ArrayGetMapName](#), [ArrayGetMapNameByAn](#), [ArrayGetString](#), [ArrayGetStringByAn](#), [ArrayIsDirty](#), [ArraySetInt](#), [ArraySetIntByAn](#), [ArraySetIsDirty](#), [ArraySetString](#), [ArraySetStringByAn](#), [ArraySwap](#), [DspArrayByAn](#)

## Example

```
hArray = ArrayGetArrayByAn(hAn);
```

## See Also

[Array Functions](#)

## ArrayGetInfo

Retrieves the size of the x-, y-, or z-dimension for an array.

## Syntax

**INT ArrayGetInfo(INT hArray, INT nType)**

*hArray:*

The handle of the array.

*nType:*

0 = the array's x-dimension

1 = the array's y-dimension

2 = the array's z-dimension

## Return Value

The size of the x-, y-, or z-dimension, if successful. If unsuccessful, -1 is returned. The error code can be obtained by calling the [IsError](#) Cicode function.

## Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExists](#), [ArrayExistsByAn](#), [ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInt](#), [ArrayGetIntByAn](#), [ArrayGetMapName](#), [ArrayGetMapNameByAn](#), [ArrayGetString](#), [ArrayGetStringByAn](#), [ArrayIsDirty](#), [ArraySetInt](#), [ArraySetIntByAn](#), [ArraySetIsDirty](#), [ArraySetString](#), [ArraySetStringByAn](#)

[ArraySetIsDirty](#), [ArraySetString](#), [ArraySetStringByAn](#), [ArraySwap](#), [DspArrayByAn](#)

## Example

```
...
hArray = ArrayCreate("TestArray", 3, 4, 5);
x=ArrayGetInfo(hArray,0); //x-dimension expected = 3
y=ArrayGetInfo(hArray,1); //y-dimension expected = 4
z=ArrayGetInfo(hArray,2); //z-dimension expected = 5
```

## See Also

[Array Functions](#)

## ArrayGetInt

Retrieves an integer value from an array.

## Syntax

**INT ArrayGetInt(INT *hArray*, INT *x* [, INT *y* [, INT *z*]])**

*hArray*:

The handle of the array.

*x*:

The index for the x–dimension.

*y*:

The index for the y–dimension.

*z*:

The index for the z–dimension.

## Return Value

The requested integer, if successful. If unsuccessful, the error code can be obtained by calling the [IsError](#) Cicode function.

## Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExists](#), [ArrayExistsByAn](#), [ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetIntByAn](#), [ArrayGetMapName](#), [ArrayGetMapNameByAn](#), [ArrayGetString](#), [ArrayGetStringByAn](#), [ArrayIsDirty](#), [ArraySetInt](#), [ArraySetIntByAn](#), [ArraySetIsDirty](#), [ArraySetString](#), [ArraySetStringByAn](#), [ArraySwap](#), [DspArrayByAn](#)

## Example

```
...
```

```
// Set the integer value at Array(1,2,3) to be 444
ArraySetInt(hArray, 444, 1, 2, 3);
// Get the integer value at Array(1,2,3)
value = ArrayGetInt(hArray, 1, 2, 3);
// value = 444
...
```

## See Also

[Array Functions](#)

### ArrayGetIntByAn

Retrieves an integer value from an array associated with a specified AN.

## Syntax

```
INT ArrayGetIntByAn(INT nAN, INT x [, INT y [, INT z]])
```

*nAN*:

The AN number.

*x*:

The index for the x–dimension.

*y*:

The index for the y–dimension.

*z*:

The index for the z–dimension.

## Return Value

The requested integer, if successful. If unsuccessful, –1 is returned. The error code can be obtained by calling the [IsError](#) Cicode function.

## Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExists](#), [ArrayExistsByAn](#), [ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetMapName](#), [ArrayGetMapNameByAn](#), [ArrayGetString](#), [ArrayGetStringByAn](#), [ArrayIsDirty](#), [ArraySetInt](#), [ArraySetIntByAn](#), [ArraySetIsDirty](#), [ArraySetString](#), [ArraySetStringByAn](#), [ArraySwap](#), [DspArrayByAn](#)

## Example

```
...
// Set the integer value at (1,2,3) of the array associated with hAn to be 444
ArraySetIntByAn(hAn, 444, 1, 2, 3);
// Get the integer value at (1,2,3) of the array associated with hAn
value = ArrayGetIntByAn(hAn, 1, 2, 3);
```

```
// value = 444
...
```

## See Also

[Array Functions](#)

### ArrayGetMapName

Retrieves the name of the map associated with an array.

## Syntax

```
STRING ArrayGetMapName(INT hArray)
```

*hArray*:

The handle of the array.

## Return Value

If successful, the requested map name as a string. If unsuccessful, " " is returned. The error code can be obtained by calling the [IsError](#) Cicode function.

## Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExists](#), [ArrayExistsByAn](#),  
[ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetIntByAn](#),  
[ArrayGetMapNameByAn](#), [ArrayGetString](#), [ArrayGetStringByAn](#), [ArrayIsDirty](#), [ArraySetInt](#), [ArraySetIntByAn](#),  
[ArraySetIsDirty](#), [ArraySetString](#), [ArraySetStringByAn](#), [ArraySwap](#), [DspArrayByAn](#)

## Example

```
MapName = ArrayGetMapName(hArray1);
```

## See Also

[Array Functions](#)

### ArrayGetMapNameByAn

Retrieves the name of the map for an array associated with a specified AN.

## Syntax

```
STRING ArrayGetMapNameByAn(INT nAN)
```

*nAN*:

The AN associated with the array.

## Return Value

If successful, the requested map name as a string. If unsuccessful, " " is returned. The error code can be obtained by calling the [IsError](#) Cicode function.

## Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExists](#), [ArrayExistsByAn](#),  
[ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetIntByAn](#), [ArrayGetString](#),  
[ArrayGetStringByAn](#), [ArrayIsDirty](#), [ArraySetInt](#), [ArraySetIntByAn](#), [ArraySetIsDirty](#), [ArraySetString](#),  
[ArraySetStringByAn](#), [ArraySwap](#), [DspArrayByAn](#)

## Example

```
MapName = ArrayGetMapNameByAn(hAn);
```

## See Also

[Array Functions](#)

## ArrayGetString

Retrieves a string from an array.

## Syntax

```
STRING ArrayGetString(INT hArray, INT x [, INT y [, INT z]])
```

*hArray*:

The handle of the array.

*x*:

The index for the x-dimension.

*y*:

The index for the y-dimension.

*z*:

The index for the z-dimension.

## Return Value

The requested string, if successful. If unsuccessful, " " is returned. The error code can be obtained by calling the [IsError](#) Cicode function.

## Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExists](#), [ArrayExistsByAn](#),  
[ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetIntByAn](#),  
[ArrayGetMapName](#), [ArrayGetMapNameByAn](#), [ArrayGetStringByAn](#), [ArrayIsDirty](#), [ArraySetInt](#), [ArraySetIntByAn](#),  
[ArrayIsDirty](#), [ArraySetString](#), [ArrayGetStringByAn](#), [ArraySwap](#), [DspArrayByAn](#)

## Example

```
// Set the string value at Array(1,2,3) to be "abc"  
ArraySetString(hArray, "abc", 1, 2, 3);  
// Get the integer value at Array(1,2,3)  
value = ArrayGetString(hArray, 1, 2, 3);  
// value = "abc"  
...
```

## See Also

[Array Functions](#)

## ArrayGetStringByAn

Retrieves a string value from an array associated with a specified AN.

## Syntax

STRING **ArrayGetStringByAn**(INT *nAN*, INT *x* [, INT *y* [, INT *z*]])

*nAN*:

The AN associated with array.

*x*:

The index for the x–dimension.

*y*:

The index for the y–dimension.

*z*:

The index for the z–dimension.

## Return Value

The requested string, otherwise an error is returned.

## Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExists](#), [ArrayExistsByAn](#),  
[ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetIntByAn](#),  
[ArrayGetMapName](#), [ArrayGetMapNameByAn](#), [ArrayGetString](#), [ArrayIsDirty](#), [ArraySetInt](#), [ArraySetIntByAn](#),

[ArraySetIsDirty](#), [ArraySetString](#), [ArraySetStringByAn](#), [ArraySwap](#), [DspArrayByAn](#)

## Example

```
// Set the integer value at (1,2,3) of the array associated with hAn to be "abc"
ArraySetStringByAn(hAn, "abc", 1, 2, 3);
// Get the integer value at (1,2,3) of the array associated with hAn
value = ArrayGetStringByAn(hAn, 1, 2, 3);
// value = "abc"
...
```

## See Also

[Array Functions](#)

## ArrayIsDirty

Determines if an array is ‘dirty’ (information in the array has changed). When any array element is changed, the ‘Dirty’ flag of the array is set to be TRUE.

## Syntax

**INT ArrayIsDirty(INT hArray)**

*hArray:*

The handle of the array.

## Return Value

1 if true (is dirty), or 0 if false (is not dirty). An error is not returned by this function.

## Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExists](#), [ArrayExistsByAn](#),  
[ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetIntByAn](#),  
[ArrayGetMapName](#), [ArrayGetMapNameByAn](#), [ArrayGetString](#), [ArrayGetStringByAn](#), [ArraySetInt](#),  
[ArraySetIntByAn](#), [ArraySetIsDirty](#), [ArraySetString](#), [ArraySetStringByAn](#), [ArraySwap](#), [DspArrayByAn](#)

## Example

```
hArray = ArrayCreate("TestArray", 3, 3, 3);
error1 = ArraySetIsDirty(hArray, 0); // reset is dirty
intig1 = ArrayIsDirty(hArray); // intig1 = 0 -> reset
ArraySetInt(hArray, 444, 1, 2, 3); // Change a value
intig1 = ArrayIsDirty(hArray) ; // intig1 = 1 -> set
```

## See Also

[Array Functions](#)

### ArraySetInt

Sets an integer value within an array.

## Syntax

**INTArraySetInt(INT hArray, INT nValue, INT x [, INT y [, INT z]])**

*hArray:*

The handle of the array.

*nValue:*

The value you would like to set.

*x:*

The index for the x–dimension.

*y:*

The index for the y–dimension.

*z:*

The index for the z–dimension.

## Return Value

If unsuccessful, an error code will be returned by the [IsError](#) Cicode function.

## Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExists](#), [ArrayExistsByAn](#),  
[ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetIntByAn](#),  
[ArrayGetMapName](#), [ArrayGetMapNameByAn](#), [ArrayGetString](#), [ArrayGetStringByAn](#), [ArrayIsDirty](#),  
[ArraySetIntByAn](#), [ArraySetIsDirty](#), [ArraySetString](#), [ArraySetStringByAn](#), [ArraySwap](#), [DspArrayByAn](#)

## Example

```
...
error1= ArraySetInt(hArray, IntToStr(i), 0, 0, 0);
value1= ArrayGetInt(hArray, 0, 0, 0);
...
```

## See Also

[Array Functions](#)

## ArraySetIntByAn

Sets an integer value in an array associated with a specified AN.

## Syntax

**INTArraySetIntByAn(INT *nAN*, INT *nValue*, INT *x* [, INT *y* [, INT *z*]])**

*nAN*:

The AN associated with the array.

*nValue*:

The value you would like to set.

*x*:

The index for the x–dimension.

*y*:

The index for the y–dimension.

*z*:

The index for the z–dimension.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExists](#), [ArrayExistsByAn](#),  
[ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetIntByAn](#),  
[ArrayGetMapName](#), [ArrayGetMapNameByAn](#), [ArrayGetString](#), [ArrayGetStringByAn](#), [ArrayIsDirty](#), [ArraySetInt](#),  
[ArraySetIsDirty](#), [ArraySetString](#), [ArraySetStringByAn](#), [ArraySwap](#), [DspArrayByAn](#)

## Example

```
...
// Set the integer value at (1,2,3) of the array associated with hAn to be 444
ArraySetIntByAn(hAn, 444, 1, 2, 3);
// Get the integer value at (1,2,3) of the array associated with hAn
value = ArrayGetIntByAn(hAn, 1, 2, 3);
// value = 444
...
```

## See Also

[Array Functions](#)

## ArraySetIsDirty

Allows you to indicate that an array is dirty (meaning the information in the array has changed), or not dirty (the information in the array has not been changed).

## Syntax

**ArraySetIsDirty(INT hArray, INT IsDirty)**

*hArray*:

The handle of the array.

*IsDirty*:

0= False

1 = True

## Return Value

If unsuccessful, an error code will be returned by the [IsError](#) Cicode function.

## Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExists](#), [ArrayExistsByAn](#),  
[ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetIntByAn](#),  
[ArrayGetMapName](#), [ArrayGetMapNameByAn](#), [ArrayGetString](#), [ArrayGetStringByAn](#), [ArrayIsDirty](#), [ArraySetInt](#),  
[ArraySetIntByAn](#), [ArraySetString](#), [ArraySetStringByAn](#), [ArraySwap](#), [DspArrayByAn](#)

## Example

```
hArray = ArrayCreate("TestArray", 3, 3, 3);
error1 = ArraySetIsDirty(hArray, 0); // reset is dirty
intig1 = ArrayIsDirty(hArray); // intig1 = 0 -> reset
ArraySetInt(hArray, 444, 1, 2, 3); // Change a value
intig1 = ArrayIsDirty(hArray) ; // intig1 = 1 -> set
```

## See Also

[Array Functions](#)

## ArraySetString

Sets a string value in an array.

## Syntax

**INTArraySetString(INT hArray, STRING sValue, INT x [, INT y [, INT z]])**

*hArray*:

The handle of the array.

*sValue*:

The string value you would like to set.

*x*:

The index for the x-dimension.

*y*:

The index for the y-dimension.

*z*:

The index for the z-dimension.

## Return Value

If unsuccessful, an error code will be returned by the [IsError](#) Cicode function.

## Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExists](#), [ArrayExistsByAn](#),  
[ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetIntByAn](#),  
[ArrayGetMapName](#), [ArrayGetMapNameByAn](#), [ArrayGetString](#), [ArrayGetStringByAn](#), [ArrayIsDirty](#), [ArraySetInt](#),  
[ArraySetIntByAn](#), [ArraySetIsDirty](#), [ArraySetStringByAn](#), [ArraySwap](#), [DspArrayByAn](#)

## Example

```
// Set the string value at Array(1,2,3) to be "abc"  
ArraySetString(hArray, "abc", 1, 2, 3);  
// Get the integer value at Array(1,2,3)  
value = ArrayGetString(hArray, 1, 2, 3);  
// value = "abc"  
...
```

## See Also

[Array Functions](#)

## ArraySetStringByAn

Sets a string value in an array associated with a specified AN.

## Syntax

**INTArraySetStringByAn(INT *nAN*, STRING *sValue*, INT *x* [, INT *y* [, INT *z*]])**

*nAN*:

The AN associated with the array.

*sValue*:

The string value you would like to set.

*x*:

The index for the x–dimension.

*y*:

The index for the y–dimension.

*z*:

The index for the z–dimension.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExists](#), [ArrayExistsByAn](#),  
[ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetIntByAn](#),  
[ArrayGetMapName](#), [ArrayGetMapNameByAn](#), [ArrayGetString](#), [ArrayGetStringByAn](#), [ArrayIsDirty](#), [ArraySetInt](#),  
[ArraySetIntByAn](#), [ArraySetIsDirty](#), [ArraySetString](#), [ArraySwap](#), [DspArrayByAn](#)

## Example

```
// Set the integer value at (1,2,3) of the array associated with hAn to be "abc"
ArraySetStringByAn(hAn, "abc", 1, 2, 3);
// Get the integer value at (1,2,3) of the array associated with hAn
value = ArrayGetStringByAn(hAn, 1, 2, 3);
// value = "abc"
...
```

## See Also

[Array Functions](#)

## ArraySwap

Swaps the contents of two arrays.

## Syntax

**INT ArraySwap(INT *hArray1*, INT *hArray2*)**

*hArray1*:

The handle of the array that you would like to swap with *hArray2*.

*hArray2*:

The handle of the array that you would like to swap with *hArray1*.

## Return Value

If unsuccessful, an error code will be returned by the [IsError](#) Cicode function.

## Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExists](#), [ArrayExistsByAn](#),  
[ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetIntByAn](#),  
[ArrayGetMapName](#), [ArrayGetMapNameByAn](#), [ArrayGetString](#), [ArrayGetStringByAn](#), [ArrayIsDirty](#), [ArraySetInt](#),  
[ArraySetIntByAn](#), [ArraySetIsDirty](#), [ArraySetString](#), [ArraySetStringByAn](#), [DspArrayByAn](#)

## Example

```
error1 = ArraySwap(hArray1, hArray2);
```

## See Also

[Array Functions](#)

## DspArrayByAn

Displays an alarm list associated with a specified AN at the location of that AN. The array associated with the AN contains the alarm data. It is typically used as part of an alarm list or generic list Genie.

## Syntax

**INTDspArrayByAn(INT nAN)**

*nAN:*

The AN associated the array that contains alarm data.

## Return Value

0 (zero). If there is an error, the error code can be obtained by calling the [IsError](#) Cicode function.

## Related Functions

[ArrayCopy](#), [ArrayCreate](#), [ArrayCreateByAn](#), [ArrayDestroy](#), [ArrayDestroyByAn](#), [ArrayExists](#), [ArrayExistsByAn](#),  
[ArrayFillFromAlarmDataByAn](#), [ArrayGetArrayByAn](#), [ArrayGetInfo](#), [ArrayGetInt](#), [ArrayGetIntByAn](#),  
[ArrayGetMapName](#), [ArrayGetMapNameByAn](#), [ArrayGetString](#), [ArrayGetStringByAn](#), [ArrayIsDirty](#), [ArraySetInt](#),  
[ArraySetIntByAn](#), [ArraySetIsDirty](#), [ArraySetString](#), [ArraySetStringByAn](#), [ArraySwap](#)

## Example

```
DspArrayByAn(hAn);
```

## See Also

[Array Functions](#)

## Clipboard Functions

Following are functions relating to the Windows clipboard:

<a href="#">ClipCopy</a>	Copies a string to the Windows clipboard.
<a href="#">ClipPaste</a>	Pastes a string from the Windows clipboard.
<a href="#">ClipReadLn</a>	Reads a line of text from the Windows clipboard.
<a href="#">ClipSetMode</a>	Sets the format of data sent to the Windows clipboard.
<a href="#">ClipWriteLn</a>	Writes a line of text to the Windows clipboard.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## ClipCopy

Copies a string to the Windows clipboard. When the string is in the clipboard, you can paste it to any Windows program.

## Syntax

**ClipCopy**(*sText*[, *bAppendNewLine*])

*sText*:

The string to copy to the clipboard.

*bAppendNewLine*:

Determines if new line characters (carriage return + line feed) are appended to text when it is added to the clipboard. You can set this argument to FALSE if you need to paste text into a field that does not accept these characters.

TRUE (default) = new line characters are included.

FALSE = new line characters are not included.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[ClipWriteLn](#)

## Example

```
ClipCopy("put this in clipboard");
```

See Also

[Clipboard Functions](#)

## ClipPaste

Pastes a string from the Windows clipboard.

## Syntax

**ClipPaste()**

## Return Value

The contents of the clipboard (as a string). If the clipboard is empty, an empty string is returned.

## Related Functions

[ClipReadLn](#), [ClipCopy](#)

## Example

```
/* Get string from clipboard into sText. */  
sText = ClipPaste();
```

See Also

[Clipboard Functions](#)

## ClipReadLn

Reads a single line of text from the Windows clipboard. With this function, you can read a block of text from the clipboard - line by line. Call the function once to read each line of text from the clipboard. When the end of the clipboard is reached, an empty string is returned.

## Syntax

**ClipReadLn()**

## Return Value

One line of text from the clipboard (as a string). If the clipboard is empty, an empty string is returned.

## Related Functions

[ClipPaste](#)

## Example

```
/* Get first line of text from clipboard. */  
sText = ClipReadLn();  
WHILE StrLength(sText) > 0 DO  
! Do something with text  
...  
! Read next line of clipboard  
sText = ClipReadLn();  
END
```

### See Also

[Clipboard Functions](#)

## ClipSetMode

Sets the format of data sent to the Windows clipboard.

## Syntax

**ClipSetMode(*nMode*)**

*nMode*:

The mode of the data:

1 - ASCII Text

2 - CSV (Comma separated values) format

You can select multiple modes by adding modes together.

## Return Value

The value of the previous mode.

## Related Functions

[ClipCopy](#), [ClipWriteLn](#)

## Example

```
/* Set the clipboard to CSV mode, write two values, and reset the
```

```
clipboard to the original mode. */
nOldMode = ClipSetMode(2);
ClipCopy("100,200");
ClipSetMode(nOldMode);
```

**See Also**[Clipboard Functions](#)

## ClipWriteLn

Writes a line of text to the Windows clipboard. With this function, you can write any amount of text to the clipboard. Call this function once for each line of text. To terminate the block of text, call this function and pass an empty string.

### Syntax

**ClipWriteLn(*sText*)**

*sText*:

The line of text to write to the clipboard, or an empty string (" ") to end the write operation.

### Return Value

0 (zero) if successful, otherwise an error code is returned.

### Related Functions

[ClipCopy](#)

### Example

```
ClipWriteLn("first line of text");
ClipWriteLn("second line of text");
ClipWriteLn(""); ! End of write operation
```

**See Also**[Clipboard Functions](#)

## Cluster Functions

Following are functions relating to clusters:

<a href="#">ClusterActivate</a>	Allows the user to activate an inactive cluster.
<a href="#">ClusterDeactivate</a>	Allows the user to deactivate an active cluster.
<a href="#">ClusterFirst</a>	Allows the user to retrieve the first configured cluster in the project.

<a href="#">ClusterGetName</a>	Deprecated in this version
<a href="#">ClusterIsActive</a>	Allows the user to determine if a cluster is active.
<a href="#">ClusterNext</a>	Allows the user to retrieve the next configured cluster in the project.
<a href="#">ClusterServerTypes</a>	Allows the user to determine which servers are defined for a given cluster.
<a href="#">ClusterSetName</a>	Deprecated in this version
<a href="#">ClusterStatus</a>	Allows the user to determine the connection status from the client to a server on a cluster.
<a href="#">ClusterSwapActive</a>	Allows the user to deactivate an active cluster at the same time as activating a deactive cluster.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## ClusterActivate

This function allows the user to activate an inactive cluster. When a cluster is made active, all data associated with that cluster is available to the client, and hardware alarms will occur if no connections can be made to the servers in the cluster.

## Syntax

**ClusterActivate(*sClusterName*)**

*sClusterName*:

The name of the cluster to activate enclosed in quotation marks "".

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[ClusterDeactivate](#), [ClusterFirst](#), [ClusterIsActive](#), [ClusterNext](#), [ClusterServerTypes](#), [ClusterStatus](#), [ClusterSwapActive](#), [TaskCluster](#)

## See Also

[Cluster Functions](#)

### ClusterDeactivate

This function allows the user to deactivate an active cluster. When a cluster is made inactive, no data associated with that cluster is available to the client, and hardware alarms will not occur if no connections can be made to the servers in the cluster.

## Syntax

**ClusterDeactivate(*sClusterName*)**

*sClusterName*:

The name of the cluster to deactivate enclosed in quotation marks "".

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[ClusterActivate](#), [ClusterFirst](#), [ClusterIsActive](#), [ClusterNext](#), [ClusterServerTypes](#), [ClusterStatus](#), [ClusterSwapActive](#), [TaskCluster](#)

## See Also

[Cluster Functions](#)

### ClusterFirst

This function allows the user to retrieve the first configured cluster in the project.

## Syntax

**ClusterFirst()**

## Return Value

The name of the first configured cluster.

## Related Functions

[ClusterActivate](#), [ClusterDeactivate](#), [ClusterIsActive](#), [ClusterNext](#), [ClusterServerTypes](#), [ClusterStatus](#),

[ClusterSwapActive](#), [TaskCluster](#)

## See Also

[Cluster Functions](#)

### ClusterGetName

ClusterGetName is deprecated in this version of Plant SCADA.

## Syntax

**ClusterGetName(*sPrimary*, *sStandby*, *nMode*)**

*sPrimary*:

The name of the cluster's primary server (that is that which was set as sPrimary using the ClusterSetName() function). Must be a String type variable.

*sStandby*:

The name of the cluster's standby server (that is that which was set as sStandby using the ClusterSetName() function). Must be a String type variable.

*nMode*:

The mode is for future expansion of the function - set to 0 (zero).

## Return Value

The status of the get name.

## Related Functions

[ClusterSetName](#)

## Example

```
// Return and display the server names.//  
ClusterGetName(sPrimary, sStandby, 0);  
Prompt("Name of Cluster" + sPrimary);
```

## See Also

[Cluster Functions](#)

### ClusterIsActive

This function allows the user to determine if a cluster is active.

## Syntax

**ClusterIsActive(ClusterName)**

*sClusterName*:

The name of the cluster to query enclosed in quotation marks " ".

## Return Value

TRUE if active, FALSE otherwise. If the cluster name was invalid, this function will return FALSE and a hardware alarm will be generated.

## Related Functions

[ClusterActivate](#), [ClusterDeactivate](#), [ClusterFirst](#), [ClusterNext](#), [ClusterServerTypes](#), [ClusterStatus](#),  
[ClusterSwapActive](#), [TaskCluster](#)

## See Also

[Cluster Functions](#)

## ClusterNext

This function allows the user to retrieve the next configured cluster in the project.

## Syntax

**ClusterNext(ClusterName)**

*sClusterName*:

Any configured cluster name enclosed in quotation marks "", this will usually be the name of the previous cluster as returned from [ClusterFirst](#), or a previous call to ClusterNext.

## Return Value

The name of the next configured cluster or an empty string if there is no more clusters.

## Related Functions

[ClusterActivate](#), [ClusterDeactivate](#), [ClusterFirst](#), [ClusterIsActive](#), [ClusterServerTypes](#), [ClusterStatus](#),  
[ClusterSwapActive](#), [TaskCluster](#)

## See Also

[Cluster Functions](#)

## ClusterServerTypes

This function allows the user to determine which servers are defined for a given cluster.

## Syntax

**ClusterServerTypes(ClusterName)**

*sClusterName:*

The name of the cluster to query enclosed in quotation marks "".

## Return Value

Logical OR of the following server flags:

- 0001 - 1st bit set means an Alarm Server is configured
- 0010 - 2nd bit set means a Trend Server is configured
- 0100 - 3rd bit set means a Report Server is configured
- 1000 - 4th bit set means an IO Server is configured

For example, a return value of 14 indicates an I/O Server, a Report Server, and a Trend Server are configured.

## Related Functions

[ClusterActivate](#), [ClusterDeactivate](#), [ClusterFirst](#), [ClusterIsActive](#), [ClusterNext](#), [ClusterStatus](#), [ClusterSwapActive](#), [TaskCluster](#)

## See Also

[Cluster Functions](#)

## ClusterSetName

ClusterSetName is deprecated in this version of Plant SCADA.

## Syntax

**ClusterSetName(*sPrimary*, *sStandby*, *nMode*)**

*sPrimary:*

The name of the cluster's primary server (Reports Server, Alarms Server etc.), as defined using the Computer Setup Wizard. When the ClusterSetName() function is used, Plant SCADA will attempt to connect to this server.

*sStandby:*

The name of the cluster's standby server (Reports Server, Alarms Server etc.), as defined using the Computer Setup Wizard. If the *sPrimary* server is unavailable when the ClusterSetName() function is used, Plant SCADA will attempt to connect to this server.

If there is no standby server, enter an empty string for *sStandby*.

*nMode*:

The mode of the connection:

0 - If you select this mode, Plant SCADA will renew the last connection. If it was connected to the *sPrimary* server, when this function was last used, it will attempt to connect to it again. If it was last connected to the *sStandby* server, it will attempt to connect to it again. This mode is useful when a server is known to be unavailable, as it facilitates faster cluster switching.

1 - Plant SCADA will attempt to connect to the *sPrimary* server first, each time this function is used. If the *sPrimary* server is unavailable, Plant SCADA will try the *sStandby* server.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[ClusterGetName](#)

## Example

```
// Connect to Cluster A, with server CITECTA1 as primary server,  
and CITECTA2 as standby.//  
ClusterSetName("CITECTA1", "CITECTA2", 0);  
// Display the menu page for Cluster A Project.//  
PageDisplay("MenuA");
```

## See Also

[Cluster Functions](#)

## ClusterStatus

This function allows the user to determine the connection status from the client to a server on a cluster.

## Syntax

**ClusterStatus**(*clusterName*, *serverType*)

*clusterName*:

The name of the cluster to query enclosed in quotation marks "".

*serverType*:

The type of server (not a bit mask):

1 - Alarm Server

2 - Trend Server

4 - Report Server

## 8 - IO Server

### Return Value

One of the following values:

- -1 - if the cluster does not contain a server of the given type.
- -2 - if the cluster does not exist"
- 0 - if the cluster contains the server but the cluster is inactive.
- 1 - if the cluster is active but the connection to the server is offline.
- 2 - if the cluster is active and the connection to the server is online.

### Related Functions

[ClusterActivate](#), [ClusterDeactivate](#), [ClusterFirst](#), [ClusterIsActive](#), [ClusterNext](#), [ClusterServerTypes](#),  
[ClusterSwapActive](#), [TaskCluster](#)

### See Also

[Cluster Functions](#)

### ClusterSwapActive

This function allows the user to deactivate an active cluster at the same time as activating an inactive cluster. The arguments may be passed in any order, but one cluster needs to be active and the other needs to be inactive.

### Syntax

**ClusterSwapActive**(*clusterNameA*, *clusterNameB*)

*clusterNameA*:

The name of the cluster to activate or deactivate enclosed in quotation marks "".

*clusterNameB*:

The name of the cluster to activate or deactivate enclosed in quotation marks "".

### Return Value

0 (zero) if successful, otherwise an error code is returned.

### Related Functions

[ClusterActivate](#), [ClusterDeactivate](#), [ClusterFirst](#), [ClusterIsActive](#), [ClusterNext](#), [ClusterServerTypes](#), [ClusterStatus](#),  
[TaskCluster](#)

## See Also

[Cluster Functions](#)

## Color Functions

Following are functions relating to colors:

<a href="#">CitectColourToPackedRGB</a>	Converts a Plant SCADA color into a packed RGB color value that can be used by an ActiveX object.
<a href="#">GetBlueValue</a>	Returns the Blue component of a packed RGB color.
<a href="#">GetGreenValue</a>	Returns the Green component of a packed RGB color.
<a href="#">GetRedValue</a>	Returns the Red component of a packed RGB color.
<a href="#">MakeColour</a>	Creates a color from red, green and blue component parts.
<a href="#">PackedRGB</a>	Returns a packed RGB color based on specified red, green, and blue values.
<a href="#">PackedRGBToCitectColour</a>	Converts a packed RGB color into the nearest equivalent Plant SCADA color.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

### CitectColourToPackedRGB

Converts a Plant SCADA color value into a packed RGB color value that can be understood by an ActiveX object.

## Syntax

**CitectColourToPackedRGB(*nCitectColor*)**

*nCitectColor*:

The Plant SCADA color value to be converted into a packed RGB color. Plant SCADA colors are defined in the labels database, or calculated by the function [MakeColour](#).

## Return Value

The packed RGB color value - if successful, otherwise an error code is returned.

## Related Functions

[PackedRGBToCitectColour](#)

## See Also

[Color Functions](#)

### GetBlueValue

Returns the blue component of a packed RGB color.

## Syntax

**GetBlueValue(*nPackedRGB*)**

*nPackedRGB*:

The packed RGB color.

## Return Value

The red value (0-255) - if successful, otherwise an error is returned.

## Related Functions

[GetRedValue](#), [GetGreenValue](#)

## See Also

[Color Functions](#)

### GetGreenValue

Returns the green component of a packed RGB color.

## Syntax

**GetGreenValue(*nPackedRGB*)**

*nPackedRGB*:

The packed RGB color.

## Return Value

The red value (0-255) - if successful, otherwise an error is returned.

## Related Functions

[GetRedValue](#), [GetBlueValue](#)

## See Also

[Color Functions](#)

### GetRedValue

Returns the red component of a packed RGB color.

## Syntax

**GetRedValue(*nPackedRGB*)**

*nPackedRGB*:

The packed RGB color.

## Return Value

The red value (0-255) - if successful, otherwise an error is returned.

## Related Functions

[GetGreenValue](#), [GetBlueValue](#)

## See Also

[Color Functions](#)

### MakeColour

Creates a color from red, green and blue component parts.

---

**Note:** To define a transparent color, use the label TRANSPARENT.

---

## Syntax

**MakeColour(*nRed*,*nGreen*,*nBlue*)**

*nRed*:

The color value for red, from 0-255

*nGreen*:

The color value for green, from 0-255

*nBlue*:

The color value for blue, from 0-255

## Return Value

An integer that is an encoded representation of the color created.

## Examples

```
! creates the color red
MakeColour(255,0,0)
! creates the color white
MakeColour(255,255,255)
```

**Note:** For backwards compatibility, you can refer to this function using the alias "MakeCitectColour".

## See Also

[Color Functions](#)

## PackedRGB

Returns a packed RGB color based on specified red, green, and blue values.

## Syntax

**PackedRGB(*nRed*, *nGreen*, *nBlue*)**

*nRed*:

The red component of the desired packed RGB color.

*nGreen*:

The green component of the desired packed RGB color.

*nBlue*:

The blue component of the desired packed RGB color.

## Return Value

The packed RGB color value - if successful, otherwise an error is returned.

## Related Functions

[CitectColourToPackedRGB](#)

## See Also

[Color Functions](#)

## PackedRGBToCitectColour

Converts a packed RGB color into a calculated Plant SCADA color value.

## Syntax

**PackedRGBToCitectColour(*nPackedRGB*)**

*nPackedRGB*:

The packed RGB color.

## Return Value

The Plant SCADA color value if successful; otherwise an error is returned.

## Related Functions

[CitectColourToPackedRGB](#)

## See Also

[Color Functions](#)

## Communication Functions

Following are functions relating to communications:

<a href="#">ComClose</a>	Closes a communication port.
<a href="#">ComOpen</a>	Opens a communication port for access.
<a href="#">ComRead</a>	Reads characters from a communication port.
<a href="#">ComReset</a>	Resets the communication port.
<a href="#">ComWrite</a>	Writes characters to a communication port.
<a href="#">SerialKey</a>	Redirects serial characters from a port to the keyboard.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## ComClose

This function flags a port for close and reopen when the next ComOpen (Port mode 0) or driver open occurs. It does not actually close the port when called. Any Cicode tasks that are waiting for a read or write operation to complete (or that are retrying to read or write) return with a range error.

Plant SCADA automatically closes all communication ports at shutdown.

This function can only be called from an I/O server.

## Syntax

**ComClose(*hPort*)**

*hPort*:

The communication port handle, returned from the [ComOpen](#) function. This handle identifies the table where all data on the associated communication port is stored.

## Return Value

0 if the port is successfully closed, or an error if the port is already closed or if the port number is invalid.

## Related Functions

[ComOpen](#), [ComRead](#), [ComWrite](#)

## Example

See [ComOpen](#).

## See Also

[Communication Functions](#)

## ComOpen

Opens a communication port for access. The board and port need to both be defined in the database (using the Boards and Ports settings).

If you try to open the same COM port twice with [ComOpen](#), the second open will not succeed and return -1. If this is passed without checking other Com functions, the COM port may not do anything. For this reason, do not open COM ports twice, and always check the return value from ComOpen.

The communication system should be used for low speed communications only. You should not use the communication functions to communicate with high speed PLCs - the performance may not be adequate.

It is highly recommended that you only use ComOpen in the context where no existing driver is using the same port.

For slower connections where latency might exist, it is recommended that a sleep is configured after ComOpen(). This delay will ensure that ComRead or ComWrite do not execute before the device is connected. The sleep

should be twice as long as the time it takes for communication to reach its destination and return (for example, ping time for TCPIP ports). For local serial ports, a sleep will not be needed.

This function can only be called from an I/O server.

## Syntax

**ComOpen(*sPort*, *nMode*)**

*sPort*:

The port name as specified in the Ports database.

*nMode*:

The mode of the open:

0 - Take control of the port from Plant SCADA. In this non-shared mode, you have complete access to the port - Plant SCADA cannot use the port. Communication will be restored when the port is closed. In this mode, fresh port opens will occur each call.

1 - Share the port with Plant SCADA. In this mode, you can write to the port, and Plant SCADA can also use it. Please be aware that ComRead will be unreliable if the communication port is opened as shared. In this mode, if the port is found to be connected, a reconnect does not occur. If not connected, then the port is connected.

It is recommended that mode 1 be used where possible.

## Return Value

A communication port handle if the communication system is opened successfully, otherwise -1 is returned. The handle identifies the table where all data on the associated port is stored. You can use the handle in the other communication functions, to send and receive characters from the port.

## Related Functions

[ComClose](#), [ComRead](#), [ComWrite](#)

## Example

```
INT
FUNCTION
StartSerial(STRING sPort)
    INT hPort;
    hPort = ComOpen(sPort, 0);
    IF hPort < 0 THEN
        Prompt("Cannot open port " + sPort);
        RETURN -1;
    END
    TaskNew("SerialRead", hPort, 0);
    TaskNew("SerialWrite", hPort, 0);
    ComClose(hPort);
    RETURN 0;
END
INT
FUNCTION
```

```
SerialWrite(INT hPort)
    STRING buffer;
    INT SerialWriteError;
    INT length;
    INT error;
    WHILE 1 DO
        ! put data into buffer and set length
        .
        .
        ErrSet(1);
        SerialWriteError = ComWrite(hPort, buffer, length, 1);
        error = IsError() // if 7 then a timeout occurred
        IF SerialWriteError THEN
            Prompt("Error Writing port");
            ComReset(hPort);
        END
    END
    RETURN 0;
END
INT
FUNCTION
SerialRead(INT hPort)
    STRING buffer;
    INT length;
    INT total;
    INT SerialReadError;
    total = 0;
    WHILE 1 DO
        length = 128; ! need to set length as read modifies
        SerialReadError = ComRead(hPort, buffer, length, 1);
        IF SerialReadError THEN
            // Only flag error if data expected
            Prompt("Error from port " + SerialReadError : #####);
            ComReset(hPort);
        ELSE
            ! get data from buffer, length is set to number read
            .
            .
        END
    END
    RETURN 0;
END
```

## See Also

[Communication Functions](#)

## ComRead

Reads characters from a communication port. The characters are read from the communication port into a string buffer. If no characters have arrived after the specified timeout, the function returns with a timeout error. If the timeout is 0, the function gets any characters that have arrived from the last call, and returns immediately.

You use the *iLength* variable to specify the length of the buffer, or the maximum number of characters to read

when ComRead is called. When ComRead returns, *iLength* is set to the actual number of characters read. Because *iLength* is modified by this function, you need to reset it before each call.

You should not treat the string buffer as a normal string - it has no string terminator. Use the StrGetChar function to extract characters from the buffer.

ComRead should not be used if the port is in use by an existing driver.

For high speed transports like TCPIP, it is recommended that timeout be set to 0.

It is strongly recommended not to call ComRead while another ComRead is still pending on the same port, because it can produce unexpected results.

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

Do not call ComRead() if another instance of ComRead() is still pending on the same port.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

This function is a blocking function. It blocks the calling Cicode task until the operation is complete. This function can only be called from an I/O Server.

## Syntax

**ComRead(*hPort*, *sBuffer*, *iLength*, *iTimeOut*)**

*hPort*:

The communication port handle, returned from the ComOpen() function. This handle identifies the table where the data on the associated communication port is stored.

*sBuffer*:

The buffer into which to put the characters. The actual number of characters read is returned in *iLength*. Must be a String type variable.

*iLength*:

The number of characters to read into the buffer. The maximum length you may read in one call is 128 characters. When the function returns, this variable is set to the actual number of characters read. Must be a Long type variable.

*iTimeOut*:

The timeout for the read to complete in seconds:

- If *iTimeOut* = 0 (zero), the function checks for characters in the buffer and returns.
- If *iTimeOut* > 0, the function returns after this number of seconds in Port mode 0 (and sets error 7 which is the task timeout code), or as soon as bytes are read in Port mode 1. In mode 1 if the read occurred before the timeout, there will be no error.
- If *iTimeOut* < 0, the function waits forever for characters.

## Return Value

0 (zero) if the read is successful, otherwise an error code is returned.

## Related Functions

[ComOpen](#), [ComClose](#), [ComWrite](#), [StrGetChar](#)

## Example

See [ComOpen](#).

### See Also

[Communication Functions](#)

## ComReset

Resets the communication port. This function can only be called from an I/O Server.

## Syntax

**ComReset(*hPort*)**

*hPort*:

The communication port handle, returned from the [ComOpen](#) function. This handle identifies the table where all data on the associated communication port is stored.

## Return Value

0 (zero) if the write is successful, otherwise an error code is returned.

## Related Functions

[ComOpen](#), [ComClose](#), [ComRead](#), [StrGetChar](#)

## Example

See [ComOpen](#).

### See Also

[Communication Functions](#)

## ComWrite

Writes characters to a communication port. The characters are written from the string buffer to the port. If the characters have not been transmitted after the specified timeout, the function returns with a timeout error. If the timeout is 0, the function returns immediately and the characters are transmitted in the background.

ComWrite does not treat the buffer as a true string, but rather as an array of characters of the length specified - you can send any character to the communication port. Use the StrSetChar function to build the buffer. Do not

call ComWrite while another ComWrite is still pending on the same port, because it can produce unexpected results.

You use the *iLength* variable to specify the length of the buffer, or the maximum number of characters to write when ComWrite is called. When ComWrite returns, *iLength* is reset to zero.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

It is highly recommended that you only use ComWrite in the context where no existing driver is using the same port.

For high speed transports like TCPIP, it is recommended that timeout be set to 0.

This function can only be called from an I/O server.

## Syntax

**ComWrite(*hPort*, *sBuffer*, *iLength*, *iTimeOut*)**

*hPort*:

The communication port handle, returned from the ComOpen() function. This handle identifies the table where all data on the associated communication port is stored.

*sBuffer*:

The buffer from which to write the characters. Must be a String type variable.

*iLength*:

The number of characters to write from the buffer. The maximum number of characters you can write is 128. Must be a Long type variable.

*iTimeOut*:

The timeout for the write to complete in seconds.

- If *iTimeOut* = 0 (zero), the characters are copied to the communication buffer and the function returns immediately - the characters are transmitted in the background.
- If *iTimeOut* > 0, the function returns after this number of seconds in Port mode 0 (and sets error 7 which is the task timeout code), or as soon as the write occurs in Port mode 1. In mode 1 if the write occurred, there will be no error.
- If *iTimeOut* < 0, the function waits forever to transmit the characters.

## Return Value

0 (zero) if the write is successful, otherwise an error code is returned.

## Related Functions

[ComOpen](#), [ComClose](#), [ComRead](#), [StrGetChar](#)

## Example

See [ComOpen](#).

## See Also

[Communication Functions](#)

### SerialKey

Redirects all serial characters from a port to the keyboard. If using a keyboard attached to a serial port, you should call this function at startup, so that Plant SCADA copies all characters (read from the port) to the keyboard. The Port needs to be defined in the Ports database.

If the port is not on an I/O server, you need to create a dummy I/O server record (for example, name the server DServer1). Complete the Boards and Ports records. Set the following parameters in the CITECT.INI file:

[IOServer]Name to the server name (for example, DServer1)  
[IOServer]Server to 0

This method enables the port without making the computer an I/O server. (If the I/O server is enabled (and not required as an I/O server), extra overhead and memory are used.)

This function can only be called from an I/O server.

## Syntax

**SerialKey(*sPort*)**

*sPort*:

The name of the port connected to the serial keyboard.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[ComOpen](#)

## Example

```
SerialKey("Port1"); ! enable the serial keyboard
```

## See Also

[Communication Functions](#)

## Dynamic Data Exchange Functions

Following are functions relating to Dynamic Data Exchange:

DDEExec	Executes a command in an external DDE compliant
---------	---

	Windows application.
DDEPost	Makes a Plant SCADA variable available for DDE linking by other DDE compliant Windows applications.
DDERead	Reads a variable from a DDE compliant Windows application.
DDEWrite	Writes a variable to a DDE compliant Windows application.
DDEhExecute	Executes a command in an external DDE compliant Windows application.
DDEhGetLastError	Gets the latest Windows DDE error code.
DDEhInitiate	Starts a DDE conversation with an external DDE compliant Windows application.
DDEhPoke	Writes data to a DDE compliant Windows application.
DDEhReadLn	Reads a line of text from a DDE Conversion.
DDEhRequest	Requests data from a DDE compliant Windows application.
DDEhSetMode	Set the mode of a DDE conversation.
DDEhTerminate	Closes a DDE conversation with a Windows application.
DDEhWriteLn	Writes a line of text to the DDE conversation.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## DDEExec

Executes a command in an external Windows application running on the same computer. With this function, you can control other applications that support DDE. Refer to the documentation provided with the external Windows application to determine if DDE is supported and what functions can be called.

You cannot use DDEExec() to call macros on a remote computer or to call Access SQLs. For these calls, Network DDE needs to pass the *sDocument* argument, so you need to use the *DDEh...* functions, passing *sDocument* in the DDEhInitiate() function.

## Syntax

**DDEExec(*sApplication*, *sCommand*)**

*sApplication*:

Application name (.EXE filename), for example, "WinWord".

*sCommand*:

The command that the application will execute.

## Return Value

1 (one) if successful, otherwise an error code is returned.

## Related Functions

[DDEPost](#), [DDERead](#), [DDEWrite](#), [DDEhExecute](#)

## Example

```
/* Instruct the Excel application to recalculate its spreadsheet
immediately. */
DDEExec("Excel", "[Calculate.Now()]");
```

## See Also

[Dynamic Data Exchange Functions](#)

## DDEhExecute

Executes a command in an external Windows application. You need to first start a conversation with the DDEhInitiate function, and use the handle returned by that function to identify the conversation.

With this function, you can control other applications that support DDE. Refer to the documentation provided with your other Windows application to determine if DDE is supported and what functions can be called.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

## Syntax

**DDEhExecute(*Handle*, *sCommand*)**

*Handle*:

The integer handle that identifies the DDE conversation, returned from the DDEhInitiate function.

*sCommand*:

The command that the application will execute.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[DDEhInitiate](#), [DDEhRequest](#), [DDEhPoke](#), [DDEhTerminate](#), [DDEhGetLastError](#)

## Example

See [DDEhInitiate](#).

## See Also

[Dynamic Data Exchange Functions](#)

## DDEhGetLastError

Gets the latest error code issued from Windows for the conversation identified by the handle.

## Syntax

**DDEhGetLastError(Handle)**

*Handle*:

The integer handle that identifies the DDE conversation, returned from the [DDEhInitiate](#) function.

## Return Value

The error code last issued from Windows DDEML (for that conversation):

DMLERR_ADVACKTIMEOUT	0x4000
DMLERR_BUSY	0x4001
DMLERR_DATAACKTIMEOUT	0x4002
DMLERR_DLL_NOT_INITIALIZED	0x4003
DMLERR_DLL_USAGE	0x4004
DMLERR_EXECACKTIMEOUT	0x4005
DMLERR_INVALIDPARAMETER	0x4006
DMLERR_LOW_MEMORY	0x4007
DMLERR_MEMORY_ERROR	0x4008

DMLERR_NOTPROCESSED	0x4009
DMLERR_NO_CONV_ESTABLISHED	0x400a
DMLERR_POKEACKTIMEOUT	0x400b
DMLERR_POSTMSG_FAILED	0x400c
DMLERR_REENTRANCY	0x400d
DMLERR_SERVER_DIED	0x400e
DMLERR_SYS_ERROR	0x400f
DMLERR_UNADVACKTIMEOUT	0x4010
DMLERR_UNFOUND_QUEUE_ID	0x4011

## Related Functions

[DDEhInitiate](#), [DDEhRequest](#), [DDEhPoke](#), [DDEhTerminate](#)

## Example

See [DDEhInitiate](#).

## See Also

[Dynamic Data Exchange Functions](#)

## DDEhInitiate

Starts a conversation with an external Windows application. When the data exchange is complete, you should terminate the conversation to free system resources.

## Syntax

**DDEhInitiate(*sApplication*, *sDocument*)**

*sApplication*:

The application name (.EXE filename), for example, "WinWord".

*sDocument*:

The document, topic, or file name.

## Return Value

An integer handle for the conversation between Plant SCADA and the other application, or -1 if the conversation

is not started successfully. The handle is used by the other *DDEh...* functions, to identify the conversation.

## Related Functions

[DDEhExecute](#), [DDEhRequest](#), [DDEhPoke](#), [DDEhTerminate](#), [DDEhGetLastError](#)

## Example

```
! Read from Excel spreadsheet
STRING FUNCTION GetExcelData();
    INT hChannel;
    STRING sData;
    hChannel = DDEhInitiate("EXCEL", "DATA.XLS");
    IF hChannel > -1 THEN
        sData = DDEhRequest(hChannel, "R1C1");
        DDEhTerminate(hChannel);
        hChannel = -1;
    END;
    RETURN sData;
END

! Write to Excel spreadsheet
FUNCTION SetExcelData(STRING sData);
    INT hChannel;
    hChannel = DDEhInitiate("EXCEL", "DATA.XLS");
    IF hChannel > -1 THEN
        DDEhPoke(hChannel, "R1C1", sData);
        DDEhTerminate(hChannel);
        hChannel = -1;
    END;
END

! Execute Excel Macro
FUNCTION DoExcelMacro();
    INT hChannel;
    hChannel = DDEhInitiate("EXCEL", "DATA.XLS");
    IF hChannel > -1 THEN
        DDEhExecute(hChannel, "[RUN(^"TestMacro^")]");
        DDEhTerminate(hChannel);
        hChannel = -1;
    END;
END
```

## See Also

[Dynamic Data Exchange Functions](#)

### DDEhPoke

Writes a value to an external Windows application, for example, an Excel spreadsheet. The value is written once to the application. (To write the value dynamically, you need to call this function at the rate at which the data needs to be updated.)

You need to first start a conversation with the [DDEhInitiate](#) function, and use the handle returned by that

function to identify the conversation.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

## Syntax

**DDEhPoke**(*Handle*, *sItem*, *sValue*)

*Handle*:

The integer handle that identifies the DDE conversation, returned from the [DDEhInitiate](#) function.

*sItem*:

A unique name for the item; for example, the variable name, field name, or spreadsheet cell position.

*sValue*:

The value of the item.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[DDEhInitiate](#), [DDEhExecute](#), [DDEhRequest](#), [DDEhTerminate](#), [DDEhGetLastError](#)

## Example

See [DDEhInitiate](#).

## See Also

[Dynamic Data Exchange Functions](#)

## DDEhReadLn

Reads a line of text from a DDE Conversion, for example, from an Excel spreadsheet. You need to first start a conversation with the [DDEhInitiate](#) function, and use the handle returned by that function to identify the conversation. This function allows you to read a large amount of data via DDE. Keep calling the function until an empty string is returned to verify that all the data has been read.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

## Syntax

**DDEhReadLn**(*Handle*, *sTopic*)

*Handle*:

The integer handle that identifies the DDE conversation, returned from the [DDEhInitiate](#) function.

*sTopic*:

A unique topic name for the item; for example, the variable name, field name, or spreadsheet cell position.

## Return Value

A line of data, or an empty string when all data has been read.

## Related Functions

[DDEhSetMode](#), [DDEhWriteLn](#), [DDEhInitiate](#), [DDEhExecute](#), [DDEhRequest](#), [DDEhTerminate](#), [DDEhGetLastError](#)

## Example

See [DDEhWriteLn](#).

## See Also

[Dynamic Data Exchange Functions](#)

## DDEhRequest

Reads a value from an external Windows application, for example, from an Excel spreadsheet. You need to first start a conversation with the [DDEhInitiate](#) function, and use the handle returned by that function to identify the conversation.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

## Syntax

**DDEhRequest**(*Handle*, *sItem*)

*Handle*:

The integer handle that identifies the DDE conversation, returned from the [DDEhInitiate](#) function.

*sItem*:

A unique name for the item; for example, the variable name, field name, or spreadsheet cell position.

## Return Value

A string of data, or an empty string if the function cannot read the value.

## Related Functions

[DDEhInitiate](#), [DDEhExecute](#), [DDEhPoke](#), [DDEhTerminate](#), [DDEhGetLastError](#)

## Example

See [DDEhInitiate](#).

## See Also

[Dynamic Data Exchange Functions](#)

### DDEhSetMode

Set the mode of the DDE conversation. The default mode of a DDE conversation is to use TEXT data format - a simple string of data. This function allows you to set the mode to CSV (Comma Separated Values). Some Windows applications support this mode of data as it helps them to separate the data. For example, when you send CSV format to Excel, each value will be placed into a unique cell. If you use TEXT mode all the data will be placed into the same cell.

## Syntax

**DDEhSetMode**(*Handle*, *sMode*)

*Handle*:

The integer handle that identifies the DDE conversation, returned from the DDEhInitiate function.

*sMode*:

The mode of the DDE conversation:

1 - Text (default)

2 - CSV

## Return Value

The error code.

## Related Functions

[DDEhInitiate](#), [DDEhExecute](#), [DDEhRequest](#), [DDEhTerminate](#), [DDEhPoke](#), [DDEhReadLn](#), [DDEhWriteLn](#), [DDEhSetMode](#)

## Example

See [DDEhWriteLn](#).

## See Also

[Dynamic Data Exchange Functions](#)

### DDEhTerminate

Closes the conversation identified by the handle, and frees the resources associated with that conversation. After you call this function, the handle is no longer valid.

With Network DDE, you might need to terminate and re-initiate a conversation. For example, if you delete rows

on an MS Access sheet, the deleted rows display as #DELETED until you terminate and re-initiate the conversation.

## Syntax

**DDEhTerminate(*Handle*)**

*Handle*:

The integer handle that identifies the DDE conversation, returned from the [DDEhInitiate](#) function.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[DDEhInitiate](#), [DDEhExecute](#), [DDEhPoke](#), [DDEhRequest](#),

## Example

See [DDEhInitiate](#).

## See Also

[Dynamic Data Exchange Functions](#)

## DDEhWriteLn

Writes a line of text to the DDE conversation. With this function, you can write any amount of text to the DDE conversation. Call this function once for each line of text. To terminate the block of text, call this function and pass an empty string.

## Syntax

**DDEhWriteLn(*Handle*, *sTopic*, *sData*)**

*Handle*:

The integer handle that identifies the DDE conversation, returned from the [DDEhInitiate](#) function.

*sTopic*:

A unique name for the topic the data will be written to; for example, the spreadsheet cell position. The topic is only used when you complete the write by passing an empty string for data.

*sData*:

The line of data to write. To terminate the data and make Plant SCADA send the data, set the data to an empty string.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[DDEhInitiate](#), [DDEhExecute](#), [DDEhRequest](#), [DDEhTerminate](#), [DDEhGetLastError](#), [DDEhPoke](#), [DDEhReadLn](#),  
[DDEhWriteLn](#), [DDEhSetMode](#)

## Example

```
! Write to Excel spreadsheet
! write the numbers 1..8 into 8 unique cells in Excel.
FUNCTION WriteExcelData(STRING sData);
    INT hChannel;
    hChannel = DDEhInitiate("EXCEL", "DATA.XLS");
    IF hChannel > -1 THEN
        // set to CSV mode so EXCEL will put each value in a cell
        DDEhSetMode(hChannel, 2);
        DDEhWriteLn(hChannel, "", "1,2,3,4");
        DDEhWriteLn(hChannel, "R1C1:R2C4", "5,6,7,8");
        DDEhWriteLn(hChannel,"R1C1:R2C4","");
        DDEhTerminate(hChannel);
        hChannel = -1;
    END;
END
```

## See Also

[Dynamic Data Exchange Functions](#)

## DDEPost

Makes a Plant SCADA variable value available for DDE linking (which means it posts a DDE link so that it can be read by other DDE compliant applications running on the same computer). This sets up Plant SCADA to behave as a DDE Server for this DDE channel.

**Note:** This function is disabled unless the parameter [DDE]Enable is set to 1 in the Citect.ini file.

After a value is posted, other Windows applications running on the same computer can read the value by using their own DDE Client functions. If the value of the posted variable changes, any linked applications are informed of the new value.

To link to this value from any DDE Client applications running on the same computer, they need to appropriately use the DDE Client syntax with:

- "Citect" as the **<DDE Server application name>**
- "Data" as the **<DDE Topic name>**
- The name used for the first parameter **sItem** in this DDEPost() function as the **<DDE data item name>**.

Unlike the DDERead() and DDEWrite() Cicode functions which are static, the DDEPost() function can be used to

create a dynamic DDE link, providing the DDE Client applications appropriately set their side of the DDE channel to be automatically updated.

## Syntax

**DDEPost(*sItem*, *sValue*)**

*sItem*:

A unique name for the item; for example, the variable name, field name, or spreadsheet cell position.

*sValue*:

The value of the item.

## Return Value

The value that is posted, or empty string if the function does not succeed in posting the link.

## Related Functions

[DDEExec](#), , [DDEWrite](#)

## Example

```
! In Cicode, post a string "PV1" for external DDE
applications to connect with DDEPost("TAGONE",PV1);
/* To link to this posted tag from a cell in Excel, set the cell to
=Citect|Data!TAGONE. This will set the value of the Excel cell to the string "PV1". */
/* To link to this posted tag from a field in Word, set the field
to{DDEAuto Citect Data TAGONE}. This will set the value of the
field link to the string "PV1". */
```

### See Also

[Dynamic Data Exchange Functions](#)

## DDERead

Reads values from an external DDE compliant Windows application running on the same computer, (for example, from an Excel spreadsheet cell or a Word document).

This is a one-way static communication which is read once from the application per call. To read the value dynamically, call this function at the rate at which the data is required to be updated.

Use this function when you want precise control over exactly what you want from the DDE exchange.

## Syntax

**DDERead(*sApplication*, *sDocument*, *sItem* [, *Mode*] )**

*sApplication*:

The application name (.EXE filename), for example, "WinWord".

*sDocument:*

The document, topic, or file name.

*sItem:*

A unique name for the item; for example, the variable name, field name, or spreadsheet cell position.

*Mode:*

A flag that tells the application whether or not to set up an advise loop:

0 - Do not set up advise loop.

1 - Set up advise loop (default).

## Return Value

The value (from the external application) as a string, or an empty string if the function cannot read the desired values.

## Related Functions

[DDEExec](#), [DDEPost](#), [DDEWrite](#)

## Example

```
/* Read the value from R1C1 (Row1,Column1) of an Excel spreadsheet
named "Sheet1". */
DDERead("Excel","Sheet1","R1C1");
/* Read the value from the Item1 bookmark of the Word document
named "Recipes.doc". */
DDERead("Winword","Recipes","Item1");
```

## See Also

[Dynamic Data Exchange Functions](#)

## DDEWrite

Writes a value to an external Windows application, for example, to an Excel spreadsheet. The value is written once to the application. To write the value dynamically, you need to call this function at the rate at which the data needs to be updated.

Use DDEWrite() to cause Plant SCADA runtime to initiate the DDE conversation with a DDE compliant application running on the same computer.

## Syntax

**DDEWrite(*sApplication*, *sDocument*, *sItem*, *sValue*)**

*sApplication:*

The application name (.EXE filename), for example, "WinWord".

*sDocument:*

The document, topic, or file name.

*sItem:*

A unique name for the item; for example, the variable name, field name, or spreadsheet cell position.

*sValue:*

The value of the item.

## Return Value

The value that is sent to the other application, or an empty string if the function does not successfully write the value.

## Related Functions

[DDEExec](#), [DDEPost](#)

## Example

```
/* Write the value of a Plant SCADA variable named
TAGONE to R1C1 (Row1,Column1) of an Excel spreadsheet named
"Sheet1". The value is in string format. */
DDEWrite("Excel","Sheet1","R1C1",TAGONE);
```

## See Also

[Dynamic Data Exchange Functions](#)

## Device Functions

Following are functions relating to devices:

<a href="#">DevAppend</a>	Appends a blank record to the end of a device.
<a href="#">DevClose</a>	Closes a device.
<a href="#">DevControl</a>	Controls a dBASE or SQL device.
<a href="#">DevCurr</a>	Gets the current device number.
<a href="#">DevDelete</a>	Deletes the current record in a database device.
<a href="#">DevDisable</a>	Disables (and re-enables) a device from any access.
<a href="#">DevEOF</a>	Checks for the end of a file.
<a href="#">DevFind</a>	Finds a record in a device.

<a href="#">DevFirst</a>	Finds the first record in a device.
<a href="#">DevFlush</a>	Flushes buffered data to a device.
<a href="#">DevGetField</a>	Gets field data from the current record.
<a href="#">DevHistory</a>	Renames a device file and any subsequent history files.
<a href="#">DevInfo</a>	Gets device information.
<a href="#">DevModify</a>	Modifies the attributes of a device.
<a href="#">DevNext</a>	Gets the next record in a device.
<a href="#">DevOpen</a>	Opens a device for access.
<a href="#">DevOpenGrp</a>	This function has been deprecated.
<a href="#">DevPrev</a>	Gets the previous record in a device.
<a href="#">DevPrint</a>	Prints free-format data to a group of devices.
<a href="#">DevRead</a>	Reads characters from a device.
<a href="#">DevReadLn</a>	Reads a line of characters from a device.
<a href="#">DevRecNo</a>	Gets the current record number of a device.
<a href="#">DevSeek</a>	Moves to any record in a device.
<a href="#">DevSetField</a>	Sets new field data in the current record.
<a href="#">DevSize</a>	Gets the size of a device.
<a href="#">DevWrite</a>	Writes a string to a device.
<a href="#">DevWriteLn</a>	Writes a string with a newline character to a device.
<a href="#">DevZap</a>	Zaps a device.
<a href="#">Print</a>	Prints a string in a report.
<a href="#">PrintLn</a>	Prints a string with a newline character in a report.
<a href="#">PrintFont</a>	Changes the printing font on the current device.

## See Also

[Cicode Function Categories](#)[Using Cicode Functions](#)

## DevAppend

Appends a blank record to the end of a device. After the record is appended, you can use the [DevSetField](#) function to add data to fields in the record.

**Note:** For SQL devices this function is a blocking Cicode function. Data should be added to all requested fields in the row via the Cicode function [DevSetField](#) before [DevAppend](#) is applied.

You need to first call the [DevOpen](#) function to get the device handle (hDev).

## Syntax

**DevAppend(hDev)**

*hDev*:

The device handle, returned from the [DevOpen](#) function. The device handle identifies the table where all data on the associated device is stored.

## Return Value

0 (zero) if the record is successfully appended, otherwise an error code is returned.

## Related Functions

[DevOpen](#), [DevSetField](#)

## Example

```
INT FUNCTION WriteAlarmCount( INT hDevice, STRING sAlarm, INT iCount, INT iTime )
    DevAppend(hDevice);
    DevSetField(hDevice, "ALARM", sAlarm);
    DevSetField(hDevice, "TIME", IntToStr(iTime));
    DevSetField(hDevice, "COUNT", IntToStr(iCount));
END
```

For SQL devices the above example will not work. Instead use the following example:

```
INT FUNCTION WriteAlarmCount( INT hSqlDevice, STRING sAlarm, INT iCount, INT iTime )
    DevSetField(hSqlDevice, "ALARM", sAlarm);
    DevSetField(hSqlDevice, "TIME", IntToStr(iTime));
    DevSetField(hSqlDevice, "COUNT", IntToStr(iCount));
    DevAppend(hSqlDevice);
END
```

## See Also

[Device Functions](#)

## DevClose

Closes a device. Any data in the buffer is flushed to the device before it is closed. After a device is closed, its

device handle becomes invalid and cannot be used.

For SQL devices, this function is a blocking Cicode function.

## Syntax

**DevClose(*hDev*, *Mode*)**

*hDev*:

The device handle, returned from the [DevOpen](#) function. The device handle identifies the table where data on the associated device is stored.

*Mode*:

The mode of the close:

0 - Close the device in user mode - the default mode if none is specified. A device opened by Cicode function DevOpen need to be closed in this mode.

1 - Close the device in remove logging mode - under this mode, the current device will be rolled over to history files immediately. You should only use this mode in a report.

2 - Close the device in keep logging mode - under this mode, the current device will not be rolled over to history files. This allows subsequent messages to be written to the same file. This mode is used internally in a report written in rich text format (rtf).

---

**Note:** Do not call DevClose to the current device in an rtf report. This may make the output file unreadable.

---

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[DevOpen](#)

## Example

```
DevClose(hDev);
```

## See Also

[Device Functions](#)

## DevControl

Controls a dBASE or SQL device. You can pack a dBASE device to physically remove deleted records, or re-index a dBASE device to regenerate the keys. You can issue queries to an SQL device, or get the error status of the last SQL query.

For SQL devices, this function is a blocking Cicode function.

## Syntax

**DevControl(*hDev*, *Type* [, *sData*])**

*hDev*:

The device handle, returned from the [DevOpen](#) function. The device handle identifies the table where all data on the associated device is stored.

*nType*:

The type of command:

0 - Re-index the device based on the key defined in the device record (dBASE devices only).

1 - Pack the database file - all deleted records are removed (dBASE devices only).

2 - Issue a direct SQL query to the device (SQL devices only).

3 - Get error status of the last SQL query (SQL devices only).

---

**Note:** ASCII files and printers are not supported.

---

*sData*:

The command data, that is the SQL query to be issued. Used only for Type 2 commands.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[DevOpen](#), [DevZap](#)

## Example

```
! pack a dBASE file device
DevControl(hDev, 1, "");
```

## See Also

[Device Functions](#)

## DevCurr

Gets the current device handle. You can only call this function in a report, to get the handle of the device where the report is logging. You can then use the other device functions (for example, [DevPrint](#)) to access that logging device. (To get the handle of a device other than a logging device, you need to use the [DevOpen](#) function.)

If the report is logging to a group of devices, this function will return the group handle. However, not all device functions support group handles, for example, you cannot read from a group of devices.

## Syntax

**DevCurr()**

## Return Value

The current device handle or group handle.

If no device is configured, -1 is returned.

## Related Functions

[DevOpen](#), [DevPrint](#)

## Example

```
! Get the report device number.  
hDev=DevCurr();
```

## See Also

[Device Functions](#)

## DevDelete

Deletes the current record in a dBASE database device. The record is not physically deleted, but is marked for deletion. You can physically delete the record by packing the database with the [DevControl](#) function.

For SQL devices, this function is a blocking Cicode function.

## Syntax

**DevDelete(*hDev*)**

*hDev*:

The device handle, returned from the [DevOpen](#) function. The device handle identifies the table where all data on the associated device is stored.

## Return Value

0 (zero) if the record is successfully deleted, otherwise an error code is returned.

## Related Functions

[DevOpen](#), [DevClose](#), [DevControl](#)

## Example

```
! Delete the current record.  
DevDelete(hDev);
```

## See Also

[Device Functions](#)

### DevDisable

Disables (and re-enables) a device from all access, and discards any data written to the device. When a device is disabled, it cannot be opened, and data cannot be read from the device. Use this function to disable logging to a database or printer.

The *State* argument is a toggle. A *State* of 1 disables the device(s), but you can then re-enable the device(s) by repeating the function with *State* = 0.

## Syntax

**DevDisable(*sName*, *State*)**

*sName*:

The device name, or \* (asterisk) for all devices.

*State*:

The disable state:

0 - Enable the device.

1 - Disable the device.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[DevOpen](#)

## Example

```
! Disable the AlarmLog device.  
DevDisable("AlarmLog",1);  
:  
DevDisable("AlarmLog",0); ! Re-enable the device.
```

## See Also

[Device Functions](#)

## DevEOF

Gets the status of the end of file (EOF) flag for a device. When you use the [DevPrev](#), [DevNext](#), or [DevSeek](#) function, the start or end of the device will eventually be reached, and the EOF flag will be set. Use this function to test the EOF flag.

## Syntax

**DevEOF(*hDev*)**

*hDev*:

The device handle, returned from the [DevOpen\(\)](#) function. The device handle identifies the table where all data on the associated device is stored.

## Return Value

1 if the EOF flag has been set, otherwise 0 (zero).

## Related Functions

[DevOpen](#), [DevPrev](#), [DevNext](#), [DevSeek](#), [DevReadLn](#)

## Example

```
hDev = DevOpen("Log", 0);
WHILE NOT DevEOF(hDev) DO
    Prompt(DevGetField(hDev, "Tag"));
    DevNext(hDev);
END
DevClose(hDev);
```

## See Also

[Device Functions](#)

## DevFind

Searches a device for a record that contains specified data in a specified field. The search starts at the current record and continues forward until the matched data is found or the end of the database is reached. If the file has a keyed index, an indexed search is used.

For SQL devices, this function is a blocking Cicode function.

## Syntax

**DevFind(*hDev*, *sFind*, *sField*)**

*hDev*:

The device handle, returned from the DevOpen() function. The device handle identifies the table where all data on the associated device is stored.

*sFind:*

The data to find in *sField*, as a string.

For SQL devices: The DevFind() function can distinguish between numbers, strings, and dates, so you do not need to enclose the data in quote marks. Dates and times need to be in the correct format:

- Date: YYYY-MM-DD
- Time: HH:MM:SS
- DateTime: YYYY-MM-DD HH:MM:SS[.F...] (The fraction .F... is optional.)

*sField:*

The field name to match.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[DevOpen](#), [DevSeek](#)

## Example

```
! Find the Ice cream recipe.  
DevNotFound=DevFind(hDev,"Ice cream","Recipe");  
IF DevNotFound=0 THEN  
    ! Get the recipe values.  
    ..  
ELSE  
    Prompt("Ice cream not found");  
END
```

## See Also

[Device Functions](#)

## DevFirst

Finds the first record in a device.

For SQL devices, this function is a blocking Cicode function.

## Syntax

**DevFirst(*hDev*)**

*hDev:*

The device handle, returned from the [DevOpen](#) function. The device handle identifies the table where all data on the associated device is stored.

## Return Value

The first indexed record (if the device is an indexed database), otherwise the first record in the device.

## Related Functions

[DevOpen](#), [DevClose](#)

## Example

```
! Find the first record.  
FirstRec = DevFirst(hDev);
```

### See Also

[Device Functions](#)

## DevFlush

Flushes buffered data to the physical device. Plant SCADA normally optimizes the writing of data for maximum performance, so use this function only if it is really necessary.

## Syntax

**DevFlush(*hDev*)**

*hDev*:

The device handle, returned from the [DevOpen](#) function. The device handle identifies the table where data on the associated device is stored.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[DevOpen](#), [DevClose](#)

## Example

```
! Flush device to disk.  
DevFlush(hDev);
```

## See Also

[Device Functions](#)

### DevGetField

Gets field data from the current record in a device.

## Syntax

**DevGetField(*hDev*, *sField*)**

*hDev*:

The device handle, returned from the [DevOpen](#) function. The device handle identifies the table where all data on the associated device is stored.

*sField*:

The field name, as a string of up to 10 characters. (The dBASE file format limits all field names to a maximum of 10 characters.)

## Return Value

The field data (as a string). If the field is not found an empty string is returned.

## Related Functions

[DevOpen](#), [DevSetField](#)

## Example

```
INT
FUNCTION
GetRecipe(STRING sName)
    INT hDev;
    hDev = DevOpen("Recipe", 0);
    IF hDev >= 0 THEN
        DevSeek(hDev, 1);
        IF DevFind(hDev, sName, "NAME") = 0 THEN
            PLC_FLOUR = DevGetField(hDev, "FLOUR");
            PLC_WATER = DevGetField(hDev, "WATER");
            PLC_SALT = DevGetField(hDev, "SALT");
            PLC_MILK = DevGetField(hDev, "MILK");
        ELSE
            DspError("Cannot Find Recipe " + sName);
        END
        DevClose(hDev);
    ELSE
        DspError("Cannot open recipe database");
    END
END
```

## See Also

[Device Functions](#)

### DevHistory

Renames a device file and any subsequent history files. The current device is closed and renamed as the first history file. For example, the device file 'Templog.txt' is renamed as 'Templog.001'. If a history file 'Templog.001' already exists, it is renamed as 'Templog.002', and so on. The next time data is written to the device, a new device file is created.

---

**Note:** If the device file has not been created (that is data has not been written to the device), only existing history files are renamed. Use this function for direct control of the device history process.

The DevHistory function does not support SQL devices.

## Syntax

**DevHistory(*hDev*)**

*hDev*:

The device handle, returned from the [DevOpen](#) function. The device handle identifies the table where all data on the associated device is stored.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[DevOpen](#), [DevControl](#)

## Example

```
! Create history file
DevHistory(hDev);
```

## See Also

[Device Functions](#)

### DevInfo

Gets information on a device.

## Syntax

**DevInfo(*hDev*, *nType*)**

*hDev*:

The device handle, returned from the [DevOpen](#) function. The device handle identifies the table where all data on the associated device is stored.

*nType*:

Type of information:

-*n*: Name of field *n* (where *n* is any number up to the total number of fields). For example, if there are 10 fields, -7 will return the name of field 7.

- (*Total no. of fields + n*): Length of field *n* (where *n* is any number up to the total number of fields). For example, if there are 10 fields, -15 will return the length of field 5.

0: Device Name

1: Format

2: Header

3: File Name

4: Number of history files

5: Form length

6: Number of fields

7: Disable flag

8: Device type

9: Record size

10: Format number

11: Type of history schedule:

- 0: Event triggered
- 1: Daily
- 2: Weekly
- 3: Monthly
- 4: Yearly

12: The history period, in seconds, or week day, month or year, for example, if history is weekly then this is the day of the week, that is 1 to 7

13: Synchronisation time of day of the history in seconds, for example, 36000 (that is, 10:00:00)

14: The time the next history file will be created in seconds

## Return Value

The device information as a string if successful, otherwise an empty string is returned.

## Related Functions

[DevControl](#)

## Example

```
! Get the number of fields in a device.  
NoFields=DevInfo(hDev,6);  
FOR I=1 TO NoFields DO  
    ! Get and display the name of each field.  
    sField=DevInfo(hDev,-I);  
    nLength=DevInfo(hDev,-I - NoFields);  
    Prompt("Field Name "+sField + "Length " + nLength:##);  
END
```

## See Also

[Device Functions](#)

### DevModify

Modifies the attributes of a device. The device needs to be closed before you can modify a device.

This function allows you to dynamically change the file name or other attributes of a device at run time. You can use a single device to access many files. For example, you can create a device called Temp with a file name of TEMP.DBF. Using this function you could dynamically change the file name to access any dBASE file.

This function is useful in conjunction with the FormOpenFile or FormSaveAsFile functions. (These functions allow the operator to select file names easily.)

When using this function, you should be careful that no other Cicode function is already using the same device. Check the return value of this function before opening the device or you will destroy the data in the device to which it is already attached. If the device is already open, calling DevModify will return an error (and raise a hardware alarm to notify user).

If DevModify returns error, it means it has not modified the device and the device parameters will remain as they were before the call to DevModify.

Use a semaphore to help protect your Cicode.

For SQL devices, this function is a blocking Cicode function.

## Syntax

**DevModify(*sName*, *Format*, *Header*, *FileName*, *nType*)**

*Name:*

The name of the device.

*Format:*

A new format for the device or "\*" to use the existing format. See Format Templates for more information.

*Header:*

A new header for the device or "\*" to use the existing header.

*FileName:*

A new file name for the device or "\*" (asterisk) to use the existing filename.

*nType:*

A new device type, or -1 to use the existing device type.

Device Type	Device
ASCII_DEV	ASCII file
PRINTER_DEV	Printer
dBASE_DEV	dBASE file
SQL_DEV	SQL database

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[DevOpen](#), [DevClose](#), [DevSetField](#), [DevInfo](#), [DevAppend](#), [FormOpenFile](#)

## Example

```
! change the file name of MyDev
DevModify("MyDev", "*", "*", "c:\data\newfile.dbf", -1);
! change the fields and file name of MyDev
DevModify("MyDev", "{time}{date}{tags}", "*",
"C:\DATA\OLDFILE.DBF", -1);
! change the device to TXT file
DevModify("MyDev", "*", "*", "C:\DATA\OLDFILE.TXT", ASCII_DEV);
```

## See Also

[Device Functions](#)

## DevNext

Gets the next record in a device. If the end of the database is reached, the EOF flag is set and an error code is returned.

For SQL devices, this function is a blocking Cicode function.

## Syntax

**DevNext(*hDev*)**

*hDev*:

The device handle, returned from the [DevOpen](#) function. The device handle identifies the table where all data on the associated device is stored.

## Return Value

0 if the next record is read, or an error if the end of the database is reached.

## Related Functions

[DevEOF](#), [DevPrev](#)

## Example

```
Status=0;  
I = 0;  
hDev = DevOpen("Log", 0);  
WHILE Status = 0 DO  
    DspText(20 + I, 0, DevGetField(hDev,"Tag"));  
    I = I + 1;  
    Status = DevNext(hDev);  
END  
DevClose(hDev);
```

## See Also

[Device Functions](#)

## DevOpen

Opens a device and returns the device handle. The device needs to be defined in the Plant SCADA database. If the device cannot be opened, and user error checking is not enabled, the current Cicode task is halted.

You can use this function to return the handle of a device that is already open. The DevOpen function does not physically open another device - it returns the same device handle as when the device was opened. The mode of the second open call is ignored. To re-open an open device in a different mode, you need to first close the device and then re-open it in the new mode.

For SQL devices, this function is a blocking Cicode function. When using an ODBC driver to connect to an SQL server or database, experience has shown that connecting only once on startup and not closing the device yields the best performance. ODBC connection is slow and if used on demand may affect your system's performance. Also, some ODBC drivers may leak memory on each connection and may cause errors after a number of re-connects.

---

**Note:** If you are opening a database device in indexed mode (nMode=2), an index file will automatically be created by Plant SCADA if one does not already exist. If you feel a device index has become corrupt, delete the existing index file and a new one will be created the next time the DevOpen function is run.

---

## Syntax

**DevOpen(sName [, nMode] )**

*Name:*

The name of the device.

***nMode:***

The mode of the open:

0 - Open the device in shared mode - the default mode when opening a device if none is specified.

1 - Open the device in exclusive mode. In this mode only one user can have the device open. The open will return an error if another user has the device open in shared or exclusive mode.

2 - Open the device in indexed mode. In this mode the device will be accessed in index order. This mode is only valid if the device is a database device and has an index configured in the Header field at the Devices form. Please be aware that specifying mode 2 when opening an ASCII device is ignored internally.

4 - Open the device in 'SQL not select' mode. If opened in this mode, you need to not attempt to read from an SQL device.

8 - Open the device in logging mode. In this mode the history files will be created automatically.

16 - Open the device in read only mode. In this mode data can be viewed, but not written. This mode is supported only by DBF and ASCII files - it is ignored by printers and SQL/ODBC databases.

## Return Value

The device handle. If the device cannot be opened, -1 is returned. The device handle identifies the table where all data on the associated device is stored.

## Related Functions

[DevClose](#)

## Example

```
FUNCTION
PrintRecipe(STRING sCategory)
    STRING sRecipe;
    INT hRecipe, hPrinter;
    ErrSet(1); ! enable user error checking
    hRecipe = DevOpen("Recipe", 0);
    IF hRecipe = -1 THEN
        DspError("Cannot open recipe");
        RETURN FALSE;
    END
    hPrinter = DevOpen("Printer1", 0);
    IF hPrinter = -1 THEN
        DspError("Cannot open printer");
        RETURN FALSE;
    END
    ErrSet(0); ! disable user error checking
    WHILE NOT DevEof(hRecipe) DO
        sRecipe = DevReadLn(hRecipe);
        DevWriteLn(hPrinter, sRecipe);
    END
    DevClose(hRecipe);
    DevClose(hPrinter);
    RETURN TRUE;
END
```

## See Also

[Device Functions](#)

### DevPrev

Gets the previous record in a device. If the start of the database is reached, the EOF flag is set and an error code is returned.

## Syntax

**DevPrev(*hDev*)**

*hDev*:

The device handle, returned from the [DevOpen](#) function. The device handle identifies the table where all data on the associated device is stored.

## Return Value

0 if the record is read successfully, or an error code if the start of the database is reached.

## Related Functions

[DevOpen](#), [DevEOF](#), [DevNext](#)

## Example

```
Status=0;
I = 0;
hDev = DevOpen("Log", 0);
iError = DevSeek(hDev, DevSize(hDev)); ! seek to end
WHILE iError = 0 DO
    DspText(20 + I, 0, DevGetField(hDev,"Tag"));
    I = I + 1;
    iError = DevPrev(hDev);
END
DevClose(hDev);
```

## See Also

[Device Functions](#)

### DevPrint

Prints free-format data to groups of devices. Using this function, you can write data to many devices at the same time. You would normally use this function in a report.

## Syntax

**DevPrint(*hGrp*, *sData*, *NewLine*)**

*hGrp*:

The device handle, or the group handle for a group of devices.

*sData*:

The data to print to the group of devices.

*NewLine*:

The newline flag:

0 - Do not insert a newline character.

1 - Insert a newline character.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[DevWriteLn](#), [DevCurr](#)

## Example

```
! Get the report device number or group number (for a group of
devices).
hGrp=DevCurr();
! Print PV123 to a group of devices.
DevPrint(hGrp,"PV123="+PV123:###,1);
```

## See Also

[Device Functions](#)

## DevRead

Reads characters from a device. If the device is record-based, the current field is read. If the device is free-format, the specified number of characters is read. If the number of characters specified is greater than the number of characters remaining in the device, only the remaining characters are read.

The DevRead function does not support SQL devices. Use the [DevGetField](#) function for these devices.

## Syntax

**DevRead(*hDev*, *Length*)**

*hDev*:

The device handle, returned from the [DevOpen](#) function. The device handle identifies the table where all data on

the associated device is stored.

*Length:*

The number of characters to read.

## Return Value

The data (in string format). If the end of the device is found, an empty string is returned.

## Related Functions

[DevOpen](#), [DevReadLn](#), [DevFind](#)

## Example

```
! Read 20 characters from a device.  
Str=DevRead(hDev,20);
```

## See Also

[Device Functions](#)

## DevReadLn

Reads data from the current record of a device until the end of the line, or end of the record. If the device is record-based, the record number is incremented. The carriage return and newline characters are not returned.

The DevReadLn function does not support SQL devices. Use the [DevGetField](#) function for these devices.

## Syntax

**DevReadLn(*hDev*)**

*hDev:*

The device handle, returned from the [DevOpen](#) function. The device handle identifies the table where all data on the associated device is stored.

## Return Value

The data (in string format). If the end of the device is found, an empty string is returned and the EOF flag is set.

## Related Functions

[DevOpen](#), [DevRead](#), [DevEOF](#), [DevFind](#)

## Example

```
Str=DevReadLn(hDev);
```

## See Also

[Device Functions](#)

### DevRecNo

Gets the current record number of a device. If the device is record-based, the record number ranges from 1 to the maximum size of the file. If the device is free-format, the record number ranges from 0 to the maximum byte size -1.

The DevRecNo function does not support SQL devices. For these devices -1 is returned.

## Syntax

**DevRecNo(*hDev*)**

*hDev*:

The device handle, returned from the [DevOpen](#) function. The device handle identifies the table where all data on the associated device is stored.

## Return Value

The record number. If an error is detected while getting the record number, -1 is returned.

## Related Functions

[DevOpen](#), [DevSeek](#)

## Example

```
! Get the current record number.  
Rec=DevRecNo(hDev);
```

## See Also

[Device Functions](#)

### DevSeek

Moves the device pointer to a specified position in the device. If the device is a database, and it is opened in indexed mode, DevSeek will seek to the record number - not through the index. To locate the first record in an indexed device, call the [DevFirst](#) function.

## Syntax

**DevSeek(*hDev*, *Offset*)**

*hDev*:

The device handle, returned from the DevOpen() function. The device handle identifies the table where all data on the associated device is stored.

*Offset*:

The offset in the device. If the device is a database device, the offset is the record number. If the device is a binary device, the offset is in bytes (from 0 to the maximum file size -1).

**Note:** If offset causes a seek past the end of the file, DevSeek returns no error, but sets the EOF flag (that is, a subsequent DevEOF call will return true). For SQL devices, the function can use only either 0 or 1 offset (beginning of the table).

---

## Return Value

0 (zero) if the seek was successful, otherwise an error code is returned.

## Related Functions

[DevOpen](#), [DevEOF](#), [DevRecNo](#), [DevFirst](#)

## Example

```
hDev=DevOpen("Log", 0);
DevSeek(hDev,100);
DevGetField(hDev,"Tag");
! Gets the value of the "Tag" field at record 100.
```

## See Also

[Device Functions](#)

## DevSetField

Sets new field data in the current record in a device.

## Syntax

**DevSetField(*hDev*, *sField*, *sData*)**

*hDev*:

The device handle, returned from the [DevOpen](#) function. The device handle identifies the table where all data on the associated device is stored.

*sField*:

The field name, as a string of up to 10 characters. (The dBASE file format limits all field names to a maximum of

10 characters.)

*sData:*

New field data, in string format. Plant SCADA converts any other data type into a string before setting the data.

## Return Value

0 (zero) if the data is successfully set, otherwise an error code is returned.

## Related Functions

[DevOpen](#), [DevAppend](#), [DevGetField](#)

## Example

```
! Set the fields in the "Recipe" device.  
hDev=DevOpen("Recipe", 0);  
DevSeek(hDev, 1);  
DevSetField(hDev, "Name", "WhiteBread");  
DevSetField(hDev, "Flour", IntToStr(iFlour));  
DevSetField(hDev, "Water", iWater:####);  
DevSetField(hDev, "Salt", iSalt);  
DevClose(hDev);
```

## See Also

[Device Functions](#)

## DevSize

Gets the size of a physical device.

The DevSize function does not support SQL devices. For these devices -1 will be returned.

## Syntax

**DevSize(*hDev*)**

*hDev:*

The device handle, returned from the [DevOpen](#) function. The device handle identifies the table where all data on the associated device is stored.

## Return Value

If the device is a database device, the number of records is returned. If the device is a binary device, the number of bytes in the file is returned. If an error is detected, -1 is returned.

## Related Functions

[DevRecNo](#), [DevSeek](#)

## Example

```
INT NoRec;  
NoRec=DevSize(hDev);  
! Seek to the last record.  
DevSeek(hDev,NoRec);
```

## See Also

[Device Functions](#)

## DevWrite

Writes a string to a device. If the device is free-format, the data is written to the device as specified. If the device is record-based, the data is written to the current field, and the field pointer is moved to the next field.

Writing to a DBF device appends the data to the device.

Writing to a SQL device appends the data to the device only when all fields of the row have been written.

## Syntax

**DevWrite(*hDev, sData*)**

*hDev*:

The device handle, returned from the [DevOpen](#) function. The device handle identifies the table where all data on the associated device is stored.

*sData*:

The data to write, as a string.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DevOpen](#), [DevWriteLn](#)

## Example

```
! Write PV123 to the device.  
DevWrite(hDev,"PV123="+PV123:###.#);
```

For SQL devices: The DevWrite function can distinguish between numbers, strings, and dates, so you do not need to enclose the data in quote marks. Dates and times need to be in the correct format:

- Date: YYYY-MM-DD
- Time: HH:MM:SS
- DateTime: YYYY-MM-DD HH:MM:SS[.F...] (The fraction .F... is optional.)

## See Also

[Device Functions](#)

### DevWriteLn

Writes a string to a device. If the device is free-format, the data is written to the device, followed by a newline character. If the device is record-based, a new record is appended to the device and the data is written to this record. The record pointer is then moved to the next record.

## Syntax

**DevWriteLn(*hDev*, *sData*)**

*hDev*:

The device handle, returned from the [DevOpen](#) function. The device handle identifies the table where all data on the associated device is stored.

*sData*:

The data to write, as a string.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DevOpen](#), [DevWrite](#)

## Example

```
/* Write PV123 to the device followed by a newline character */
DevWriteLn(hDev, "PV123="+PV123:###.#);
```

## See Also

[Device Functions](#)

### DevZap

Zaps a device. If a database device is zapped, all records are deleted. If an ASCII file is zapped, the file is truncated to 0 (zero) length. Use this function when you want to delete all records in a database or file without

deleting the actual file.

For SQL devices, this function is a blocking Cicode function.

## Syntax

**DevZap(*hDev*)**

*hDev*:

The device handle, returned from the [DevOpen](#) function. The device handle identifies the table where all data on the associated device is stored.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DevDelete](#)

## Example

```
! Delete all records in the alarm log database.  
hDev = DevOpen("AlarmLog", 0);  
DevZap(hDev);
```

## See Also

[Device Functions](#)

## Print

Prints a string on the current device. You should call this function only in a report. The output is sent to the device (or group of devices) defined in the Reports database (in the output device field).

---

**Note:** To print a new line in an RTF report, use the "\par" special character. For example, Print("String" + "\par").

---

## Syntax

**Print(*String*)**

*String*:

The string (data) to print.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[PrintLn](#), [DevCurr](#), [DevPrint](#), [DevWrite](#)

## Example

```
! Print "Testvar" and stay on the same line.  
Print("Value of Testvar="+Testvar:##.#);
```

## See Also

[Device Functions](#)

## PrintFont

Changes the printing font on the current device. You should call this function only in a report. It will change the font style for the device (or group of devices) defined in the Reports database (output device field). It has effect only on reports being printed to a PRINTER\_DEV - it has no effect on other types of devices, such as ASCII\_DEV and dBASE\_DEV.

## Syntax

**PrintFont(*Font*)**

*Font*:

The Plant SCADA font (defined in the Fonts database).

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[Print](#)

## Example

The following report file...

```
{! example.rpt }  
-----  
AN example Report  
-----  
{CICODE}  
PrintFont("HeadingFont");  
{END}  
Plant Area 1  
{CICODE}
```

```
PrintFont("ReportFont");
{END}
{Time(1) } {Date(2) }
PV_1 {PV_1:#####.##}
PV_2 {PV_2:#####.##}
-----End of Report-----
...will print as...
-----
AN example Report
-----
Plant Area 1
04:41:56 19-10-93
PV_1 49.00
PV_2 65.00
-----End of Report-----
```

## See Also

[Device Functions](#)

## PrintLn

Prints a string on the current device, followed by a newline character. You should call this function only in a report. The output will be sent to the device or group of devices defined in the Reports database (in the output device field).

**Note:** To print a new line in an RTF report, use the "\par" special character. For example, PrintLn("String" + "\par").

## Syntax

**PrintLn(*String*)**

*String*:

The string (data) to print.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[Print](#), [DevCurr](#), [DevPrint](#), [DevWrite](#)

## Example

```
! Print "Testvar" followed by a new line.
PrintLn("Value of Testvar="+Testvar:##.#);
```

## See Also

[Device Functions](#)

## Display Functions

Following are functions relating to the display of graphics pages and objects:

<a href="#">DspAnCreateControlObject</a>	Creates a new instance of an ActiveX object. If the object already exists for the given Animation Point Number, then that object will be used (a new object is not created).
<a href="#">DspAnFree</a>	Frees (removes) an AN from the current page.
<a href="#">DspAnGetArea</a>	Gets the area configured for an object at a specific AN (animation-point number).
<a href="#">DspAnGetMetadata</a>	Retrieves the field value of the specified metadata entry.
<a href="#">DspAnGetMetadataAt</a>	Retrieves metadata information at the specified index.
<a href="#">DspAnGetPos</a>	Gets the x and y coordinates of an AN (animation-point number).
<a href="#">DspAnGetPrivilege</a>	Gets the privileges configured for an object at a specific AN (animation-point number).
<a href="#">DspAnInfo</a>	Gets information on the state of the animation at an AN.
<a href="#">DspAnInRgn</a>	Checks if an AN is within a specified region.
<a href="#">DspAnMove</a>	Moves an AN.
<a href="#">DspAnMoveRel</a>	Moves an AN relative to its current position.
<a href="#">DspAnSetName</a>	Sets the animation name using a valid AN and Name.
<a href="#">DspAnNew</a>	Creates an AN at the specified x and y coordinates.
<a href="#">DspAnNewRel</a>	Creates an AN relative to another AN.
<a href="#">DspAnSetMetadata</a>	Non-blocking function, that sets the value of the specified metadata entry.
<a href="#">DspAnSetMetadataAt</a>	Sets the value of a metadata entry.
<a href="#">DspBar</a>	Displays a bar graph at an AN.

DspBmp	Displays a bitmap at a specified AN.
DspButton	Displays a button at an AN and puts a key into the key command line (when the button is selected).
DspButtonFn	Displays a button at an AN and calls a function when the button is selected.
DspClearClip	Clears any previously specified clip rectangle for an AN.
DspChart	Displays a chart at an AN.
DspCol	DspCol is deprecated in this version.
DspDel	Deletes the objects at an AN.
DspDelayRenderBegin	Delays screen updating until DspDelayRenderEnd() is called.
DspDelayRenderEnd	Ends the screen update delay set by DspDelayRenderBegin().
DspDirty	Forces an update to an AN.
DspError	Displays an error message at the prompt AN.
DspFile	Defines the screen attributes for displaying a text file.
DspFileGetInfo	Gets the attributes of a file to screen display.
DspFileGetName	Gets the name of the file being displayed in the display "window".
DspFileScroll	Scrolls a file (displayed in the display "window") by a number of characters.
DspFileSetName	Sets the name of the file to display in the display "window".
DspFont	Creates a font.
DspFontHnd	Gets a font handle.
DspFullScreen	Enables or disables the fullscreen mode of the active window.
DspGetAnBottom	Gets the bottom extent of the object at the specified AN.
DspGetAnCur	Gets the current AN.

DspGetAnExtent	Gets the extent of the object at a specified AN.
DspGetAnFirst	Returns the first AN on the current page.
DspGetAnFromName	Returns the AN for an object using a valid name.
DspGetAnFromNameRelative	Name used to retrieve the AN of an object on the page relative to a given AN.
DspGetAnFromPoint	Gets the AN of the object at a specified set of screen coordinates.
DspGetAnHeight	Gets the height of the object at a specified AN.
DspGetAnLeft	Gets the left extent of the object at the specified AN.
DspGetAnNext	Returns the AN following a specified AN.
DspGetAnRight	Gets the right extent of the object at the specified AN.
DspGetAnTop	Gets the top extent of the object at the specified AN.
DspGetAnWidth	Gets the width of the object at a specified AN.
DspGetEnv	Gets a page environment variable.
DspGetMetadataFromName	Name used to retrieve the metadata of an object on the page.
DspGetMetadataFromNameRelative	Name used to retrieve the metadata of an object on the page relative to a given AN.
DspGetMouse	Gets the mouse position.
DspGetMouseOver	Determines if the mouse is within the boundaries of a given AN.
DspGetNameFromAn	Retrieves the Animation name of an object using a valid AN.
DspGetNearestAn	Gets the nearest AN.
DspGetParentAn	Gets the parent animation number (if any), for the specified AN.
DspGetSlider	Gets the current position (value) of a slider at an AN.
DspGetTip	Gets the tool tip text associated with an AN.
DspGrayButton	Greys and disables a button.
DspInfo	Gets object display information from an AN.

DspInfoDestroy	Deletes an object information block created by DspInfoNew().
DspInfoField	Gets stored and real-time data for a variable tag.
DspInfoNew	Creates an object information block for an AN.
DspInfoValid	Checks if an object information block is still valid.
DspIsButtonGray	Gets the current status of a button.
DspKernel	Displays the Kernel window.
DspMarkerMove	Moves a trend or chart marker to a specified position.
DspMarkerNew	Creates a new trend marker.
DspMCI	Controls a multimedia device.
DspPlaySound	Plays a waveform (sound).
DspPopupConfigMenu	Displays the contents of a menu node as a pop-up (context) menu, and runs the command associated with the selected menu item.
DspPopupMenu	Creates a menu consisting of a number of menu items.
DspRichText	Creates a Rich Text object at the animation point.
DspRichTextEdit	Enables/disables editing of the contents of a rich text object.
DspRichTextEnable	Enables/disables a rich text object.
DspRichTextGetInfo	Returns size information about a rich text object.
DspRichTextLoad	Loads a copy of a rich text file into a rich text object.
DspRichTextPgScroll	Scrolls the contents of a rich text object by one page length.
DspRichTextPrint	Prints the contents of a rich text object.
DspRichTextSave	Saves the contents of a rich text object to a file.
DspRichTextScroll	Scrolls the contents of a rich text object by a user defined amount.
DspRubEnd	Ends a rubber band selection.
DspRubMove	Moves a rubber band selection to the new position.
DspRubSetClip	Sets the clipping region for the rubber band display.

DspRubSetColor	Sets the color of the rubber band tool that is used to select a section within a trend object.
DspRubStart	Starts a rubber band selection (used to rescale a trend with the mouse).
DspSetClip	Sets the clip rectangle for an AN.
DspSetCurColor	Sets the color of the focus rectangle.
DspSetMetadataFromName	Name used to set the metadata of an object on the page.
DspSetMetadataFromNameRelative	Name used to set the metadata of an object on the page relative to the given AN.
DspSetPopupMenuFont	Sets the font for the text of the popup menu.
DspSetSlider	Sets the current position of a slider at the specified AN.
DspSetTip	Sets tool tip text associated with an AN.
DspSetTooltipFont	Sets the font for tool tip text.
DspStatus	Sets the communication status error for a specified animation number.
DspStr	Displays a string at an AN.
DspSym	Displays a symbol at an AN.
DspSymAnm	Displays a series of animated symbols at an AN.
DspSymAnmEx	Displays a series of animated symbols at an AN.
DspSymAtSize	Displays a symbol at a scale and offset from an AN.
DspText	Displays text at an AN.
DspTipMode	Switches the display of tool tips on or off.
DspTrend	Displays a trend at an AN.
DspTrendInfo	Gets information on a trend definition.
DspWebContentGetURL	Gets the current URL for a Web Content control on a graphics page.
DspWebContentSetURL	Sets the URL for a Web Content control on a graphics page.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

### DspAnCreateControlObject

Creates a new instance of an ActiveX object. If the object already exists for the given Animation Point Number, then that object will be used, that is a new object will not be created, the existing object will merely be refreshed.

AN object created using this function remains in existence until the page is closed or the associated Cicode Object is deleted.

## Syntax

**DspAnCreateControlObject(*nAN, sClass, Width, Height [, sEventClass]*)**

*nAN*:

The animation-point number.

*sClass*:

The class of the object. You can use the object's human readable name, its program ID, or its GUID. If the class does not exist, the function will return an error.

For example:

- "Calendar Control 8.0" - human readable name
- "MSCAL.Calendar.7" - Program ID
- "{8E27C92B-1264-101C-8A2F-040224009C02}" - GUID

*Width*:

The width of the ActiveX object.

*Height*:

The height of the ActiveX object.

*sEventClass*:

The string you would like to use as the event class for the object.

## Return Value

The newly created object, if successful, otherwise an error is generated.

## Related Functions

[CreateObject](#), [CreateControlObject](#)

## Example

See [CreateControlObject](#).

## See Also

[Display Functions](#)

### DspAnFree

Frees (removes) an AN from the current page. If an animation exists at the animation number, it is deleted before the AN is freed. Use this function to free existing ANs or ANs created with the DspAnNew() function. Please be aware that the ANs are only freed in memory - the change is not persistent. The next time the page is opened it will display the AN.

---

**Note:** This function was designed to only be used on Cicode objects, or the animation point object from the version 3.xx/4.xx toolbox.

---

## Syntax

**DspAnFree(*nAN*)**

*nAN*:

The animation-point number.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspAnNew](#)

## Example

```
/* Remove AN20 from the current page. */
DspAnFree(20);
```

## See Also

[Display Functions](#)

### DspAnGetArea

Gets the area configured for an object at a specific AN (animation-point number). The area is returned as an integer.

---

**Note:** This function does not return the areas of keyboard commands associated with the object.

---

## Syntax

**DspAnGetArea(*nAN*)**

*nAN*:

The animation-point number.

## Return Value

The area if successful, otherwise an error is returned. If the object is configured with 'Same area as page' checked, the area of the page will be returned. AN area of 0 (zero) means no areas are configured for the object.

## Related Functions

[DspAnGetPrivilege](#)

## Example

```
/* Get the area configured for the object at AN60. /
DspAnGetArea(60);
```

## See Also

[Display Functions](#)

## DspAnGetMetadata

Retrieves the field value of the specified metadata entry.

## Syntax

**DspAnGetMetadata(*nAN*, *sMetaName*)**

*nAN*:

An animation number that uniquely identifies an object. This object contains the list of metadata definitions that will be used to perform the association operations. When -2 is specified, it is equivalent to using DspGetAnCur(). (See for usage and limitations.)

*sMetaName*:

The name of the metadata entry for which to search.

---

**Note:** Before calling this function, it may be worthwhile to call ErrSet(1) to disable error checking as this function will generate a hardware error for any object that does not have a metadata entry 'sMetaName', and the cicode task will stop executing.

---

## Return Value

Value for the specified metadata. Returns empty string if a matching metadata entry is not defined and error code if unsuccessful.

## Related Functions

[DspAnGetMetadataAt](#), [DspAnSetMetadata](#), [DspAnSetMetadataAt](#)

## See Also

[Display Functions](#)

### DspAnGetMetadataAt

Retrieves metadata information at the specified index.

## Syntax

**DspAnGetMetadataAt(*nAN*, *nIndex* ,*sField*)**

*nAn*:

An animation number that uniquely identifies an object. This object contains the list of metadata definitions that will be used to perform the association operations. When -2 is specified, it is equivalent to using DspGetAnCur. (See [DspGetAnCur](#) for usage and limitations.)

*nIndex*:

The index of the metadata in the animation point. The index is 0-based; i.e. the first metadata entry has an index of 0, the next 1, and so on.

*sField*:

The name of the field from which to retrieve the information for the metadata. Supported fields are:

- Name
- Value

## Return Value

The field value string. If there is an error, an empty string is returned. The error code can be obtained by calling the IsError Cicode function.

## Related Functions

[DspAnGetMetadata](#), [DspAnSetMetadata](#), [DspAnSetMetadataAt](#)

## See Also

[Display Functions](#)

## DspAnGetPos

Gets the x and y coordinates of an AN, in pixels, relative to the top-left corner of the window.

## Syntax

**DspAnGetPos(*nAN*, *X*, *Y* [, *nMode*])**

*nAN*:

The animation-point number.

*X*, *Y*:

Variables used to store the x and y pixel coordinates of the AN, returned from this function.

*nMode*:

0 – (Default mode) Gets coordinates of an AN on the page. If the AN has not been displayed yet, e.g. default visibility state is hidden, (0, 0) is returned from calling the function.

1 – Gets coordinates of an AN from its configuration. i.e. page coordinates at which the object was originally inserted in Graphics Builder is returned from calling the function.

## Return Value

0 (zero) if successful, otherwise an error is returned. The *X* and *Y* variables are set to the AN's position if successful, or to -1 if an error has been detected.

## Related Functions

[DspAnMove](#), [DspAnInRgn](#), [DspGetAnCur](#), [DspGetMouse](#), [DspGetNearestAn](#), [PageTransformCoords](#)

## Example

```
/* Get the position of AN20 into X and Y. /
DspAnGetPos(20,X,Y);
```

## See Also

[Display Functions](#)

## DspAnGetPrivilege

Gets the privileges configured for an object at a specific AN (animation-point number). The privilege is returned as an integer.

**Note:** This function does not return the privileges of keyboard commands associated with the object.

## Syntax

**DspAnGetPrivilege(*nAN*)**

*nAN:*

The animation-point number.

## Return Value

The privilege if successful, otherwise an error is returned. A privilege of 0 (zero) means no privileges are configured for the object.

## Related Functions

[DspAnGetArea](#)

## Example

```
/* Get the privileges of the object at AN45. /
DspAnGetPrivilege(45);
```

## See Also

[Display Functions](#)

## DspAnInfo

Gets information on an AN - the type or state of the animation that is currently displayed.

## Syntax

**DspAnInfo(*nAN*, *nType*)**

*nAN:*

The animation-point number.

*nType:*

The type of information:

**0** - The type of animation currently displayed at the AN. The following is returned:

- **0** - No animation is displayed.
- **1** - Color is displayed.
- **2** - A bar graph is displayed.
- **3** - Text is displayed.
- **4** - A symbol is displayed.
- **5** - AN animation symbol is displayed.
- **6** - A trend is displayed.
- **7** - A button is displayed.
- **8** - A slider is displayed.

- 9 - A plot is displayed.

1 - The state of the animation currently displayed. If color is displayed, the color is returned. If a bar graph, trend, or symbol is displayed, the bar, trend, or symbol name is returned. If text is displayed, the font handle is returned.

2 - The value of the text or the name of a button at the given AN point is returned.

3 - The type of animation currently displayed at the AN for Mode 0 plus the following:

- 10 - Rich Edit
- 11 - Bitmap
- 12 - Straight Line
- 13 - Free hand Line
- 14 - Rectangle
- 15 - Ellipse
- 16 - Spark
- 17 - Group
- 18 - Windows Meta File
- 19 - Poly Line
- 20 - Dynamic object
- 21 - Pipe
- 22 - Symbol Set
- 23 - OCX
- 24 - Basic AN
- 25 - Number
- 26 - Advanced AN
- 27 - Keyboard AN
- 28 - Sizemove AN
- 29 - Touch AN

## Return Value

The animation information, which depends on the type passed argument, as described above, as a string.

## Related Functions

[DspGetAnCur](#)

## Example

```
IF DspAnInfo(25,0) = "1" THEN
    /* If color on AN 25, then get the color */
    col = DspAnInfo(25,1);
END
```

## See Also

[Display Functions](#)

### DspAnInRgn

Checks if an AN is within a region bounded by two ANs.

Will return False if any of the specified ANs are completely clipped. A partially clipped AN is treated as being fully visible.

## Syntax

**pAnInRgn(nAN, One, Two)**

*nAN*:

The animation-point number.

*One, Two*:

One - the AN at a corner of the region; two - the AN at the opposite corner of the region.

## Return Value

1 if the AN is within the region, or 0 (zero) if it is not.

## Example

```
DspGetMouse(X,Y);
DspAnMove(250,X,Y);
IF DspAnInRgn(250,20,30) THEN
    Prompt("Mouse in region bounded by AN20 and AN30");
ELSE
    Prompt("Mouse not in region");
END
```

## See Also

[Display Functions](#)

### DspAnMove

Moves an AN to a new position. Any animation at this AN is also moved.

---

**Note:** This function was designed to only be used on Cicode objects, or the animation point object from the version 3.xx/4.xx toolbox.

---

## Syntax

**DspAnMove(nAN, X, Y)**

*nAN:*

The animation-point number.

*X:*

The x pixel coordinates of the new position.

*Y:*

The y pixel coordinates of the new position.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspAnMoveRel](#)

## Example

```
DspAnMove(25,100,200);  
! Moves AN25 to pixel location 100,200.
```

## See Also

[Display Functions](#)

## DspAnMoveRel

Moves an AN relative to its current position. Any animation at this AN is also moved.

---

**Note:** This function was designed to only be used on Cicode objects, or the animation point object from the version 3.xx/4.xx toolbox.

---

## Syntax

**DspAnMoveRel(*nAN*, *X*, *Y*)**

*nAN:*

The animation-point number.

*X:*

The number of pixels to move the AN in the x plane.

*Y:*

The number of pixels to move the AN in the y plane.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspAnMove](#)

## Example

```
DspAnMoveRel(25,10,20);
/* Moves AN25 by 10 pixels to the right and 20 pixels downward,
relative to its current position. */
```

## See Also

[Display Functions](#)

## DspAnNew

Creates an AN at the specified x and y coordinates.

## Syntax

**DspAnNew(X, Y)**

X:

The x pixel coordinate where the new AN is created.

Y:

The y pixel coordinate where the new AN is created.

## Return Value

If successful, the new AN is returned. If the AN cannot be created, -1 is returned. If an AN already exists at this location, that AN is returned.

## Related Functions

[DspAnNewRel](#), [DspAnFree](#)

## Example

```
AN=DspAnNew(100,200);
DspSym(AN,20);
/* Displays symbol 20 at pixel location 100,200 */
```

## See Also

[Display Functions](#)

### DspAnNewRel

Creates an AN at a distance of x,y pixels from a specified AN.

**Note:** This function was designed to only be used on Cicode objects, or the animation point object from the version 3.xx/4.xx toolbox.

## Syntax

**DspAnNewRel(*nAN*, *X*, *Y*)**

*nAN*:

The AN used as a reference for the new AN.

*X*:

The distance in the x plane (in pixels) from the reference AN to the new AN.

*Y*:

The distance in the y plane (in pixels) from the reference AN to the new AN.

## Return Value

If successful, the new AN is returned. If the AN cannot be created, -1 is returned. If an AN already exists at this location, that AN is returned.

## Related Functions

[DspAnNew](#), [DspGetAnCur](#)

## Example

```
AN=DspAnNewRel(20,100,200);
/* Creates an AN at 100x and 200y pixels from AN20 */
```

## See Also

[Display Functions](#)

### DspAnSetName

Using a valid AN set the name of an animation object.

## Syntax

**DspAnSetName(*hAN*, *sName*)**

*hAN*:

The AN used as a reference for the new Animation Name.

*sName*

Animation Name to be set

## Return Value

Returns an error if given name is already in use.

## Related Functions

[DspGetAnFromName](#), [DspGetNameFromAn](#)

## See Also

[Display Functions](#)

## DspAnSetMetadata

Non-blocking function, that sets the value of the specified metadata entry.

---

**Note:** Metadata items can only be set using Cicode if the name is configured in the object properties -metadata tab and saved with the page.

---

## Syntax

**DspAnSetMetadata(*nAn*, *sMetaName*, *sValue*)**

*nAn*:

An animation number that uniquely identifies an object. This object contains the list of metadata definitions that will be used to perform the association operations. When -2 is specified, it is equivalent to using DspGetAnCur(). (See [DspGetAnCur](#) for usage and limitations.)

*sMetaName*:

The name of metadata entry for which to search.

*sValue*:

The value for the metadata to be set.

---

**Note:** Before calling this function, it may be worthwhile to call ErrSet(1) to disable error checking as this function will generate a hardware error for any object that does not have a metadata entry called '*sMetaName*', and the cicode task will stop executing

---

## Return Value

0 if successful, error code if unsuccessful

## Related Functions

[DspAnSetMetadataAt](#), [DspAnGetMetadata](#), [DspAnGetMetadataAt](#)

## See Also

[Display Functions](#)

### DspAnSetMetadataAt

Non-blocking function, that sets the value of a metadata entry.

**Note:** Metadata items can only be set using Cicode if the name is configured in the object properties -metadata tab and saved with the page.

## Syntax

**DspAnSetMetadataAt(*nAN*, *nIndex*, *sField*, *sFieldValue*)**

*nAn*:

An animation number that uniquely identifies an object. This object contains the list of metadata definitions that will be used to perform the association operations. When -2 is specified, it is equivalent to using DspGetAnCur(). (See [DspGetAnCur](#) for usage and limitations.)

*nIndex*:

The index of the metadata in the animation point.

*sField*:

The name of the field in which to set the information for the metadata. Supported fields are:

- Name
- Value

*sFieldValue*:

The value to set in the specified field of the metadata entry.

**Note:** Clusters should be configured either directly by specifying a full tag name such as C1.TagA or indirectly via the function calls (such as WinNewAt(...)) or via the page configuration parameter.

## Return Value

0 if successful, error code if unsuccessful

## Related Functions

[DspAnSetMetadata](#), [DspAnGetMetadata](#), [DspAnGetMetadataAt](#)

## See Also

[Display Functions](#)

### DspBar

Displays a bar graph (on a graphics page) at a specified AN. To scale a tag into the correct range, use the EngToGeneric() function.

**Note:** This function was designed to only be used on Cicode objects, or the animation point object from the version 3.xx/4.xx toolbox.

## Syntax

**DspBar(*nAN*, *Bar*, *Value*)**

*nAN*:

The AN where the bar graph will be displayed.

*Bar*:

The name of the bar graph to display in the format <[LibName.]BarName>. If you do not specify the library name, a bar graph from the Global library displays (if it exists). To display a Version 1.xx bar graph, specify the bar definition (1 to 255). For example, if you specify bar 1, Plant SCADA displays the bar graph Global.Bar001.

*Value*:

The value to display on the bar graph. The value needs to be from 0 to 32000 to give 0 to full-scale range on the bar.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[EngToGeneric](#)

## Example

```
DspBar(25,"Bars.Loops",320);
/* Displays a value of 320 (that is 10%) on the loops bar (from the
bars library) at AN25. */
DspBar(25,3,320);
/* Displays a value of 320 (that is 10%) on bar definition 3
(Plant SCADA Version 1.xx) at AN25. */
DspBar(26,"Loops_Bar",EngToGeneric(Tag1,0,100));
/* Displays Tag1 on the loops_bar (from the global library) at
AN26. Tag1 has an engineering scale of 0 to 100. */
```

## See Also

[Display Functions](#)

### DspBmp

Displays a bitmap at a specified AN. This function allows you to display any bitmap file at run time. (You can get a new bitmap file from operator input or from the plant, and display it dynamically.)

**Note:** This function was designed to only be used on Cicode objects, or the animation point object from the version 3.xx/4.xx toolbox.

## Syntax

**DspBmp(*nAN*, *sFile*, *Mode*)**

*nAN*:

The animation-point number.

*sFile*:

The name of the bitmap (.BMP) file. The file needs to be in the user project path. (Only .bmp files are supported. Other image formats like .png or .jpg are not supported.)

*Mode*:

The mode of bitmap display:

0 - Erase the existing bitmap and display this bitmap.

1 - Do not erase the existing bitmap, just draw the new bitmap. (This mode provides smoother animation than Mode 0, but the bitmaps needs to be the same size).

2 - Do not erase the existing bitmap, just draw the new bitmap. This mode is similar to mode 1, but it displays the bitmap about 3 times faster. However, the bitmap should not contain any transparent color, or it will display as a random color. Use this mode for fast, smooth animation.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspDel](#)

## Example

```
// Display the bitmap "MyImage.bmp" at AN 60
DspBMP(60, "MyImage.bmp", 0)
```

## See Also

[Display Functions](#)

## DspButton

Displays a button at a specified AN. When the button is selected, the key definition is put into the key command line. The font, width, height, and down and repeat keys of the button are optional. If you do not specify a width and height, the button adjusts to the size of the button *sName*.

---

**Note:** This function was designed to only be used on Cicode objects, or the animation point object from the version 3.xx/4.xx toolbox.

---

## Syntax

**DspButton(*nAN*, *UpKey*, *Name* [, *hFont*] [, *Width*] [, *Height*] [, *DownKey*] [, *RepeatKey*] [, *Style*])**

*nAN*:

The animation-point number.

*UpKey*:

The key generated when the command button is selected (when the mouse button is released after being clicked down). This is the default operation for commands activated by a button.

*Name*:

The name to display on the button.

*hFont*:

The handle of the font used to display the button name. Use the DspFont() function to create a new font and return the font handle. Use the DspFontHnd() function to return the font handle of an existing font. The Windows button font is used if the font is omitted or is not defined in the database.

*Width*:

The width of the button in pixels.

*Height*:

The height of the button in pixels.

*DownKey*:

The key generated when the mouse button is clicked down (over the command button). Normally this parameter is not used, because most buttons are configured to activate a command when the mouse button is released (returning to the 'up' position).

*RepeatKey*:

The key generated repetitively, while the mouse button is being held down (over the command button).

*Style*:

A number indicating the visibility style of the button:

0 - NORMAL: The button appears as a standard button.

1 - BORDER\_3D: The button is drawn with only the 3-D border (transparent face).

2 - BORDER: The button is drawn with only a thin line border.

3 - TARGET: The button is totally transparent - this constitutes a screen target.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspButtonFn](#), [KeySetSeq](#), [DspFont](#), [DspFontHnd](#)

## Example

```
/* Display a self-sizing button at AN20 using the default font.  
The button is named "Help". When selected, the Key Code "KEY_F1"  
is put into the key command line. */  
DspButton(20,KEY_F1,"Help");  
/* Display the same button at AN20, but in an existing font called  
"BigFont". */  
DspButton(20,KEY_F1,"Help",DspFontHnd("BigFont"));
```

## See Also

[Display Functions](#)

## DspButtonFn

Displays a button at a specified AN. When the button is selected, a user function is called. If the width and height are 0 (zero), then the button adjusts to the size of the button *sName*.

---

**Note:** This function was designed to only be used on Cicode objects, or the animation point object from the version 3.xx/4.xx toolbox.

---

## Syntax

**DspButtonFn(*nAN*, *UpFunction*, *Name* [, *hFont*] [, *Width*] [, *Height*] [, *DownFunction*] [, *RepeatFunction*] )**

*nAN*:

The animation-point number.

*UpFunction*:

The user function called when the command button is selected (when the mouse button is released after being clicked down). This is the default operation for commands activated by a button. This callback function can have no arguments, so specify the function with no parentheses (). The callback function needs to return INT as its return data type. You cannot specify a Plant SCADA built-in function for this argument.

*Name*:

The name to display on the button.

*hFont*:

The handle of the font used to display the button name. Use the DspFont() function to create a new font and return the font handle. Use the DspFontHnd() function to return the font handle of an existing font. The Windows button font is used if the font is omitted or is not defined in the database.

**Width:**

The width of the button in pixels.

**Height:**

The height of the button in pixels.

**DownFunction:**

The user function called when the mouse button is clicked down (over the command button). Normally this parameter is not used, because most buttons are configured to activate when the mouse button is released (returning to the 'up' position). The callback function needs to have no arguments, so specify the function with no parentheses (). The callback function needs to return INT as its return data type. You cannot specify a Plant SCADA built-in function for this argument.

**RepeatFunction:**

The user function called repetitively, while the mouse button is being held down (over the command button). The callback function needs to have no arguments, so specify the function with no parentheses (). The callback function needs to return INT as its return data type. You cannot specify a Plant SCADA built-in function for this argument.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspButton](#), [DspFont](#), [DspFontHnd](#)

## Example

```
DspButtonFn(20,MyFunc,"Help",0,50,10);
! Call this function when the button is selected.
INT
FUNCTION
MyFunc()
    PageDisplay("Help");
    RETURN 0;
END
```

## See Also

[Display Functions](#)

## DspChart

Displays a chart at an AN. Charts are trend lines with markers on them. Values are plotted on the chart pens. You need to specify *Value1*, but *Value2* to *Value8* are optional.

If more values (than the configured pens) are specified, the additional values are ignored. If fewer values (than the configured pens) are specified, the pens that have no values are not displayed.

You should use this function only if you want to control the display of charts directly.

**Note:** This function was designed to only be used on Cicode objects, or the animation point object from the version 3.xx/4.xx toolbox.

## Syntax

**DspChart(*nAN*, *Chart*, *Value1* [, *Value2* ... *Value8*] )**

*nAN*:

The AN where the chart will be displayed.

*Chart*:

The chart to display.

*Value1*:

The value to display on Pen 1 of the chart.

*Value2* ... *8*:

The values to display on Pen 2...Pen 8 of the chart. These values are optional.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspDel](#), [DspTrend](#)

## Example

```
/* Using chart definition 5 at AN25, display a value of 10 on
Pen1, 20 on Pen2, 30 on Pen3 and 40 on Pen4 of the chart. */
DspChart(25,5,10,20,30,40);
/* Using chart definition 6 at AN26, display a value of 100 on Pen1
and 500 on Pen2 of the chart. */
DspChart(26,6,100,500);
```

## See Also

[Display Functions](#)

## DspClearClip

Clears the clipping rectangle surrounding the object or group of objects.

## Syntax

**DspClearClip(INT nAN)**

*nAn:*

The animation-point number of the object or groups within the clipping rectangle.

## Return Value

If nAN is invalid will return 0

When cleared will return 0 and change co-ordinates for clipping rectangle to 0.

## Example

```
DspClearClip(17)
```

## See Also

[Display Functions](#)

## DspCol

DspCol is deprecated in this version of Plant SCADA.

## Syntax

**DspCol(nAN, Color)**

*nAN:*

The animation-point number.

*Color:*

The color to display at the AN.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspDel](#)

## Example

```
DspCol(25,RED);  
/* Displays the color red at AN25. */
```

## See Also

[Display Functions](#)

### DspDel

Deletes all objects from a specified AN.

**Note:** This function was designed to only be used on Cicode objects, or the animation point object from the version 3.xx/4.xx toolbox.

## Syntax

**DspDel(*nAN*)**

*nAN*:

The animation-point number.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspDirty](#)

## Example

```
DspDel(25);
! Deletes all animation at AN25.
```

## See Also

[Display Functions](#)

### DspDelayRenderBegin

Delays screen updating until DspDelayRenderEnd is called. This function should be used with DspDelayRenderEnd() to "sandwich" Cicode that will modify the appearance of a page. The code should be preceded by DspDelayRenderBegin(), and followed by DspDelayRenderEnd(). This will reduce screen update times, because the modifying code is given time to execute before the page is updated with the changes, and the changes are all made in a single re-draw.

**Note:** If you have not changed the [Page]DelayRenderAll parameter from its default (TRUE), then you do not need to use this function.

You can call this function as many times in a row as you like, as long as each is ended with a call to DspDelayRenderEnd.

Because your display will stop updating while the "sandwiched" code runs, you should try to make that code as efficient as possible. Do not call Sleep() or any other Cicode functions that will take a long time to run.

Do not call WinSelect within the "sandwiched" code. Do not call this function directly from the Kernel.

## Syntax

**DspDelayRenderBegin()**

## Related Functions

[DspDelayRenderEnd](#)

## Example

```
/* Begin delay so the following code can be executed before the
images are re-drawn. */
DspDelayRenderBegin();
DspBMP(50, "Image1.bmp", 0) ! Display the bitmap "Image1.bmp"
at AN 50
DspBMP(100, "Image2.bmp", 0) ! Display the bitmap "Image2.bmp"
at AN 100
DspBMP(150, "Image3.bmp", 0) ! Display the bitmap "Image3.bmp"
at AN 150
DspBMP(200, "Image4.bmp", 0) ! Display the bitmap "Image4.bmp"
at AN 200
DspBMP(250, "Image5.bmp", 0) ! Display the bitmap "Image5.bmp"
at AN 250
/* End delay so the images can be re-drawn. */
DspDelayRenderEnd();
```

## See Also

[Display Functions](#)

## DspDelayRenderEnd

Ends the screen update delay set by DspDelayRenderBegin. This function should be used with DspDelayRenderBegin() to "sandwich" Cicode that will modify the appearance of a page. The code should be preceded by DspDelayRenderBegin(), and followed by DspDelayRenderEnd(). This will reduce screen update times, because the modifying code is given time to execute before the page is updated with the changes, and the changes are all made in a single re-draw.

Because your display will stop updating while the "sandwiched" code runs, you should try to make that code as efficient as possible. Do not call Sleep() or any other Cicode functions that will take a long time to run.

Do not call WinSelect within the "sandwiched" code. Do not call this function directly from the Kernel.

---

**Note:** If you have not changed the [\[Page\]DelayRenderAll](#) parameter from its default (TRUE), then you do not need to use this function.

## Syntax

```
DspDelayRenderEnd()
```

## Return Value

No value is returned.

## Related Functions

[DspDelayRenderBegin](#)

## Example

```
/* Begin delay so the following code can be executed before the
images are re-drawn. */
DspDelayRenderBegin();
DspBMP(50, "Image1.bmp", 0) ! Display the bitmap "Image1.bmp"
at AN 50
DspBMP(100, "Image2.bmp", 0) ! Display the bitmap "Image2.bmp"
at AN 100
DspBMP(150, "Image3.bmp", 0) ! Display the bitmap "Image3.bmp"
at AN 150
DspBMP(200, "Image4.bmp", 0) ! Display the bitmap "Image4.bmp"
at AN 200
DspBMP(250, "Image5.bmp", 0) ! Display the bitmap "Image5.bmp"
at AN 250
/* End delay so the images can be re-drawn. */
DspDelayRenderEnd();
```

## See Also

[Display Functions](#)

## DspDirty

Forces Plant SCADA to update an AN. Normally, Plant SCADA updates the animation on the AN only if the data has changed. This function tells Plant SCADA to update the AN the next time it animates the AN - even if the data has not changed.

Use this function when you have complex animations that overlap. If two or more animations overlap, you should use the DspDel() or DspDirty() function on their ANs, and then display them in the same order (when they need to be updated).

---

**Note:** This function was designed to only be used on Cicode objects, or the animation point object from the version 3.xx/4.xx toolbox.

## Syntax

**DspDirty(*nAN*)**

*nAN*:

The animation-point number.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspDel](#)

## Example

```
DspDirty(20);  
! Forces an update of AN20.
```

## See Also

[Display Functions](#)

## DspError

Displays an error message at the prompt AN on the operator's computer. You can disable the error message display (of this function) by setting the Cicode execution mode in the CodeSetMode() function.

## Syntax

**DspError(*String*)**

*String*:

The message to be displayed.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[CodeSetMode](#), [Prompt](#)

## Example

```
DspError("Error found");
! Displays "Error found" at the prompt AN.
```

## See Also

[Display Functions](#)

## DspFile

Defines the screen attributes for displaying a text file. This function defines a "window" where the file will be displayed. You should call this function before any file-to-screen function.

You need to define sequential ANs for each line of text in the display. The file is displayed starting at the specified AN, then the next (highest) AN, and so on. You should not use proportionally-spaced fonts, because the columns of text might not be aligned.

You would normally call this function as the entry function for a graphics page. Use the DspFileSetName() function to specify the file to be displayed. This function is a low level animation function - it controls exactly how the file is to display. If you just want to display a file, use the PageFile() function.

## Syntax

**DspFile**(*nAN, hFont, Height, Width*)

*nAN*:

The AN where the file display window will be positioned.

*hFont*:

The handle for the font that is used to display the file, returned from the DspFont() or DspFontHnd() function. The font handle identifies the table where data on the associated font is stored.

*Height*:

The maximum number of lines to display on one page of the file display window.

*Width*:

The width of the file display window, in characters.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[PageFile](#), [DspFileGetInfo](#), [DspFileGetName](#), [DspFileScroll](#), [DspFileSetName](#), [DspFont](#), [DspFontHnd](#)

## Example

```
DspFile(20,0,20,80);
/* Defines the attributes of a screen display to start at AN20,
using the default font, with a window size of 20 lines x 80
columns. */
```

## See Also

[Display Functions](#)

### DspFileGetInfo

Gets the attributes of a file-to-screen display (used for displaying text files).

## Syntax

**DspFileGetInfo(*nAN*, *Type*)**

*nAN*:

The AN where the file display window will be located. This AN needs to be the same as the AN specified with the DspFile() function.

*nType*:

The type of data required:

0 - The width of the file display window, in characters.

1 - The maximum number of lines that can display in one page of the file display window.

2 - The file-to-screen row offset number.

3 - The file-to-screen column offset number.

4 - The number of lines in the displayed file.

## Return Value

The attributes of the "window" as an integer. If an incorrect AN is specified, an error is returned.

## Related Functions

[DspFile](#), [DspFileGetName](#), [DspFileScroll](#), [DspFileSetName](#)

## Example

```
! Display the page number of the file display.
PageNumber=IntToStr(DspFileGetInfo(20,2)/DspFileGetInfo(20,1)+1);
DspText(12,0,"Page No "+PageNumber);
```

## See Also

[Display Functions](#)

### DspFileGetName

Gets the name of the file being displayed in the display "window". You can use this function to display the file name on the screen.

## Syntax

**DspFileGetName(*nAN*)**

*nAN*:

The animation-point number.

## Return Value

The name of the file (as a string). If an incorrect AN is specified, an error is returned.

## Related Functions

[DspFile](#), [DspFileGetInfo](#), [DspFileScroll](#), [DspFileSetName](#)

## Example

```
DspText(11,0,DspFileGetName(20));  
! Displays the name of the file displayed at AN20.
```

## See Also

[Display Functions](#)

### DspFileScroll

Scrolls a file (displayed in the display "window") by a number of characters.

## Syntax

**DspFileScroll(*nAN, Direction, Characters*)**

*nAN*:

The animation-point number.

*Direction*:

The direction in which to scroll:

1 - Left

2 - Right

3 - Up

4 - Down

*Characters:*

The number of characters to scroll. To page up or page down through the file, scroll by the height of the file-to-screen window (returned by DspFileGetInfo(AN, 1)).

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspFile](#), [DspFileGetInfo](#), [DspFileSetName](#), [DspFileGetName](#)

## Example

### Page Keyboard

Key Sequence	PgUp
Command	DspFileScroll(20,3,10)
Comment	Scroll up 10 lines

## See Also

[Display Functions](#)

## DspFileSetName

Sets the name of the file to display in the display "window". You should call the DspFile() function first (as the entry function for a graphics page) to define the attributes of the display. You can then use the DspFileSetName() function (as a keyboard command) to display a user-specified file. When you call this function, the specified file name is read from disk and displayed on the screen.

## Syntax

**DspFileSetName(*nAN*, *sName*)**

*nAN*:

The animation-point number.

*sName*:

The name of the file to display.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspFile](#), [DspFileGetInfo](#), [DspFileGetName](#), [DspFileScroll](#)

## Example

### Pages

Page Name	FilePage
Entry Command	DspFile(20,0,20,80)
Comment	Defines a file to screen display to commence at AN20

### Page keyboard

Key Sequence	##### Enter
Command	DspFileSetName(20, Arg1)
Comment	Displays a specified file on the page

```
DspFile(20,0,20,80);
/* Defines the file-to-screen display to commence at AN20 using
the default font, with a window size of 20 lines x 80 columns. */
DspFileSetName(20,"C:\AUTOEXEC.BAT");
! Displays file C:\AUTOEXEC.BAT.
```

## See Also

[Display Functions](#)

## DspFont

Creates a font and returns a font handle. If the requested font already exists, its font handle is returned. You can use this font handle in the functions that display text, buttons, and text files.

If the exact font size does not exist, the closest font size is used.

## Syntax

**DspFont**(*FontType*, *PixelSize*, *ForeOnColor*, *BackOnColor* [, *ForeOffColor*] [, *BackOffColor*] )

*FontType*:

The font type, for example, "Helv".

***PixelSize:***

The font size, as a positive number for pixels, or a negative number for points.

***ForeOnColor:***

The foreground color used for the text. If implementing flashing color, this is the initial color that will be used. Select a color from the list of Predefined Color Names and Codes or create an RGB-based color using the function MakeColour.

***BackOnColor:***

The color used for the background of text. If implementing flashing color, this is the initial color that will be used. Select a color from the list of Predefined Color Names and Codes or create an RGB-based color using the function MakeColour.

***ForeOffColor:***

An optional argument only required if implementing flashing color for the font foreground. It represents the secondary color used. Select a color from the list of Predefined Color Names and Codes or create an RGB-based color using the function MakeColour.

***BackOffColor:***

An optional argument only required if implementing flashing color for the font background. It represents the secondary color used. Select a color from the list of Predefined Color Names and Codes or create an RGB-based color using the function MakeColour.

## Return Value

The font handle as an integer. If the font cannot be created, -1 is returned. The font handle identifies the table where all data on the associated font is stored.

## Related Functions

[DspFontHnd](#), [DspText](#), [DspButton](#), [DspButtonFn](#), [DspFile](#), [DspSetFontPopupMenuFont](#)

## Example

```
Font = DspFont("Helv", -12, White, Red);
DspText(20, Font, "Text in Helv Font");
/* Displays "Text in Helv Font" in 12-point Helvetica font in
white on red at AN20. */
Font = DspFont("Helv", 24, White, Red, Black);
DspText(20, Font, "Text in Helv Font");
/* Displays "Text in Helv Font" in 24 pixel Helvetica font in
flashing black and white on red at AN20. */
```

## See Also

[Display Functions](#)

## DspFontHnd

Gets the font handle of a font that is defined in the Fonts database. You can use this font handle in the functions that display text, buttons, and text files.

## Syntax

**DspFontHnd(*sName*)**

*Name:*

The font name in the fonts database.

## Return Value

The font handle as an integer. If the font cannot be found, -1 is returned. The font handle identifies the table where the data on the associated font is stored.

## Related Functions

[DspFont](#), [DspText](#), [DspButton](#), [DspButtonFn](#), [DspFile](#), [DspSetPopupMenuFont](#)

## Example

Font Name	BigFont
Font Type	Helv
Pixel Size	24
Foreground Color	Blue
Background Color	-1
Comment	Defines a font

```
hBigFont=DspFontHnd("BigFont");
DspText(20,hBigFont,"Text in Big Font");
/* Displays "Text in Big Font" in 24-point Helvetica font in blue
on an unchanged background at AN20. */
```

## See Also

[Display Functions](#)

## DspFullScreen

Disables or enables the fullscreen mode of the currently active window. This function does not resize the

window when it is called; it merely sets the mode flag. The next time the window is displayed, its size (on screen) changes to reflect the setting of the flag. This function overrides the [Animator]FullScreen parameter setting.

If [Page]DynamicSizing is turned on, a page in fullscreen state takes up the entire display area (assuming this does not affect its aspect ratio), and it cannot be resized. Also, a fullscreen page will display without a title bar unless **Title Bar** is checked in Page Properties (or was checked when the page was created). Resizing pages can result in lower picture quality. If this is unacceptable, you should re-design the page using the desired resolution.

If [Page]DynamicSizing is turned off, fullscreen will have the same limitations as it had in versions of Plant SCADA prior to V5.10. In other words, for a page to be displayed in fullscreen, the size of the page needs to be the same size as the display (or bigger). If the page is smaller than the display, the title bar will still display even if fullscreen mode is enabled. Check the size of the graphic pages in CtDraw Tools | Page Attributes Dialog to verify that it is the same as the display resolution. For example 640x480 for VGA, 800x600 for SVGA and 1024x768 for XGA.

## Syntax

**DspFullScreen(*Mode*)**

*Mode*:

Fullscreen mode:

0 - Disable fullscreen mode.

1 - Enable fullscreen mode without title bar

2 – Enable fullscreen mode with title bar.

Return Value

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[WinMode](#)

## Example

```
/*Minimize the Window, Enable fullscreen mode and then maximize  
the window.*/  
WinMode(6);  
DspFullScreen(1);  
WinMode(3);
```

## See Also

[Display Functions](#)

## DspGetAnBottom

Gets the bottom extent of the object at the specified AN.

### Syntax

**DspGetAnBottom(*nAN*)**

*nAN*:

The animation-point number.

### Return Value

The y coordinate of the bottom extent of the object at the AN. If no object exists at the AN, -1 is returned.

### Related Functions

[DspGetAnBottom](#), [DspGetAnWidth](#), [DspGetAnHeight](#), [DspGetAnLeft](#), [DspGetAnRight](#), [DspGetAnTop](#),  
[DspGetAnNext](#), [DspGetAnExtent](#)

### Example

```
nBottom = DspGetAnBottom(30);
```

### See Also

[Display Functions](#)

## DspGetAnCur

Gets the AN of the current graphics object. This function should only be used by expressions or Cicode functions called from the condition fields of a graphics object, excluding input/command fields. If you need to know the AN that triggered the input/command, the [KeyGetCursor](#) function may be used as it returns the AN where the cursor is currently positioned.

You cannot call this function from the Button or Keyboard forms.

### Syntax

**DspGetAnCur()**

### Return Value

The AN associated with the current graphics object. If this function is called outside the page animation system or from an input/command field, -1 will be returned.

## Example

### Numbers

AN	20
Expression	MyFunc(PV_10)
Comment	Display the value of PV_10 at AN20

```
/* Function displays a number at the current AN and returns the
value supplied in the call */
INT
FUNCTION
MyFunc(INT value)
    INT AN, hNew;
    AN = DspGetAnCur();
    hNew = DspAnNewRel(AN, 0, 20);
    DspStr(hNew, "Default", VALUE:###.#);
    RETURN value;
END
```

## See Also

[Display Functions](#)

### DspGetAnExtent

Gts the extent (the page co-ordinates) of the object at the specified AN.

## Syntax

**DspGetAnExtent(*nAN, Top, Left, Bottom, Right*)**

*nAN*:

The AN at which the object is positioned.

*Top*:

A buffer that contains the top-most extent of the object.

*Left*:

A buffer that contains the left-most extent of the object.

*Bottom*:

A buffer that contains the bottom-most extent of the object.

*Right*:

A buffer that contains the right-most extent of the object.

## Return Value

0 (zero) if successful, otherwise an error is returned. The *Top*, *Left*, *Bottom*, and *Right* arguments contain the

extents of the object, in pixels.

## Related Functions

[DspGetAnRawExtent](#), [DspGetAnWidth](#), , [DspGetAnLeft](#), [DspGetAnRight](#), [DspGetAnBottom](#), [DspGetAnTop](#),  
[PageTransformCoords](#)

## Example

```
// Get extents at AN 25.  
DspGetAnExtent(25, Top, Left, Bottom, Right);
```

## See Also

[Display Functions](#)

### DspGetAnRawExtent

Gets the extent of the object from the graphics page at the specified AN. The extent represents the page coordinates at which the object was originally inserted.

**Note:** If the object has movement configured, use the [DspGetAnExtent](#) function to get the extent of the object.

## Syntax

**DspGetAnRawExtent(*nAN*, *Top*, *Left*, *Bottom*, *Right*)**

*nAN*:

The AN at which the object is positioned.

*Top*:

A buffer that contains the top-most extent of the object.

*Left*:

A buffer that contains the left-most extent of the object.

*Bottom*:

A buffer that contains the bottom-most extent of the object.

*Right*:

A buffer that contains the right-most extent of the object.

## Return Value

0 (zero) if successful, otherwise an error is returned. The *Top*, *Left*, *Bottom*, and *Right* arguments contain the extents of the object, in pixels.

## Related Functions

[DspGetAnExtent](#), [DspGetAnWidth](#), [DspGetAnHeight](#), [DspGetAnLeft](#), [DspGetAnRight](#), [DspGetAnBottom](#),  
[DspGetAnTop](#), [PageTransformCoords](#)

## Example

```
// Get extents at AN 25.  
DspGetAnRawExtent(25, Top, Left, Bottom, Right);
```

## See Also

[Display Functions](#)

### DspGetAnFirst

Gets the first AN on the current page, based on the order in which the ANs were stored by Graphics Builder.

## Syntax

`DspGetAnFirst()`

## Return Value

The value for the first AN, otherwise an error is returned.

## Related Functions

[DspGetAnNext](#)

## See Also

[Display Functions](#)

### DspGetAnFromName

Name used to retrieve the AN of an object on the page. Use the following relative path syntax:

[RelativePath].[Group].[Group].Control

This will browse the group hierarchy to a specific starting point, and then drill down to other groups to find the named control or object.

If you do not use a relative path the system will first look for an object within the groups specified relative to the object the cicode is running on, and if the name is not found, it will then search the page for the first existence of the object name. See [Referencing an object using a name at runtime](#) for more information.

**Note:** When retrieving the AN for an object within a genie or super genie you need to use a relative path.

## Syntax

**DspGetAnFromName(*sName*)**

*sName*

The Name of the object used as a reference for the AN.

## Return Value

AN of object or -1 if not found.

## Related Functions

[DspAnSetName](#), [DspGetNameFromAn](#)

## Example

```
// Example of relative pathing syntax
// Reference from a current group, symbol or genie
DspGetAnFromName("./Object1")
// Navigate three levels up and from the root level search for Object1.
DspGetAnFromName("../..\\Object1")
```

## See Also

[Display Functions](#)

## DspGetAnFromNameRelative

Name used to retrieve the animation number (AN) of an object on the page relative to a given Animation Number(AN). Use the following syntax:

[RelativePath].[Group].[Group].Control

This will browse the group hierarchy to a specific starting point, and then drill down to other groups to find the named control or object.

See [Referencing an Object using a Name at Runtime](#) more information on the relative path syntax.

**Note:** When retrieving the AN for an object within a genie or super genie you need to use a relative path.

## Syntax

**DspGetAnFromNameRelative(*hAN*, *sName*)**

*hAN*

AN used as the starting point for the search.

*sName*

The Name used as a reference for the AN.

**Note:** With an AN you can start the search from any level and not just the group you are in.

## Return Value

AN of object or -1 if not found.

## Related Functions

[DspAnSetName](#), [DspGetNameFromAn](#)

## Example

```
// example of relative pathing syntax
DspGetAnFromNameRelative(625, "..\..\..\Object1")
//Using another AN as browse starting point
DspGetAnFromNameRelative(639, "..\..\Object1")
```

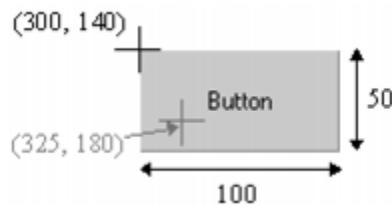
## See Also

[Display Functions](#)

### DspGetAnFromPoint

Gets the AN of the object at a specified set of screen coordinates. If the X and Y coordinates given are within the extents of an object, then the AN number of the object will be returned.

For example, if there is a button at coordinates (300, 140), and it is 100 wide, 50 high, this function would return the AN if it uses X between 300 & 400 and Y between 140 and 190, such as DspGetAnFromPoint(325,180).



If you are using groups and the specified coordinates point to an object that is part of a group, the AN of the object is returned, not the AN of the group.

## Syntax

**DspGetAnFromPoint(X, Y [, PrevAN] )**

**X:**

The x coordinate of the screen point.

**Y:**

The y coordinate of the screen point.

**PrevAN:**

Retrieves the previous AN (in z-order) in situations where a number of objects overlap at the specified point. The default of 0 (zero) specifies no previous AN. A non-zero value should only ever be passed if it is the result of a

previous call to DspGetAnFromPoint.

## Return Value

The AN or 0 (zero) if no object exists at the point.

For clipped objects the AN will not be returned if the mouse is over the clipped region.

## Example

```
DspGetMouse(X,Y);
// GetMouse position
AN = DspGetAnFromPoint(X,Y);
// Gets AN if mouse is over the object
Prompt("AN of object =" + nAN:###);
!Displays the object's AN at the prompt line
```

If several objects overlap each other at the specified point, the PrevAN argument can be used to produce a list of the associated ANs. This is achieved by using PrevAN to pass the previous result into another call of the function until a zero return is given.

```
INT nAn;
nAn = DspGetAnFromPoint(100,100)
WHILE nAn <> 0 DO
    //Do Something
    nAn = DspGetAnFromPoint(100,100,nAn);
END
```

## See Also

[Display Functions](#)

## DspGetAnHeight

Gets the height of the object at a specified AN.

## Syntax

**DspGetAnHeight(*nAN*)**

*nAN*:

The animation-point number.

## Return Value

The height of the object (in pixels). If no object exists at the AN, -1 is returned.

## Related Functions

[DspGetAnWidth](#), [DspGetAnLeft](#), [DspGetAnRight](#), [DspGetAnTop](#)

## Example

```
nHeight = DspGetAnHeight(30);
```

## See Also

[Display Functions](#)

### DspGetAnLeft

Gets the left extent of the object at the specified AN.

## Syntax

**DspGetAnLeft(*nAN*)**

*nAN*:

The animation-point number.

## Return Value

The x coordinate of the left extent of the object at the AN. If no object exists at the AN, -1 is returned.

## Related Functions

[DspGetAnWidth](#), [DspGetAnHeight](#), [DspGetAnRight](#), [DspGetAnBottom](#), [DspGetAnTop](#), [DspGetAnExtent](#)

## Example

```
nLeft = DspGetAnLeft(30);
```

## See Also

[Display Functions](#)

### DspGetAnNext

Returns the AN that follows the specified AN, based on the order in which the ANs were stored on a page by Graphics Builder.

## Syntax

**DspGetAnNext(*nAN*)**

*nAN*:

The animation-point number.

## Return Value

The value for the next AN. If -1 is returned, it means the specified AN is invalid or it is the last AN on the page.

## Related Functions

[DspGetAnFirst](#)

## See Also

[Display Functions](#)

## DspGetAnRight

Gets the right extent of the object at the specified AN.

## Syntax

**DspGetAnRight(*nAN*)**

*nAN*:

The animation-point number.

## Return Value

The x coordinate of the right extent of the object at the AN. If no object exists at the AN, -1 is returned.

## Related Functions

[DspGetAnWidth](#), [DspGetAnHeight](#), [DspGetAnLeft](#), [DspGetAnTop](#), [DspGetAnExtent](#)

## Example

```
nRight = DspGetAnRight(30);
```

## See Also

[Display Functions](#)

## DspGetAnTop

Gets the top extent of the object at the specified AN.

## Syntax

**DspGetAnTop(*nAN*)**

*nAN*:

The animation-point number.

## Return Value

The y coordinate of the top extent of the object at the AN. If no object exists at the AN, -1 is returned.

## Related Functions

[DspGetAnWidth](#), [DspGetAnHeight](#), [DspGetAnLeft](#), [DspGetAnRight](#), [DspGetAnBottom](#), [DspGetAnExtent](#)

## Example

```
nTop = DspGetAnTop(30);
```

## See Also

[Display Functions](#)

## DspGetAnWidth

Gets the width of the object at a specified AN.

## Syntax

**DspGetAnWidth(*nAN*)**

*nAN*:

The animation-point number.

## Return Value

The width of the object (in pixels). If no object exists at the AN, -1 is returned.

## Related Functions

[DspGetAnHeight](#), [DspGetAnLeft](#), [DspGetAnRight](#), [DspGetAnBottom](#), [DspGetAnTop](#), [DspGetAnExtent](#)

## Example

```
nWidth = DspGetAnWidth(30);
```

## See Also

[Display Functions](#)

### DspGetEnv

Gets a page environment variable.

## Syntax

**DspGetEnv(*sName*)**

*sName*:

The name of the variable (set using the page environment dialog).

**Note:** Page environment variables are case sensitive.

## Return Value

The value of the variable (as a string).

## Example

```
FUNCTION
PageGroup()
    PageDisplay(DspGetEnv("GroupMenu"));
END
```

## See Also

[Display Functions](#)

### DspGetMetadataFromName

Name used to retrieve the metadata of an object on the page. Use the following relative path syntax:

[RelativePath].[Group].[Group].Control

This will browse the group hierarchy to a specific starting point, and then drill down to other groups to find the named control or object and metadata. See [Referencing an Object using a Name at Runtime](#) for more information.

## Syntax

**DspGetMetadataFromName(*sName*, *sMetaName*)**

*sName*

The Name used as a reference for the object.

*sMetaName*

The name of the metadata to be returned.

The name belonging to the metadata name/value pair defined in the object's properties metadata tab. See the topic Metadata in the main help for more information.

## Return Value

The value of the metadata or blank.

## Related Functions

[DspSetMetadataFromName](#)

## Example

```
// example of relative pathing syntax
// Relative pathing varies according to the starting point of the search
DspGetMetadataFromName("Object1", "Pump")
DspGetMetadataFromName(".\Object1", "Pump")
DspGetMetadataFromName("../..\..\Object1", "Pump")
```

## See Also

[Display Functions](#)

## DspGetMetadataFromNameRelative

Name used to retrieve the metadata of an object on the page relative to a given Animation Number (AN). Use the following syntax:

[RelativePath].[Group].[Group].Control

This will browse the group hierarchy from the given AN, and then drill down to other groups to find the named control or object. See Referencing an Object using a Name at Runtime for more information.

## Syntax

**DspGetMetadataFromNameRelative(*hAN*, *sName*, *sMetaName*)**

*hAN*

AN used as the starting point for the search.

With an AN you can start the search from any level and not just the group you are in.

***sName***

The name of the AN used as a reference for the object.

***sMetaName***

The name of the metadata to be returned.

The name belonging to the metadata name/value pair defined in the object's properties metadata tab. See the topic **Metadata** in the main help for more information.

## Return Value

The value of the metadata or blank.

## Related Functions

[DspSetMetadataFromNameRelative](#)

## Example

```
// example of relative pathing syntax
DspGetMetadataFromNameRelative(639, "..\..\Object1", "Pump")
//Using another AN as browse starting point
DspGetMetadataFromNameRelative(625, "..\..\..\Object1", "Pump")
```

## See Also

[Display Functions](#)

## DspGetMouse

Gets the x and y coordinates of the mouse position, relative to the top left corner of the window.

## Syntax

**DspGetMouse(X, Y)**

X:

A locally declared variable used to store the x pixel coordinate of the mouse position, returned from this function.

Y:

A locally declared variable used to store the y pixel coordinate of the mouse position, returned from this function.

---

**Note:** Locally declared Cicode variables need to be used for X and Y, otherwise an "Incompatible Type" compile error may be generated.

## Return Value

0 (zero) if successful, otherwise an error is returned. The X and Y variables are set to the mouse position.

## Related Functions

[DspAnGetPos](#), [DspGetMouseOver](#), [DspGetNearestAn](#), [PageTransformCoords](#)

## Example

```
! If the mouse cursor is at x,y pixel coordinate 43,20;  
DspGetMouse(X,Y);  
! Sets X to 43 and Y to 20.
```

## See Also

[Display Functions](#)

## DspGetMouseOver

Determines if the mouse is within the boundaries of a given AN.

## Syntax

**DspGetMouseOver(*nAN*)**

*nAN*

The AN of the animation you wish to check, or -1 for the current AN. Defaults to -1.

## Return Value

1 if within the specified AN, 0 if not.

Will return False if the mouse is over the clipped part of an AN.

## Related Functions

[KeyGetCursor](#), [DspAnGetPos](#), [DspGetMouse](#), [DspGetNearestAn](#)

## See Also

[Display Functions](#)

## DspGetNameFromAn

Using a valid animation number (AN) the animation name of the object is returned.

## Syntax

**DspGetNameFromAn(*hAN*)**

*hAN*:

The AN used as a reference for the name of the object.

## Return Value

The animation name of the object, or blank

## Related Functions

[DspGetAnFromName](#), [DspAnSetName](#)

## See Also

[Display Functions](#)

## DspGetNearestAn

Gets the AN nearest to a specified x,y pixel location.

If using groups and the nearest object to the specified coordinates is part of a group, the AN of the object is returned, not the AN of the group.

If using clipping objects completely clipped ANs will be ignored when determining the nearest AN to the specified x, y pixel location. Partially clipped ANs are treated as being fully visible.

## Syntax

**DspGetNearestAn(*X, Y*)**

*X*:

The x coordinate (in pixels).

*Y*:

The y coordinate (in pixels).

## Return Value

The animation point number (AN). A value of -1 is returned if no AN is found.

## Related Functions

[DspGetMouse](#), [DspAnGetPos](#), [DspGetAnFromPoint](#)

## Example

```
DspGetMouse(X,Y);
! Gets mouse position.
AN=DspGetNearestAn(X,Y);
! Gets AN nearest to the mouse.
Prompt("Mouse At AN"+nAN:###);
! Displays AN nearest to the mouse.
```

## See Also

[Display Functions](#)

### DspGetParentAn

Gets the parent animation number (if any), for the specified animation number. AN animation point will have a parent animation point if it corresponds to an object in a group.

## Syntax

**DspGetParentAn(*nAN*)**

*AN:*

The animation-point number.

## Return Value

The parent animation point number (AN). If no parent animation exists or an invalid animation number is passed, 0 (zero) is returned.

## Related Functions

[DspGetAnCur](#)

## Example

```
// Get the parent animation for object 89 (part of a symbol set)
AN = DspGetParentAn(89);
```

## See Also

[Display Functions](#)

### DspGetSlider

Gets the current position (value) of a slider at an AN. You can call this function in the slider event to find the new

position of the slider.

**Note:** This function was designed to only be used on Cicode objects, or the animation point object from the version 3.xx/4.xx toolbox.

## Syntax

**DspGetSlider(*nAN*)**

*nAN*:

The animation-point number.

## Return Value

The value of the slider from 0 to 32000. If no animation exists at the AN, -1 is returned.

## Related Functions

[DspSetSlider](#)

## Example

```
// Get the position of the slider at AN 30
nPos = DspGetSlider(30);
```

## See Also

[Display Functions](#)

## DspGetTip

Gets the tool tip text associated with an AN.

## Syntax

**DspGetTip(*nAN*, *Mode*)**

*nAN*:

The AN from which to get the tool tip text. If no object is configured at the AN, the function will return an empty string.

*Mode*:

0 - Tool tips from all animation records configured at the AN. Tips are concatenated with a newline character between each string. (This mode is only used for V3.xx and V4.xx animations, and has been subsequently superseded.)

1 - The tool tip from the object configured at the AN.

## Return Value

The tool tip text (as a string). If no user tip is available, an empty string is returned.

## Related Functions

[DspSetTip](#), [DspTipMode](#)

## Example

```
!Display the tool tip text on AN19
DspText(19, 0, DspGetTip(KeyGetCursor(), 1));
```

## See Also

[Display Functions](#)

## DspGrayButton

Grays and disables a button. If the button is a symbol, the symbol is overwritten with a gray mask. (When a button is grayed, it cannot be selected.) If the **Disabled** field in the Buttons database is blank, the button is enabled unless you use this function. If the **Disabled** field in the Buttons database contains an expression, this function will not override the expression.

---

**Note:** This function was designed to only be used on Cicode objects, or the animation point object from the version 3.xx/4.xx toolbox.

---

## Syntax

**DspGrayButton(*nAN*, *nMode*)**

*nAN*:

The AN where the button is located.

*nMode*:

The mode of the operation:

0 - Ungray the button.

1 - (GRAY\_SUNK) Recess the text or symbol (the text or symbol on the button is recessed and shadowed).

2 - (GRAY\_PART) This mode is now obsolete - it now has the same effect as GRAY\_ALL.

3 - (GRAY\_ALL) - Mask the entire button (a gray mask displays over the face of the button).

## Return Value

0 (zero) if successful, otherwise, -1 (if no AN is found).

## Related Functions

[DspButton](#), [DspButtonFn](#), [DsplsButtonGray](#)

## Example

```
! Disable button at AN21
DspGrayButton(21, GRAY_SUNK);
```

## See Also

[Display Functions](#)

## DsplInfo

Extracts individual pieces of object information from an AN. Each AN can have multiple expressions associated with it, and each expression can have multiple variables associated with it. You use an index to refer to each individual expressions or variables. Typically, you would query the number of expressions, then the number of variables in a given expression, then the details of a given variable tag.

**Note:** Before calling this function you need to first use **DsplInfoNew()** to create a handle to the information block from which you want to extract information.

## Syntax

**DsplInfo(*hInfo*, *Type*, *Index*)**

*hInfo*:

The object information block handle, as returned by **DsplInfoNew()**. This handle identifies the table (or block) where all object data is stored.

*nType*:

The type of data to extract:

0 - Object title (the name of the object type)

1 - Object expression text

2 - Object expression result text

3 - The variable tag name

4 - Not supported.

5 - The engineering value associated with the variable

6 - The Cicode context. Calling **DsplInfo** with this Type will return a string describing the context in which the Cicode expression is contained. For example, if it appears on the horizontal movement tab it would return "Move X".

7 - The number of Cicode expressions. Calling **DsplInfo** with this Type will return the number of Cicode expressions associated with this animation point.

8 - The number of tags in the expression. Calling **DsplInfo** with this Type will return the number of tags that appear in the given Cicode expression.

9 - Name of the cluster in which the variable tag resides.

10 - Full name of the variable tag in the form *cluster.tagname*.

*Index:*

An index to the variable within the information block. The required index changes according to the Type as follows:

- For Types 0 to 2, 6 and 8, the index needs to be set to the index of the expression that you wish to query.
- For Types 3 to 5, the index needs to be set to the index of the tag that you wish to query. When one of these types is used, DsplInfo will query the tag in the most recently queried expression (otherwise expression 0).
- For Type 7, the index is ignored.

---

**Note:** Getting the raw value using DsplInfo is no longer supported. To get the raw value of a tag, use the TagSubscribe function, specifying a value of "Raw" for the *sScaleMode* parameter. When using TagSubscribe, you can either call SubscriptionGetAttribute to obtain the value whenever required or register a callback Cicode function to run when the value changes. See [TagSubscribe](#) for more details.

---

## Return Value

The object information (as a string). A blank string is returned if you specify a non-existent expression or variable.

## Related Functions

[DsplInfoNew](#), [DsplInfoField](#), [DsplInfoDestroy](#), [TagSubscribe](#), [SubscriptionAddCallback](#), [SubscriptionGetAttribute](#)

## Example

```
INT hInfo;
INT iEngineeringValue;
INT iNumberOfExpressions;
INT iNumberOfTags;
INT iExpressionIndex;
INT iTagIndex;
STRING sObjectType;
STRING sExpressionText;
STRING sExpressionResult;
STRING sExpressionContext;
STRING sTagName;
hInfo = DsplInfoNew(AN);
IF (hInfo > -1) THEN
    sObjectType = DsplInfo(hInfo, 0, 0);
    iNumberOfExpressions = StrToInt(DsplInfo(hInfo, 7, 0));
    FOR iExpressionIndex = 0 TO iExpressionIndex < iNumberOfExpressions DO
        sExpressionText = DsplInfo(hInfo, 1, iExpressionIndex);
        sExpressionResult = DsplInfo(hInfo, 2, iExpressionIndex);
        sExpressionContext = DsplInfo(hInfo, 6, iExpressionIndex);
        iNumberOfTags = StrToInt(DsplInfo(hInfo, 8, iExpressionIndex));
        FOR iTagIndex = 0 TO iTagIndex < iNumberOfTags DO
            sTagName = DsplInfo(hInfo, 3, iTagIndex);
            iEngineeringValue = StrToInt(DsplInfo(hInfo, 5, iTagIndex));
        ..
    END
```

```
    ..  
END  
END
```

## See Also

[Display Functions](#)

### DspInfoDestroy

Destroys an object information block created by DspInfoNew(). You should destroy an object information block when you no longer need it, to free Plant SCADA resources.

When the page (with which the object is associated) is closed, Plant SCADA automatically destroys the object information block.

## Syntax

**DspInfoDestroy(*hInfo*)**

*hInfo*:

The object information block handle, as returned by DspInfoNew(). This handle identifies the table (or block) where all object data is stored.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspInfo](#), [DspInfoNew](#), [DspInfoField](#), [DspInfoValid](#)

## Example

```
hInfo=DspInfoNew(20);  
! Do animation operation  
DspInfoDestroy(hInfo);
```

## See Also

[Display Functions](#)

### DspInfoField

Obtains static and real-time data from a variable tag. You get static data from the Variable Tags database. The additional field "Eng\_Value", returns dynamic real-time data for the variable tag. To get this real-time data, you need to first call the DspInfoNew() function to get the information block handle *hInfo*.

Getting the raw value of a variable tag using `DspInfoField` is no longer supported. To get the raw value of a tag, use the `TagSubscribe` function, specifying a value of "Raw" for the `sScaleMode` parameter. When using `TagSubscribe`, you can either call `SubscriptionGetAttribute` to obtain the value whenever required or register callback cicode function to run when the value changes. See `TagSubscribe` for more details.

## Syntax

`DspInfoField(hInfo, sTag, sField [, sClusterName] )`

*hInfo*:

The object information block handle, as returned by `DspInfoNew()`. This handle identifies the table (or block) where data on the object is stored. Set this handle to 0 (zero) if you do not require real-time data.

*sTag*:

The name of the variable tag. The name of the tag can be prefixed by the name of the cluster that is "ClusterName.Tag". This argument does not support arrays. If array syntax is used, the information will be retrieved for only the tag name.

*sField*:

The name of the field from which to extract the data:

- *Cluster* - Name of the cluster in which the Tag resides
- *Comment* - Variable tag comment
- *Eng\_Full* - Engineering Full Scale
- *Eng\_Zero* - Engineering Zero Scale
- *Eng\_Units* - Engineering Units
- *Eng\_Value* - Scaled engineering value - Dynamic
- *Field* - Description
- *FullName* - Full name of the tag in the form *cluster.tagname*.
- *Name* - Variable Tag Name
- *Type* - Data Type
- *Unit* - I/O Device Name

*sClusterName*:

Specifies the name of the cluster in which the Tag resides. This is optional if you have one cluster or are resolving the tag via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

The data (as a string).

## Related Functions

[DspInfo](#), [DspInfoNew](#), [DspInfoDestroy](#), [SubscriptionGetAttribute](#), [SubscriptionAddCallback](#), [TagSubscribe](#)

## Example

```
! Get the I/O device that Variable Tag "PV123" belongs to.  
IODev=DspInfoField(0,"PV123","Unit");  
! Get the real-time engineering value of a tag.  
hInfo=DspInfoNew(20);  
sTag=DspInfo(hInfo,3,0);  
EngValue=DspInfoField(hInfo,sTag,"Eng_Value");
```

## See Also

[Display Functions](#)

### DspInfoNew

Creates an object information block. Use this function with the associated low-level animation information functions to get and process object information on an AN.

---

**Note:** When you have finished with the object information block, you need to destroy it with the DspInfoDestroy() function. There are limited number of info 383 blocks that can be allocated, if they are not freed properly DspInfoNew will return -1.

---

If you need simple animation help, use the InfoForm() or the InfoFormAn() functions.

---

**Note:** DspInfoNew, InfoForm and InfoFormAn will not work if the parameter[General]SymbolicInfo = 0.

---

## Syntax

**DspInfoNew(*nAN*)**

*nAN*:

The AN for which object information is provided.

## Return Value

The object information block handle. If no object data is available, then -1 is returned.

## Related Functions

[DspInfo](#), [DspInfoField](#), [InfoForm](#), [InfoFormAn](#)

## Example

```
/*This example creates a form, with the title "Tag Info" and a  
size of 25 x 5 characters. It creates an information block for the  
AN closest to the mouse cursor and then extracts the name, I/O  
device, and engineering value for the first tag in the object  
expression.*/  
INT hInfo;  
STRING sTag;
```

```
hInfo=DspInfoNew(DspGetNearestAN());
IF hInfo>-1 THEN
    FormNew("Tag Info",25,5,2);
    sTag=DspInfo(hInfo,3,0);
    FormPrompt(0,0,sTag);
    FormPrompt(0,16,DspInfoField(hInfo,sTag,"Unit"));
    FormPrompt(0,32,DspInfoField(hInfo,sTag,"Eng_Value"));
    FormRead(0);
    DspInfoDestroy(hInfo);
END
```

## See Also

[Display Functions](#)

### DspInfoValid

Checks if an object information block handle is valid. An object information block handle becomes invalid after it is destroyed, or if the user closes the page it is associated with. Use this function if background Cicode is using the object information block, and the operator closes the page.

## Syntax

**DspInfoValid(*hInfo*)**

*hInfo*:

The object information block handle, as returned by DspInfoNew(). This handle identifies the table (or block) where all object data is stored.

## Return Value

1 if the information block handle is valid, otherwise 0 (zero).

## Related Functions

[DspInfoNew](#), [DspInfoField](#), [DsplInfoDestroy](#)

## Example

```
IF DspInfoValid(hInfo) THEN
    EngValue=DspInfoField(hInfo,sTag,"Eng_Value");
END
```

## See Also

[Display Functions](#)

## DspIsButtonGray

Gets the current status of a button.

**Note:** This function was designed to only be used on Cicode objects, or the animation point object from the version 3.xx/4.xx toolbox.

## Syntax

**DspIsButtonGray(nAN)**

*nAN*:

The AN for which object information is provided.

## Return Value

The current mode of the button:

- 0 - The button is active (not grayed).
- 1 - (SUNK\_GRAY) The button is inactive (the text or symbol on the button is recessed).
- 2 - (PART\_GRAY) This mode is now obsolete. The button will be inactive even if part\_gray is returned.
- 3 - (ALL\_GRAY) The button is inactive (the entire button is masked).

## Related Functions

[DspButton](#), [DspButtonFn](#), [DspGrayButton](#)

## Example

```
! Check the status of the button at AN21
status = DspIsButtonGray(21);
```

## See Also

[Display Functions](#)

## DspKernel

Displays the Kernel window.

To display the Kernel window, the user that is currently logged in needs to be assigned to a Role that has the **Kernel Access** property set to "Full Access" or "Read Only". A Read Only user can access the Kernel, but cannot perform some privileged commands like running Cicode. See [Display the Kernel Window](#).

**Note:** The special Kernel user that enabled Kernel access in versions prior to Plant SCADA 2023 R2 no longer has default access to the Kernel window.

Kernel access should be restricted to authorised personnel only as once they are in the Kernel, they can execute any Cicode function without further privilege restrictions and therefore have total control of Plant SCADA (and

subsequently the plant and equipment). Please be aware that you can also open the Kernel by setting the [Debug]Menu parameter to 1 and, when your system is running, selecting **Kernel** from the system menu.

You should be experienced with Plant SCADA and Cicode before attempting to use the Kernel as these facilities are powerful, and if used incorrectly, can corrupt your system.

**Note:**

- You should only use the Kernel for diagnostics and debugging purposes, and not for normal Plant SCADA operation.
- Kernel access should be restricted to authorised personnel only as once they are in the Kernel, they can execute any Cicode function without further privilege restrictions and therefore have total control of Plant SCADA (and subsequently the plant and equipment).

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

- Do not use the Kernel for normal Plant SCADA operation. The Kernel is only for diagnostics and debugging purposes.
- Configure your security so that only approved personnel can view or use the Kernel.
- Do not view or use the Kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of AVEVA Support.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Syntax

**DspKernel(*nMode*)**

*nMode*:

The display mode of Kernel:

1 - Display the Kernel. If the Kernel is already displayed and *nMode*=1, the keyboard focus is changed to the Kernel.

0 - Hide the Kernel

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[KerCmd](#), [TraceMsg](#)

## Example

```
DspKernel(1);  
!Display the Runtime Kernel window
```

## See Also

[Display Functions](#)

### DspMarkerMove

Moves a trend or chart marker to a specified position.

## Syntax

**DspMarkerMove(*nAN*, *hMarker*, *Offset*)**

*nAN*:

The AN where the trend or chart is positioned.

*hMarker*:

The marker handle, as returned from the DspMarkerNew() function. The marker handle identifies the table where all data on the associated marker is stored.

*Offset*:

The offset by which to move the marker. Vertical markers have an offset from 0 (zero) to the maximum number of samples in the trend. Horizontal markers have a offset of 0 (zero) to 32000.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspMarkerNew](#), [OnEvent](#)

## Example

See [DspMarkerNew](#).

## See Also

[Display Functions](#)

### DspMarkerNew

Creates a new trend marker. A trend marker is used to show cursor values or limits on a trend. You can use up to 10 markers on a single trend or chart.

If you add markers to a trend or chart that Plant SCADA is animating, you need to repaint them using the trend paint event (OnEvent(Window,22)). (Otherwise Plant SCADA will delete any markers displayed when the trend is updated.)

## Syntax

**DspMarkerNew(*nAN*, *Mode*, *Color*)**

*nAN*:

The animation-point number.

*Mode*:

The mode of the marker:

*0* - A vertical marker

*1* - A horizontal marker

*Color*:

The color of the marker (flashing color not supported). Select a color from the list of [Predefined Color Names and Codes](#) or create an RGB color using the function [MakeColour](#).

## Return Value

The marker handle, or -1 if the function is unsuccessful. The marker handle identifies the table where data on the associated marker is stored.

## Related Functions

[DspMarkerMove](#), [OnEvent](#)

## Example

```
INT offset; ! offset of marker
INT hMarker; ! marker handle
hMarker = DspMarkerNew(40, 0, WHITE);
! create a new marker, vertical WHITE
offset = 100;
DspMarkerMove(40, hMarker, offset);
! Moves marker to offset 100
OnEvent(22, MyTrendPaint);
! set trend paint event, needs to stop event when change pages
! this function is called when Plant SCADA updates the trend
INT
FUNCTION
MyTrendPaint()
    DspMarkerMove(40, hMarker, offset);
    RETURN 0;
END
```

## See Also

[Display Functions](#)

## DspMCI

Controls a multimedia device. The Media Control Interface (MCI) is a high-level command interface to multimedia devices and resource files. MCI provides applications with device-independent capabilities for controlling audio and visual peripherals (for example, for playing multimedia devices and recording multimedia resource files).

Using this function, you can control multimedia devices by using simple commands like open, play, and close. MCI commands are a generic interface to multimedia devices. You can control any supported multimedia device, including audio playback and recording. For a full overview of MCI, see the Windows *Multimedia Programmer's Guide*.

## Syntax

**DspMCI(*sCommand*)**

*sCommand*:

The MCI command. See Microsoft Windows documentation for details.

## Return Value

A string message with the status of the MCI command.

## Related Functions

[DspPlaySound](#)

## Example

```
DspMCI("open cdaudio")
DspMCI("set cdaudio time format tmsf")
DspMCI("play cdaudio from 6 to 7")
DspMCI("close cdaudio")
/*Plays track 6 of an audio CD*/
DspMCI("open c:\mmdata\purplefi.wav type waveaudio alias finch")
DspMCI("set finch time format samples")
DspMCI("play finch from 1 to 10000")
DspMCI("close finch")
/*Plays the first 10,000 samples of a waveform audio file*/
```

## See Also

[Display Functions](#)

## DspPlaySound

Plays a waveform (sound). Wave form sound files \*.wav are provided with Windows and by third-party developers, or you can record them yourself to play long (and complex) sound sequences.

**⚠ WARNING****LOSS OF CONTROL**

- Do not use Plant SCADA's alarming system as the primary alert mechanism for safety-related alarms or notifications (that is, those classified as critical to process safety, the protection of the plant and equipment, or the protection of human life).
- Do not use Plant SCADA's audible alarm functionality as a replacement for safety-related warning systems critical to process safety, the protection of the plant and equipment or the protection of human life.
- Safety-related alarms and notifications should be independent and separate from the process control system. They should be implemented in the form of a stand-alone physical alarm system driving individual discrete alarm annunciators.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

When the *sSoundname* is a sound file path and "Mode" is "0", this function loads the .wav file from the specified path (Sound Files) and plays the associated waveform. If the specified sound file is not available in the mentioned path, an error message will be returned and a hardware alarm will be reported. Use of a system sound in this mode is not supported and will result in an error message and a hardware alarm.

When the *sSoundname* is a system sound name and *nMode* is "1", this function searches the system for the entry with the specified name and plays the associated waveform. If the name cannot be found, there is no error message indicated as either a return value or hardware alarm. For this reason, the use of mode 0 with waveform files is recommended over mode 1 with system sounds. Further, use of a valid wave form sound file in this mode instead of mode "0" will result in an error message and a hardware alarm informing the wrong mode is in use.

## Syntax

**DspPlaySound(*sSoundname*, *nMode*)**

*sSoundname*:

Sound File Path: Sound file to be played from the specified path.

System Sound Name: The predefined system waveform to be played. Examples of predefined sounds include:

- SystemAsterisk
- SystemExclamation
- SystemQuestion
- SystemDefault
- SystemHand
- SystemExit
- SystemStart

*nMode*:

- Mode 0: For use when the "*sSoundname*" is a sound file path.
- Mode 1: For use when the "*sSoundname*" is a system sound name.

## Return Value

TRUE if successful, otherwise FALSE (if an error is detected).

## Related Functions

[Beep](#)

## Example

```
DspPlaySound("C:\WINNT\MEDIA\Notify.wav",0);
DspPlaySound("SystemStart",1);
```

## See Also

[Display Functions](#)

## DspPopupConfigMenu

Displays the contents of a menu node as a pop-up (context) menu, and run the command associated with the selected menu item. You can specify the contents of a menu using the menu configuration dialog at design time, or using the Menu family of Cicode functions at runtime.

## Syntax

**DspPopupConfigMenu(*hParent*, [, *bNonRecursive* [, *XPos* [, *YPos*]]])**

*hParent*

The parent node of the menu tree returned from any of the following functions:

- [MenuGetGenericNode\(\)](#), [MenuGetPageNode\(\)](#) or [MenuGetWindowNode\(\)](#) - used to get the parent node of menu tree for a page.
- [MenuGetFirstChild\(\)](#), [MenuGetNextChild\(\)](#), [MenuGetPrevChild\(\)](#), [MenuGetParent\(\)](#) - used to traverse to other nodes in a menu tree

*bNonRecursive*

Whether not to recursively transverse child tree nodes and list them as sub-menus in the pop-up menu. This parameter is optional. If it is left unspecified, its value will be defaulted to 0 (recursive). When it is set to 1, only the immediate child nodes of the specified menu handle will be listed. In this mode, tree nodes will be listed as normal menu items (instead of submenus) in the pop-up menu.

*XPos*

The x-coordinate (relative to the page) at which the menu will be displayed. This parameter is optional. If it is left unspecified, the menu will display at the cursor's current position.

*YPos*

The y-coordinate (relative to the page) at which the menu will be displayed. This parameter is optional. If it is left unspecified, the menu will display at the cursor's current position.

## Return Value

0 if the selected menu command is run or error code if menu command cannot run.

## See Also

[Display Functions](#)

### DspPopupMenu

Creates a popup menu consisting of a number of menu items. Multiple calls to this function enable you to add new items and create sub menus, building a system of linked, Windows-style menus.

Menu items can be displayed as checked and/or disabled. You can also specify a bitmap to display as a menu icon.

This function is first called to build the menu's items and links, and then called again to display it on the screen. In this final call, you have the option to specify the coordinates at which the menu will display, or let it default to the current cursor position.

## Syntax

**DspPopupMenu(*iMenuNumber*, *sMenuItems* [, *XPos*] [, *YPos*] )**

*iMenuNumber*:

An integer representing the menu you are adding items to. The first menu created is Menu 0. If left unspecified, this parameter defaults to -1, causing the menu to be displayed on the screen.

Multiple function calls with the same *iMenuNumber* allow you to build up entries in a particular menu. For example, the following four function calls with *iMenuNumber* = 1 build up 8 entries in Menu 1:

- DspPopupMenu(1, "Selection A>2, Selection B>3");
- DspPopupMenu(1, "Selection C>2, Selection D");
- DspPopupMenu(1, "Selection E>2, Selection F>3");
- DspPopupMenu(1, "Selection G>2, Selection H");

*sMenuItems*:

A comma-separated string defining the items in each menu. The default value for this parameter is an empty string, which will get passed to the function in the call to display the menu.

The (!), (~), and (,) symbols control display options for menu items.

For example, !Item1 disables Item1; ~Item2 checks Item2; and ,Item3 inserts a separator above Item3. To insert a link from a menu item to a sub menu, use the (>) symbol. For example, : Item4>1 means Item4 links to menu 1.

To insert a bitmap to the left of a menu item as its icon, use the following notation: [Icon]Item5 Inserts the bitmap Icon.BMP to the left of Item5. [Icon] needs to be placed before the Item name, but after any disable (!) or check (~) symbols you may wish to specify.

Bitmap files used for menu icons need to be saved in the project directory so that they can be found by Plant SCADA.

*XPos*:

The x-coordinate (relative to the page) at which the menu will be displayed. This parameter is optional. If it is left

unspecified, the menu will display at the cursor's current position.

*YPos:*

The y-coordinate (relative to the page) at which the menu will be displayed. This parameter is optional. If it is left unspecified, the menu will display at the cursor's current position.

## Return Value

The selected menu item as an integer. This comprises the menu number (return value div 100), and the position of the item in the menu (return value mod 100). For example, a return value of 201 indicates that the first item in Menu 2 was selected, and a return value of 3 indicates that the third item in Menu 0 was selected.

The return value is limited to a maximum of 65535, that is 655 menus and 35 items on the menu. Above this limit the function returns 0.

**Note:** Links to sub menus are not counted as menu items. For example, if your menu consists of 10 links and one unlinked item, the function will return only when the unlinked item is selected.

## Example 1

```
!Example 1 illustrates one menu with three menu items.  
FUNCTION BuildPopupMenu()  
    INT iSelection;  
    DspPopupMenu(0, "Item 1,!Item 2,~Item 3");  
    iSelection = DspPopupMenu(-1, "", 150, 300);  
    ! The above builds a menu with three items:  
    ! 'Item 1' will be shown as normal, 'Item 2' will be shown as disabled,  
    ! and 'Item 3' will be shown as checked.  
    ! The menu will be displayed at position (150, 300).  
END
```

## Example 2

```
!Example 2 illustrates the creation of two menus which are linked.  
FUNCTION BuildLinkedPopupMenu()  
    INT iSelection;  
    DspPopupMenu(0, "Item A,Item B>1,Item C");  
    DspPopupMenu(1, "Item B1,,[Trend]Item B2,,Item B3");  
    iSelection = DspPopupMenu();  
    ! The above will build two menus - Menu 0 and Menu 1  
    ! Item B on Menu 0 links to Menu 1.  
    ! 'Item B2' will be shown with Trend.BMP at its left.  
    ! The menu will be displayed at the cursor's position.  
    ! If 'Item A' is selected, iSelection will equal 1  
    ! If 'Item C' is selected, iSelection will equal 2  
    ! If 'Item B1' is selected, iSelection will equal 101  
    ! If 'Item B2' is selected, iSelection will equal 102  
    ! If 'Item B3' is selected, iSelection will equal 103  
END
```

## See Also

[Display Functions](#)

## DspRichText

Creates a Rich Text object of the given dimensions at the animation point *nAN*. This object can then be used to display an RTF file (like an RTF report) called using the DspRichTextLoad function.

## Syntax

**DspRichText(*nAN*, *iHeight*, *iWidth*, *nMode*)**

*nAN*:

The AN at which the rich text object will display when the DspRichText command is run.

*iHeight*:

The height of the rich text object in pixels. The height is established by measuring down from the animation point.

*iWidth*:

The width of the rich text object in pixels. The width is established by measuring across to the right from the animation point.

*nMode*:

The display mode for the rich text object. The mode can be any combination of:

0 - **Disabled** - should be used if the rich text object is to be used for display purposes only.

1 - **Enabled** - allows you to select and copy the contents of the RTF object (for instance an RTF report), but you will not be able to make changes.

2 - **Read/Write** - allows you to edit the contents of the RTF object. Remember, however, that the object needs to be enabled before it can be edited. If it has already been enabled, you can just enter Mode 2 as your argument. If it is not already enabled, you will need to enable it. By combining Mode 1 and Mode 2 in your argument (3), you can enable the object, and make it read/write at the same time.

Because the content of the rich text object is just a copy of the original file, changes will not affect the actual file, until saved using the [DspRichTextSave](#) function.

## Return Value

0 if successful, otherwise an error is returned.

## Related Functions

[DspRichTextLoad](#), [PageRichTextFile](#)

## Example

```
//This will produce a rich text object at animation point 57,  
which is 200 pixels high, and 200 pixels wide. This object will be  
for display purposes only (that is read only)//  
DspRichText(57,200,200,0);
```

## See Also

[Display Functions](#)

### DspRichTextEdit

Enables editing of the contents of the rich text object at *nAN* if *nEdit* = TRUE, and disables editing if *nEdit* = FALSE.

## Syntax

**DspRichTextEdit(*nAN*, *bEdit*)**

*nAN*:

The reference AN for the rich text object.

*bEdit*:

The value of this argument determines whether you will be able to edit the contents of the rich text object at AN. Enter TRUE to enable editing, or enter FALSE to make the contents read-only.

Changes made to the contents of the object will not be saved until the DspRichTextSave function is used.

## Return Value

0 if successful, otherwise an error is returned.

## Related Functions

[PageRichTextFile](#), [DspRichTextEnable](#), [DspRichTextSave](#)

## Example

```
// Enables editing of the rich text object at AN 25 - if one
exists. Otherwise an error will be returned to iResult //
iResult = DspRichTextEdit(25,TRUE);
```

## See Also

[Display Functions](#)

### DspRichTextEnable

Enables the rich text object at *nAN* if *nEnable* = TRUE, and disables the object if *nEnable* = FALSE. When the object is disabled, its contents cannot be selected or copied etc.

## Syntax

**DspRichTextEnable(*nAN*, *bEnable*)**

*nAN*:

The reference AN for the rich text object.

*bEnable*:

The value of this argument determines whether the rich text object at AN will be enabled or disabled. Enter TRUE to enable the object (that is you can select and copy the contents of the RTF object, but you can't make changes). Enter FALSE to disable the object (that is make it display only).

## Return Value

0 if successful, otherwise an error is returned.

## Related Functions

[DspRichTextEdit](#)

## Example

```
// This line disables the rich text object at AN 25 - if one
// exists. Otherwise an error will be returned to iResult //
iResult = DspRichTextEnable(25,FALSE);
```

## See Also

[Display Functions](#)

## DspRichTextGetInfo

Retrieves size information about the rich text object at animation point *nAN*.

## Syntax

**DspRichTextGetInfo(*nAN*, *iType*)**

*nAN*:

The reference AN for the rich text object.

*iType*:

The following size information (in pixels) can be returned about the specified rich text object:

- 0 - Height
- 1 - Width

## Return Value

The requested information as a string (units = pixels).

## Related Functions

[PageRichTextFile](#)

## Example

```
! Gets the height of the rich text object at AN 25 - if one exists.  
iHeight = DspRichTextGetInfo(25,0);  
! Gets the width of the rich text object at AN 423.  
iWidth = DspRichTextGetInfo(423,1);
```

## See Also

[Display Functions](#)

## DspRichTextLoad

Loads a copy of the file *Filename* into the rich text object) at animation point *nAN*. (The rich text object may have been created using either the DspRichTextLoad function or the PageRichTextFile function.)

## Syntax

**DspRichTextLoad(*nAN*, *sFilename*)**

*nAN*:

The animation point at which a copy of the rich text file (for example, an RTF report) will display. This AN needs to match that of a rich text object (created using either the DspRichText function, or the PageRichTextFile function), or the copy of the file will not be loaded into anything, and will not display.

*sFilename*:

The name of the file to be copied and loaded into the rich text object at the specified animation point. The filename needs to be entered in quotation marks "".

The maximum file size that can be loaded is 512kb.

If you are loading a copy of an RTF report, the report needs to already have been run and saved to a file. Remember that the filename for the saved report comes from the File Name field in the Devices form. The location of the saved file needs to also be included as part of the filename. For example, if the filename in the Devices form listed [Data];RepDev.rtf, then you would need to enter "[Data]\repdev.rtf" as your argument. Alternatively, you can manually enter the path, for example, "c:\MyApplication\data\repdev.rtf".

If you are keeping a number of history files for the report, instead of using the extension rtf, you need to change it to reflect the number of the desired history file, for example, 001.

## Return Value

0 if successful, otherwise an error is returned.

## Related Functions

[DspRichText](#), [PageRichTextFile](#)

## Example

```
// This will look in the [Data] path (as specified in the
Citect.ini file), and load a copy of the file DayRep.rtf into the
rich text object at animation point 57. //
DspRichTextLoad(57, "[Data]\DayRep.rtf");
// This will look in the [Data] path (as specified in the
Citect.ini file), and load a copy of the history file DayRep.003
into the rich text object at animation point 908. //
DspRichTextLoad(908, "[Data]\DayRep.003");
// This will load a copy of the history file
f:\MyApplication\data\DayRep.006, into the rich text object at animation
point 908. //
DspRichTextLoad(908, "f:\MyApplication\data\DayRep.006");
```

## See Also

[Display Functions](#)

## DspRichTextPgScroll

Scrolls the contents of the rich text object displayed at *nAN*, by one page length in the direction given in *direction*.

## Syntax

**DspRichTextPgScroll(*nAN*, *iDirection*)**

*nAN*:

The reference AN for the rich text object.

*iDirection*:

The direction in which you want to scroll each time this function is run. You can choose from the following:

1 - Left

2 - Right

3 - Up

4 - Down

8 - Scroll to top

16 - Scroll to bottom

## Return Value

0 if successful, otherwise an error is returned.

## Related Functions

[PageRichTextFile](#), [DspRichTextEdit](#), [DspRichTextScroll](#)

## Example

```
// This line scrolls the contents of the rich text object at AN 25
// down one page. Otherwise an error will be returned to iResult //
iResult = DspRichTextPgScroll(25,4);
// This line scrolls the contents of the rich text object at AN 423
// right one page. Otherwise an error will be returned to iResult //
iResult = DspRichTextPgScroll(423,2);
```

## See Also

[Display Functions](#)

## DspRichTextPrint

Prints the contents of the rich text object at animation point *nAN*, to the port *PortName*.

## Syntax

**DspRichTextPrint(*nAN*, *sPortName*)**

*nAN*:

The reference AN for the rich text object.

*sPortName*:

The name of the printer port to which the contents of the rich text object will be printed. This name needs to be enclosed within quotation marks "". For example "LPT1", to print to the local printer, or "\\Pserver\canon1" using UNC to print to a network printer.

## Return Value

0 if successful, otherwise an error is returned.

## Related Functions

[DspRichText](#), [FileRichTextPrint](#)

## Example

```
! This lines prints
DspRichTextPrint(25,"LPT1:");
```

## See Also

[Display Functions](#)

### DspRichTextSave

Saves the contents of the rich text object at animation point *nAN*, to the file *Filename*.

## Syntax

**DspRichTextSave(*nAN*, *sFilename*)**

*nAN*:

The reference AN for the rich text object.

*sFilename*:

The name under which the contents of the rich text object will be saved. This name needs to be enclosed within quotation marks "", and needs to include the destination path. For example "[Data]\saved.rtf".

## Return Value

0 if successful, otherwise an error is returned.

## Related Functions

[DspRichText](#), [PageRichTextFile](#), [DspRichTextLoad](#), [DspRichTextEdit](#)

## Example

```
// These lines show two different ways of saving the contents of
// the rich text object (at AN 25) to file DayRep.rtf//
DspRichTextSave(25,"[Data]\DayRep.rtf");
DspRichTextSave(25,"c:\MyApplication\data\DayRep.rtf");
```

## See Also

[Display Functions](#)

### DspRichTextScroll

Scrolls the contents of the rich text object displayed at *nAN*, in the direction given in *direction*, by the number of

lines/units given in *amount*. Remember that the height of a line varies according to the font used, therefore if you need to scroll absolute distances, it might be advisable to use the DspRichTextPgScroll function.

## Syntax

**DspRichTextScroll(*nAN*, *iDirection*, *iAmount*)**

*nAN*:

The reference AN for the rich text object.

*iDirection*:

The direction in which you want to scroll each time this function is run. You can choose from the following:

1 - Left

2 - Right

3 - Up

4 - Down

8 - Scroll to top

16 - Scroll to bottom

*iAmount*:

The amount by which you would like to scroll each time this function is run. Enter the number of lines (for a vertical direction) or units (for a horizontal direction) by which you would like to scroll.

## Return Value

0 if successful, otherwise an error is returned.

## Related Functions

[PageRichTextFile](#), [DspRichTextEdit](#), [DspRichTextPgScroll](#)

## Example

```
DspRichTextScroll(25,4,8);
DspRichTextScroll(423,2,1);
```

## See Also

[Display Functions](#)

## DspRubEnd

Ends the rubber band selection, and returns the coordinates of the rubber band selection. The meaning of the *cx* and *cy* values depend on the *nMode* you specify in the DspRubStart() function.

## Syntax

**DspRubEnd(x, y, cx, cy)**

*x,y:*

The x and y coordinates of the start position. Must be a Long type variable.

*cx,cy:*

The x and y coordinates of the end position. Must be a Long type variable.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspRubStart](#), [DspRubMove](#), [DspRubSetClip](#)

## Example

See [DspRubStart](#).

## See Also

[Display Functions](#)

## DspRubMove

Moves the rubber band selection to the new position. You need to first have defined a rubber band selection using the DspRubStart() and DspRubEnd() functions.

This function will erase the existing rubber band and then redraw it in the new position. You would normally move the rubber band by mouse input, but you can get input from the keyboard or any other Cicode to control the rubber band.

## Syntax

**DspRubMove(x, y)**

*x,y:*

The x and y coordinates of the current position.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspRubStart](#), [DspRubEnd](#), [DspRubSetClip](#)

## Example

See [DspRubStart](#).

## See Also

[Display Functions](#)

## DspRubSetClip

Sets the clipping region for the rubber band display. If you enable the clipping region, the rubber band will not move outside of the clip region. This allows you to restrict the rubber band to within some constrained region. (For example, to prevent an operator from dragging the rubber band outside of the trend display when zooming the trend.)

You need to call this function (to enable the clipping region) before you can start the rubber band selection (with the DspRubStart() function).

## Syntax

**DspRubSetClip(x1, y1, x2, y2)**

*x1,y1,x2,y2:*

The x and y coordinates of the clipping region.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspRubStart](#), [DspRubEnd](#), [DspRubMove](#)

## Example

```
// Set the clipping region to a rectangle starting at 100, 100 to
// 200, 300
DspRubSetClip(100, 100, 200, 300);
// Start the rubber band display with clipping mode on
DspRubStart(x, y, 4);
```

## See Also

[Display Functions](#)

### DspRubSetColor

Sets the color of the rubber band tool that is used to select a section within a trend object.

## Syntax

**DspRubSetColor(INT Color)**

*Color:*

An integer representing the color to use for the rubber band. The default value is white (0xFFFFFFFF).

There are a set of labels available in the Include project that you can use:

BLACK — 0x00000000

BLUE — 0x00000080

GREEN — 0x00008000

CYAN — 0x00008080

RED — 0x00800000

MAGENTA — 0x00800080

BROWN — 0x00808000

GREY — 0x00BFBFBF

DARK\_GREY — 0x007F7F7F

LIGHT\_BLUE — 0x000000FF

LIGHT\_GREEN — 0x0000FF00

LIGHT\_CYAN — 0x0000FFFF

LIGHT\_RED — 0x00FF0000

LIGHT\_MAGENTA — 0x00FF00FF

YELLOW — 0x00FFFF00

WHITE — 0x00FFFFFF

---

**Note:** If your project is based on the Situational Awareness Starter Project, this value will be overwritten by the color theme setting specified for the project.

---

## Return Value

The previous rubber band color as an integer.

## Example

```
DspRubSetColor(WHITE); // The rubber band will be white (the default)
DspRubSetColor(MakeColour(221, 221, 221)); // The rubber band will be grey
```

```
DspRubSetColor(MakeColour(255, 0, 0)); // The rubber band will be red
DspRubSetColor(MakeColour(0, 255, 0)); // The rubber band will be green
DspRubSetColor(MakeColour(0, 0, 255)); // The rubber band will be blue
```

## See Also

[Display Functions](#)

### DspRubStart

Starts the rubber band selection. Call this function when the left mouse button is pressed - the rubber band is displayed at the starting position. Call the DspRubEnd() function to end the selection, when the mouse button is released. The DspRubMove() function moves the selection to the new position.

This function is used by the trend templates for the trend zoom function. Use the rubber band functions whenever you want the operator to select a region on the screen or display a dynamic rectangle on the screen.

You can only display one rubber band per page. If you display a second rubber band, the first rubber band is erased. To move a rubber band with the mouse, use the OnEvent() function to get notification of the mouse movement, and then the DspRubMove() function. Because these are generic rubber-band display functions, you can get input from the keyboard, Cicode variables, the I/O device, and the mouse.

## Syntax

**DspRubStart(*x*, *y*, *nMode*)**

*x,y*:

The x and y coordinates of the current position.

*nMode*:

The mode of the rubber banding operation:

0 - cx,cy as absolute pixel positions

1 - cx,cy in pixels relative to x,y

2 - (x,y) the distance from top left to (cx,cy)

4 - enable the rubber band selection using the clipping region defined by DspRubSetClip().

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspRubEnd](#), [DspRubMove](#), [DspRubSetClip](#), [OnEvent](#)

## Example

See also the ZOOM.CI file in the Include project for details.

```
INT xRub, yRub, cxRub, cyRub;
```

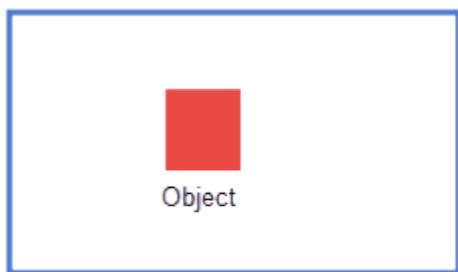
```
/* Call this function on left mouse button down. */
FUNCTION
StartSelection()
    INT x,y;
    DspGetMouse(x,y); ! Get the current mouse position
    DspRubStart(x,y,0); ! Start the rubber banding
    OnEvent(0,MouseEvent); ! Attach mouse move event
END
/* Call this function on left mouse button up. */
FUNCTION
EndSelection()
    ! Stop the rubber banding and get sizes into the ..Rub variables
    DspRubEnd(xRub,yRub,cxRub,cyRub);
    OnEvent(0,0); ! Stop mouse move event
END
INT
FUNCTION
MouseEvent()
    INT x,y;
    DspGetMouse(x,y); ! Get mouse position
    DspRubMove(x,y); ! Move the rubber band
    RETURN 0
END
```

## See Also

[Display Functions](#)

### DspSetClip

Sets the clip rectangle for the AN of an object or group of objects, defined by the top, left, right and bottom coordinates.



Clipping Rectangle



Object is not rendered outside clipping rectangle

If an AN has a clip rectangle specified, then any part (object) of the AN that is outside of the rectangle will not be rendered. You can use a clip rectangle for "control" type genies like treeviews that have scrollbars, to clip the contents of the control to the control's area when the scrollbar is moved.

A clip rectangle can be set for a group which will affect all the children of the group (and their children).

A clip rectangle set on a child will override the clip set on the group (this also applies to nested groups).

When an AN is clipped (i.e. it is outside the clip rectangle) then it will not respond to mouse input i.e. touch commands will not be invoked, tooltips will not be displayed, and the kobject will not be given the focus rectangle for keyboard input. If it is partially clipped, then this only applies to the clipped part, the visible part still responds as normal.

## Syntax

**DspSetClip(INT nAN, INT nLeft, INT nTop, INT nRight, INT nBottom)**

*nAN:*

The animation-point number of the object or group of objects inside the clipping region.

*nLeft:*

The x coordinate of the left side of the clipping boundary

*nTop:*

The x coordinate of the top of the clipping boundary

*nRight:*

The y co-ordinate of the right side of the clip rectangle.

*nBottom:*

The y co-ordinate of the bottom of the clip rectangle.

## Return Value

It returns an error if nAN is invalid, or the specified rectangle is invalid i.e. nLeft >= nRight or nTop >= nBottom.

---

**Note:** Will return an error if nAN specifies an ActiveX control.

---

## Example

```
DspSetClip(17, 200, 200, 800, 500);
```

## See Also

[Display Functions](#)

## DspSetCurColor

Sets the color of the focus rectangle. The focus rectangle is used to highlight a selectable object, tab or list item as the mouse moves over it.

## Syntax

**DspSetCurColor(INT Color)**

**Color:**

An integer representing the color to use for the focus rectangle. The default value is white (0x00FFFFFF).

There are a set of labels available in the Include project that you can use:

BLACK — 0x00000000  
BLUE — 0x00000080  
GREEN — 0x00008000  
CYAN — 0x00008080  
RED — 0x00800000  
MAGENTA — 0x00800080  
BROWN — 0x00808000  
GREY — 0x00BFBFBF  
DARK\_GREY — 0x007F7F7F  
LIGHT\_BLUE — 0x000000FF  
LIGHT\_GREEN — 0x0000FF00  
LIGHT\_CYAN — 0x0000FFFF  
LIGHT\_RED — 0x00FF0000  
LIGHT\_MAGENTA — 0x00FF00FF  
YELLOW — 0x00FFFF00  
WHITE — 0x00FFFFFF  
TRANSPARENT - 0xFF000000

---

**Note:** If your project is based on the Situational Awareness Starter Project, this value will be overwritten by the color theme setting specified for the project.

---

## Return Value

The previous focus rectangle color as an integer.

## Example

```
DspSetCurColor(WHITE); // The focus rectangle will be white (the default)
DspSetCurColor(MakeCitectColour(221, 221, 221)); // The focus rectangle will be grey
DspSetCurColor(MakeCitectColour(255, 0, 0)); // The focus rectangle will be red
DspSetCurColor(MakeCitectColour(0, 255, 0)); // The focus rectangle will be green
DspSetCurColor(MakeCitectColour(0, 0, 255)); // The focus rectangle will be blue
```

## See Also

[Display Functions](#)

### DspSetMetadataFromName

Name used to set the metadata of an object on the page. Use the following syntax:

[RelativePath].[Group].[Group].Control

## Syntax

**DspSetMetadataFromName(*sName*, *sMetaName*, *sValue*)**

*sName*

The Name used as a reference for the object.

*sMetaName*

The Name of the metadata to be returned.

*sValue*

The value of the metadata to be returned.

## Return Value

The value of the metadata or error code.

## Related Functions

[DspSetMetadataFromName](#)

## See Also

[Display Functions](#)

## DspSetMetadataFromNameRelative

Name used to set the metadata of an object on the page relative to the given AN. Use the following syntax:

[RelativePath].[Group].[Group].Control

This will browse the group hierarchy to the given AN, and then drill down to other groups to find the named control or object.

See [Referencing an Object Using a Name at Runtime](#) for more information.

## Syntax

**DspSetMetadataFromNameRelative(*hAN*, *sName*, *sMetaName*, *sValue*)**

*hAN*

AN used as the starting point for the search of the object name.

*sName*

The Name used as a reference for the object.

*sMetaName*

The Name of the metadata to be returned.

*sValue*

The Name of the metadata to be returned.

## Return Value

The value of the metadata or error code

## Related Functions

[DspGetMetadataFromNameRelative](#)

## See Also

[Display Functions](#)

## DspSetFont

Allows you to set the font and font size for text that appears in popup menus at runtime. For example, you can use this function to adjust the size of the text that appears in context menus to suit a particular screen resolution.

**Note:** Background color and foreground color properties are not supported on popup menus.

## Syntax

**DspSetFont(*hFont*)**

*hFont*:

The handle of font to be used.

## Return Value

No return value.

## Related Functions

[DspSetTooltipFont](#), [DspFont](#), [DspFontHnd](#)

## Example

```
IF (StrEndsWith(sPage, "_HD1080") = TRUE) THEN
    hFont = DspFontHnd("SA_Menu");
ELSE
    IF (StrEndsWith(sPage, "_UHD4K") = TRUE) THEN
        hFont = DspFontHnd("SA_Menu4K");
END
IF (hFont <> -1) THEN
    DspSetFont(hFont);
END
```

## See Also

[Display Functions](#)

### DspSetSlider

Sets the current position of a slider at the specified AN. You can use this function to move a slider, and adjust the value of the variable associated with the slider.

**Note:** This function was designed to only be used on Cicode objects, or the animation point object from the version 3.xx/4.xx toolbox.

## Syntax

**DspSetSlider(*nAN*, *nPos*)**

*nAN*:

The animation-point number.

*nPos*:

The position of the slider from 0 to 32000 where 0 is the zero position of the slider and 32000 if full position of the slider.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspGetSlider](#)

## Example

```
// Set the position of the slider at AN 30 to 1/2 scale
DspSetSlider(30, 16000);
```

## See Also

[Display Functions](#)

### DspSetTip

Sets tool tip text associated with an AN. Any existing text associated with the AN will be replaced with the new text.

## Syntax

**DspSetTip(*nAN*, *sText*)**

*nAN*:

The animation-point number.

*sText*:

The tool tip text to set for the AN.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspGetTip](#), [DspTipMode](#)

## Example

```
!Set a tool tip for AN19
DspSetTip(19, "Start Slurry Pump");
```

## See Also

[Display Functions](#)

## DspSetTooltipFont

Sets the font for tool tip text.

The parameter [Animator]TooltipFont also specifies the tool tip font. The parameter is checked at startup, and if it is set, the font is set accordingly. You can then use DspSetTooltipFont() to override the parameter until the next time you start Plant SCADA.

## Syntax

**DspSetTooltipFont(*sName* [, *nPointSize*] [, *sAttribs*] )**

*sName*:

The name of the Windows font to be used, enclosed by quotation marks ". A value for this parameter is required, however specifying an empty string "" will set the tooltip font to the default of MS Sans Serif.

*nPointSize*:

The size of the font in points. If you do not specify a value, the point size defaults to 12.

*sAttribs*:

A string specifying the format of the font. Use one or all of the following, enclosed by quotation marks " ":

- *B* to specify Bold
- *I* to specify Italics
- *U* to specify Underline

If you don't specify a value for this parameter, it will default to an empty string and no formatting will be applied.

## Return Value

No return value.

## Related Functions

[DspGetTip](#), [DspTipMode](#), [DspSetPopupMenuFont](#)

## Example

```
!Set the tool tip font to Bold, Italic, Times New Roman, with a
point size of 12
DspSetTooltipFont("Times New Roman", 12, "BI");
```

## See Also

[Display Functions](#)

## DspStatus

Determines whether the object at the specified AN will be grayed (hatch pattern) in the event communication attempts are unsuccessful.

## Syntax

**DspStatus**(*nAN*, *nMode*)

*nAN*:

The animation-point number.

*nMode*:

0 - Normal display when communication attempts are unsuccessful

1 - Gray the object (with a hatch pattern) when communication attempts are unsuccessful

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Example

```
DspStatus(67, 1)
```

```
// Disable the animation at AN 67
```

## See Also

[Display Functions](#)

### DspStr

Displays a string at a specified AN.

**Note:** This function was designed to only be used on Cicode objects, or the animation point object from the version 3.xx/4.xx toolbox.

## Syntax

**DspStr(*nAN*, *sFont*, *sText* [, *iLength*] [, *iAlignMode*] [, *iLengthMode*] [, *bTooltip*])**

*nAN*:

The AN where the text will be displayed.

*sFont*:

The name of the font that is used to display the text. The Font Name needs to be defined in the Fonts database. If the font is not found, the default font is used.

*sText*:

The text to display.

*iLength*:

Length of the text to display, either in characters or pixels depending on Mode (default -1, no truncation)

*iAlignMode*:

The alignment of the text string:

0 - Left Justified (default)

1 - Right Justified.

2 - Center Justified.

*iLengthMode*:

The length mode of the text string:

0 - Length as pixels truncated (default)

1 - Length as pixels truncated with ellipsis

2 - Length interpreted as characters.

*bTooltip*:

Determines if the original text is provided as tool tip when the displayed text becomes truncated.

0 - Tool tip is not available (default).

1 - Tool tip is available.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspText](#)

## Example

```
DspStr(25,"RedFont","Display this text");
/* Displays "Display this text" using "RedFont" at AN25. "RedFont"
needs to be defined in the Fonts database. */
```

## See Also

[Display Functions](#)

## DspSym

Displays a symbol or a Genie at a specified AN. This dynamically displayed object will disappear at page navigation.

To display a Genie, the Genie should be configured to be used with this function. See "Create a New Genie" topic for more information.

## Syntax

**DspSym**(*iAN*, *sSymbol* [, *iMode*] [, *iType*] )

*iAN*:

The AN where the symbol will be displayed.

*sSymbol*:

The name of the symbol to display in the format <[LibName.]SymName>. If you do not specify the library name, a symbol from the Global library will display (if it exists).

*iMode*:

Not used. The mode is always set to 1, which means do not erase the existing symbol, just draw the new symbol.

*iType*:

Symbol Type:

0 - Symbol. *sSymbol* parameter is interpreted as a symbol. If you do not specify the library name, a symbol from the Global library will display (if it exists).

1 - Dynamic Genie, *sSymbol* parameter is interpreted as a Genie.

2 - Priority Genie. *sSymbol* is interpreted as a priority which is a numeric key in the alarm priority table and the function selects "Library Name.Genie Name" from the table.

3 - Priority Genie Thumbnail. *sSymbol* is interpreted as a priority which is a numeric key in the alarm priority

table and the function selects "Library Name.Thumbnail Name" from the table.

4 - Mode Genie. sSymbol is interpreted as a mode which is a display name in the alarm mode table. The function selects "Library Genie Name" from the table.

5 - .Mode Genie Thumbnail. sSymbol is interpreted as a mode which is a display name in the alarm mode table. The function selects "Library Genie Name" from the table.

---

**Note:** Type 0 symbols do not erase existing objects whilst type 1, 2, 3, 4, 5 symbols erase existing objects at the AN.

---

## Return Value

0 (zero) if successful, otherwise one of the following errors will be returned. A hardware alarm is also raised against DspSym when there is an error message.

- If an invalid type is passed, for example, DspSym(600, "lib1.genie1", 1, 99), the error 274 "Invalid Argument Passed" is returned.
- If a genie name which does not exist is passed, or, the option "I want to use this Genie with the Cicode function DspSym" is not selected in the Graphics Builder, the error 350 "RDB file not found" is returned.
- If an invalid animation number is passed, for example, DspSym(-99, "lib1.genie1", 1, 1), the error 274 "Invalid Argument Passed" is returned.
- If an animation number which does not exist is passed, the error 303 "Invalid animation number" is returned.
- If the AN where the genie will be displayed is a part (child) of the genie that was dynamically created, the error 408 "Databrowse not supported" is returned.
- For other runtime errors, for example, it did not create an animation for one of the objects of the genie, the error 272 "Out of memory" is returned.

## Related Functions

[DspDel](#)

## Example

```
! Display the centrifuge symbol (from the pumps library) at AN25.  
DspSym(25,"Pumps.Centrifuge");  
  
! Display the centrifuge symbol (from the global library) at AN26.  
DspSym(26,"Centrifuge");  
! Display the dspgenie1 genie from testlib library at an animation number indicated by iAn1.  
DspSym(iAn1, "testlib.dspgenie1", 1, 1)  
! Display the genie specified in the record of priority 1 on alarm priority table,  
specified by Library Name and Genie Name fields, at an animation number indicated by iAn4.  
iResult = DspSym(iAn4, "1", 1, 2);  
! Display the genie thumbnail specified in the record of priority 1 on alarm priority  
table, specified by Library Name and Thumbnail Name fields, at an animation number  
indicated by iAn9.  
DspSym(iAn9, "3", 1, 3);  
! Display the genie specified in the record of "Shelved / Disabled" on Alarm Modes table,
```

```
specified by Library Name and Genie Name fields, at an animation number indicated by iAn4.  
iResult = DspSym(iAn4, "Shelved / Disabled", 1, 4);  
! Display the genie specified in the record of "Shelved / Disabled" on Alarm Modes table,  
specified by Library Name and Thumbnail Name fields, at an animation number indicated by  
iAn7.  
iResult = DspSym(iAn7, "Shelved / Disabled", 1, 5);
```

## See Also

[Display Functions](#)

### DspSymAnm

Animates a series of symbols at an AN. *Sym1* displays first, then *Sym2*, *Sym3* . . . *Sym8* and then *Sym1* displays again, etc. When the next symbol in the sequence is displayed, the current symbol is not erased, but is overwritten to provide a smoother animation. The symbols should all be the same size.

The frequency of changing the symbols is determined by the [Page]AnmDelay parameter.

You only need to call this function once to keep the animation going. To stop the animation call the DspDel() function, or call this function again with different symbols (to change the animation).

---

**Note:** This function was designed to only be used on Cicode objects, or the animation point object from the version 3.xx/4.xx toolbox.

---

## Syntax

**DspSymAnm(*nAN*, *Sym1* [, *Sym2* ... *Sym8*] [, *iDisplayMode*] [, *sSym9*] )**

*nAN*:

The AN where the animation will occur.

*Sym1*:

The name of the first symbol to animate in the format <[LibName.]SymName>. If you do not specify the library name, a symbol from the Global library will display (if it exists). Sym1 needs to be specified.

*Sym2..Sym8*:

The names of the symbols to animate in frames 2 to 8 in the format <[LibName.]SymName>. If you do not specify the library name, a symbol from the Global library will display (if it exists).

*iDisplayMode*:

Not used. Always set to -1, which means **Soft animation**. The background screen (a rectangular region beneath the symbol) is restored with the original image. Any objects that are within the rectangular region are destroyed when the background is restored. Use this mode when each animation symbol is a different size.

*Sym9*:

Not all symbols have to be specified. If for example, only two symbols are to display, specify Sym1 and Sym2.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspSym](#)

## Example

```
DspSymAnm(25, "Pumps.Centrifuge", "Pumps.Floataion");
! Alternately displays the centrifuge symbol and the flotation symbol
(from the pumps library) at AN25.
```

## See Also

[Display Functions](#)

### DspSymAnmEx

Animates a series of symbols at an AN. *Sym1* displays first, then *Sym2*, *Sym3* . . . *Sym9* and then *Sym1* displays again, etc. When the next symbol in the sequence is displayed, the current symbol is not erased, but is overwritten to provide a smoother animation. The symbols should all be the same size.

The frequency of changing the symbols is determined by the [Page]AnmDelay parameter.

You only need to call this function once to keep the animation going. To stop the animation call this function again with a different *Mode*.

**Note:** This function was designed to only be used on Cicode objects, or the animation point object from the version 3.xx/4.xx toolbox.

## Syntax

**DspSymAnmEx(*nAN*, *Mode*, *Sym1* [, *Sym2* ... *Sym9*] )**

*nAN*:

The AN where the animation will occur.

*Mode*:

Not used. Always set to -1, which means **Soft animation**. The background screen (a rectangular region beneath the symbol) is restored with the original image. Any objects that are within the rectangular region are destroyed when the background is restored. Use this mode when each animation symbol is a different size.

*Sym1*:

The name of the first symbol to animate in the format <[LibName.]SymName>. If you do not specify the library name, a symbol from the Global library will display (if it exists). Sym1 needs to be specified.

*Sym2..Sym9*:

The names of the symbols to animate in frames 2 to 9 in the format <[LibName.]SymName>. If you do not specify the library name, a symbol from the Global library will display (if it exists).

Not all symbols have to be specified. If for example, only two symbols are to display, specify Sym1 and Sym2.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspSym](#)

## Example

```
DspSymAnmEx(25,-1,"Pumps.Centrifuge","Pumps.Floatation");
! Alternately displays the centrifuge symbol and the flotation symbol
(from the pumps library) at AN25.
```

## See Also

[Display Functions](#)

## DspSymAtSize

Displays a symbol or a genie at the specified scale and offset from the AN position.

By calling this function continuously, you can move symbols or genies around the screen and change their size and shape, to simulate trippers, elevators, and so on. You change the PositionX, PositionY values to change the position of the symbol/genie, the SizeX, SizeY values to change its size, or the symbol/Genie itself to change its shape.

You can only use this function at a blank AN, or an AN with a symbol defined without symbols configured. The AN needs to not be attached to any other animation object.

To display a Genie, the Genie should be configured to be used with this function. See "Create a New Genie" topic for more information.

## Syntax

**DspSymAtSize(*nAN, sSym, PositionX, PositionY, SizeX, SizeY, Mode [,Type]*)**

*nAN:*

The AN where the symbol will be animated.

*sSym:*

The name of the symbol to display, move, or size. If *sSym* is 0 (zero), any existing symbol at the AN is erased.

*PositionX:*

The horizontal offset position (from the AN) of the symbol (in pixels).

*PositionY:*

The vertical offset position (from the AN) of the symbol (in pixels).

*SizeX, SizeY:*

The horizontal and vertical scaling factors for the symbol (0 - 32000). For example, if *SizeX* and *SizeY* are both

32000, the symbol is displayed at its normal size. Please be aware that symbols can only be reduced in size.

*Mode:*

The mode of the display:

**-1 - Soft animation.** The background screen (a rectangular region beneath the symbol) is restored with the original image. Any objects that are within the rectangular region are destroyed when the background is restored. Use this mode when each animation symbol is a different size.

**0 - Overlap animation.** The background screen (beneath the symbol) is not erased - the next symbol is displayed on top. Transparent color is supported in this mode, allowing for symbol overlap. For this mode to display correctly, each symbol needs to be the same size.

**1 - Animate animation.** The background screen (beneath the symbol) is not erased - the next symbol is displayed on top. This mode provides the fastest animation. For this mode to display correctly, each symbol needs to be the same size. Transparent color is not supported in this mode.

**8 - Stops animation at last symbol displayed.** Use this mode where you want to freeze your animation at the end of the sequence.

**16 - Stops animation at current symbol displayed.** Use this mode where you want to freeze your animation instantly.

*Type:*

Symbol Type:

**0 - Symbol.** sSym parameter is interpreted as a symbol. If you do not specify the library name, a symbol from the Global library will display (if it exists).

**1 - Dynamic Genie.** sSym parameter is interpreted as a Genie.

**2 - Priority Genie.** sSym is interpreted as a priority which is a numeric key in the alarm priority table and the function selects "Library Name.Genie Name" from the table.

**3 - Priority Genie Thumbnail.** sSym is interpreted as a priority which is a numeric key in the alarm priority table and the function selects "Library Name.Thumbnail Name" from the table.

**4 - Mode Genie.** sSym is interpreted as a mode which is a display name in the alarm mode table. The function selects "Library Genie Name" from the table.

**5 - .Mode Genie Thumbnail.** sSym is interpreted as a mode which is a display name in the alarm mode table. The function selects "Library Genie Name" from the table.

---

**Note:**

- Type 0 symbols do not erase existing objects whilst type 1, 2, 3, 4, 5 symbols erase existing objects at the AN.
  - When using type 1, 2, 3, 4, 5, only mode 0 is supported
- 

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspAnMove](#), [DspAnMoveRel](#), [DspSym](#)

## Example

```
! Display tripper moving in x axis at normal size.  
DspSymAtSize(21, "lib.tripper", x, 0, 32000, 32000, 0);  
! Display elevator going up and down.  
DspSymAtSize(22, "lib.elevator", 0, y, 32000, 32000, 0);  
! Display can getting bigger and smaller.  
DspSymAtSize(23, "lib.can", 0, 0, size, size, 0);  
!Display genie1 50% smaller than its original size  
DspSymAtSize(24, "lib.genie1", 0, 0, 16000, 16000, 0, 1);
```

## See Also

[Display Functions](#)

### DspText

Displays text at a specified AN location. This function does the same operation as DspStr(), however it uses a font number rather than a font name.

---

**Note:** This function was designed to only be used on Cicode objects, or the animation point object from the version 3.xx/4.xx toolbox.

---

## Syntax

**DspText(*hAN*, *iFont*, *sText* [, *iLength*] [, *iAlignMode*] [, *iLengthMode*] [, *bTooltip*])**

*hAN*:

The AN where the text will be displayed.

*iFont*:

The font number that is used to display the text. (To use the default font, set to -1.)

*sText*:

The text to display.

*iLength*:

Length of the Text to display, either in characters or pixels depending on Mode (default -1, no truncation).

*iAlignMode*:

The alignment of the text string:

0 - Left Justified (default).

1 - Right Justified.

2 - Center Justified.

*iLengthMode*:

The length mode of the text string:

0 - Length as pixels truncated (default).

1 - Length as pixels truncated with ellipsis.

2 - Length interpreted as characters.

*bTooltip:*

Determines if the original text is provided as tool tip when the displayed text becomes truncated.

0 - Tool tip is not available (default).

1 - Tool tip is available.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspStr](#), [DspFont](#), [DspFontHnd](#)

## Example

```
/* Displays "Display this text" at AN25 using the font defined as
BigFont. */
hBigFont=DspFontHnd("BigFont");
DspText(25,hBigFont,"Display this text");
```

## See Also

[Display Functions](#)

## DspTipMode

Switches the display of tool tips on or off. This function overrides the setting in the [\[Page\]TipHelp](#) parameter.

## Syntax

**DspTipMode(*nMode*)**

*nMode:*

The display mode:

0 - Off

1 - On

2 - Toggle the tool tip mode

3 - Do not change the mode, just return the current value

## Return Value

The old mode.

## Related Functions

[DspSetTip](#), [DspGetTip](#)

## Example

```
DspTipMode(1); //Switch on tool tips
```

## See Also

[Display Functions](#)

## DspTrend

Displays a trend at an AN. Values are plotted on the trend pens. You need to specify *Value1*, but *Value2* to *Value8* are optional. If more values (than configured pens) are specified, the additional values are ignored. If fewer values (than configured pens) are specified, the pens that have no values are not displayed.

`DspTrend()` is optimized so that it will not display the trend until a full set of samples has been collected. For example, if you have defined 100 samples for your trend, the trend will not display until value 100 is entered.

You should use this function only if you want to control the display of trends directly. If you use the standard Trends (defined in the Trends database) this function is called automatically.

---

**Note:** This function was designed to only be used on Cicode objects, or the animation point object from the version 3.xx/4.xx toolbox.

---

## Syntax

**DspTrend(*nAN*, *Trend*, *Value1* [,*Value2* ... *Value8*])**

*nAN*:

The AN where the trend will be displayed.

*Trend*:

The name of the trend to display in the format <[LibName.]TrnName>. If you do not specify the library name, a trend from the Global library will display (if it exists).

To display a Version 1.xx trend, specify the trend number (0 to 255). For example, if you specify trend 1, Plant SCADA displays the trend Global.Trn001.

*Value1*:

The value to display on Pen 1 of the trend.

*Value2...8*:

The values to display on Pen 2...Pen 8 of the trend (optional).

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspDel](#)

## Example

```
/* Using the main_loop trend (from the trends library) at AN25,
display a value of 10 on Pen1, 20 on Pen2, 30 on Pen3 and 40 on
Pen4 of the trend. */
DspTrend(25,"Trends.Main_Loop",10,20,30,40);
/* Using trend definition 5 (Plant SCADA Version 1.xx)
at AN25, display a value of 10 on Pen1, 20 on Pen2, 30 on Pen3 and
40 on Pen4 of the trend. */
DspTrend(25,5,10,20,30,40);
/* Using the loops trend (from the global library) at AN26,
display a value of 100 on Pen1 and 500 on Pen2 of the trend. */
DspTrend(26,"Loops",100,500);
/* Display a trend configured with 100 samples immediately. The
data for the first 100 samples is stored in an array -
MyData[100]. On first display, grab all the data and call
DspTrend().*/
FOR i = 0 to 100 DO
    DspTrend(AN, "Loops", MyData[i]);
END
// display new samples every 300ms
WHILE TRUE DO
    // Shift MyData down and grab new sample
    TableShift(MyData, 100, 1);
    MyData[99] = MyFastData;
    DspTrend(AN, "Loops", MyData[99]);
    SleepMS(300);
END
/* Display a trend configured with 100 samples immediately. Dummy
data is pushed into the first 100 samples to fill the trend. Once
these values are entered, the trend will be updated each time a
new sample value is entered.*/
// fill up the trend.
FOR i = 0 to 100 DO
    DspTrend(AN, "Loops", 0);
END
// display new samples every 300ms
WHILE TRUE DO
    DspTrend(AN, "Loops", MyFastData);
    SleepMS(300);
END
```

## See Also

[Display Functions](#)

## DspTrendInfo

Get information on a trend definition.

## Syntax

**DspTrendInfo(*hTrend*, *Type*, *AN*)**

*hTrend*:

The name of the trend in the format <[LibName.]TrnName>. If you do not specify the library name, a trend from the Global library is assumed.

To get information on a Version 1.xx trend, specify the trend number (0 to 255). For example, if you specify trend 1, Plant SCADA obtains information from the trend Global.Trn001.

*nType*:

Type of trend info:

0 - Type of trend:

- 0 = line
- 1 = bar

1 - Number of samples in trend

2 - Height of trend (in pixels)

3 - Width of trend sample (in pixels)

4 - Number of trend pens

11 - Color of pen 1. If the pen uses flashing color, the initial color used. (Use type 19 to determine the secondary flashing color for pen 1.)

12 - Color of pen 2. If the pen uses flashing color, the initial color used. (Use type 20 to determine the secondary flashing color for pen 2.)

13 - Color of pen 3. If the pen uses flashing color, the initial color used. (Use type 21 to determine the secondary flashing color for pen 3.)

14 - Color of pen 4. If the pen uses flashing color, the initial color used. (Use type 22 to determine the secondary flashing color for pen 4.)

15 - Color of pen 5. If the pen uses flashing color, the initial color used. (Use type 23 to determine the secondary flashing color for pen 5.)

16 - Color of pen 6. If the pen uses flashing color, the initial color used. (Use type 24 to determine the secondary flashing color for pen 6.)

17 - Color of pen 7. If the pen uses flashing color, the initial color used. (Use type 25 to determine the secondary flashing color for pen 7.)

18 - Color of pen 8. If the pen uses flashing color, the initial color used. (Use type 26 to determine the secondary flashing color for pen 8.)

19 - The secondary color used for pen 1, if flashing color is used.

20 - The secondary color used for pen 2, if flashing color is used.

21 - The secondary color used for pen 3, if flashing color is used.

22 - The secondary color used for pen 4, if flashing color is used.

23 - The secondary color used for pen 5, if flashing color is used.

24 - The secondary color used for pen 6, if flashing color is used.

25 - The secondary color used for pen 7, if flashing color is used.

26 - The secondary color used for pen 8, if flashing color is used.

*nAN:*

The AN where the trend is displayed.

## Return Value

The trend information (as an integer). If Pen Color (*Types 11 - 18*) is requested from a bar trend, the return value is -1.

## Related Functions

[DspTrend](#)

## Example

```
! get the number of samples for the main_loop trend (from the
trends library).
nSamples = DspTrendInfo("Trends.Main_Loop", 1);
! get the number of samples for trend 3 (Plant SCADA
Version 1.xx).
nSamples = DspTrendInfo(3, 1);
```

## See Also

[Display Functions](#)

## DspWebContentGetURL

Gets the current URL for a Web Content control on a graphics page.

## Syntax

**DspWebContentGetURL(INT *nAN*)**

*nAN:*

The AN where the Web Content control is displayed.

## Return Value

The current URL of the specified Web Content control as a string.

The following hardware alarms will be generated if not successful:

- Cicode error message 303 will indicate that the AN is invalid or does not contain a Web Content object.

- Cicode error message 583 will indicate that the URL is longer than 255 characters.

In both cases, an empty string will be returned.

## Related Functions

[DspWebContentSetURL](#)

## See Also

[Display Functions](#)

### DspWebContentSetURL

Sets the URL for a Web Content control on a graphics page.



#### LOSS OF CONTROL

- Do not use the Plant SCADA Web Content control to display an untrusted or potentially unsafe web site, as this may leave your runtime system vulnerable to malicious activity.
- Do not enter a URL in the Web Content control that will allow users to navigate to an untrusted or potentially unsafe web site.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Before you implement the Web Content control as part of your runtime system, you should carefully consider the security implications of the content that will be accessible. If using an internet-based website or application, you will be externally exposing your SCADA system and network.

## Syntax

**DspWebContentSetURL(INT *nAN*, STRING *sURL*)**

*nAN*:

The AN where the Web Content control is displayed.

*sURL*:

A valid URL (maximum 255 characters).

## Return Value

0 (zero) if successful, otherwise an error is returned.

If the AN is invalid or does not contain a Web Content object, Cicode error message 303 will be returned (Invalid AN). This will also trigger a hardware alarm for error code 303.

## Related Functions

[DspWebContentGetURL](#)

## See Also

[Display Functions](#)

## DLL Functions

Following are functions relating to DLLs:

<a href="#">DLLCall</a>	Calls a DLL function.
<a href="#">DLLCallEx</a>	Calls a DLL function, and passes the specified arguments to that function.
<a href="#">DLLClose</a>	Closes a link to a DLL function.
<a href="#">DLLOpen</a>	Opens a link to a DLL function.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## DLLCall

Calls a DLL function, and passes a string of arguments to that function. Plant SCADA converts these arguments (where required) into the type specified in the DLLOpen call. If an argument cannot be converted, it is set to zero (0) or an empty string "".

You need to first open the DLL with the DLLOpen function.

Only one call to the DLLCall function can be made at a time, which means runtime will wait for the called function to return before doing anything else. If the called function takes too long to return, it won't let other tasks execute. Therefore, care needs to be taken so that one call returns before the next is made.

Good programming practice requires that functions which are not expected to complete in a short time are run as separate Windows threads and return a value immediately to Plant SCADA.

## Syntax

**DLLCall(*hFunction*, *sArgs*)**

*hFunction*:

The DLL function handle, returned from DLLOpen.

*sArgs*:

The string of arguments to pass to the DLL function. The argument string contains all the arguments for the function, separated by commas (,). Enclose string arguments in quote marks "", and use the string escape character (^) to put a string delimiter within a string. This syntax is the same as the syntax for the TaskNew() function

## Return Value

The result of the function, as a string.

## Related Functions

[DLLOpen](#), [DLLClose](#)

## Example

See [DLLOpen](#).

## See Also

[DLL Functions](#)

## DLLCallEx

Calls a DLL function, and passes the specified arguments to that function.

You need to first open the DLL with the [DLLOpen](#) function.

Only one call to the DLLCallEx() function can be made at a time, which means runtime will wait for the called function to return before doing anything else. If the called function takes too long to return, it won't let other tasks execute. Therefore, care needs to be taken so that one call returns before the next is made.

Good programming practice requires that functions which are not expected to complete in a short time are run as separate Windows threads and return a value immediately to Plant SCADA.

## Syntax

**DLLCallEx(*hFunction*, *vParameters*)**

*hFunction*:

The DLL function handle, returned from DLLOpen().

*vParameters*:

A variable length parameter list of method arguments. The parameters will be passed to the function in the order that you enter them.

Specifying too few or too many parameters will generate an Invalid Argument hardware error. An Invalid Argument hardware error will also be generated if you specify a parameter to the DLL function with the wrong type.

---

**Note:** This function does not support the passing of global variables into the *vParameters* argument.

## Return Value

The result of the function. If the DLL function returns a string then your Cicode return variable should be of type STRING. All other types will be INT.

## Related Functions

[DLLOpen](#), [DLLClose](#)

## Example

```
/* This function is called when Plant SCADA starts up,
to initialize all the DLLs that are called */
INT hAnsiUpper;
INT hGlobalAlloc;
FUNCTION InitMyDLLs()
    ! Open DLL to AnsiUpper
    hAnsiUpper = DLLOpen("USER.DLL", "AnsiUpper", "CC");
    hGlobalAlloc = DLLOpen("Kernel", "GlobalAlloc", "IIJ");
    END
/* This is the Cicode entry point into the DLL function call. This
function hides the DLL interface from the rest of Plant SCADA. */
STRING
FUNCTION AnsiUpper(STRING sString)
    STRING sResult;
    sResult = DLLCallEx(hAnsiUpper, sString);
    RETURN sResult;
END
/* Allocate memory and return memory handle */
INT
FUNCTION GlobalAlloc(INT Mode, INT Length)
    INT hMem;
    hMem = DLLCallEx(hGlobalAlloc, Mode, Length);
    RETURN hMem;
END
```

## See Also

[DLL Functions](#)

## DLLClose

Closes the link to a DLL function, and frees the memory allocated for that function link. When the link is closed, you cannot call the function. Plant SCADA automatically closes all function links at shutdown.

## Syntax

**DLLClose(*hFunction*)**

*hFunction*:

The DLL function handle, returned from DLLOpen().

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DLLOpen](#), [DLLCall](#)

## Example

See [DLLOpen](#).

## See Also

[DLL Functions](#)

## DLLOpen

Opens a link to a DLL function, by loading the specified DLL library into memory and attaching it to the named function. After you open the function link, you can call the function with the DLLCall function. You pass the function number returned from the DLLOpen function as an argument in the DLLCall function.

Plant SCADA only supports DLL functions that accept arguments via stack-based calling conventions. For example, in Win32 only cdecl and stdcall are supported.

---

**Note:** A DLL compiled for 32 bit cannot be called on an alarm server operating in Extended Memory mode. To allow this to work, you will need to recompile the DLL to target 64 bit.

One accepted method for interfacing with a DLL function is to write a Cicode function file. This file contains the DLLOpen() function to initialize the functions, and one Cicode function for each DLL function, as an interface. In this way, you can hide the DLL interface in this file. Any other Cicode function will call the Cicode interface, and the call to the DLL remains transparent.

Please be aware that DLLs need to be on the path. The file extension is not required.

---

**Note:** You need to specify the arguments to the function correctly. Plant SCADA has no way of checking the number and type of arguments to the function. If you specify the number of arguments incorrectly, your computer may display unexpected behavior. You should test your interface thoroughly before using it on a live system.

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

Ensure that you specify the arguments to the DLLOpen() function correctly according to the following list.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Syntax

**DLLOpen(*sLib*, *sName*, *sArgs*)**

*sLib*:

The DLL library name.

*sName*:

The function name. An underscore (\_) is required in the function name for a 'C' function, but not for a Pascal function. When you call a DLL from a Cicode function, *sName* needs to be the same as the name defined in the .DEF file used to link the DLL. The file extension is not required.

*sArgs*:

The string specifying the function arguments. The first character in the string is the return value of the function.

*A* - Logical.

*B* - IEEE 8 byte floating point number.

*C* - Null terminated string. Maximum string length 255 characters.

*D* - Byte counted string. First byte contains the length of the string, maximum string length 255 characters.

*H* - Unsigned 2 byte integer.

*I* - Signed 2 byte integer.

*J* - Signed 4 byte integer.

## Return Value

The DLL function handle, or -1 if the library or function could not be found or loaded.

## Related Functions

[DLLCall](#), [DLLClose](#)

## Example

```
/* This function is called when Plant SCADA starts up,
to initialize the DLLs that are called */
INT hAnsiUpper;
INT hGlobalAlloc;
FUNCTION InitMyDLLs()
    ! Open DLL to AnsiUpper
    hAnsiUpper = DLLOpen("USER.DLL", "AnsiUpper", "CC");
    hGlobalAlloc = DLLOpen("Kernel", "GlobalAlloc", "IIJ");
END
/* This is the Cicode entry point into the DLL function call. This
function hides the DLL interface from the rest of Plant SCADA. */
STRING
FUNCTION AnsiUpper(STRING sString)
    STRING sResult;
    sResult = DLLCall(hAnsiUpper, "^" + sString + "^");
    RETURN sResult;
END
```

```
/* Allocate memory and return memory handle */
INT
FUNCTION GlobalAlloc(INT Mode, INT Length)
    STRING sResult;
    INT hMem;
    sResult = DLLCall(hGlobalAlloc, Mode : ##### + "," + Length : #####);
    hMem = StrToInt(sResult);
    RETURN hMem;
END
```

## See Also

[DLL Functions](#)

## Equipment Database Functions

The following are functions relate to the equipment database.

<a href="#">EquipBrowseClose</a>	Closes an equipment database browse session.
<a href="#">EquipBrowseFirst</a>	Gets the first equipment database entry in the browse session.
<a href="#">EquipBrowseGetField</a>	Gets the field indicated by the cursor position in the browse session.
<a href="#">EquipBrowseNext</a>	Gets the next equipment database entry in the browse session.
<a href="#">EquipBrowseNumRecords</a>	Returns the number of records in the current browse session.
<a href="#">EquipBrowseOpen</a>	Opens an equipment database browse session.
<a href="#">EquipBrowsePrev</a>	Gets the previous equipment database entry in the browse session.
<a href="#">EquipCheckUpdate</a>	Checks if the equipment database file has been updated, and provides the facility to reload it.
<a href="#">EquipGetParameter</a>	Reads a runtime parameter of an equipment database record from the EQPARAM.RDB database file.
<a href="#">EquipGetProperty</a>	Reads a property of an equipment database record from the EQUIP.DBF file.
<a href="#">EquipRefBrowseClose</a>	Closes an equipment database browse session.
<a href="#">EquipRefBrowseFirst</a>	Gets the first equipment database entry in the browse session.

<a href="#">EquipRefBrowseGetField</a>	Gets the field indicated by the cursor position in the browse session.
<a href="#">EquipRefBrowseNext</a>	Gets the next equipment database entry in the browse session.
<a href="#">EquipRefBrowseNumRecords</a>	Returns the number of records in the current browse session.
<a href="#">EquipRefBrowseOpen</a>	Opens an equipment database browse session.
<a href="#">EquipRefBrowsePrev</a>	Gets the previous equipment database entry in the browse session.
<a href="#">EquipSetProperty</a>	Sets the property of an item of equipment.
<a href="#">EquipStateBrowseClose</a>	Terminates a browsing session and cleans up the resources used by the session.
<a href="#">EquipStateBrowseFirst</a>	Places the data browse cursor at the first record.
<a href="#">EquipStateBrowseGetField</a>	Returns the value of the particular field in a record to which the data browse cursor is currently referencing.
<a href="#">EquipStateBrowseNext</a>	Places the data browse cursor at the next available record.
<a href="#">EquipStateBrowseNumRecords</a>	Returns the number of records that match the current filter criteria.
<a href="#">EquipStateBrowseOpen</a>	Initiates a new session for browsing the equipment states configured.
<a href="#">EquipStateBrowsePrev</a>	Places the data browse cursor at the previous record.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## EquipBrowseClose

The EquipBrowseClose function terminates an active data browse session and cleans up resources associated with the session.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

**INT EquipBrowseClose(LONG Session)**

### Session

The handle to a browse session previously returned by a EquipBrowseOpen call.

## Return Value

0 (zero) if the equipment database browse session exists, otherwise an error code is returned.

## Related Functions

[EquipBrowseFirst](#), [EquipBrowseGetField](#), [EquipBrowseNext](#), [EquipBrowseOpen](#), [EquipBrowsePrev](#),  
[EquipCheckUpdate](#), [EquipSetProperty](#)

## See Also

[Equipment Database Functions](#)

### EquipBrowseFirst

The EquipBrowseFirst function places the data browse cursor at the first record.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **EquipBrowseFirst**(LONG Session)

*Session:*

The handle to a browse session previously returned by a EquipBrowseOpen call.

## Return Value

0 (zero) if the equipment database browse session exists, otherwise an error code is returned.

## Related Functions

[EquipBrowseClose](#), [EquipBrowseGetField](#), [EquipBrowseNext](#), [EquipBrowseNumRecords](#), [EquipBrowseOpen](#),  
[EquipBrowsePrev](#), [EquipCheckUpdate](#), [EquipGetProperty](#)

## See Also

[Equipment Database Functions](#)

### EquipBrowseGetField

The EquipBrowseGetField function retrieves the value of the specified field from the record the data browse cursor is currently referencing.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

STRING **EquipBrowseGetField**(LONG *Session*, STRING *FieldName*)

*Session*:

The handle to a browse session previously returned by a EquipBrowseOpen call.

*FieldName*:

The name of the field that references the value to be returned. Supported fields are:

Name, Cluster, Type, Area, Location, IODevice, Page, Help, Comment, Composite, Parent, Custom1, Custom2, Custom3, Custom4, Custom5, Custom6, Custom7, Custom8.

See [Browse Function Field Reference](#) for information about fields.

## Return Value

The value of the specified field as a string. An empty string may or may not be an indication that an error has been detected. The last error should be checked in this instance to determine if an error has actually occurred.

## Related Functions

[EquipBrowseClose](#), [EquipBrowseFirst](#), [EquipBrowseNext](#), [EquipBrowseNumRecords](#), [EquipBrowseOpen](#),  
[EquipBrowsePrev](#), [EquipCheckUpdate](#), [EquipGetProperty](#)

## Example

```
STRING fieldValue = "";
STRING fieldName = "TYPE";
INT errorCode = 0;
...
fieldValue = EquipBrowseGetField(iSession, sFieldName);
IF fieldValue <> "" THE
    // Successful case
ELSE
    // Function returned an error
END
...
```

## See Also

[Equipment Database Functions](#)

## EquipBrowseNext

The EquipBrowseNext function moves the data browse cursor forward one record. If you call this function after you have reached the end of the records, error 412 is returned (Databrowse session EOF).

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT EquipBrowseNext(LONG Session)**

*Session*

The handle to a browse session previously returned by a EquipBrowseOpen call.

## Return Value

0 (zero) if the equipment database browse session exists, otherwise an error is returned.

## Related Functions

[EquipBrowseClose](#), [EquipBrowseFirst](#), [EquipBrowseGetField](#), [EquipBrowseNumRecords](#), [EquipBrowseOpen](#),  
[EquipBrowsePrev](#), [EquipCheckUpdate](#), [EquipGetProperty](#)

## See Also

[Equipment Database Functions](#)

## EquipBrowseNumRecords

The EquipBrowseNumRecords function returns the number of records that match the filter criteria.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

**LONG EquipBrowseNumRecords(LONG Session)**

*Session:*

The handle to a browse session previously returned by a EquipBrowseOpen call.

## Return Value

The number of records that have matched the filter criteria. A value of 0 denotes that no records have matched. A value of -1 denotes that the browse session is unable to provide a fixed number. This may be the case if the data being browsed changed during the browse session.

## Related Functions

[EquipBrowseClose](#), [EquipBrowseFirst](#), [EquipBrowseGetField](#), [EquipBrowseNext](#), [EquipBrowseOpen](#),  
[EquipBrowsePrev](#), [EquipCheckUpdate](#), [EquipGetProperty](#)

## Example

```
INT numRecords = 0;
```

```
...
numRecords = EquipBrowseNumRecords(iSession);
IF numRecords <> 0 THEN
    // Have records
ELSE
    // No records
END
...
```

## See Also

[Equipment Database Functions](#)

### EquipBrowseOpen

The EquipBrowseOpen function initiates a new browse session and returns a handle to the new session that can be used in subsequent data browse function calls.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

LONG **EquipBrowseOpen**(STRING *Filter*, STRING *Fields* [, STRING *Clusters*] )

### *Filter:*

A filter expression specifying the records to return during the browse. An empty string indicates that all records will be returned. Where a fieldname is not specified in the filter, it is assumed to be equipment name. For example, the filter "AAA" is equivalent to "name=AAA".

The following regular expressions are supported: \*expr, expr\*, and \*expr\*. To specify an exclusion filtering condition, use the NOT keyword after the = operator.

### *Fields:*

Specifies via a comma delimited string the columns to be returned during the browse. An empty string indicates that the server will return all available columns. Supported fields are:

ALIAS, AREA, CLUSTER, COMMENT, CONTENT, CUSTOM1...8, DEFSTATE, DEVSCHEDE, HIDDEN, LOCATION, NAME, PAGE, PARENT, SCHEDID, SCHEDULED, TYPE.

See [Browse Function Field Reference](#) for information about fields.

### *Clusters:*

An optional parameter that specifies via a comma delimited string the subset of the clusters to browse. An empty string indicates that all connected clusters will be browsed.

## Return Value

Returns an integer handle to the browse session. Returns -1 when an error is detected.

The returned entries will be ordered alphabetically by name.

## Related Functions

[EquipBrowseClose](#), [EquipBrowseFirst](#), [EquipBrowseGetField](#), [EquipBrowseNext](#), [EquipBrowseNumRecords](#),  
[EquipBrowseOpen](#), [EquipCheckUpdate](#), [EquipGetProperty](#)

## Example

```
INT iSession;
...
iSession = EquipBrowseOpen("NAME=ABC*", "NAME,AREA",
"ClusterA,ClusterB");
IF iSession <> -1 THEN
    // Successful case
ELSE
    // Function returned an error
END
...
```

## See Also

[Equipment Database Functions](#)

## EquipBrowsePrev

The EquipBrowsePrev function moves the data browse cursor back one record. If you call this function after you have reached the beginning of the records, error 412 is returned (Databrowse session EOF).

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT EquipBrowsePrev(LONG Session)**

*Session:*

The handle to a browse session previously returned by a EquipBrowseOpen call.

## Return Value

0 (zero) if the equipment database browse session exists, otherwise an error is returned.

## Related Functions

[EquipBrowseClose](#), [EquipBrowseFirst](#), [EquipBrowseGetField](#), [EquipBrowseNext](#), [EquipBrowseNumRecords](#),  
[EquipBrowseOpen](#), [EquipCheckUpdate](#), [EquipGetProperty](#)

## See Also

[Equipment Database Functions](#)

## EquipCheckUpdate

The runtime environment will automatically check if the equipment database has been updated before each page open and before running an equipment browse open function ([EquipBrowseOpen](#)). It does this by checking the version of the database file. You may also update the database periodically in a background task to reload the database even if the page is not changed.

The EquipCheckUpdate function checks if the equipment database file has been updated, and provides the facility to reload it. The reload can only be performed if there are no open browse sessions.

## Syntax

```
INT EquipCheckUpdate(INT Reload[, STRING sCluster])
```

*Reload*:

1 (TRUE) if you want to reload the database.

0 (FALSE) checks the status of the database without reloading it.

*Cluster*:

The cluster name to check and/or reload the equipment database for. All clusters will be checked if omitted or an empty string is set.

## Return Value

0 (FALSE) if the equipment database has not changed, or 1 (TRUE) if it has been changed, otherwise an error is returned.

## Related Functions

[EquipBrowseClose](#), [EquipBrowseGetField](#), [EquipBrowseFirst](#), [EquipBrowseNext](#), [EquipBrowseNumRecords](#),  
[EquipBrowseOpen](#), [EquipBrowsePrev](#), [EquipGetProperty](#)

## See Also

[Equipment Database Functions](#)

## EquipGetParameter

Reads a runtime parameter of an equipment database record from the EQPARAM.RDB database file.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

```
EquipGetParameter(ClusterName, EquipmentName, ParameterName)
```

*ClusterName*

The name of the cluster (optional for a single cluster system)

*EquipmentName*

The name of the equipment from which to get information. The name of the equipment can be prefixed by the name of the cluster that is "ClusterName.Equipment".

*ParameterName*

The name of the parameter for which to retrieve a value.

**Return Value**

The value of the specified parameter as a string. If the parameter is a tag reference, the value of the tag shall be returned.

---

**Note:** The value field of the Equipment Runtime Parameters table can work like a tag reference when the Is Tag column is set to true or default. It can contain Variable tag, or equipment and item name reference of a variable tag (using equipment.item notation) that will result in the tag value being returned by this function.

---

An empty string may or may not be an indication that an error has been detected. The last error should be checked in this instance to determine if an error has actually occurred.

**Related Functions**

[EquipBrowseFirst](#), [EquipBrowseGetField](#), [EquipBrowseNext](#), [EquipBrowseNumRecords](#), [EquipBrowseOpen](#),  
[EquipBrowsePrev](#), [EquipCheckUpdate](#), [EquipGetProperty](#)

**See Also**

[Equipment Database Functions](#)

**EquipGetProperty**

This function reads a property of an equipment database record from the EQUIP.RDB database file.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

**Syntax**

STRING **EquipGetProperty**(STRING *Name*, STRING *Field*, INT *Mode*, STRING *Cluster*)

*Name:*

The name of the equipment from which to get information. The name of the equipment can be prefixed by the name of the cluster that is "ClusterName.Equipment".

*Field:*

The field to read. Supported fields are:

Name, Cluster, Type, Area, Location, IODevice, Page, Parent, Composite, Help, Comment, Alias, Content, Hidden, Custom1, Custom2, Custom3, Custom4, Custom5, Custom6, Custom7, Custom8, Defstate, Scheduled

- *Name* — The name of the equipment (254 characters).
- *Cluster* — The cluster to which the equipment belongs (16 characters).
- *Type* — The equipment-specific type of device (254 characters).

- *Area* — Area number (integer) (16 characters).
- *Location* — Equipment specific field (254 characters).
- *IODevice* — I/O Device name(s) (254 characters).
- *Page* — Page name (254 characters).
- *Parent* — The name of the parent equipment derived from the name of the equipment (maximum 254 characters).
- *Help* — Help context (254 characters).
- *Comment* — User comment (254 characters).
- *Custom1..8* — User definable fields (254 characters each).
- *Composite* — The equipment specific composite name (maximum 254 characters).
- *Defstate* — The default state.
- *Scheduled* — Specifies if the equipment participates in a schedule.
- *Alias* — Meaningful name of equipment (254 characters).
- *Content* — Workspace content associated with equipment instance.
- *Hidden* — Hide from equipment tree at runtime.

Runtime fields:

- *MODE* — the current mode of the equipment. 0 = automatic; 1= Manual Inherited; 2= Manual.
- *DRMODE* — the current DR mode level for the equipment. The value will be a positive integer to represent the current DR level or zero if inactive.
- *STATE* — the current state of the equipment. Although you can only set state for equipment in Override mode, it is still possible to get state for equipment in normal and inherited override mode.

*Mode:*

0 Block — values are retrieved from the report server, the value will also be stored in local memory cache.

1 Cache — values are first retrieved from local memory cache, if the property is missing from the memory cache, a request will be sent to the report server and the result will then be cached. If the property is missing from the memory cache or if a previous request is in progress, an empty string will be returned.

2 Local — values are retrieved from the local RDB. Error 348 (Property does not exist) will be returned if the property requested is not available. Requesting runtime fields will also return this error.

3 CacheThenLocal — values are first retrieved from local memory cache, if the property is missing from the memory cache, a request will be sent to the report server and the result will then be cached. If the property is missing from the memory cache or if a previous request is in progress, the local RDB will be accessed to retrieve the property value.

*Cluster*

The name of the cluster (optional for a single cluster system).

---

**Note:** Mode 1 and 3 will access the report server at least once, there will be a once off performance penalty on the report server, operating in this mode will always guarantee equipment properties being the latest. Mode 2 will be the fastest but cannot guarantee equipment will be up-to-date (see [Client Side Online Changes](#)). Mode 0 will be the slowest as each request is sent across the wire and the function will block until the report server responds.

---

## Return Value

The value of the specified field as a string. An empty string may or may not be an indication that an error has been detected. The last error should be checked in this instance to determine if an error has actually occurred.

## Related Functions

[EquipBrowseClose](#), [EquipBrowseFirst](#), [EquipBrowseGetField](#), [EquipBrowseNext](#), [EquipBrowseNumRecords](#),  
[EquipBrowseOpen](#), [EquipBrowsePrev](#), [EquipCheckUpdate](#), [EquipSetProperty](#)

## See Also

[Equipment Database Functions](#)

### EquipRefBrowseClose

The EquipRefBrowseClose function terminates an active data browse session and cleans up resources associated with the session.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

INT **EquipRefBrowseClose**(LONG *Session*)

*Session*:

The handle to a browse session previously returned by a EquipBrowseOpen call.

## Return Value

0 (zero) if the equipment database browse session exists, otherwise an error code is returned.

## Related Functions

[EquipRefBrowseFirst](#), [EquipRefBrowseGetField](#), [EquipRefBrowseNext](#), [EquipRefBrowseNumRecords](#),  
[EquipRefBrowseOpen](#), [EquipRefBrowsePrev](#), [EquipCheckUpdate](#), [EquipGetProperty](#)

## See Also

[Equipment Database Functions](#)

### EquipRefBrowseFirst

The EquipRefBrowseFirst function places the reference browse cursor at the first record.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

```
INT EquipRefBrowseFirst(LONG Session)
```

*Session:*

The handle to a browse session previously returned by a EquipRefBrowseOpen call.

## Return Value

0 (zero) if the equipment database browse session exists, otherwise an error code is returned.

## Related Functions

[EquipRefBrowseClose](#), [EquipRefBrowseGetField](#), [EquipRefBrowseNext](#), [EquipRefBrowseNumRecords](#),  
[EquipRefBrowseOpen](#), [EquipRefBrowsePrev](#), [EquipCheckUpdate](#), [EquipGetProperty](#)

## See Also

[Equipment Database Functions](#)

## EquipRefBrowseGetField

The EquipRefBrowseGetField function retrieves the value of the specified field from the record the data browse cursor is currently referencing.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

```
STRING EquipRefBrowseGetField(LONG Session, STRING FieldName)
```

*Session:*

The handle to a browse session previously returned by a EquipRefBrowseOpen call.

*FieldName:*

The name of the field that references the value to be returned. Supported fields are:

ASSOC, CLUSTER, COMMENT, CUSTOM1...8, EQUIP, ORDER, REFCAT, REFCLUST, REFEQUIP, REFITEM.

See [Browse Function Field Reference](#) for information about fields.

## Return Value

The value of the specified field as a string. An empty string may or may not be an indication that an error has been detected. The last error should be checked in this instance to determine if an error has actually occurred.

## Related Functions

[EquipRefBrowseClose](#), [EquipRefBrowseFirst](#), [EquipRefBrowseNext](#), [EquipRefBrowseNumRecords](#),

[EquipRefBrowseOpen](#), [EquipRefBrowsePrev](#), [EquipGetProperty](#)

## Example

```
STRING fieldValue = "";
STRING fieldName = "TYPE";
INT errorCode = 0;
...
fieldValue = EquipRefBrowseGetField(iSession, sFieldName);
IF fieldValue <> "" THEN
    // Successful case
ELSE
    // Function returned an error
END
...
```

## See Also

[Equipment Database Functions](#)

## EquipRefBrowseNext

The EquipRefBrowseNext function moves the data browse cursor forward one record. If you call this function after you have reached the end of the records, error 412 is returned (Databrowse session EOF).

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **EquipRefBrowseNext**(LONG *Session*)

*Session*

The handle to a browse session previously returned by a EquipRefBrowseNext call.

## Return Value

0 (zero) if the equipment reference database browse session exists, otherwise an error is returned.

## Related Functions

[EquipRefBrowseClose](#), [EquipRefBrowseFirst](#), [EquipRefBrowseGetField](#), [EquipRefBrowseNumRecords](#),  
[EquipRefBrowseOpen](#), [EquipRefBrowsePrev](#), [EquipCheckUpdate](#), [EquipGetProperty](#)

## See Also

[Equipment Database Functions](#)

## EquipRefBrowseNumRecords

The EquipRefBrowseNumRecords function returns the number of records that match the filter criteria.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

```
LONG EquipRefBrowseNumRecords(LONG Session)
```

*Session:*

The handle to a browse session previously returned by a EquipRefBrowseOpen call.

## Return Value

The number of records that have matched the filter criteria. A value of 0 denotes that no records have matched. A value of -1 denotes that the browse session is unable to provide a fixed number. This may be the case if the data being browsed changed during the browse session.

## Related Functions

[EquipRefBrowseClose](#), [EquipRefBrowseFirst](#), [EquipRefBrowseNext](#), [EquipRefBrowseOpen](#), [EquipRefBrowsePrev](#),  
[EquipCheckUpdate](#), [EquipGetProperty](#)

## Example

```
INT numRecords = 0;  
...  
numRecords = EquipRefBrowseNumRecords(iSession);  
IF numRecords <> 0 THEN  
    // Have records  
ELSE  
    // No records  
END  
...
```

## See Also

[Equipment Database Functions](#)

## EquipRefBrowseOpen

The EquipRefBrowseOpen function initiates a new browse session and returns a handle to the new session that can be used in subsequent equipment reference browse function calls.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

LONG EquipRefBrowseOpen(STRING *Filter*, STRING *Fields* [, STRING *Sort*] )

*Filter:*

A filter expression specifying the records to return during the browse. An empty string indicates that all records will be returned. Where a fieldname is not specified in the filter, it is assumed to be equipment name. For example, the filter "AAA" is equivalent to "name=AAA".

All string fields can be filtered based on regular expressions. Using an operator other than = will cause strings to not match the filter criteria. The following regular expressions are supported \*expr, expr\*, and \*expr\*.

*Fields:*

Specifies via a comma delimited string the columns to be returned during the browse. An empty string indicates that the server will return all available columns. Supported fields are:

ASSOC, CLUSTER, COMMENT, CUSTOM1...8, EQUIP, ORDER, REFCAT, REFCLUST, REFEQUIP, REFITEM.

See [Browse Function Field Reference](#) for information about fields.

*Sort:*

Comma delimited list of record fields to be returned in order of sorting preferences. For example(GROUP:D, ORDER:A)

## Return Value

Returns an integer handle to the browse session. Returns -1 when an error is detected.

The returned entries will be ordered alphabetically by name.

## Related Functions

[EquipRefBrowseClose](#), [EquipRefBrowseFirst](#), [EquipRefBrowseGetField](#), [EquipRefBrowseNext](#),  
[EquipRefBrowseNumRecords](#), [EquipRefBrowsePrev](#), [EquipCheckUpdate](#), [EquipGetProperty](#)

## Example

```
INT iSession;
...
iSession = EquipRefBrowseOpen("NAME=ABC*", "NAME,AREA","GROUP");
IF iSession <> -1 THEN
    // Successful case
ELSE
    // Function returned an error
END
...
```

## See Also

[Equipment Database Functions](#)

## EquipRefBrowsePrev

The EquipRefBrowsePrev function moves the data browse cursor back one record. If you call this function after you have reached the beginning of the records, error 412 is returned (Databrowse session EOF).

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT EquipRefBrowsePrev(LONG Session)**

*Session:*

The handle to a browse session previously returned by a EquipRefBrowseOpen call.

## Return Value

0 (zero) if the equipment database browse session exists, otherwise an error is returned.

## Related Functions

[EquipRefBrowseClose](#), [EquipRefBrowseFirst](#), [EquipRefBrowseGetField](#), [EquipRefBrowseNext](#),  
[EquipBrowseNumRecords](#), [EquipRefBrowseOpen](#), [EquipCheckUpdate](#), [EquipGetProperty](#)

## See Also

[Equipment Database Functions](#)

## EquipSetProperty

The EquipSetProperty function sets the property of an item of equipment.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT EquipSetProperty(STRING Name, STRING Field, STRING Value, STRING Cluster)**

*Name:*

The name of the equipment.

*Field:*

Only properties coming from the scheduler engine can be set:

- MODE: (the current mode of the equipment)
- STATE: (the current state of the equipment. The state only can be set for equipment in "Override" mode)
- DRMODE: (the DR mode of the equipment)

*Value:*

The value to be set:

- MODE: 0, automatic; 2 Manual
- STATE: Any state configured in the system for this equipment, also the mode set the mode to Manual otherwise an error will be sent.
- DRMODE: the current DR mode level for the equipment. The value will be a positive integer to represent the current DR level or zero if inactive.

*Cluster:*

The name of the cluster (optional)

## Return Value

Returns 0 if successful otherwise it returns an error.

## Related Functions

[EquipGetProperty](#), [EquipBrowseOpen](#), [EquipBrowseClose](#), [EquipBrowseFirst](#), [EquipBrowseGetField](#),  
[EquipBrowseNext](#), [EquipBrowseNumRecords](#), [EquipBrowsePrev](#)

## See Also

[Equipment Database Functions](#)

## EquipStateBrowseClose

The EquipStateBrowseClose function terminates a browsing session and cleans up the resources used by the session. This function uses iSession as the argument which is previously returned by the EquipStateBrowseOpen function.

## Syntax

INT **EquipStateBrowseClose**(LONG Session )

*Session:*

The handle to a browse session previously returned by a EquipStateBrowseOpen call

## Return Value

Returns 0 if the equipment state database browse session has been closed successfully, otherwise an error is returned.

## Related Functions

[EquipStateBrowseOpen](#), [EquipStateBrowseFirst](#), [EquipStateBrowseGetField](#), [EquipStateBrowseNext](#),  
[EquipStateBrowseNumRecords](#), [EquipStateBrowsePrev](#), [EquipGetProperty](#)

## Example

```
ErrSet(1);
    INT error;
    fieldValue = EquipStateBrowseClose(iSession);
    error = IsError();

    IF(error <> 0) THEN
        //return error
    ELSE
        //successful
    ErrSet(0)
    end
```

## See Also

[Equipment Database Functions](#)

### EquipStateBrowseFirst

The EquipStateBrowseFirst function places the data browse cursor at the first record.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **EquipBrowseFirst**(LONG Session)

*Session:*

The handle to a browse session previously returned by a EquipStateBrowseOpen call.

## Return Value

0 (zero) if the equipment state database browse session exists, otherwise an error is returned.

## Related Functions

[EquipStateBrowseOpen](#), [EquipStateBrowseFirst](#), [EquipStateBrowseGetField](#), [EquipStateBrowseNext](#),  
[EquipStateBrowseNumRecords](#), [EquipStateBrowsePrev](#), [EquipGetProperty](#)

## See Also

[Equipment Database Functions](#)

### EquipStateBrowseGetField

The EquipStateBrowseGetField function returns the value of the particular field in a record to which the data browse cursor is currently referencing. This function uses the field name whose value needs to be returned.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

STRING **EquipStateBrowseGetField**(LONG *Session*, STRING *Field*)

*Session*:

The handle to a browse session previously returned by a EquipStateBrowseOpen call.

*Field*:

The name of the field that references the value to be returned. Available fields are:

- Name – the state name
- Equipment – equipment name
- Delay – entry command delay
- Period – repeat command period
- Priority – the state priority
- Description – state description
- DRmode – demand-response mode of this state

## Return Value

The value of the specified field as a string. An empty string may or may not be an indication that an error has been detected. The last error should be checked in this instance to determine if an error has actually occurred.

## Related Functions

[EquipStateBrowseOpen](#), [EquipStateBrowseFirst](#), [EquipStateBrowseNext](#), [EquipStateBrowseNumRecords](#),  
[EquipStateBrowsePrev](#), [EquipGetProperty](#)

## Example

```
ErrSet(1);
    INT error;
    fieldValue = EquipStateBrowseGetField(iSession, sFieldName);
    error = IsError();

    IF(error <> 0) THEN
        //return error
    ELSE
        //successful
    ErrSet(0)
    end
```

## See Also

[Equipment Database Functions](#)

## EquipStateBrowseNext

The EquipStateBrowseNext function places the data browse cursor at the next available record. This function will return an error if called when the data cursor is at end of the records.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT EquipStateBrowseNext(LONG Session)**

*Session*

The handle to a browse session previously returned by a EquipStateBrowseOpen call.

## Return Value

0 (zero) if the equipment state database browse session exists, otherwise an error is returned.

## Related Functions

[EquipStateBrowseOpen](#), [EquipStateBrowseFirst](#), [EquipStateBrowseGetField](#), [EquipStateBrowseNumRecords](#),  
[EquipStateBrowsePrev](#), [EquipGetProperty](#)

## See Also

[Equipment Database Functions](#)

## EquipStateBrowseNumRecords

The EquipStateBrowseNumRecords function returns the number of records that match the current filter criteria.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

**LONG EquipStateBrowseNumRecords(LONG Session)**

*Session:*

The handle to a browse session previously returned by a EquipStateBrowseOpen call.

## Return Value

Returns the number of records which matches the current filter criteria. 0 means no records were found.

## Related Functions

[EquipStateBrowseOpen](#), [EquipStateBrowseFirst](#), [EquipStateBrowseGetField](#), [EquipStateBrowseNext](#),  
[EquipStateBrowsePrev](#)

## See Also

[Equipment Database Functions](#)

### EquipStateBrowseOpen

The EquipStateBrowseOpen function initiates a new session for browsing the equipment states configured. It returns a handle for the browsing session which can be used for further browsing operations.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

LONG **EquipStateBrowseOpen**(STRING *Filter*, STRING *Fields* [, STRING *Clusters*] )

### *Filter*:

A filter expression specifying the records to return during the browse. An empty string indicates that every record will be returned. Where a field name is not specified in the filter, it is assumed to be tagname. For example, the filter "AAA" is equivalent to "name=AAA".

### *Fields*:

Specifies via a comma delimited string the columns to be returned during the browse. An empty string indicates that the server will return every available column. Available fields are:

CLUSTER, DELAY, DRMODE, EQUIPMENT, NAME, PERIOD, PRESTATE, PRIORITY.

See [Browse Function Field Reference](#) for information about fields.

### *Clusters*:

An optional parameter that specifies via a comma delimited string the subset of the clusters to browse. An empty string indicates that every connected cluster will be browsed.

## Return Value

Returns an integer handle to the browse session. Returns -1 when an error is detected.

## Related Functions

[EquipStateBrowseClose](#), [EquipStateBrowseFirst](#), [EquipStateBrowseGetField](#), [EquipStateBrowseNext](#),  
[EquipStateBrowseNumRecords](#), [EquipStateBrowsePrev](#), [EquipGetProperty](#)

## See Also

[Equipment Database Functions](#)

### EquipStateBrowsePrev

The EquipStateBrowsePrev function places the data browse cursor at the previous record. This function will return an error if called when the data cursor is at beginning of the records.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT EquipStateBrowsePrev(LONG Session)**

*Session:*

The handle to a browse session previously returned by a EquipStateBrowseOpen call.

## Return Value

0 (zero) if the equipment state browse session exists, otherwise an error is returned.

## Related Functions

[EquipStateBrowseClose](#), [EquipStateBrowseFirst](#), [EquipStateBrowseGetField](#), [EquipStateBrowseNext](#),  
[EquipStateBrowseNumRecords](#), [EquipStateBrowseOpen](#), [EquipGetProperty](#)

## See Also

[Equipment Database Functions](#)

## Error Functions

The following functions relate to errors:

<a href="#">ErrCom</a>	Gets the communication status for the current Cicode task.
<a href="#">ErrDrv</a>	Gets a protocol-specific error message.
<a href="#">ErrGetHw</a>	Gets a hardware error code.
<a href="#">ErrHelp</a>	This function is obsolete.
<a href="#">ErrInfo</a>	Gets error information.
<a href="#">ErrLog</a>	Logs a system error.
<a href="#">ErrMsg</a>	Gets the error message associated with a hardware error.
<a href="#">ErrSet</a>	Sets the error mode.
<a href="#">ErrSetHw</a>	Sets a hardware error.
<a href="#">ErrSetLevel</a>	Sets the error level.
<a href="#">ErrTrap</a>	Generates an error trap.

IsError	Checks for an error.
---------	----------------------

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## ErrCom

Gets the communication status for the current Cicode task. You can call this function in reports, Cicode that is associated with an object, and in any Cicode task.

## Syntax

`ErrCom()`

## Return Value

0 (zero) if all I/O device data associated with the task is valid, otherwise an error is returned.

## Related Functions

[CodeSetMode](#)

## Example

```
IF ErrCom()<>0 THEN
    Prompt("I/O device data is bad");
END
```

In a report format:

```
{CICODE}
IF ErrCom()<>0 THEN
    PrintLn("This Report contains bad data");
END
{END}
```

## See Also

[Error Functions](#)

## ErrDrv

Gets a protocol-specific error message and native error code.

## Syntax

**ErrDrv(*sProtocol*, *sField*, *nError*)**

*sProtocol*:

The Plant SCADA protocol.

*sField*:

The field in the PROTERR.DBF database:

- PROTOCOL
- MASK
- ERROR
- MESSAGE
- REFERENCE
- ACTION
- COMMENT

*nError*:

The protocol specific error code. This field needs to be a variable as it also the place where the returned error code is stored.

Since the first 34 specific error codes are standard for all protocols, Plant SCADA may add 'masking' to make the error code unique. For example, if an I/O device returns errors 1 to 10 (which are already used), the driver may add 0x100000 to its error codes. When this function is called, the mask will be removed before the result is returned to this variable.

## Return Value

The error message (as a string), or an empty string ("") if the error is not found. The error code is returned into the *nError* variable.

## Related Functions

[ErrInfo](#), [ErrHelp](#)

## Example

```
// Get the error message and number associated with error 108
nError = 108;
sError = ErrDrv("TIWAY", "MESSAGE", nError);
```

## See Also

[Error Functions](#)

## ErrGetHw

Gets the current hardware error status for an I/O device.

I/O devices can be grouped into 2 distinct categories: Those that are created by the system engineer, and those that are created by Plant SCADA itself.

I/O devices that are created by the system engineer include any I/O device listed in the Plant SCADA I/O devices database, and any device visible as a record in the I/O Device form in the Project Editor.

I/O devices that are created by Plant SCADA include Generic, LAN, Cicode, Animation, Reports Server, Alarms Server, Trends Server, and I/O Server, and are those specifically not created by the system engineer.

The argument's values you supply in this function are used by Plant SCADA to determine which type of device hardware alarm you want to work with.

---

**Note:** Do not use this function if you have more than 511 I/O devices in your project and the flag [\[Code\]BackwardCompatibleErrHw](#) is set to 1. You may retrieve the hardware error status for the wrong I/O device.

---

## Syntax

**ErrGetHw(Device, DeviceType)**

*Device:*

For I/O devices that are created by the system engineer, select the IODevNo as the argument value.

To determine the **IODevNo** of a physical I/O device in your project, use the I/O device record number from the I/O Device form in the Plant SCADA Studio. When using an **IODevNo**, the **DeviceType** argument needs to be set to 2.

For I/O devices that are created by Plant SCADA itself, select one of the following options as the argument value:

- Generic
- LAN
- Cicode
- Animation
- Reports Server
- Alarms Server
- Trends Server
- I/O Server

*DeviceType:*

Select a value from the following options to indicate the 'Type of Device' used in the **Device** argument:

0 - for I/O devices that are created by Plant SCADA itself (Generic, LAN, Cicode, Animation, etc).

2 - for I/O devices that are created by the system engineer.

The **DeviceType** argument was added to this function in V5.40 and later. Earlier versions did not pass a value for the **DeviceType** argument (as it did not exist). Versions prior to V5.40 identified an I/O device by passing the IODevNo (masked with the value of 8192) to the function as the **Device** argument, in the structure:

IODevNo + 8192

This was for versions of Plant SCADA permitting a maximum limit of 4095 I/O devices.

Versions prior to V5.20 masked the IODevNo with a value of 512. The backward compatibility flag for using this mask needs to be set in the Citect.INI file (see [\[Code\]BackwardCompatibleErrHw](#)).

## Return Value

The detected error.

## Related Functions

[ErrHelp](#), [ErrInfo](#), [ErrMsg](#), [ErrSetHw](#)

## Example

```
Error=ErrGetHw(3,0);
! Sets Error to the current error status for the animation device.
IF Error=0 THEN
    DspText(4,0,"");
ELSE
    DspText(4,0,"Hardware error");
END
```

## See Also

[Error Functions](#)

## ErrInfo

Gets extended error information on the last error that was detected.

## Syntax

**ErrInfo(*nType*)**

*nType*:

The type of error information. If type is 0 (zero), function returns the animation number where the error occurred.

## Return Value

The error information.

## Example

```
! Get the animation number where the last error occurred
AN = ErrInfo(0);
```

## See Also

[Error Functions](#)

### ErrLog

Logs a message to the Plant SCADA system log file.

This function is useful for logging errors in user functions, and for debugging user functions. The Plant SCADA system log file 'SYSLOG.DAT' is created in the local Windows directory of the computer, %PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Logs.

## Syntax

**ErrLog(*Message*)**

*Message*:

The message to log. This field can also contain control (such as /n) and formatting characters.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DebugMsg](#), [DebugMsgSet](#), [CodeTrace](#), [TraceMsg](#), [Halt](#)

## Example

```
FUNCTION MyFunc(INT Arg)
    IF Arg<0 THEN
        ErrLog("Invalid arg in Myfunc");
        Halt();
    END
END
```

## See Also

[Error Functions](#)

### ErrMsg

Gets the error message associated with a detected hardware error.

## Syntax

**ErrMsg(*nError*)**

*nError:*

The hardware error number returned from the IsError() function.

## Return Value

The error message (as a string). A null value is returned if *nError* is not in the range of Cicode errors.

## Related Functions

[IsError](#), [ErrHelp](#), [ErrInfo](#), [ErrTrap](#)

## Example

```
//Get the message of the last hardware error
sMsg = ErrMsg(IsError());
```

## See Also

[Error Functions](#)

## ErrSet

Sets the error-checking mode. When *Mode* is set to 0 and an error occurs that causes a component to stop executing, Plant SCADA halts the execution of the Cicode task that caused the error, and generates a hardware error.

You can perform error checking by setting *Mode* to 1 and using the IsError() function to trap errors. When the type of error is determined, you can control what happens under particular error conditions. Errset(1) can be used for parts of Cicode when task termination is undesirable, working around the underlying issue in the Cicode. After executing ErrSet(1), Plant SCADA will not automatically check for errors, and the user can manually check for errors using the Cicode function [IsError\(\)](#) after a line of code is executed.

---

**Note:** ErrSet(1) is NOT global, and when used it will only affect all functions within the particular thread in which it has been called. It is recommended that you apply ErrSet() around specific small segments of Cicode that are problematic (i.e. single Cicode statements), and not around large sections of code.

The operation of the ErrSet function is unique to each Cicode task. If you enable user error checking for one task, it does not enable error checking for any other tasks.

---

**Note:** This has changed from previous versions of Plant SCADA where this feature used to affect all Cicode tasks.

It is important to note that ErrSet(1) and ErrSet(0) are intended to be used in pairs. When a thread starts, the error check level is 0 (Plant SCADA will halt your code on errors). Each time ErrSet(1) is called, the level is incremented by 1 (with a maximum of 1000). When ErrSet(0) is called the level is decremented by 1 (with a minimum of 0). Plant SCADA will only halt your code on errors if the error check level equals 0. It does not cause a problem to call ErrSet(1) when the level is already 1000, or to call ErrSet(0) when the level is already 0. The code will just continue to run at the current error check level.

If each function calls ErrSet(1) and ErrSet(0) in pairs, there won't be a problem. However, much user Cicode does not do this reliably. For example, if there are multiple RETURN statements in a function it would need to have an ErrSet(0) before each return to make sure the level was reset. It's also common practice to put ErrSet(1) at the

beginning of each function with no corresponding ErrSet(0). To avoid problems, you can use the [ErrSetLevel](#) function instead of ErrSet(). It allows you to set the level to any number (0 - 1000), but could be used to just set it to 0 or 1 to enable or disable Plant SCADA's automatic error checking instead of letting the error check level build up to higher numbers.

## Syntax

**ErrSet(*Mode*)**

*Mode*:

Error-checking mode:

0 - default - Plant SCADA will check for errors.

1 - The user needs to check for errors.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[IsError](#), [ErrSetHw](#), [ErrSetLevel](#)

## Example 1

```
ErrSet(1);
Test=Var/0;
DevWrite(hDevice, "data to write");
nError=IsError();
! Sets Error to 273 (divide by zero).
```

The nError variable will be set to 273 (Divide by 0 Error) if DevWrite is successful, as the internal error variable has not been reset. If DevWrite fails, the error code will correctly be set to a Cicode error such as 269 (bad handle specified).

## Example 2

If you wish to find code that does not correctly use ErrSet(), you can set the citect.ini parameter [Code]EnableErrorLogging=1 This parameter is available in Plant SCADA 7.50 Service Pack 1, Patch 7 and later. When enabled, it will log an error message and the Cicode function stack to the syslog file when the error check level reaches 100 or 1000:

```
ErrSet: Error check level has reached 100; usually this means there are mismatches between
ErrSet(1) and ErrSet(0). Call Stack:
ReadFile(10001 {Good});
hFile = 100 {Good, 2016-10-21T10:55:26}
nRow = 100 {Good, 2016-10-21T10:55:26}
ErrSet: Error check level has reached 1000; usually this means there are mismatches between
ErrSet(1) and ErrSet(0). It won't be increased any more. Call stack:
```

```
ReadFile(10001 {Good});  
hFile = 1000 {Good, 2016-10-21T10:55:26}  
nRow = 1000 {Good, 2016-10-21T10:55:26}
```

## Example 3

Since the ErrSet() state persists until the thread terminates, it is good to avoid leaving it in a different state than it was when your function started. For example, the following code checks if a page exists (which would cause a fatal error if it doesn't exist). This could be called from various other Cicode functions in the project. So, it uses ErrSetLevel() but saves the previous level number that is returned. At the end of the function it sets the previous error level again.

```
// Returns TRUE if the specified page exists, otherwise returns FALSE  
INT FUNCTION PageExists(STRING sPage)  
INT nErrLevelPrev = ErrSetLevel(1);  
IsError(); //Reset the last error number  
PageFileInfo(sPage, 0);  
ErrSetLevel(nErrLevelPrev);  
RETURN IsError() = 0; //Any error indicates PageFileInfo() failed  
END
```

## See Also

[Error Functions](#)

### ErrSetHw

Sets the hardware error status for a hardware device. Call this function to generate a hardware error.

I/O devices can be grouped into two distinct categories: those created by the system engineer, and those created by Plant SCADA itself.

I/O devices that are created by the system engineer, are any I/O device listed in the Plant SCADA I/O devices database, visible as records in the I/O Device form in the Project Editor.

I/O devices that are created by Plant SCADA, including Generic, LAN, Cicode, Animation, Reports Server, Alarms Server, Trends Server, and I/O Server (are those specifically not created by the system engineer).

The arguments values you supply in this function are used by Plant SCADA to determine the type of device hardware alarm you want to work with.

---

**Note:** To use this function, you need to set [\[Code\]BackwardCompatibleErrHw](#) to 1. You cannot use this function if you have more than 511 I/O devices in your project.

## Syntax

**ErrSetHw**(*Device*, *Error*, *DeviceType*)

*Device*:

For I/O devices that are created by the system engineer, select the IODevNo as the argument value.

To determine the **IODevNo** of a physical I/O device in your project, use the I/O device record number from the I/O Device form in the Plant SCADA Studio. When using an **IODevNo**, the *DeviceType* argument needs to be set

to 2.

For I/O devices that are created by Plant SCADA itself, select one of the following options as the argument value:

- 0 - Generic
- 1 - LAN
- 2 - Cicode
- 3 - Animation
- 4 - Reports Server
- 5 - Alarms Server
- 6 - Trends Server
- 7 - I/O Server

*Error:*

The error code.

*DeviceType:*

Select a value from the following options to indicate the 'Type of Device' used in the *Device* argument:

0 - For I/O devices that are created by Plant SCADA itself (Generic, LAN, Cicode, Animation, etc).

2 - For I/O devices that are created by the system engineer.

The *DeviceType* argument was added to this function in V5.40 and later. Earlier versions did not pass a value for the *DeviceType* argument (as it did not exist). Versions prior to V5.40 identified an I/O device by passing the IODevNo (masked with the value of 8192) to the function as the *Device* argument, in the structure:

IODevNo + 8192

This was for versions of Plant SCADA that permitted a maximum limit of 4095 I/O devices.

Versions prior to V5.20 masked the IODevNo with a value of 512. The backward compatibility flag for using this mask needs to be set in the Citect.INI file (see [\[Code\]BackwardCompatibleErrHw](#)).

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[ErrHelp](#) [ErrMsg](#) [ErrSet](#) [ErrSetHw](#)

## Example

```
ErrSetHw(4,273,0);
! Generates a divide by zero error (273) on the report device.
ErrSetHw(3,0,0)
! Resets any error on the animation device.
```

## See Also

[Error Functions](#)

## ErrSetLevel

Sets the nesting error level to enable Plant SCADA error checking inside a nested function (when Plant SCADA error checking has been disabled). This function returns the old error level and sets a new error level.

The nesting error level is incremented every time the ErrSet(1) function is called.

## Syntax

**ErrSetLevel(*Level*)**

*Level*:

The nesting error level.

## Return Value

Returns the old error level and sets a new error level.

## Related Functions

[ErrSet](#)

## Example

```
! ErrorLevel 0 defaults to ErrSet(0) - enables Plant SCADA error-checking.
FUNCTION MainFn()
    ErrSet(1);
    ! ErrorLevel 1 - disables Plant SCADA error checking.
    Fn1();
    ErrSet(0);
    ! Enables Plant SCADA error checking.
END
FUNCTION Fn1()
    ErrSet(1);
    ! ErrorLevel 2 - disables Plant SCADA error checking.
    Test=Var/0;
    Error=IsError();
    ! Sets Error to 273 (divide by zero).
    Fn2();
    ErrSet(0);
    ! Enables Plant SCADA error checking.
END
FUNCTION Fn2()
    OldErrorLevel=ErrSetLevel(0);
    ! Sets nesting error level to 0 to enable Plant SCADA error-checking.
    Test=Var/0;
    ! Cicode halts and a hardware alarm is generated.
    ErrSetLevel(OldErrorLevel)
    ! Resets nesting error level to disable Plant SCADA error-checking.
END
```

## See Also

[Error Functions](#)

### ErrTrap

Generates an error trap. If Plant SCADA error checking is enabled, this function will generate a hardware error and may halt Cicode execution (see *bHalt* argument). If user error checking is enabled, the user function specified in OnEvent(2,Fn) is called.

## Syntax

**ErrTrap(*Error*, *bHalt*)**

*Error*:

The error number to trap.

*bHalt*:

Determines whether the Cicode execution will be halted.

0 - Cicode execution is not halted

1 - Cicode execution is halted

## Return Value

0 (zero) if successful, otherwise an error is returned.

[ErrSetHw](#), [ErrSet](#), [ErrSetLevel](#), [OnEvent](#)

## Example

```
IF Tag=0 THEN
    ErrTrap(273); ! Traps a divide by zero error.
ELSE
    Value=10/Tag;
END
```

## See Also

[Error Functions](#)

### IsError

Gets the current error value. The error value is set when any error is detected, and is reset after this function is called. You can call this function if user error-checking is enabled or disabled.

You should call this function as soon as possible after the operation to be checked, because the error code could be changed by the next error.

## Syntax

```
IsError()
```

## Return Value

The current error value. The current error is reset to 0 after this function is called.

## Related Functions

[ErrSet](#)

## Example

```
! Enable user error-checking.  
ErrSet(1);  
! Invalid ArcSine.  
Ac=ArcSin(20.0);  
! Sets ErrorVariable to 274 (invalid argument passed).  
ErrorVariable=IsError()
```

## See Also

[Error Functions](#)

## Event Functions

Following are functions used to trap and process asynchronous events:

<a href="#">CallEvent</a>	Calls the event function for an event type.
<a href="#">ChainEvent</a>	Calls an event function, by function number.
<a href="#">GetEvent</a>	Gets the function number of the current callback event.
<a href="#">OnEvent</a>	Sets an event callback function, by event type.
<a href="#">SetEvent</a>	Sets an event callback function, by function number.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## CallEvent

Simulates an event, triggering any OnEvent() function that has the same Type argument specified.

Plant SCADA starts running the function immediately, without reading any data from the I/O devices. Any I/O device variable that you use will contain either 0 (zero) or the last value read.

## Syntax

**CallEvent(*Window*, *nType*)**

*Window*:

The number of the window, returned from the [WinNew\(\)](#), [WinNewAt\(\)](#), or [WinNumber\(\)](#) function.

*nType*:

The type of event:

- 0 - The mouse has moved. When the mouse moves the callback function is called. The return value must be 0.
- 1 - A key has been pressed. When the user presses a key, the callback function is called after Plant SCADA checks for hot keys. If the return value is 0, Plant SCADA checks for key sequences. If the return value is not 0, Plant SCADA assumes that you will process the key and does not check the key sequence. It is up to you to remove the key from the key command line. If you are using a right mouse button click as an event, you should read about the ButtonOnlyLeftClick parameter.
- 2 - Error event. This event is called if an error is detected in Cicode, so you can write a single error function to check for your errors. If the return value is 0, Plant SCADA continues to process the error and generates a hardware error - it may then halt the Cicode task. If the return value is not 0, Plant SCADA assumes that you will process the error, and continues the Cicode without generating a hardware error.
- 3 - Page user communication error. A communication error has been detected in the data required for this page. If the return value is 0 (zero), Plant SCADA still animates the page. If the return value is not zero, it does not update the page.
- 4 - Page user open. A new page is being opened. This event allows you to define a single function that is called when all pages are opened. The return value must be 0.
- 5 - Page user close. The current page is being closed. This event allows you to define a single function that is called when all pages are closed. The return value must be 0.
- 6 - Page user always. The page is active. This event allows you to define a single function that is called when all pages are active. The return value must be 0.
- 7 - Page communication error. A communication error has been detected in the data required for this page. Reserved for use by Plant SCADA.
- 8 - Page open. This event is called each time a page is opened. Reserved for use by Plant SCADA.
- 9 - Page close. This event is called each time a page is closed. Reserved for use by Plant SCADA.
- 10 - Page always. This event is called while a page is active. Reserved for use by Plant SCADA.
- 11..17 - Undefined.
- 18 - Report start. The report server is about to start a new report. This event is called on the report server. The return value must be 0.
- 19 - Device history. A device history has just completed. The return value must be 0.
- 20 - Login. A user has just logged in.
- 21 - Logout. A user has just logged out.

- 22 - Trend needs repainting. This event is called each time Plant SCADA re-animates a real-time trend or scrolls an historical trend. You should use this event to add additional animation to a trend, because Plant SCADA deletes all existing animation when a trend is re-drawn. (For example, if you want to display extra markers, you must use this event.)
- 23 - Hardware error has been detected.
- 24 - Keyboard cursor moved. This event is called each time the keyboard command cursor moves. The cursor can be moved by the cursor keys, the mouse, or the Cicode function KeySetCursor(). Note that you can find where the keyboard command cursor is located by calling the function KeyGetCursor().
- 25 - Network shutdown. A Shutdown network command has been issued.
- 26 - Runtime system shutdown and restart. (Required because of configuration changes.)
- 27 - Event. An event has occurred.
- 28 - Accumulator. An accumulator has logged a value.
- 29 - Slider. A slider has been selected.
- 30 - Slider. A slider has moved.
- 31 - Slider. A slider has been released (that is stopped moving).

While responding to slider events 29, 30, and 31, you can set any variables but you cannot call functions that cause immediate changes to animations on the page (for example, DspText() and DspSym()). Types 29, 30, & 31 relate only to V3.xx and V4.xx animations, and will be superseded in future releases.

- 32 - Shutdown. Plant SCADA is being shutdown.

- 33 - Reserved for Plant SCADA internal use.

34 - 41 - Plant SCADA Confirmation Events. Reserved for Plant SCADA internal use. For the confirmation events, two sets of event type code are defined. The runtime calls the Plant SCADA event handler first, and conditionally proceed to the user's event handler depending on the return value of the Plant SCADA event handler.

- 34 -Plant SCADA Event: Child Window Close Confirmation.
- 35 - Plant SCADA Event: Main Window Close Confirmation.
- 36 - Plant SCADA Event: Maximize Window Confirmation.
- 37 - Plant SCADA Event: Minimize Window Confirmation.
- 38 - Plant SCADA Event: Restore Window Confirmation.
- 39 - Plant SCADA Event: Move Window Confirmation.
- 40 - Plant SCADA Event: Size Window Confirmation.
- 41 - Plant SCADA Event: Shutdown Confirmation Confirmation.

42 to 49 - User Confirmation Events. These functions are called when a specific event (mainly from Window title bar) occur and before the runtime performs the intended action. This gives a chance for the user to decide what to do with the event. If the return value is 0, the event will be passed on to the default handler so the intended action will be performed. If the return value is not 0, the event will be ignored and no further action will be taken.

- 42 - Child Window Close Confirmation, when the close button of the windows' title bar is clicked or an equivalent Windows' message is received.
- 43 - Main Window Close Confirmation, when close button of the windows' title bar is clicked which will cause the process to shutdown.
- 44 - Maximize Window Confirmation, when the maximize button of the windows' title bar is clicked or an equivalent Windows' message is received.

- 45- Minimize Window Confirmation, when the minimize button of the windows' title bar is clicked or an equivalent Windows' message is received.
  - 46 - Restore Window Confirmation, when the restore button of the windows' title bar is clicked or an equivalent Windows' message is received.
  - 47 - Move Window Confirmation, when the window is being dragged or an equivalent Windows' message is received.
  - 48 - Size Window Confirmation, when the windows is being resized or an equivalent Windows' message is received.
  - 49 - Shutdown Confirmation, when shutdown() function is called.
- 50 - 127 - Reserved for future Plant SCADA use.
- 128 - 256 - User-defined events. These events are for your own use.

## Return Value

0 (zero) if successful, otherwise an error is set. To view the error, use the IsError() function.

## Related Functions

[OnEvent](#), [GetEvent](#), [WinNew](#), [WinNewAt](#), [WinNumber](#), [IsError](#)

## Example

```
! Call Event Type 1 - key has been pressed in the current window.  
Number=WinNumber();  
CallEvent(Number,1);
```

## See Also

[Event Functions](#)

## ChainEvent

Calls an event function using the function handle. This creates a chain of event handlers from a single event. Use the GetEvent() function to get the function number of the current event handler.

## Syntax

**ChainEvent(*hFn*)**

*hFn*:

The function handle, as returned from the GetEvent() function.

## Return Value

The return value of the called event function.

## Related Functions

[OnEvent](#), [GetEvent](#)

## Example

See [GetEvent](#).

## See Also

[Event Functions](#)

## GetEvent

Gets the function handle of the existing callback event handler. You can use this function handle in the ChainEvent() function to chain call the existing event function, or in the SetEvent() function to restore the event handler.

## Syntax

**GetEvent(*nType*)**

*nType*:

The type of event:

- 0 - The mouse has moved. When the mouse moves the callback function is called. The return value must be 0.
- 1 - A key has been pressed. When the user presses a key, the callback function is called after Plant SCADA checks for hot keys. If the return value is 0, Plant SCADA checks for key sequences. If the return value is not 0, Plant SCADA assumes that you will process the key and does not check the key sequence. It is up to you to remove the key from the key command line. If you are using a right mouse button click as an event, you should read about the ButtonOnlyLeftClick parameter.
- 2 - Error event. This event is called if an error is detected in Cicode, so you can write a single error function to check for your errors. If the return value is 0, Plant SCADA continues to process the error and generates a hardware error - it may then halt the Cicode task. If the return value is not 0, Plant SCADA assumes that you will process the error, and continues the Cicode without generating a hardware error.
- 3 - Page user communication error. A communication error has been detected in the data required for this page. If the return value is 0 (zero), Plant SCADA still animates the page. If the return value is not zero, it does not update the page.
- 4 - Page user open. A new page is being opened. This event allows you to define a single function that is called when all pages are opened. The return value must be 0.
- 5 - Page user close. The current page is being closed. This event allows you to define a single function that is called when all pages are closed. The return value must be 0.
- 6 - Page user always. The page is active. This event allows you to define a single function that is called when all pages are active. The return value must be 0.
- 7 - Page communication error. A communication error has been detected in the data required for this page.

Reserved for use by Plant SCADA.

- 8 - Page open. This event is called each time a page is opened. Reserved for use by Plant SCADA.
- 9 - Page close. This event is called each time a page is closed. Reserved for use by Plant SCADA.
- 10 - Page always. This event is called while a page is active. Reserved for use by Plant SCADA.
- 11..17 - Undefined.
- 18 - Report start. The report server is about to start a new report. This event is called on the report server. The return value must be 0.
- 19 - Device history. A device history has just completed. The return value must be 0.
- 20 - Login. A user has just logged in.
- 21 - Logout. A user has just logged out.
- 22 - Trend needs repainting. This event is called each time Plant SCADA re-animates a real-time trend or scrolls an historical trend. You should use this event to add additional animation to a trend, because Plant SCADA deletes all existing animation when a trend is re-drawn. (For example, if you want to display extra markers, you must use this event.)
- 23 - Hardware error has been detected.
- 24 - Keyboard cursor moved. This event is called each time the keyboard command cursor moves. The cursor can be moved by the cursor keys, the mouse, or the Cicode function KeySetCursor(). Note that you can find where the keyboard command cursor is located by calling the function KeyGetCursor().
- 25 - Network shutdown. A Shutdown network command has been issued.
- 26 - Runtime system shutdown and restart. (Required because of configuration changes.)
- 27 - Event. An event has occurred.
- 28 - Accumulator. An accumulator has logged a value.
- 29 - Slider. A slider has been selected.
- 30 - Slider. A slider has moved.
- 31 - Slider. A slider has been released (that is stopped moving).

While responding to slider events 29, 30, and 31, you can set any variables but you cannot call functions that cause immediate changes to animations on the page (for example, DspText() and DspSym()). Types 29, 30, & 31 relate only to V3.xx and V4.xx animations, and will be superseded in future releases.

- 32 - Shutdown. Plant SCADA is being shutdown.
- 33 - Reserved for Plant SCADA internal use.

34 - 41 - Plant SCADA Confirmation Events. Reserved for Plant SCADA internal use. For the confirmation events, two sets of event type code are defined. The runtime calls the Plant SCADA event handler first, and conditionally proceed to the user's event handler depending on the return value of the Plant SCADA event handler.

- 34 -Plant SCADA Event: Child Window Close Confirmation.
- 35 - Plant SCADA Event: Main Window Close Confirmation.
- 36 - Plant SCADA Event: Maximize Window Confirmation.
- 37 - Plant SCADA Event: Minimize Window Confirmation.
- 38 - Plant SCADA Event: Restore Window Confirmation.
- 39 - Plant SCADA Event: Move Window Confirmation.
- 40 - Plant SCADA Event: Size Window Confirmation.
- 41 - Plant SCADA Event: Shutdown Confirmation Confirmation.

42 to 49 - User Confirmation Events. These functions are called when a specific event (mainly from Window title

bar) occur and before the runtime performs the intended action. This gives a chance for the user to decide what to do with the event. If the return value is 0, the event will be passed on to the default handler so the intended action will be performed. If the return value is not 0, the event will be ignored and no further action will be taken.

- 42 - Child Window Close Confirmation, when the close button of the windows' title bar is clicked or an equivalent Windows' message is received.
  - 43 - Main Window Close Confirmation, when close button of the windows' title bar is clicked which will cause the process to shutdown.
  - 44 - Maximize Window Confirmation, when the maximize button of the windows' title bar is clicked or an equivalent Windows' message is received.
  - 45- Minimize Window Confirmation, when the minimize button of the windows' title bar is clicked or an equivalent Windows' message is received.
  - 46 - Restore Window Confirmation, when the restore button of the windows' title bar is clicked or an equivalent Windows' message is received.
  - 47 - Move Window Confirmation, when the window is being dragged or an equivalent Windows' message is received.
  - 48 - Size Window Confirmation, when the windows is being resized or an equivalent Windows' message is received.
  - 49 - Shutdown Confirmation, when shutdown() function is called.
- 50 - 127 - Reserved for future Plant SCADA use.
- 128 - 256 - User-defined events. These events are for your own use.

## Return Value

The function handle of the existing callback event handler, or -1 if there are no event handlers.

## Related Functions

[OnEvent](#), [CallEvent](#), [ChainEvent](#), [SetEvent](#)

## Example

```
! Get existing event handler.  
hFn=GetEvent(0);  
! Trap mouse movements.  
OnEvent(0,MouseFn);  
..  
! Restore old event handler.  
SetEvent(0,hFn);  
INT  
FUNCTION MouseFn()  
..  
! Chain call old event handler.  
RETURN ChainEvent(hFn);  
END
```

## See Also

[Event Functions](#)

## OnEvent

Sets an event callback function for an event type. The callback function is called when the event occurs.

Using callback functions removes the need for polling or checking for events. Callback functions have no arguments and needs to return an integer. They also need to be non-blocking.

Plant SCADA starts running the function immediately, without reading any data from the I/O devices. Any I/O device variable that you use will contain either 0 (zero) or bad quality. Only local variables are supported.

The return value of the callback will depend on the type of the event. Set the Fn argument to 0 (zero) to disable the event.

### Note:

- For event type 42..49, a windows system event is received. When the user clicks any button of the Windows tile bar, or size/move the window, or shutting down a process, the callback function is called. If the return value is 0, the event will be processed by Plant SCADA in default mode which is the original behavior. If the return value is not 0, Plant SCADA assumes that you will process the event and discard the message internally.
- If the event handler is non-interactive with instant return value, it can be called directly.
- Since the event callback function cannot block, any blocking function calls or long loops should be moved to a separate function. The callback function can start with the TaskNew() command. For confirmation events, the event callback function should return a non-zero value to tell Plant SCADA not to continue processing the event. The new task can call the appropriate function to process the event, such as the Shutdown() function for a Shutdown Confirmation event.
- The "Shutdown Confirmation" event is raised in the following cases:
  - Calling shutdown() without setting the callEvent parameter to zero.
  - Closing the top level application window - after raising the event "Main Window Close Confirmation".
  - Receiving WM\_QUIT message what can come from any source both internal and external.
- The "Shutdown Confirmation" event is not raised in other cases including, but not limited to, the following known scenarios:
  - The RuntimeManager is doing stop, restart the program, or doing shutdown all.
  - The Windows Task Manager is doing End Task, End Process, or End Process Tree.

## Syntax

**OnEvent(*Type*, *Fn*)**

*nType*:

The type of event:

- 0 - The mouse has moved. When the mouse moves the callback function is called. The return value must be 0.
- 1 - A key has been pressed. When the user presses a key, the callback function is called after Plant SCADA checks for hot keys. If the return value is 0, Plant SCADA checks for key sequences. If the return value is not 0,

Plant SCADA assumes that you will process the key and does not check the key sequence. It is up to you to remove the key from the key command line. If you are using a right mouse button click as an event, you should read about the ButtonOnlyLeftClick parameter.

- 2 - Error event. This event is called if an error is detected in Cicode, so you can write a single error function to check for your errors. If the return value is 0, Plant SCADA continues to process the error and generates a hardware error - it may then halt the Cicode task. If the return value is not 0, Plant SCADA assumes that you will process the error, and continues the Cicode without generating a hardware error.
- 3 - Page user communication error. A communication error has been detected in the data required for this page. If the return value is 0 (zero), Plant SCADA still animates the page. If the return value is not zero, it does not update the page.
- 4 - Page user open. A new page is being opened. This event allows you to define a single function that is called when all pages are opened. The return value must be 0.
- 5 - Page user close. The current page is being closed. This event allows you to define a single function that is called when all pages are closed. The return value must be 0.
- 6 - Page user always. The page is active. This event allows you to define a single function that is called when all pages are active. The return value must be 0.
- 7 - Page communication error. A communication error has been detected in the data required for this page. Reserved for use by Plant SCADA.
- 8 - Page open. This event is called each time a page is opened. Reserved for use by Plant SCADA.
- 9 - Page close. This event is called each time a page is closed. Reserved for use by Plant SCADA.
- 10 - Page always. This event is called while a page is active. Reserved for use by Plant SCADA.
- 11..17 - Undefined.
- 18 - Report start. The report server is about to start a new report. This event is called on the report server. The return value must be 0.
- 19 - Device history. A device history has just completed. The return value must be 0.
- 20 - Login. A user has just logged in.
- 21 - Logout. A user has just logged out.
- 22 - Trend needs repainting. This event is called each time Plant SCADA re-animates a real-time trend or scrolls an historical trend. You should use this event to add additional animation to a trend, because Plant SCADA deletes all existing animation when a trend is re-drawn. (For example, if you want to display extra markers, you must use this event.)
- 23 - Hardware error has been detected.
- 24 - Keyboard cursor moved. This event is called each time the keyboard command cursor moves. The cursor can be moved by the cursor keys, the mouse, or the Cicode function KeySetCursor(). Note that you can find where the keyboard command cursor is located by calling the function KeyGetCursor().
- 25 - Network shutdown. A Shutdown network command has been issued.
- 26 - Runtime system shutdown and restart. (Required because of configuration changes.)
- 27 - Event. An event has occurred.
- 28 - Accumulator. An accumulator has logged a value.
- 29 - Slider. A slider has been selected.
- 30 - Slider. A slider has moved.
- 31 - Slider. A slider has been released (that is stopped moving).

While responding to slider events 29, 30, and 31, you can set any variables but you cannot call functions that cause immediate changes to animations on the page (for example, DspText() and DspSym()). Types 29, 30, & 31 relate only to V3.xx and V4.xx animations, and will be superseded in future releases.

- 32 - Shutdown. Plant SCADA is being shutdown.
- 33 - Reserved for Plant SCADA internal use.

34 - 41 - Plant SCADA Confirmation Events. Reserved for Plant SCADA internal use. For the confirmation events, two sets of event type code are defined. The runtime calls the Plant SCADA event handler first, and conditionally proceed to the user's event handler depending on the return value of the Plant SCADA event handler.

- 34 -Plant SCADA Event: Child Window Close Confirmation.
- 35 - Plant SCADA Event: Main Window Close Confirmation.
- 36 - Plant SCADA Event: Maximize Window Confirmation.
- 37 - Plant SCADA Event: Minimize Window Confirmation.
- 38 - Plant SCADA Event: Restore Window Confirmation.
- 39 - Plant SCADA Event: Move Window Confirmation.
- 40 - Plant SCADA Event: Size Window Confirmation.
- 41 - Plant SCADA Event: Shutdown Confirmation Confirmation.

42 to 49 - User Confirmation Events. These functions are called when a specific event (mainly from Window title bar) occur and before the runtime performs the intended action. This gives a chance for the user to decide what to do with the event. If the return value is 0, the event will be passed on to the default handler so the intended action will be performed. If the return value is not 0, the event will be ignored and no further action will be taken.

- 42 - Child Window Close Confirmation, when the close button of the windows' title bar is clicked or an equivalent Windows' message is received.
- 43 - Main Window Close Confirmation, when close button of the windows' title bar is clicked which will cause the process to shutdown.
- 44 - Maximize Window Confirmation, when the maximize button of the windows' title bar is clicked or an equivalent Windows' message is received.
- 45- Minimize Window Confirmation, when the minimize button of the windows' title bar is clicked or an equivalent Windows' message is received.
- 46 - Restore Window Confirmation, when the restore button of the windows' title bar is clicked or an equivalent Windows' message is received.
- 47 - Move Window Confirmation, when the window is being dragged or an equivalent Windows' message is received.
- 48 - Size Window Confirmation, when the windows is being resized or an equivalent Windows' message is received.
- 49 - Shutdown Confirmation, when shutdown() function is called.

50 - 127 - Reserved for future Plant SCADA use.

128 - 256 - *User-defined events. These events are for your own use.*

*Fn:*

The function to call when the event occurs. This callback function needs to have no arguments, so you specify the function with no parentheses (). The callback function needs to return INT as its return data type. You cannot specify a Plant SCADA built-in function as a callback function.

Set *Fn* to 0 to disable the event.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[GetEvent](#), [CallEvent](#), [ChainEvent](#)

## Examples

**Example 1** - Calls a function called KeyFn to determine if the ESC key has been pressed on a key press event.

```
OnEvent(1,KeyFn);
INT
FUNCTION KeyFn()
    INT Key;
    Key=KeyPeek(0);
    IF Key=27 THEN
        Prompt("ESC pressed");
        RETURN 1;
    ELSE
        RETURN 0;
    END
END
```

**Example 2** - Calls a function called MouseFn to display the position of the mouse whenever it is moved.

```
OnEvent(0,MouseFn);
INT
FUNCTION MouseFn()
    INT X,Y;
    DspGetMouse(X,Y);
    RETURN 0;
END
```

**Example 3** - Presents a user with a confirmation dialog box when the main window close button is pressed.

```
sFUNCTION XYZStartup()
    OnEvent(43, ConfirmShutdown);
END

INT FUNCTION ConfirmShutdown()
    TaskNew("_ShutdownDlg", "", 2+8);
    RETURN 1;
END

FUNCTION _ShutdownDlg()
    STRING sMsg = "Are you sure ?";
    INT nRC;

    nRC = Message("Close this window and shutdown", sMsg, 1+32);
```

```
IF nRC = 0 THEN
    Shutdown("", "", 1, "", 0);
END
END
```

## See Also

[Event Functions](#)

## SetEvent

Sets an event callback function by specifying a function handle. You can use this function with the GetEvent function to restore an old event handler.

## Syntax

**SetEvent(*nType*, *hFn*)**

*nType*:

The type of event:

- 0 - The mouse has moved. When the mouse moves the callback function is called. The return value must be 0.
- 1 - A key has been pressed. When the user presses a key, the callback function is called after Plant SCADA checks for hot keys. If the return value is 0, Plant SCADA checks for key sequences. If the return value is not 0, Plant SCADA assumes that you will process the key and does not check the key sequence. It is up to you to remove the key from the key command line. If you are using a right mouse button click as an event, you should read about the ButtonOnlyLeftClick parameter.
- 2 - Error event. This event is called if an error is detected in Cicode, so you can write a single error function to check for your errors. If the return value is 0, Plant SCADA continues to process the error and generates a hardware error - it may then halt the Cicode task. If the return value is not 0, Plant SCADA assumes that you will process the error, and continues the Cicode without generating a hardware error.
- 3 - Page user communication error. A communication error has been detected in the data required for this page. If the return value is 0 (zero), Plant SCADA still animates the page. If the return value is not zero, it does not update the page.
- 4 - Page user open. A new page is being opened. This event allows you to define a single function that is called when all pages are opened. The return value must be 0.
- 5 - Page user close. The current page is being closed. This event allows you to define a single function that is called when all pages are closed. The return value must be 0.
- 6 - Page user always. The page is active. This event allows you to define a single function that is called when all pages are active. The return value must be 0.
- 7 - Page communication error. A communication error has been detected in the data required for this page. Reserved for use by Plant SCADA.
- 8 - Page open. This event is called each time a page is opened. Reserved for use by Plant SCADA.
- 9 - Page close. This event is called each time a page is closed. Reserved for use by Plant SCADA.
- 10 - Page always. This event is called while a page is active. Reserved for use by Plant SCADA.
- 11..17 - Undefined.

- 18 - Report start. The report server is about to start a new report. This event is called on the report server. The return value must be 0.
- 19 - Device history. A device history has just completed. The return value must be 0.
- 20 - Login. A user has just logged in.
- 21 - Logout. A user has just logged out.
- 22 - Trend needs repainting. This event is called each time Plant SCADA re-animates a real-time trend or scrolls an historical trend. You should use this event to add additional animation to a trend, because Plant SCADA deletes all existing animation when a trend is re-drawn. (For example, if you want to display extra markers, you must use this event.)
- 23 - Hardware error has been detected.
- 24 - Keyboard cursor moved. This event is called each time the keyboard command cursor moves. The cursor can be moved by the cursor keys, the mouse, or the Cicode function KeySetCursor(). Note that you can find where the keyboard command cursor is located by calling the function KeyGetCursor().
- 25 - Network shutdown. A Shutdown network command has been issued.
- 26 - Runtime system shutdown and restart. (Required because of configuration changes.)
- 27 - Event. An event has occurred.
- 28 - Accumulator. An accumulator has logged a value.
- 29 - Slider. A slider has been selected.
- 30 - Slider. A slider has moved.
- 31 - Slider. A slider has been released (that is stopped moving).

While responding to slider events 29, 30, and 31, you can set any variables but you cannot call functions that cause immediate changes to animations on the page (for example, DspText() and DspSym()). Types 29, 30, & 31 relate only to V3.xx and V4.xx animations, and will be superseded in future releases.

- 32 - Shutdown. Plant SCADA is being shutdown.

- 33 - Reserved for Plant SCADA internal use.

34 - 41 - Plant SCADA Confirmation Events. Reserved for Plant SCADA internal use. For the confirmation events, two sets of event type code are defined. The runtime calls the Plant SCADA event handler first, and conditionally proceed to the user's event handler depending on the return value of the Plant SCADA event handler.

- 34 -Plant SCADA Event: Child Window Close Confirmation.
- 35 - Plant SCADA Event: Main Window Close Confirmation.
- 36 - Plant SCADA Event: Maximize Window Confirmation.
- 37 - Plant SCADA Event: Minimize Window Confirmation.
- 38 - Plant SCADA Event: Restore Window Confirmation.
- 39 - Plant SCADA Event: Move Window Confirmation.
- 40 - Plant SCADA Event: Size Window Confirmation.
- 41 - Plant SCADA Event: Shutdown Confirmation Confirmation.

42 to 49 - User Confirmation Events. These functions are called when a specific event (mainly from Window title bar) occur and before the runtime performs the intended action. This gives a chance for the user to decide what to do with the event. If the return value is 0, the event will be passed on to the default handler so the intended action will be performed. If the return value is not 0, the event will be ignored and no further action will be taken.

- 42 - Child Window Close Confirmation, when the close button of the windows' title bar is clicked or an

equivalent Windows' message is received.

- 43 - Main Window Close Confirmation, when close button of the windows' title bar is clicked which will cause the process to shutdown.
- 44 - Maximize Window Confirmation, when the maximize button of the windows' title bar is clicked or an equivalent Windows' message is received.
- 45 - Minimize Window Confirmation, when the minimize button of the windows' title bar is clicked or an equivalent Windows' message is received.
- 46 - Restore Window Confirmation, when the restore button of the windows' title bar is clicked or an equivalent Windows' message is received.
- 47 - Move Window Confirmation, when the window is being dragged or an equivalent Windows' message is received.
- 48 - Size Window Confirmation, when the windows is being resized or an equivalent Windows' message is received.
- 49 - Shutdown Confirmation, when shutdown() function is called.

50 - 127 - Reserved for future Plant SCADA use.

128 - 256 - *User-defined events. These events are for your own use.*

*hFn*

The function handle, as returned from the GetEvent function.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[GetEvent](#)

## Example

See [GetEvent](#).

## See Also

[Event Functions](#)

## File Functions

The following functions relate to file operations:

<a href="#">FileClose</a>	Closes a file.
<a href="#">FileCopy</a>	Copies a file or group of files.
<a href="#">FileDelete</a>	Deletes a file.

<a href="#">FileEOF</a>	Checks for the end of a file.
<a href="#">FileExist</a>	Checks if a file exists.
<a href="#">FileFind</a>	Finds a file that matches a specified search criteria.
<a href="#">FileFindClose</a>	Closes a find (started with FileFind) that did not run to completion.
<a href="#">FileGetTime</a>	Gets the time on a file.
<a href="#">FileMakePath</a>	Creates a file path string from individual components.
<a href="#">FileOpen</a>	Opens or creates an ASCII file.
<a href="#">FilePrint</a>	Prints a file on a device.
<a href="#">FileRead</a>	Reads characters from a file.
<a href="#">FileReadBlock</a>	Reads a block of characters from a file.
<a href="#">FileReadLn</a>	Reads a line from a file.
<a href="#">FileRename</a>	Renames a file.
<a href="#">FileRichTextPrint</a>	Prints a rich text file.
<a href="#">FileSeek</a>	Seeks a position in a file.
<a href="#">FileSetTime</a>	Sets the time on a file.
<a href="#">FileSize</a>	Gets the size of a file.
<a href="#">FileSplitPath</a>	Splits a file path into individual string components.
<a href="#">FileWrite</a>	Writes characters to a file.
<a href="#">FileWriteBlock</a>	Writes a block of characters to a file.
<a href="#">FileWriteLn</a>	Writes a line to a file.

## See Also

[Cicode Function Categories](#)  
[Using Cicode Functions](#)

## FileClose

Closes a file. All data written to the file is flushed to disk when the file is closed, and the file number becomes invalid.

## Syntax

**FileClose(*File*)**

*File*:

The file number.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FileOpen](#)

## Example

```
File=FileOpen("C:\Data\Report.Txt","r");
..
! Do file operations.
..
! Close the file.
FileClose(File);
```

## See Also

[File Functions](#)

## FileCopy

Copies a file. You can use the DOS wild card characters (\*) and (?) to copy groups of files. In asynchronous mode, this function will return immediately and the copy will continue in the background. Copying files does not interfere with the operation of other Plant SCADA tasks, because this function is time-sliced.

If synchronous mode is selected, this function becomes a blocking function.

## Syntax

**FileCopy(*sSource*, *sDest*, *nMode*)**

*sSource*:

The name of the source file to copy.

*sDest*:

The name of destination file to copy to. To copy a file to the printer, specify the name as "LPT1.DOS".

*nMode*:

The copy mode:

0 - Normal

1 - Copy only if the file time is different.

2 - Copy in asynchronous mode.

Multiple modes can be selected by adding them together (for example, set *Mode* to 3 to copy in asynchronous mode if the file time is different).

## Return Value

0 (zero) if successful, otherwise an error is returned. However, if you copy in asynchronous mode, the return value does not reflect whether the copy operation was successful or not, because the function returns before the actual copy is complete.

## Related Functions

[FileDelete](#)

## Example

```
! Copy Report.Txt to Report.Bak.  
FileCopy ("C:\Data\Report.Txt", "C:\Data\Report.Bak",0);  
/* Copy AlarmLog.Txt to AlarmLog.Bak only if the file time is  
different. Copy in the background. */  
FileCopy ("AlarmLog.Txt", "AlarmLog.Bak",1+2);
```

## See Also

[File Functions](#)

## FileDelete

Deletes a file.

## Syntax

**FileDelete(*sName*)**

*Name:*

The name of the file to delete.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FileCopy](#)

## Example

```
! Delete old report file.  
FileDelete("C:\Data\Report.Txt");
```

## See Also

[File Functions](#)

### FileEOF

Determines if the end of the file has been reached.

## Syntax

**FileEOF(*File*)**

*File*:

The file number.

## Return Value

1 if the end of file has been reached, otherwise 0 (zero).

## Related Functions

[FileSeek](#)

## Example

```
WHILE NOT FileEOF(File) DO  
    Str=FileReadLn(File);  
END
```

## See Also

[File Functions](#)

### FileExist

Checks if a file exists. If the return value is 1, the file exists.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**FileExist(*sName*)**

**Name:**

The name of the file to check.

## Return Value

TRUE (1) if the file exists, otherwise FALSE (0).

## Related Functions

[FileOpen](#)

## Example

```
! Check if the file exists
IF FileExist("C:\Data\Report.Txt") THEN
    ! The file exists
END
```

## See Also

[File Functions](#)

## FileFind

Finds a file that matches a specified search criteria. To find a list of files, you need to first call this function with the required path and mode (to find the first file), then call the function again with an empty path and a mode of 0 (to find the remaining files). After the last file is found, an empty string is returned.

If the search is for multiple files, FileFindClose needs to be called if the search does not run to completion (for example, you do not run until an empty string is returned).

## Syntax

**FileFind(*sPath*, *nMode*)**

***sPath:***

The name of the file to check. To search for multiple files, the wildcards \* and ? can be used to match multiple entries.

***nMode:***

The type of file to check:

0 - Normal files (includes files with read-only and archived attributes)

1 - Read-only files only

2 - Hidden files only

4 - System files only

16 - Subdirectories only

32 - Archived files only

128 - Files with no attributes only

These numbers can be added together to search for multiple types of files during one search.

## Return Value

The full path and filename. If no files are found, an empty string is returned.

## Related Functions

[FileOpen](#), [FileSplitPath](#), [FileMakePath](#)

## Example

```
! Search for all dBase files in the run directory and make a backup
sPath = FileFind("[run]:\*.dbf", 0);
WHILE StrLength(sPath) > 0 DO
    FileSplitPath(sPath, sDrive, sDir, sFile, sExt);
    sBak = FileMakePath(sDrive, sDir, sFile, "BAK");
    FileCopy(sPath, sBak, 0);
    ! Find the next file
    sPath = FileFind("", 0);
END
```

## See Also

[File Functions](#)

## FileFindClose

Closes a find (started with [FileFind](#)) that did not run to completion.

## Syntax

**FileFindClose()**

## Return Value

0 if no error is detected, or a Cicode error code if an error occurred.

## Related Functions

[FileFind](#)

## Example

```
//Find the first dbf file starting with fred
```

```
sPath = FileFind("[run]:\fred*.dbf", 0);
IF (StrLength(sPath) > 0) THEN
    //Do work here
    FileFindClose();
END
```

## See Also

[File Functions](#)

### FileGetTime

Gets the time on a file.

## Syntax

**FileGetTime(*File*)**

*File*:

The file number.

## Return Value

The file time of the file (in the Plant SCADA time/date variable format).

## Related Functions

[FileOpen](#), [FileClose](#), [FileSetTime](#)

## Example

```
File = FileOpen("[data]:report.txt", "r");
! Get the time of the file
iTime = FileGetTime(File);
FileClose(File);
```

## See Also

[File Functions](#)

### FileMakePath

Creates a file path string from individual components.

## Syntax

**FileMakePath(*sDrive*, *sDir*, *sFile*, *sExt*)**

*sDrive:*

The disk drive.

*sDir:*

The directory string.

*sFile:*

The file name (without the extension).

*sExt:*

The file extension.

## Return Value

The full path as a string.

## Related Functions

[FileSeek](#), [FileFind](#), [FileSplitPath](#)

## Example

See [FileFind](#).

## See Also

[File Functions](#)

## FileOpen

Opens a file and returns a file number that can be used by other file functions. The maximum file size supported is 1 Megabyte for displaying text files.

You can also use this function to check if a file exists, by opening the file in read-only mode. A return value of -1 indicates that the file cannot be opened.

[ErrSet\(1\)](#) needs to be in the previous line of your code, else the execution stops and a hardware error is generated. If [ErrSet\(1\)](#) is used then it does not halt, and -1 is returned.

## Syntax

**FileOpen(*sName*, *nMode*)**

*sName:*

The name of the file to open.

*nMode:*

The mode of the opened file:

- "a" - Append mode. Allows you to append to the file without removing the end of file marker. The file cannot be read. If the file does not exist, it will be created.

- "a+" - Append and read modes. Allows you to append to the file and read from it. The end of file marker will be removed before writing and restored when writing is complete. If the file does not exist, it will be created.
- "r" - Read-only mode. Allows you to (only) read from the file. If the file does not exist or cannot be found, the function call will return the value -1.
- "r+" - Read/write mode. Allows you to read from, and write to, the file. If the file already exists (before the function is called), its contents will be deleted. If the file does not exist or cannot be found, the function call will return the value -1.
- "w" - Write mode. Opens an empty file for writing. If the file already exists (before the function is called), its contents will be deleted. If the file does not exist or cannot be found, the file will be created.
- "w+" - Read/write mode. Opens an empty file for both reading and writing. If the file already exists (before the function is called), its contents will be deleted. If the file does not exist or cannot be found, the file will be created.

## Return Value

The file number. If the file cannot be opened, -1 is returned and the code is halted.

## Related Functions

[FileClose](#), [FileRead](#), [FileReadLn](#), [FileWrite](#), [FileWriteLn](#)

## Example

```
! Open a file in read-only mode.  
ErrSet(1);  
File=FileOpen("C:\Data\Report.Txt","r");  
ErrSet(0);
```

## See Also

[File Functions](#)

## FilePrint

Prints a file on a device.

## Syntax

**FilePrint(*Devicename*, *Filename*)**

*Devicename*:

The name of the target device.

*Filenname*:

The name of the file to print on the device.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FileOpen](#), [FileWrite](#), [FileWriteLn](#)

## Example

```
! Print a data file on the system printer.  
FilePrint("Printer_Device","Data.txt");
```

## See Also

[File Functions](#)

## FileRead

Reads a number of characters from a file. The string can contain less characters than requested if the end of file is reached. A maximum of 255 characters can be read in each call.

## Syntax

**FileRead**(*File*, *Length*)

*File*:

The file number.

*Length*:

The number of characters to read.

## Return Value

The text from the file (as a string).

## Related Functions

[FileOpen](#), [FileClose](#), [FileReadLn](#)

## Example

```
WHILE NOT FileEOF(File) DO  
    Str=FileRead(File,20);  
END
```

## See Also

[File Functions](#)

### FileReadBlock

Reads a number of characters from a file. The buffer can contain less characters than requested if the end of file is reached. A maximum of 255 characters can be read in each call. The data should be treated as a binary data and should not be passed to string functions. You may use StrGetChar() function to extract each character from the buffer, or pass the buffer to another function which will accept binary data.

## Syntax

**FileReadBlock(*File*, *Buffer*, *Length*)**

*File*:

The file number.

*Buffer*:

The buffer to return the binary data. This may be a string or a string packed with binary data. The string terminator is ignored and the length argument specifies the number of characters to write. Must be a string variable.

*Length*:

The number of characters to read.

## Return Value

The number of characters read from the file. When the end of the file is found 0 will be returned. If an error occurs -1 will be returned and IsError() will return the error code.

## Related Functions

[FileOpen](#), [FileClose](#), [FileRead](#), [FileWriteBlock](#), [StrGetChar](#)

## Example

```
// read binary file and copy to COM port
length = FileReadBlock(File, buf, 128);
WHILE length > 0 DO
    ComWrite(hPort, buf, length, 0);
    length = FileReadBlock(File, buf, 128);
END
```

## See Also

[File Functions](#)

## FileReadLn

Reads a line from a file. Up to 255 characters can be returned. The carriage return and newline characters are not returned. If the line is longer than 255 characters, the error overflow (code 275) is returned - you should call this function again to read the rest of the line.

## Syntax

**FileReadLn(*File*)**

*File*:

The file number.

## Return Value

The text, as a string.

## Related Functions

[FileOpen](#), [FileClose](#), [FileRead](#)

## Example

```
sLine = FileReadLn(hFile);
! do stuff with the string
WHILE IsError() = 275 DO
    ! read the rest of the line
    sLine = FileReadLn(hFile);
    ! do stuff with the rest of the line
END
```

## See Also

[File Functions](#)

## FileRename

Renames a file.

## Syntax

**FileRename(*Oldname*, *Newname*)**

*Oldname*:

The original name of the file.

*Newname*:

The new name of the file.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FileCopy](#), [FileDelete](#)

## Example

```
! Rename REPORT.TXT as REPORT.OLD.  
FileRename("C:\Data\Report.Txt", "C:\Data\Report.Old");
```

## See Also

[File Functions](#)

## FileRichTextPrint

Prints the rich text file *sFilename* to the printer given by *sPortname*.

## Syntax

**FileRichTextPrint(*sFilename*, *sPortName*)**

*sFilename*:

The filename of the rich text format file. The filename needs to be entered in quotation marks "".

Remember that the filename for a saved report comes from the File Name field in the Devices form. The location of the saved file needs to also be included as part of the filename. For example, if the filename in the Devices form listed [Data];RepDev.rtf, then you would need to enter "[Data]\repdev.rtf" as your argument. Alternatively, you can manually enter the path, for example, "c:\MyApplication\data\repdev.rtf".

If you are keeping a number of history files for the report, instead of using the extension rtf, you need to change it to reflect the number of the desired history file, for example, 001.

*PortName*:

The name of the printer port to which the rich text file will be printed. This name needs to be enclosed within quotation marks "". For example "LPT1", to print to the local printer, or "\\\Pserver\canon1" using UNC to print to a network printer.

## Return Values

0 if successful, otherwise an error is returned.

## Related Functions

[PageRichTextFile](#), [DspRichTextPrint](#)

## Example

```
// This would print the file [Data]\richtext.rtf to LPT1. Remember
// that the [Data] path is specified in the Citect.ini file. The file
// richtext.rtf is the name of the output file for the report, as
// specified in the Devices form. //
iResult = FileRichTextPrint("[Data]\richtext.rtf","LPT1:");
// This would print the file f:\Plant SCADA\data\richtext.rtf to LPT1.
// The file richtext.rtf is the name of the output file for the
// report, as specified in the Devices form. //
iResult =
FileRichTextPrint("f:\Plant SCADA\data\richtext.rtf","LPT1:");
```

## See Also

[File Functions](#)

## FileSeek

Moves the file pointer to a specified position in a file.

## Syntax

**FileSeek(File, Offset)**

*File:*

The file number.

*Offset:*

The offset in bytes, from 0 to (maximum file size -1). This value needs to be  $\geq 0$ .

## Return Value

The new file position, or -1 if an error is detected.

## Related Functions

[FileSize](#)

## Example

```
! Seek to the start of the file.
FileSeek(File,0);
```

## See Also

[File Functions](#)

## FileSetTime

Sets the time on a file.

**Note:** In order for this function to work, the file needs to first be opened in write or read/write mode.

---

## Syntax

**FileSetTime(*File*, *iTime*)**

*File*:

The file number.

*iTime*:

The new file time, in the Plant SCADA time/date variable format.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FileOpen](#), [FileClose](#), [FileGetTime](#)

## Example

```
File = FileOpen("[data]:report.txt", "r+");
! set the file to the current time
FileSetTime(File, TimeCurrent());
FileClose(File);
```

## See Also

[File Functions](#)

## FileSize

Gets the size of a file.

## Syntax

**FileSize(*File*)**

*File*:

The file number.

## Return Value

The size of the file in bytes.

## Related Functions

[FileSeek](#)

## Example

```
! Get the size of the file.  
Size=FileSize(File);
```

## See Also

[File Functions](#)

## FileSplitPath

Splits a file path into individual string components. You enter the full path string as *sPath*. The individual components of the path name are returned in the arguments *sDrive*, *sDir*, *sFile*, and *sExt*. Arguments must be declared as Cicode variables. If declared as local variables or tags, an "Incompatible Types" error will be returned.

## Syntax

**FileSplitPath(*sPath*, *sDrive*, *sDir*, *sFile*, *sExt*)**

*sPath*:

The full path string.

*sDrive*:

The disk drive. Must be a String type variable.

*sDir*:

The directory string. Must be a String type variable.

*sFile*:

The file name (without the extension). Must be a String type variable.

*sExt*:

The file extension. Must be a String type variable.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FileSeek](#), [FileFind](#), [FileMakePath](#)

## Example

See [FileFind](#).

## See Also

[File Functions](#)

## FileWrite

Writes a string to a file. The string is written at the current file position. The maximum number of characters allowed is 255.

## Syntax

**FileWrite**(*File*, *String*)

*File*:

The file number.

*String*:

The string to write.

## Return Value

The number of characters written.

## Related Functions

[FileOpen](#), [FileClose](#), [FileWriteLn](#)

## Example

```
! Write to the file.  
FileWrite(File,"Data");
```

## See Also

[File Functions](#)

## FileWriteBlock

Writes a string or buffer to a file. The data is written at the current file position. You may create the binary data by using the StrSetChar function or by reading the data from some other function. This function is similar to the FileWrite() function however you specify the length of data to write to the file. The FileWrite() function will send the data to the file until the sting terminator is found. FileWriteBlock() will ignore any string terminator and copy the length of bytes to the file. This allows this function to be used for binary data transfer.

## Syntax

**FileWriteBlock(*File*, *Buffer*, *Length*)**

*File*:

The file number.

*Buffer*:

The data to write to the file. This may be a string or a string packed with binary data. The string terminator is ignored and the length argument specifies the number of characters to write. Must be a string variable.

*Length*:

The number of characters to write. The maximum number of characters you may write in one call is 255. (If you use a string without a terminator in a function that expects a string, or in a Cicode expression, you could get invalid results.) To use the string to build up a buffer, you do not need the terminating 0 (zero).

## Return Value

The number of characters written to the file. If an error is detected -1 will be returned and IsError() will return the error code.

## Related Functions

[FileOpen](#), [FileClose](#), [FileWrite](#), [FileReadBlock](#), [StrSetChar](#)

## Example

```
STRING buf;
FOR I = 0 TO 20 DO
    StrSetChar(buf, I, I); // put binary data into string
END
! Write binary data to the file.
FileWrite(File, buf, 20);
```

## See Also

[File Functions](#)

## FileWriteLn

Writes a string to a file, followed by a newline character. The string is written at the current file position. The maximum number of characters allowed is 255.

## Syntax

**FileWriteLn(*File*, *String*)**

*File*:

The file number.

*String*:

The string to write.

## Return Value

The number of characters written (including the carriage return and newline characters).

## Related Functions

[FileOpen](#), [FileClose](#), [FileWrite](#)

## Example

```
! Write a line to the file.  
FileWriteLn(File,"Line of file data");
```

## See Also

[File Functions](#)

## Form Functions

Following are functions relating to forms:

<a href="#">FormActive</a>	Checks if a form is currently active.
<a href="#">FormAddList</a>	Adds a text string to a list box or combo box.
<a href="#">FormButton</a>	Adds a button to a form.
<a href="#">FormCheckBox</a>	Adds a check box to the current form.
<a href="#">FormComboBox</a>	Adds a combo box to the current form.
<a href="#">FormCurr</a>	Gets the current form and field handles.

FormDestroy	Removes a form from the screen.
FormEdit	Adds edit fields to a form.
FormField	Adds general fields to a form.
FormGetCurrlInst	Gets data associated with a field.
FormGetData	Gets the data associated with a form.
FormGetInst	Gets data associated with a field on a form.
FormGetText	Gets field text on an active form.
FormGoto	Go to a specified form.
FormGroupBox	Adds a group box to the current form.
FormInput	Adds an input field to a form.
FormListAddText	Adds a new text entry to a combo box or a list box.
FormListBox	Adds a list box to the current form.
FormListDeleteText	Deletes existing text from combo box or list box.
FormListSelectText	Selects (highlights) a text entry in a combo box or a list box.
FormNew	Creates a new form.
FormNumPad	Provides a keypad form for the operator to add numeric values.
FormOpenFile	Displays a File Open dialog box.
FormPassword	Adds a password (no echo) input field.
FormSecurePassword	Adds a secure password (no echo) input field.
FormPosition	Sets the position of a form on the screen, before it is displayed.
FormPrompt	Adds a prompt to a form.
FormRadioButton	Adds a radio button to the current form.
FormRead	Displays a form and reads user input.
FormSaveAsFile	Displays a File Save As dialog box.
FormSelectPrinter	Displays the Select Printer dialog box.

FormSetData	Sets data in a form.
FormSetInst	Associates data to a field on a form.
FormSetText	Sets field text on an active form.
FormWndHnd	Gets the window handle for the given form.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## FormActive

Checks if a form is currently active (displayed on the screen). This function is useful when forms are being displayed in asynchronous mode and another Cicode task is trying to access the form.

## Syntax

**FormActive(*hForm*)**

*hForm*:

The form handle, returned from the FormNew() function. The form handle identifies the table where all data on the associated form is stored.

## Return Value

TRUE (1) if the form is active or FALSE (0) if it is not.

## Related Functions

[FormDestroy](#), [FormNew](#)

## Example

See [FormDestroy](#).

## See Also

[Form Functions](#)

## FormAddList

Adds a text string to a list box or combo box. You should call this function only after the FormNew() function, and

immediately after either the FormComboBox() or the FormListBox(), and before the FormRead() function otherwise an error is returned. The text is added at the end of the list box or combo box.

To add text to a form that is already displayed, use the FormListAddText() function, and use the FormListSelectText() function to highlight text on the list.

## Syntax

**FormAddList(*sText*)**

*sText*:

The text string to add to the list box or combo box.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FormNew](#), [FormRead](#), [FormListBox](#), [FormComboBox](#), [FormListAddText](#), [FormListDeleteText](#), [FormListSelectText](#)

## Example

See [FormComboBox](#) and [FormListBox](#).

## See Also

[Form Functions](#)

## FormButton

Adds a button to the current form. You can add buttons that run callback functions (specified in *Fn*) to perform any actions you need, as well as the standard buttons - an [OK] button to save the operator's entries and close the form, and a [Cancel] button to close the form but discard the changes.

You should call this function only after the FormNew() function and before the FormRead() function. The button is added to the form at the specified column and row position. The width of the button is automatically sized to suit the text.

## Syntax

**FormButton(*Col*, *Row*, *sText*, *Fn*, *Mode*)**

*Col*:

The number of the column in which the button will be placed. Enter a number from 0 (column 1) to the form width - 1. For example, to place the button in column 8, enter 7.

*Row*:

The number of the row in which the button will be placed. Enter a number from 0 (row 1) to the form height - 1.

For example, to place the button in row 6, enter 5.

*sText:*

The text to display on the button.

*Fn:*

The callback function to call when the button is selected. Set to 0 to call no function. Please be aware that the Fn parameter needs to be of type INT and the callback function cannot contain a blocking function.

*Mode:*

Button mode:

0 - **Normal** button. When this button is selected the callback function is called.

1 - **OK** button. When this button is selected, the form is closed, and all operator-entered data is copied to buffers (specified by the other form functions). FormRead() returns 0 (zero) to indicate a successful read. The callback function specified in Fn is called. Be aware that this mode destroys the form.

2 - **Cancel** button. When this button is selected, the form is closed and operator-entered data is discarded. FormRead() returns with an error 299. The callback function specified in Fn is called. Be aware that this mode destroys the form.

3 - **Disabled** button. When this button is drawn, it will be shaded and not available for use.

## Return Value

The field handle if the button is successfully added, otherwise -1 is returned.

## Related Functions

[FormNew](#), [FormRead](#)

## Example

```
! Create a form, add buttons and then display the form on the
current page
FUNCTION FnMenu()
    FormNew("MENU",20,6,1);
    FormButton(0 ,4 , " FIND ", FindMenu, 0);
    FormButton(10,4 , " TAG ", ShowTag, 0);
    FormButton(0 ,5 , " CANCEL ", KillForm, 0);
    FormButton(10,5 , " GOTO ", GotoPg, 0);
    FormRead(0);
END
```

## See Also

[Form Functions](#)

## FormCheckBox

Adds a check box to the current form. The check box is a form control that allows the operator to make individual

selections. Each check box can be either checked (true) or cleared (false).

You should call this function only after the FormNew() function and before the FormRead() function. The check box is added to the form at the specified column and row position. The width of the button is automatically sized to suit the text.

## Syntax

**FormCheckBox(*Col*, *Row*, *sText*, *sBuf*)**

*Col*:

The number of the column in which the check box will be placed. Enter a number from 0 (column 1) to the form width - 1. For example, to place the check box in column 8, enter 7.

*Row*:

The number of the row in which the check box will be placed. Enter a number from 0 (row 1) to the form height - 1. For example, to place the check box in row 6, enter 5.

*sText*:

The text associated with the check box.

*sBuf*:

The string buffer in which to put the state of the check box. You should initialize this buffer to the state of the check box. When the form returns, this buffer will contain either '1' or '0' if the box is checked.

## Return Value

The field handle if the check box is successfully added, otherwise -1 is returned.

## Related Functions

[FormNew](#), [FormRead](#)

## Example

```
! Create a form, add check boxes, and display the form.  
! The operator may select none or all of the check boxes.  
FUNCTION FnMenu()  
    STRING sNuts, sCherrys, sChocolate;  
    sNuts = "1";  
    sCherrys = "0";  
    sChocolate = "1";  
    FormNew("IceCream",20,6,1); ]  
    FormCheckBox(2 ,2,"Nuts", sNuts);  
    FormCheckBox(2 ,3,"Cherrys", sCherrys);  
    FormCheckBox(2 ,4,"Chocolate", sChocolate);  
    FormRead(0);  
    If sNuts = "1" THEN  
        ! add the nuts  
    END  
    IF sCherrys = "1" THEN  
        ! add the cherrys
```

```
END
IF sChocolate = "1" THEN
    ! add the chocolate
END
END
```

## See Also

[Form Functions](#)

### FormComboBox

Adds a combo box to the current form. A combo box is a form control that allows the operator to type a selection or make a single selection from a text list.

You should call this function only after the FormNew() function and before the FormRead() function. The combo box is added to the form at the specified column and row position with the specified width and height. If more items are placed in the list than the list can display, a scroll bar displays (to scroll to the hidden items).

Use the FormAddList() function to add items for display in the list box. If the form is already displayed, you can use the FormListAddText() and FormListSelectText() functions to add (and highlight) text in the list box.

## Syntax

**FormComboBox(*Col*, *Row*, *Width*, *Height*, *sBuf* [, *Mode*] )**

*Col*:

The number of the column in which the combo box will be placed. Enter a number from 0 (column 1) to the form width - 1. For example, to place the combo box in column 8, enter 7.

*Row*:

The number of the row in which the combo box will be placed. Enter a number from 0 (row 1) to the form height - 1. For example, to place the combo box in row 6, enter 5.

*Width*:

The width of the list box, which should be wide enough to display your widest item. Items wider than the list box are clipped.

*Height*:

The height of the list box (the number of items that can be seen in the list box without scrolling).

*sBuf*:

The string buffer in which to store the selected item. The *sBuf* parameter can also hold the starting selection for the Combo box. For example if you set the *sBuf* string to "HELLO" before calling FormComboBox, HELLO will be displayed in the box upon opening the form. Must be a String type variable.

*Mode*:

The mode in which to create the combo box:

0 - Sort the combo box elements alphabetically.

1 - Place elements in combo box in the order they were added.

Default mode is 0.

## Return Value

The field handle if the combo box is successfully added, otherwise -1 is returned.

## Related Functions

[FormNew](#), [FormRead](#), [FormAddList](#), [FormListAddText](#), [FormListSelectText](#), [FormListBox](#)

## Example

```
! Create a form, add combo box and then display the form
! the operator may type in or select one of the items from the list
FUNCTION FnMenu()
    STRING sBuf;
    FormNew("Select Item",20,6,1);
    FormComboBox(2 ,2, 15, 5, sBuf, 1);
    FormAddList("Item One");
    FormAddList("Item two");
    FormAddList("Item three");
    FormRead(0);
    ! sBuf should contain the selected item or entered text
END
```

## See Also

[Form Functions](#)

## FormCurr

Gets the form and field handles for the current form and field. You should call this function only from within a callback function. You can then use the same callback function for all forms and fields, regardless of how the boxes, buttons, etc. on the forms are used. You should use this function with the FormGetInst() function.

## Syntax

**FormCurr(*hForm*, *hField*)**

*hForm*:

Variable containing the form handle, returned from the [FormNew\(\)](#) function. The form handle identifies the table where all data on the associated form is stored.

*hField*:

Variable containing the field handle of the field currently selected.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FormGetInst](#)

## Example

See [FormGetInst](#).

## See Also

[Form Functions](#)

## FormDestroy

Destroys a form that is removes it from the screen. Use this function (from an event) to close a form.

## Syntax

**FormDestroy(*hForm*)**

*hForm*:

The form handle, returned from the [FormNew\(\)](#) function. The form handle identifies the table where all data on the associated form is stored.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FormNew](#)

## Example

```
/* Display message to the operator. If after 10 seconds the
operator has not selected OK, then destroy the form. */
hForm=FormNew("Hello",4,20,0);
FormPrompt(1,1,"Something bad has happened");
FormButton(5,2,"OK",0,1);
FormRead(1);
! Wait 10 seconds.
Sleep(10);
IF FormActive(hForm) THEN
    ! Destroy form.
    FormDestroy(hForm);
END
```

## See Also

[Form Functions](#)

## FormEdit

Adds an edit field to the current form. You should call this function only after the FormNew() function and before the FormRead() function. A user input/edit box is added to the form at the specified column and row position. The operator can enter or edit the text in the edit box. This text is then passed to this function as *Text*.

## Syntax

**FormEdit**(*Col*, *Row*, *Text*, *Width*, *Height*, *bReadOnly* [, *maxTextLength*] )

*Col*:

The number of the column in which the edit field will be placed. Enter a number from 0 (column 1) to the form width - 1. For example, to place the edit field in column 8, enter 7.

*Row*:

The number of the row in which the edit field will be placed. Enter a number from 0 (row 1) to the form height - 1. For example, to place the edit field in row 6, enter 5.

*Text*:

Variable containing the text in the edit field. Text initially contains the default text (if any) for the operator to edit. When the function closes, this argument is passed back with the operator's input.

*Width*:

The width of the edit field.

*Height*:

The height of the edit field as measured using average character height. When value specified is less than 1 or not specified, it is default to 1 (single line). When multiple lines are specified, the content of the edit box will be word wrapped.

*bReadOnly*:

Flag to indicate whether the edit box is read only. Default is FALSE if not specified.

*maxTextLength*:

This optional parameter specifies the maximum length of input text. The default value is 0 meaning the string can have the maximum length allowed in the system (Cicode allows strings of 255 characters).

## Return Value

The field handle if the string is successfully added, otherwise -1 is returned.

## Related Functions

[FormNew](#), [FormRead](#)

## Example

```
STRING Recipe;
FormNew("Recipe",5,30,0);
! Add edit field, default Recipe to "Jam".
Recipe="Jam";
FormEdit(1,2,Recipe,20);
! Read the form.
FormRead(0);
! Recipe will now contain the operator-entered data.
```

## See Also

[Form Functions](#)

### FormField

Adds a field control device (such as a button , check box, or edit field) to the current form. You should call this function only after the FormNew() function and before the FormRead() function. This function allows you to specify a control device with more detail than the other field functions.

## Syntax

**FormField**(*Col*, *Row*, *Width*, *Height*, *Type*, *Buffer*, *Label*, *Fn* [, *maxTextLength*] )

*Col*:

The number of the column in which the control will be placed. Enter a number from 0 (column 1) to the form width - 1. For example, to place the control in column 8, enter 7.

*Row*:

The number of the row in which the control will be placed. Enter a number from 0 (row 1) to the form height - 1. For example, to place the control in row 6, enter 5.

*Width*:

The width of the control device.

*Height*:

The height of the control device.

*nType*:

The type of control device:

0 - None

1 - Edit

2 - Edit Password

3 - Text

4 - Button

5 - OK button

6 - Cancel button

7 - Group box

8 - Radio button

9 - Check box

*Buffer:*

The output buffer for the field string. The default value of the control device is initialized to the value of the buffer. If you specify a Radio button or Check box, you should initialize the buffer to "0" or "1". The result of the field will also be set to "0" or "1".

*Label:*

The display label for a button, or the default label for an edit field

*Fn:*

The callback function to call when the button is selected. Set to 0 to call no function. Please be aware that the Fn parameter needs to be of type INT, and the callback function cannot contain a blocking function. For types other than 4,5, and 6, set this argument to 0.

*maxTextLength:*

This optional parameter specifies the maximum length of input text for edit fields (this parameter is ignored for other controls). The default value is 0 meaning the string can have the maximum length allowed in the system (Cicode allows strings of 255 characters).

## Return Value

The field handle if the field is successfully added, otherwise it will return -1.

## Related Functions

[FormNew](#), [FormRead](#)

## Example

```
! Display a form with check boxes to start
!! specific motors.
FUNCTION SelectMotor()
    INT hform;
    STRING check1 = "0";
    STRING check2 = "0";
    STRING check3 = "0";
    hform = FormNew("Selection Menu", 26, 22, 6);
    FormField(16, 1, 12, 1, 9, check1, "Primary ", 0);
    FormField(16, 2, 12, 1, 9, check2, "Secondary", 0);
    FormField(16, 3, 12, 1, 9, check3, "backup ", 0);
    FormButton( 9, 20, " &Cancel ", 0, 2);
    IF FormRead(0) = 0 THEN
        IF check1 = "1" THEN
            StartMotor(MOTOR_1);
        END
        IF check2 = "1" THEN
            StartMotor(MOTOR_2);
        END
        IF check3 = "1" THEN
```

```
    StartMotor(MOTOR_3);
END
END
END
```

## See Also

[Form Functions](#)

### FormGetCurrInst

Extracts data associated with a field (set by the FormSetInst() function). You should call this function only from within a field callback function. This function is the same as calling the FormCurr() function and then the FormGetInst() function.

## Syntax

**FormGetCurrInst(*iData*, *sData*)**

*iData*:

Variable containing integer data.

*sData*:

Variable containing string data.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FormCurr](#), [FormGetInst](#), [FormSetInst](#)

## Example

```
INT
FUNCTION GetNextRec()
    INT hDev;
    STRING Str;
    FormGetCurrInst(hDev,Str);
    DevNext(hDev);
    RETURN 0;
END
```

## See Also

[Form Functions](#)

## FormGetData

Gets all data associated with a form and puts it into the output string buffers. Normally the field data is copied to the output string buffers only when the user selects the [OK] button. If you want to use the data while the form is displayed, call this function to get the data. You should call this function only while the form is displayed otherwise an error is returned, for example, from a field callback function.

## Syntax

**FormGetData(*hForm*)**

*hForm*:

The form handle, returned from the FormNew() function. The form handle identifies the table where all data on the associated form is stored.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FormCurr](#), [FormNew](#)

## Example

```
! Field callback to save data.  
FUNCTION Save()  
    INT hForm,hField;  
    FormCurr(hForm,hField);  
    FormGetData(hForm);  
    ! Access all data.  
    ..  
END
```

## See Also

[Form Functions](#)

## FormGetInst

Extracts the data associated with a field (set by the FormSetInst() function). You would normally use this function in a field callback function. It allows single callback functions to know that the form and field are associated.

## Syntax

**FormGetInst(*hForm*, *hField*, *iData*, *sData*)**

*hForm*:

The form handle, returned from the FormNew() function. The form handle identifies the table where all data on the associated form is stored.

*hField*:

The field handle of the field currently selected.

*iData*:

Integer data.

*sData*:

Variable containing string data.

## Return Value

The data (as a string).

## Related Functions

[FormSetInst](#), [FormCurr](#), [FormGetCurrInst](#), [FormNew](#)

## Example

```
INT
FUNCTION GetNextRec()
    INT hDev,hForm,hField;
    STRING Str;
    ! Get field data, for example, the hDev value.
    ..
    FormCurr(hForm,hField);
    FormGetInst(hForm,hField,hDev,Str);
    DevNext(hDev);
    ! Display new record in form.
    ..
    RETURN 0;
END
```

## See Also

[Form Functions](#)

## FormGetText

Gets the current text from a form field. You should call this function only while the form is displayed; for example,, from a field callback function.

## Syntax

**FormGetText(*hForm*, *hField*)**

*hForm*:

The form handle, returned from the FormNew() function. The form handle identifies the table where all data on the associated form is stored.

*hField:*

The field handle of the field currently selected.

## Return Value

The field text (as a string).

## Related Functions

[FormSetText](#), [FormNew](#)

## Example

```
FUNCTION Search()
    INT hForm,hField;
    STRING Recipe;
    FormCurr(hForm,hField);
    Recipe=FormGetText(hForm,hField);
    ! Go and find recipe.
    ..
END
```

## See Also

[Form Functions](#)

## FormGoto

Goes to a specified form. The form is displayed on top of all windows and the keyboard focus is set to this form.

## Syntax

**FormGoto(*hForm*)**

*hForm:*

The form handle, returned from the FormNew() function. The form handle identifies the table where all data on the associated form is stored.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FormNew](#)

## Example

```
FormGoto(hForm);
```

## See Also

[Form Functions](#)

## FormGroupBox

Adds a group box to the current form. A group box is a form control box drawn to the specified size. If the box contains radio buttons, they are grouped together. You should call this function only after the FormNew() function and before the FormRead() function.

The group box is added to the form at the specified column and row position with the specified width and height. Use the FormRadioButton() function to add the radio buttons to the box, and call this function between each group of radio buttons.

## Syntax

**FormGroupBox(*Col*, *Row*, *Width*, *Height* [, *Text*] )**

*Col*:

The number of the column in which the group box will be placed. Enter a number from 0 (column 1) to the form width - 1. For example, to place the group box in column 8, enter 7.

*Row*:

The number of the row in which the group box will be placed. Enter a number from 0 (row 1) to the form height - 1. For example, to place the group box in row 6, enter 5.

*Width*:

The width of the group box, which should be wide enough to display your widest item.

*Height*:

The height of the group box.

*Text*:

The text to display as the group box label.

## Return Value

The field handle if the group box is successfully added, otherwise -1 is returned.

## Related Functions

[FormNew](#), [FormRead](#), [FormRadioButton](#)

## Example

```
! Create a form, add to radio buttons groups and then display the
form
! The operator may select one of the radio buttons from each group
FUNCTION FnMenu()
    STRING sFast, sSlow, sMedium;
    STRING sNorth, sSouth, sEast, sWest;
    FormNew("Select Item",40,7,1);
    FormGroupBox(1 ,1, 15, 5, "Speed");
    FormRadioButton(2 ,2,"Fast", sFast);
    FormRadioButton(2 ,3,"Medium", sMedium);
    FormRadioButton(2 ,4,"Slow", sSlow);
    FormGroupBox(19 ,2, 15, 6, "Direction");
    FormRadioButton(20 ,2,"North", sNorth);
    FormRadioButton(20 ,3,"South", sSouth);
    FormRadioButton(20 ,4,"East", sEast);
    FormRadioButton(20 ,5,"West", sWest);
    FormRead(0);
END
```

## See Also

[Form Functions](#)

## FormInput

Adds a prompt and edit field to the current form. You should call this function only after the FormNew() function and before the FormRead() function. When FormRead() is called, the form will display with the prompt and edit box. The operator's input is passed back as a string (*Text*).

## Syntax

**FormInput(*Col*,*Row*,*Prompt*,*Text*,*Width* [, *maxTextLength*] )**

*Col*:

The number of the column in which the prompt will be placed. Enter a number from 0 (column 1) to the form width - 1. For example, to place the prompt in column 8, enter 7.

*Row*:

The number of the row in which the prompt will be placed. Enter a number from 0 (row 1) to the form height - 1. For example, to place the prompt in row 6, enter 5.

*Prompt*:

The prompt string.

*Text*: - Output Parameter

Variable containing the output text string containing the operator's input.

---

**Note:** Only Cicode variables can be used in output parameters, variable or local tags cannot be used and will result in a compiler error if attempted.

*Width*:

The width of the edit field.

*maxTextLength:*

This optional parameter specifies the maximum length of input text. The default value is 0 meaning the string can have the maximum length allowed in the system (Cicode allows strings of 255 characters).

## Return Value

The field handle if it is added successfully, otherwise -1 is returned.

## Related Functions

[FormNew](#), [FormRead](#)

## Example

```
FormInput(1,2,"Recipe",Recipe,20);
```

## See Also

[Form Functions](#)

## FormListAddText

Adds a new text entry to a combo box or a list box while the form is displayed. It only adds the text to the list - it does not select it. Use the [FormListSelectText\(\)](#) function to select (highlight) an entry. Call this function only when the form is displayed, for example, from a field callback function.

## Syntax

**FormListAddText(*hForm*, *hField*, *Text*)**

*hForm:*

The form handle, returned from the [FormNew\(\)](#) function. The form handle identifies the table where all data on the associated form is stored.

*hField:*

The field handle of the field currently selected.

*Text:*

The output text string containing the operator's input.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FormListSelectText](#), [FormListDeleteText](#), [FormSetText](#), [FormNew](#)

## Example

```
/* create a form with a list */
hForm = FormNew("Ingredients", 40, 10, 1);
hField = FormListBox(2,2,20,5,sBuf);
FormAddList("Flour");
FormAddList("Water");
FormAddList("Salt");
FormAddList("Sugar");
/* Display the form */
FormRead(1);
..
/*Add Milk to list */
FormListAddText(hForm, hField, "Milk");
..
```

## See Also

[Form Functions](#)

## FormListBox

Adds a list box to the current form. The list box is a form control that allows the operator to select from a list of items. You should call this function only after the FormNew() function and before the FormRead() function.

The list box is added to the form at the specified column and row position with the specified width and height. If more items are placed in the list than the list can display, a scroll bar displays for scrolling to the hidden items.

Use the FormAddList() function to add items for display in the list box. If the form is already displayed, you can use the FormListAddText() and FormListSelectText() functions to add (and highlight) text in the list box.

## Syntax

**FormListBox(*Col*, *Row*, *Width*, *Height*, *sBuf* [, *Mode*] )**

*Col*:

The number of the column in which the list box will be placed. Enter a number from 0 (column 1) to the form width - 1. For example, to place the list box in column 8, enter 7.

*Row*:

The number of the row in which the list box will be placed. Enter a number from 0 (row 1) to the form height - 1. For example, to place the list box in row 6, enter 5.

*Width*:

The width of the list box, in characters. Width should be wide enough to display your widest item. Items wider than the list box are clipped.

*Height*:

The height of the list box, as the number of items that can be seen in the list box without scrolling.

*sBuf:*

The string buffer in which to store the selected item. Must be a String type variable.

*Mode:*

The mode in which to create the list box:

0 - Sort the list box elements alphabetically.

1 - Place elements in list box in the order they were added.

Mode 0 is the default.

## Return Value

The field handle if the list box is successfully added, otherwise -1 is returned.

## Related Functions

, [FormRead](#), [FormAddList](#), [FormListAddText](#), [FormListSelectText](#), [FormComboBox](#)

## Example

```
! Create a form, add list box and then display the form.  
! The operator may select one of the items from the list.  
STRING sBuf;  
FUNCTION FnMenu()  
    FormNew("Select Item",20,6,1);  
    FormListBox(2,2,15,5,sBuf,1);  
    FormAddList("Item One");  
    FormAddList("Item two");  
    FormAddList("Item three");  
    FormButton(0,0,"OK",0,1);  
    FormButton(5,0,"CANCEL",0,2);  
    FormRead(0);  
    SELECTION= sBuf;  
END
```

## See Also

[Form Functions](#)

## FormListDeleteText

Deletes an existing text entry from a combo box or a list box while the form is displayed. It only deletes the text from the list - it does not change the selection. Call this function only when the form is displayed, for example, from a field callback function.

## Syntax

**FormListDeleteText(*hForm*, *hField*, *Text*)**

*hForm*:

The form handle, returned from the FormNew() function. The form handle identifies the table where all data on the associated form is stored.

*hField*:

The field handle of the field currently selected.

*Text*:

The text to delete.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FormListSelectText](#), [FormListAddText](#), [FormNew](#)

## Example

```
/* create a form with a list */  
hForm = FormNew("Ingredients", 40, 10, 1);  
hField = FormListBox(2,2,20,5,sBuf);  
FormAddList("Flour");  
FormAddList("Water");  
FormAddList("Salt");  
FormAddList("Sugar");  
/* Display the form */  
FormRead(1);  
..  
/*Remove Sugar from the list */  
FormListDeleteText(hForm, hField, "Sugar");  
..
```

## See Also

[Form Functions](#)

## FormListSelectText

Selects (highlights) a text entry in a Combo box or a List box while the form is displayed. The text to be selected needs to exist in the list. (Use the FormListAddText() function to add a text entry to a list.) Call this function only when the form is displayed, for example, from a field callback function.

## Syntax

**FormListSelectText(*hForm*, *hField*, *Text*)**

*hForm*:

The form handle, returned from the FormNew() function. The form handle identifies the table where all data on the associated form is stored.

*hField*:

The field handle of the field currently selected.

*Text*:

The text to be selected. If this text is not present in the list, then no item will be selected (and this text will not be added).

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FormListAddText](#), [FormSetText](#), [FormNew](#)

## Example

```
/* Create a form with a list */  
hForm = FormNew("Ingredients", 40, 10, 1);  
hField = FormListBox(2,2,20,5,sBuf);  
FormAddList("Flour");  
FormAddList("Water");  
FormAddList("Salt");  
FormAddList("Sugar");  
/* Display the form */  
FormRead(1);  
..  
/*Select Flour */  
FormListSelectText(hForm, hField, "Flour");
```

## See Also

[Form Functions](#)

## FormNew

Creates a new data entry form and defines its size and mode. After the form is created, you can add fields, and then display the form.

Before you can display a form on the screen, you need to call this function to set the size and mode of the form, and then call the various form field functions, FormInput(), FormButton(), FormEdit() etc to add user input fields to the form. To display the form on the screen (to allow the user to enter data) call the FormRead() function.

## Syntax

**FormNew(*Title*, *Width*, *Height*, *Mode*)**

*Title*:

The title of the form.

*Width*:

The character width of the form (1 to 131).

*Height*:

The character height of the form (1 to 131).

*Mode*:

The mode of the form:

0 - Default font and text spacing

1 - Small font

2 - Fixed pitch font

4 - Static text compression where the vertical spacing is reduced. This can cause formatting errors if buttons are too close, because the vertical spacing will be less than the height of a button.

8 - Keep the form on top of the Plant SCADA window.

16 - The current window cannot be changed or closed until the form is finished or cancelled.

32 - Makes a form with no caption.

128 - The form will not close if the **ESC** or **ENTER** key is pressed, unless you specifically define at least one button on the form which acts as an **OK** or **Cancel** button. For a form with no buttons, the **ENTER** key normally closes the form; this mode disables that behavior.

256 – Makes a from with no system-menu (mostly appears as a single close button X) .

Multiple modes can be selected by adding them (for example, to use Modes 4 and 2, specify Mode 6).

## Return Value

The form handle if the form is created successfully, otherwise -1 is returned. The form handle identifies the table where all data on the associated form is stored.

## Related Functions

[FormDestroy](#), [FormInput](#), [FormButton](#), [FormEdit](#), [FormRead](#)

## Example

```
FormNew("Recipe",30,5,0);
FormInput(1,1,"Recipe No",Recipe,20);
FormInput(1,2,"Amount",Amount,10);
FormRead(0);
```

## See Also

[Form Functions](#)

### FormNumPad

Provides a keypad form for the operator to add numeric values. You can customize the standard form as a mathematical keypad, with the +, -, and / operators and the decimal point. For a time keypad, use the AM, PM, and : (hour/minute divider) buttons. You can also include a password edit field.

## Syntax

**FormNumPad(*Title*, *Input*, *Mode*)**

*Title*:

The title to display on the number pad form.

*Input*:

The existing or default value. This value is returned if the form is cancelled or accepted without changes.

*Mode*:

The buttons to include on the keypad form. The Mode can be a combination of the following:

0 - Standard keypad

1 - Password edit field

2 - not used

4 - With +/- button

8 - With / button

16 - With . button

32 - With : button

64 - With AM, PM buttons

128 - with Now button

512 - with 1hr, 2hr and 8hr buttons

Multiple modes can be selected by adding them. For example, to include +/- buttons and a . button, specify Mode 20 (16+4).

**Note:** Modes 128 and 512 can not be used together on one FormNumPad.

## Return Value

The string value entered by the operator. The IsError() function returns 0 (zero). If the form was cancelled, the value of *Input* is returned, and the IsError() function returns error number 299.

## Example

```
/* Set defaults first, then four keypad forms to adjust recipe. */  
Qty_Flour=FormNumPad("Add Flour", Qty_Flour, 17);
```

```
Qty_Water=FormNumPad("Add Water", Qty_Water, 17);
Qty_Salt=FormNumPad("Add Salt", Qty_Salt, 17);
Qty_Sugar=FormNumPad("Add Sugar", Qty_Sugar, 17);
```

## See Also

[Form Functions](#)

### FormOpenFile

Displays a File Open dialog box.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**FormOpenFile(*sTitle*, *sFileName*, *sFilter*, [*bResetToDefault*])**

*sFileName*:

The name of the default file to display in the "File Name" field.

*sTitle*:

A title to display at the top of the form.

*sFilter*:

A file filter list to display in the "List Files of Type" field. The file filter list has the following format:

<File Type>|<Filter>|

Where:

File Type is the text that displays in the drop-down box, for example All Files (\*.\*). Filter is the file type, for example \*.Cl

*bResetToDefault*

Boolean parameter. When set forces the path to reset to the default path specified.

True = default path specified

False = last file specified

For example, FormOpenFile(*sTitle*, *sFileName*, *sFilter* [, *bResetToDefault* = FALSE])

## Return Value

The name and full path of the selected file (as a string) or an empty string ("") if the Cancel button is selected.

## Related Functions

[FormSaveAsFile](#), [FormSelectPrinter](#)

## Example

```
// Display the Open File dialog with the following filter list:
```

```
// All Files (*.*)  
// Exe Files (*.EXE)  
// Cicode Files (*.CI)  
sFilename = FormOpenFile("Open", "*.CI", "All Files (*.*)|*.|Exe  
Files (*.EXE)|*.EXE|Cicode Files (*.CI)|*.CI|");
```

## See Also

[Form Functions](#)

### FormPassword

Adds both a password prompt and edit field to the current form. You should call this function only after the FormNew() function and before the FormRead() function. When FormRead() is called, the form will also display the password prompt and edit field.

The operator's input is not echoed in the field; a single asterisk (\*) is displayed for each character.

## Syntax

**FormPassword**(*Col*, *Row*, *Prompt*, *Password*, *Width*)

*Col*:

The number of the column in which the prompt will be placed. Enter a number from 0 (column 1) to the form width - 1. For example, to place the prompt in column 8, enter 7.

*Row*:

The number of the row in which the prompt will be placed. Enter a number from 0 (row 1) to the form height - 1. For example, to place the prompt in row 6, enter 5.

*Prompt*:

The prompt string.

*Password*:

Variable containing the password entered by the operator.

*Width*:

The width of the edit field.

## Return Value

The field handle if it is added successfully, otherwise -1 is returned.

## Related Functions

[FormEdit](#), [FormNew](#), [FormRead](#)

## Example

```
! Add Password input.
```

```
FormPassword(1,2,"Enter Password",Password,10);
```

## See Also

[Form Functions](#)

## FormPosition

Sets the position of a form on the screen, before it is displayed. You should call this function only after the FormNew() function and before the FormRead() function.

## Syntax

**FormPosition(*X, Y, Mode*)**

*X, Y:*

The x and y pixel coordinates of the form.

*Mode:*

Not used, set it to 0.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FormNew](#), [FormRead](#)

## Example

```
hForm = FormNew("title", 20, 5, 0);
! display form at x=100, y=50
FormPosition(100, 50, 0);
```

## See Also

[Form Functions](#)

## FormPrompt

Adds a prompt field to the current form. You should call this function only after the FormNew() function and before the FormRead() function.

## Syntax

**FormPrompt(*Col, Row, Prompt [,Width] [,Height]*)**

*Col:*

The number of the column in which the prompt will be placed. Enter a number from 0 (column 1) to the form width - 1. For example, to place the prompt in column 8, enter 7.

*Row:*

The number of the row in which the prompt will be placed. Enter a number from 0 (row 1) to the form height - 1. For example, to place the prompt in row 6, enter 5.

*Prompt:*

The prompt string.

*Width:*

The width of the area that the prompt string will wrap within (measured using average character width). For this setting to be applied, you also need to specify a value for Height.

*Height:*

The height of the area that the prompt string will display within (measured using average character height). For this setting to be applied, you also need to specify a value for Width.

## Return Value

The field handle if it is added successfully, otherwise -1 is returned.

## Related Functions

[FormNew](#), [FormRead](#)

## Example

```
FormPrompt(1,2,"Enter Recipe");
```

## See Also

[Form Functions](#)

## FormRadioButton

Adds a radio button to the current form, allowing the operator to make a selection from a multiple choice list. You should call this function only after the FormNew() function and before the FormRead() function.

The radio button is added to the form at the specified column and row position. The width of the button will be sized to suit the text.

By default, all radio buttons are placed into the one group. If you require separate groups, use this function in conjunction with the FormGroupBox() function.

## Syntax

**FormRadioButton(*Col*, *Row*, *sText*, *sBuf*)**

*Col:*

The number of the column in which the button will be placed. Enter a number from 0 (column 1) to the form width - 1. For example, to place the button in column 8, enter 7.

*Row:*

The number of the row in which the button will be placed. Enter a number from 0 (row 1) to the form height - 1. For example, to place the button in row 6, enter 5.

*sText:*

The text associated with the radio button.

*sBuf:*

The string buffer in which to put the state of the radio button. You should initialize this buffer to the state of the button. When the form returns, this buffer will contain either '1' or '0' if the radio button is checked.

## Return Value

The field handle if the radio button is successfully added, otherwise -1 is returned.

## Related Functions

[FormNew](#), [FormRead](#), [FormGroupBox](#), [FormCheckBox](#)

## Example

```
! Create a form, add radio buttons and then display the form.  
! The operator may only select one radio button , either Fast,  
Medium or Slow  
FUNCTION FnMenu()  
    STRING sFast, sSlow, sMedium;  
    sFast = "1";  
    sMedium = "0";  
    sSlow = "0";  
    FormNew("Speed",20,6,1);  
    FormRadioButton(2 ,2,"Fast", sFast);  
    FormRadioButton(2, 3,"Medium", sMedium);  
    FormRadioButton(2 ,4,"Slow", sSlow);  
    FormRead(0);  
    If sFast = "1" THEN  
        ! do fast stuff  
    ELSE  
        IF sMedium = "1" THEN  
            ! do Medium stuff  
        ELSE  
            IF sSlow = "1" THEN  
                ! do slow stuff  
            END  
    END  
END
```

## See Also

[Form Functions](#)

### FormRead

Displays the current form (created with the `FormNew()` function), with all the fields that were added (with the form field functions).

You can display the form and wait for the user to finish entering data by setting the `Mode` to 0. This mode is the most commonly used, with [OK] and [Cancel] buttons to either save or discard operator entries and to close the form.

To display the form and return before the user has finished, use Mode 1. This mode is used to animate the data on the form or to perform more complex operations.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

## Syntax

`FormRead(Mode)`

*Mode*:

Mode of the form:

0 - Wait for the user.

1 - Do not wait for the user.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FormNew](#)

## Example

```
! Display the form and wait for the user.  
FormRead(0);  
! Display the form and do not wait for the user.  
FormRead(1);  
! While the form is displayed, update the time every second.  
WHILE FormActive(hForm) DO  
    FormSetText(hForm,hField,Time());  
    Sleep(1);  
END
```

## See Also

[Form Functions](#)

### FormSaveAsFile

Displays a File Save As dialog box.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**FormSaveAsFile(*sTitle*, *sFileName*, *sFilter* [, *sDefExt*] )**

*sTitle*:

A title to display at the top of the form.

*sFileName*:

The name of the default file to display in the "File Name" field.

*sFilter*:

A file filter list to display in the "List Files of Type" field. The file filter list has the following format:

<File Type>|<Filter>|

where:

File Type is the text that displays in the drop-down box, for example All Files (\*.\*). Filter is the file type, for example \*.CI

*sDefExt*:

The file extension that will be used as a default when you use the FormSaveAsFile() function. If you do not specify a default extension, files will be saved without an extension.

## Return Value

The name and full path of the selected file (as a string) or an empty string ("") if the Cancel button is selected.

## Related Functions

[FormOpenFile](#), [FormSelectPrinter](#), [FormSaveAsFile](#)

## Example

```
// Display the SaveAs dialog with the following filter list:  
// All Files (*.*)  
// Exe Files (*.EXE)  
// Cicode Files (*.CI)  
sFilename = FormSaveAsFile("Save As", "Alarms", "All Files  
(*.*)|*.EXE|Cicode Files (*.CI)|*.CI|",  
"ci");
```

## See Also

[Form Functions](#)

### FormSecurePassword

Adds both a password prompt and edit field to the current form. You should call this function only after the FormNew() function and before the FormRead() function. When FormRead() is called, the form will also display the password prompt and edit field.

The operator's input is not echoed in the field; a single asterisk (\*) is displayed for each character. The function does not return the password as a plain text, it returns an encrypted password string. The user can send this string to the UserLogin or UserVerify functions.

## Syntax

**FormSecurePassword(*Col*, *Row*, *Prompt*, *Password*, *Width*)**

*Col*:

The number of the column in which the prompt will be placed. Enter a number from 0 (column 1) to the form width - 1. For example, to place the prompt in column 8, enter 7.

*Row*:

The number of the row in which the prompt will be placed. Enter a number from 0 (row 1) to the form height - 1. For example, to place the prompt in row 6, enter 5.

*Prompt*:

The prompt string. Must be a String type variable.

*Password*:

The encrypted password entered by the operator.

*Width*:

The width of the edit field.

## Return Value

The field handle if it is added successfully, otherwise -1 is returned.

## Related Functions

[FormEdit](#), [FormRead](#), [UserLogin](#), [UserVerify](#)

## Example

```
! Add Password input.  
FormSecurePassword(1,2,"Enter Password",Password,10);
```

## See Also

[Form Functions](#)

### FormSelectPrinter

Displays the Select Printer dialog box.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**FormSelectPrinter()**

## Return Value

The name of the selected printer (as a string) or an empty string ("") if the Cancel button is selected.

## Related Functions

[FormOpenFile](#), [FormSaveAsFile](#)

## Example

```
// Display the Select Printer dialog
sPrinter = FormSelectPrinter();
```

## See Also

[Form Functions](#)

### FormSetData

Sets all the edit data from the string buffers into the form. You should call this function only while the form is active.

## Syntax

**FormSetData(*hForm*)**

*hForm*:

The form handle, returned from the FormNew() function. The form handle identifies the table where all data on the associated form is stored.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FormGetData](#), [FormNew](#)

## Example

```
INT
FUNCTION MyNextRec(
    INT hForm,hField;
    FormCurr(hForm,hField);
    FormSetData(hForm);
    RETURN 0;
END
```

## See Also

[Form Functions](#)

## FormSetInst

Associates an integer and string value with each field on a form. This data could then be used by a callback function. You can use a single callback function for all fields, and use the data to perform different operations for each field.

## Syntax

**FormSetInst(*hForm*, *hField*, *iData*, *sData*)**

*hForm*:

The form handle, returned from the `FormNew()` function. The form handle identifies the table where all data on the associated form is stored.

*hField*:

The field handle of the field currently selected.

*iData*:

Integer data.

*sData*:

String data.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FormGetInst](#), [FormNew](#)

## Example

```
! Open recipe database.  
hDev=DevOpen("Recipe", 0);  
hForm=FormNew("Recipe",20,5,0);  
hField=FormButton(5,2,"Next",GetNextRec,0);  
FormSetInst(hForm,hField,hDev,"");  
/* The device handle hDev is put into the next button , so when the  
button is selected it can get hDev and get the next record. */
```

## See Also

[Form Functions](#)

## FormSetText

Sets new field text on a field. This function allows you to change field text while the form is displayed. Call this function only when the form is displayed, for example, from a field callback function.

If you are using this function on a Combo box or a List box, it will select the text from the Combo box or List box list. If no text exists in the Combo box or List box list, the function will add it.

## Syntax

**FormSetText(*hForm*, *hField*, *Text*)**

*hForm*:

The form handle, returned from the FormNew() function. The form handle identifies the table where all data on the associated form is stored.

*hField*:

The field handle of the field currently selected. If the *hField* is a handle to the secure edit field created with FormSecurePassword, the text in the secure edit field will not be changed. However, when an empty string is passed to FormSetText(), the contents of the secure edit field will be cleared.

*Text*:

New field text.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FormCurr](#), [FormListSelectText](#), [FormListAddText](#), [FormNew](#), [FormSecurePassword](#), [FormSetText](#)

## Example

```
/* Create a form with a field */
hForm = FormNew("Ingredients", 40, 10, 1);
hField = FormPrompt(2,2,"Motor1:");
/* Display the form*/
FormRead(1);
..
/* Change the text in the field */
FormSetText(hForm, hField, "Pump1:");
..
```

## See Also

[Form Functions](#)

### FormWndHnd

Gets the window handle for the given form. The window handle may be used by 'C' programs and Plant SCADA *Wn...* functions. You should call this function only after the *FormRead()* function.

The window handle is not the same as the Plant SCADA window number and cannot be used with functions that expect the Plant SCADA window number (the *Win...* functions).

## Syntax

**FormWndHnd(*hForm*)**

*hForm*:

The form handle, returned from the *FormNew()* function. The form handle identifies the table where data on the associated form is stored.

## Return Value

The window handle if successful, otherwise a 0 is returned.

## Related Functions

[FormNew](#), [FormRead](#), [WndFind](#)

## Example

```
/* Create a form with a field */
hForm = FormNew("Ingredients", 40, 10, 1);
hField = FormPrompt(2,2,"Motor1:");
/* Display the form*/
FormRead(1);
/* Get the form's window number for future reference */
hWnd = FormWndHnd(hForm);
```

## See Also

[Form Functions](#)

## Format Functions

Following are functions used for formatting data:

<a href="#">FmtClose</a>	Closes a format template.
<a href="#">FmtFieldHnd</a>	Gets the handle of a field in a format template.
<a href="#">FmtGetField</a>	Gets field data from a format template.
<a href="#">FmtGetFieldCount</a>	Retrieves the number of fields in a format object.
<a href="#">FmtGetFieldHnd</a>	Gets field data from a format template using a field handle.
<a href="#">FmtGetFieldName</a>	Retrieves the name of a particular field in a format object.
<a href="#">FmtGetWidth</a>	Retrieves the width of a particular field in a format object.
<a href="#">FmtOpen</a>	Creates a new format template.
<a href="#">FmtSetField</a>	Sets data in a field of a format template.
<a href="#">FmtSetFieldHnd</a>	Sets data in a field of a format template using a field handle.
<a href="#">FmtToStr</a>	Converts format template data to a string

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## FmtClose

Closes a format template. After it is closed, the template cannot be used. Closing the template releases system memory.

## Syntax

**FmtClose(*hFmt*)**

*hFmt*:

The format template handle, returned from the FmtOpen() function. The handle identifies the table where all data on the associated format template is stored.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FmtOpen](#)

## Example

```
FmtClose(hFmt);
```

## See Also

[Format Functions](#)

## FmtFieldHnd

Gets the handle of a field in a format template. You can then use the field handle in the FmtGetFieldHnd() and FmtSetFieldHnd() functions. By using a handle, you only need to resolve the field name once and then call other functions as required (resulting in improved performance.)

## Syntax

**FmtFieldHnd(*hFmt, Name*)**

*hFmt:*

The format template handle, returned from the FmtOpen() function. The handle identifies the table where all data on the associated format template is stored.

*sName:*

The field name.

## Return Value

The handle of the format template field, or -1 if the field cannot be found.

## Related Functions

[FmtGetFieldHnd](#), [FmtSetFieldHnd](#), [FmtOpen](#)

## Example

```
!Resolve names at startup.  
hName=FmtFieldHnd(hFmt,"Name");  
hDesc=FmtFieldHnd(hFmt,"Desc");  
!Set field data.  
FmtSetFieldHnd(hFmt,hName,"CV101");
```

## See Also

[Format Functions](#)

### FmtGetField

Gets field data from a format template. Use this function to extract data after a call to StrToFmt().

## Syntax

**FmtGetField(*hFmt, sName*)**

*hFmt*:

The format template handle, returned from the FmtOpen() function. The handle identifies the table where data on the associated format template is stored.

*sName*:

The field name.

## Return Value

The data (as a string). If the field does not contain any data, an empty string will be returned.

## Related Functions

[StrToFmt](#), [FmtOpen](#), [FmtSetField](#), [FmtToStr](#)

## Example

```
StrToFmt(hFmt,"CV101 Raw Coal Conveyor");  
Str=FmtGetField(hFmt,"Name");  
! Str will contain "CV101".
```

## See Also

[Format Functions](#)

## FmtGetFieldCount

Retrieves the number of fields in a format object.

## Syntax

**FmtGetFieldCount(*hFmt*)**

*hFmt*:

The format template handle, returned from the FmtOpen() function. The handle identifies the table where data on the associated format template is stored.

## Return Value

Number of fields in the specified format.

## Related Functions

[FmtGetField](#), [FmtOpen](#)

## See Also

[Format Functions](#)

## FmtGetFieldHnd

Gets field data from a format template. Use this function to extract data after a call to StrToFmt(). This function has the same effect as FmtGetField(), except that you use a field handle instead of the field name.

## Syntax

**FmtGetFieldHnd(*hFmt*, *hField*)**

*hFmt*:

The format template handle, returned from the FmtOpen() function. The handle identifies the table where all data on the associated format template is stored.

*hField*:

The field handle.

## Return Value

The data (as a string). If the field does not contain any data, an empty string will be returned.

## Related Functions

[FmtGetField](#), [StrToFmt](#), [FmtOpen](#), [FmtFieldHnd](#)

## Example

```
StrToFmt(hFmt,"CV101 Raw Coal Conveyor");
hField=FmtFieldHnd(hFmt,"Name");
Str=FmtGetField(hFmt,hField);
! Str will contain "CV101".
```

## See Also

[Format Functions](#)

### FmtGetFieldName

Retrieves the name of a particular field in a format object.

## Syntax

**FmtGetFieldName(*hFmt*, *hField*)**

*hFmt*:

The format template handle, returned from the FmtOpen() function. The handle identifies the table where data on the associated format template is stored.

*hField*:

The field handle.

## Return Value

Name of requested field

## Related Functions

[FmtGetField](#), [FmtOpen](#)

## See Also

[Format Functions](#)

### FmtGetWidth

Retrieves the width of a particular field in a format object.

## Syntax

**FmtGetWidth(*hFmt*, *hField*)**

*hFmt*:

The handle to a format object. The handle identifies the table where data on the associated format template is stored.

*hField*:

The field handle.

## Return Value

Width of the requested field.

## Related Functions

[FmtGetField](#), [FmtGetFieldName](#), [FmtOpen](#)

## See Also

[Format Functions](#)

## FmtOpen

Creates a format template. After you create a template, you can use it for formatting data into strings or extracting data from a string. To format a template, use the same syntax as a device format, that is {<name>[,width[,justification]]}. The maximum number of formats that can be created in a project is 512. If you attempt to create formats in excess of this number, an error code (code 258: CT\_ERROR\_BUFFER\_OVERRUN) will be generated.

**Note:** This function should be concluded with the FmtClose function, otherwise the same error code (code 258: CT\_ERROR\_BUFFER\_OVERRUN) will be generated.

## Syntax

INT **FmtOpen**(STRING *Name*, STRING *Format*, INT *Mode*)

*Name*:

The name of the format template or Alarm Category.

*Format*:

The format of the template, as {<name>[,width[,justification]]}. Not used for alarm format. See [Format Templates](#) for more information.

*Mode*:

The mode of the open:

0 - Open the existing format.

1 - Open a new format.

2 - Open Summary Format from Alarm Category specified by Name.

3 - Open Display Format from Alarm Category specified by Name.

4 – Open SOE format from Alarm Category specified by Name.

5 – Reopen a format. If the format name already exists, it will reopen with the specified format template.

Otherwise, a new format will open.

## Return Value

The format template handle, or -1 if the format cannot be created.

## Related Functions

[FmtClose](#)

## Examples

```
hFmt=FmtOpen("MyFormat", "{Name}{Desc,20}",0);
FmtSetField(hFmt, "Name", "CV101");
FmtSetField(hFmt, "Desc", "Raw Coal Conveyor");
Str =FmtToStr(hFmt);
! Str will contain "CV101 Raw Coal Conveyor".
FmtOpen("0", "", 2);
! Display Format from Alarm Category 0
FmtOpen("0", "", 3);
! Summary Format from Alarm Category 0.
```

## See Also

[Format Functions](#)

## FmtSetField

Sets data in a field of a format template. After you have set all the fields, you can build the formatted string with the FmtToStr() function.

## Syntax

**FmtSetField(*hFmt*, *Name*, *Data*)**

*hFmt*:

The format template handle, returned from the FmtOpen() function. The handle identifies the table where all data on the associated format template is stored.

*sName*:

The name of the format template.

*Data*:

Field data.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FmtGetField](#), [FmtToStr](#), [FmtOpen](#)

## Example

```
hFmt=FmtOpen("MyFormat","{Name}{Desc, 20}",0);
FmtSetField(hFmt,"Name", "CV101");
FmtSetField(hFmt,"Desc", "Raw Coal Conveyor");
Str =FmtToStr(hFmt);
! Str will contain "CV101 Raw Coal Conveyor".
```

## See Also

[Format Functions](#)

## FmtSetFieldHnd

The fields you can build the formatted string with the FmtToStr() function. This function has the same effect as FmtSetField() except that you use a field handle instead of the field name.

## Syntax

**FmtSetFieldHnd(*hFmt*, *hField*, *Data*)**

*hFmt*:

The format template handle, returned from the FmtOpen() function. The handle identifies the table where all data on the associated format template is stored.

*hField*:

The field handle.

*Data*:

Field data.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FmtFieldHnd](#), [FmtToStr](#), [FmtSetField](#), [FmtOpen](#)

## Example

```
hField=FmtFieldHnd(hFmt, "Name");
FmtSetFieldHnd(hFmt,hField, "CV101");
```

## See Also

[Format Functions](#)

### FmtToStr

Builds a formatted string from the current field data (in a format template).

## Syntax

**FmtToStr(*hFmt*)**

*hFmt*:

The format template handle, returned from the FmtOpen() function. The handle identifies the table where all data on the associated format template is stored.

## Return Value

The formatted string as defined in the format description.

## Related Functions

[StrToFmt](#), [FmtOpen](#)

## Example

```
! Get the formatted string.  
Str=FmtToStr(hFmt);
```

## See Also

[Format Functions](#)

## Fuzzy Logic Functions

The following functions relate to fuzzy logic control.

<a href="#">FuzzyClose</a>	Closes specified fuzzy instance.
<a href="#">FuzzyGetCodeValue</a>	Gets a specified Code variable from the specified instance.
<a href="#">FuzzyGetShellValue</a>	Gets a specified Shell variable from the specified instance.
<a href="#">FuzzyOpen</a>	Creates a new fuzzy instance.

<a href="#">FuzzySetCodeValue</a>	Sets a specified Code variable in the specified instance.
<a href="#">FuzzySetShellValue</a>	Sets a specified Shell variable in the specified instance.
<a href="#">FuzzyTrace</a>	Controls the tracing.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## FuzzyClose

Frees all memory and information for the specified instance. After the fuzzy instance is closed, the handle given in the *hFuzzy* parameter is no longer valid.

## Syntax

**FuzzyClose(*hFuzzy*)**

*hFuzzy*:

The fuzzy instance handle (and integer greater than 0).

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FuzzyOpen](#)

## Example

See [FuzzyOpen](#).

## See Also

[Fuzzy Logic Functions](#)

## FuzzyGetCodeValue

Gets a value for the specified input of the specified instance.

## Syntax

**FuzzyGetCodeValue(*hFuzzy*, *iIOIndex*, *NoHitFlag*)**

*hFuzzy*:

The fuzzy instance handle (and integer greater than 0).

*iIOIndex*:

Specifies the variable to receive the value. The I/O-Indices start with 0 and increment by 1 for each variable. To find the correct index for a specific variable, the variables need to be sorted in alpha-numerical order, first the inputs and then the outputs.

*NoHitFlag*:

Variable to receive the new value of the No-hit-flag. The No- hit-flag is TRUE if no rule was active for the variable specified by *iIOIndex*, otherwise it is FALSE. Must be an Integer type variable - it cannot be a constant or PLC variable tag.

## Return Value

The code value if the function was successful, otherwise -1. Use *IsError()* to find the error number if the function does not succeed.

## Related Functions

[FuzzyOpen](#), [FuzzySetCodeValue](#)

## Example

See [FuzzyOpen](#).

## See Also

[Fuzzy Logic Functions](#)

## FuzzyGetShellValue

Gets a value for the specified input of the specified instance. The variables in the instance needs to have the data type REAL (floating point values).

## Syntax

**FuzzyGetShellValue(*hFuzzy*, *iIOIndex*, *NoHitFlag*)**

*hFuzzy*:

The fuzzy instance handle (and integer greater than 0).

*iIOIndex*:

Specifies the variable to receive the value. The I/O-Indices start with 0 and increment by 1 for each variable. To

find the correct index for a specific variable, the variables need to be sorted in alpha-numerical order, first the inputs and then the outputs.

*NoHitFlag*:

Variable to receive the new value of the No-hit-flag. The No- hit-flag is TRUE if no rule was active for the variable specified by *iIOIndex*, otherwise it is FALSE. Must be an Integer type variable - it cannot be a constant or PLC variable tag.

## Return Value

The shell value if the function was successful. Use [IsError](#) to find the error number if the function does not succeed.

## Related Functions

[FuzzyOpen](#), [FuzzySetShellValue](#)

## Example

See [FuzzyOpen](#).

## See Also

[Fuzzy Logic Functions](#)

## FuzzyOpen

This function loads a \*.FTR file, allocates memory and creates a handle for this fuzzy instance. To use the FuzzyTech functions you need to be a registered user of one or more of the following fuzzyTech products: *fuzzyTECH Online Edition*, *fuzzyTECH Precompiler Edition*, or *fuzzyTECH for Business PlusC*. And you need to only use *fuzzyTECH* to generate the \*.FTR file for FTRUN.

The application needs to call the FuzzyClose function to delete each fuzzy instance handle returned by the FuzzyOpen function.

## Syntax

**FuzzyOpen(*sFile*)**

*sFile*:

Specifies the filename of the .FTR file to load.

## Return Value

The handle to the fuzzy instance, or -1 if the function cannot complete the operation. Use [IsError](#) to find the error number.

## Related Functions

[FuzzyClose](#), [FuzzyGetShellValue](#), [FuzzySetShellValue](#), [FuzzyGetCodeValue](#), [FuzzySetCodeValue](#), [FuzzyTrace](#).

## Example

```
INT hFuzzy;
INT NoHitFlag;
INT Status;
REAL MemOutput;
// open the Fuzzy Tech runtime instance
hFuzzy = FuzzyOpen
("%PROGRAMFILES(X86)%\AVEVA Plant SCADA\Bin\traffic.ftr");
Status = IsError();
IF hFuzzy <> -1 THEN
    MemOutput = PLCOutput;
    WHILE Status = 0 DO
        FuzzySetShellValue(hFuzzy, 0, 42.0);
        FuzzySetShellValue(hFuzzy, 1, 3.14150);
        MemOutput = FuzzyGetShellValue(hFuzzy, 2, NoHitFlag);
        Status = IsError();
        // Only write to PLC if output changes.
        // This reduces load on PLC communication.
        IF MemOutput <> PLCOutput THEN
            PLCOutput = MemOutput;
        END
        SleepMS(500);
    END
    FuzzyClose(hFuzzy);
END
```

## See Also

[Fuzzy Logic Functions](#)

### FuzzySetCodeValue

Sets a value for the specified input of the specified instance.

## Syntax

**FuzzySetCodeValue(*hFuzzy*, *iIOIndex*, *iCodeValue*)**

*hFuzzy*:

The fuzzy instance handle (and integer greater than 0).

*iIOIndex*:

Specifies the variable to receive the value. The I/O-Indices start with 0 and increment by 1 for each variable. To find the correct index for a specific variable, the variables need to be sorted in alpha-numerical order, first the inputs and then the outputs.

*iCodeValue*:

The value to be copied to the variable specified by *iIOIndex*.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FuzzyOpen](#), [FuzzyGetCodeValue](#)

## Example

See [FuzzyOpen](#).

## See Also

[Fuzzy Logic Functions](#)

## FuzzySetShellValue

Sets a value for the specified input of the specified instance.

## Syntax

**FuzzySetShellValue**(*hFuzzy*, *iIOIndex*, *rShellValue*)

*hFuzzy*:

The fuzzy instance handle (and integer greater than 0).

*iIOIndex*:

Specifies the variable to receive the value. The I/O-Indices start with 0 and increment by 1 for each variable. To find the correct index for a specific variable, the variables need to be sorted in alpha-numerical order, first the inputs and then the outputs.

*rShellValue*:

The value to be copied to the variable specified by *iIOIndex*.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FuzzyOpen](#), [FuzzyGetShellValue](#)

## Example

See [FuzzyOpen](#).

## See Also

[Fuzzy Logic Functions](#)

## FuzzyTrace

Controls the trace process (starting and stopping) of the specified instance.

## Syntax

**FuzzyTrace(*hFuzzy*, *TraceOn*)**

*hFuzzy*:

The fuzzy instance handle (and integer greater than 0).

*TraceOn*:

Specifies whether to start or to stop a trace process for the Fuzzy instance specified by *hFuzzy*. If this parameter is TRUE (1), the trace process is started. If this parameter is FALSE (0), the trace process is stopped.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[FuzzyOpen](#)

## Example

See [FuzzyOpen](#).

## See Also

[Fuzzy Logic Functions](#)

## Group Functions

Following are functions relating to groups of objects:

<a href="#">GrpClose</a>	Closes a group.
<a href="#">GrpDelete</a>	Deletes items from a group.

<a href="#">GrpFirst</a>	Gets the first item in a group.
<a href="#">GrpIn</a>	Tests if an item is in a group.
<a href="#">GrpInsert</a>	Inserts items into a group.
<a href="#">GrpMath</a>	Performs mathematical operations on groups.
<a href="#">GrpName</a>	Gets the name of a group from a group handle.
<a href="#">GrpNext</a>	Gets the next item in a group.
<a href="#">GrpOpen</a>	Opens a group.
<a href="#">GrpToStr</a>	Converts a group into a string

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## GrpClose

Closes a group. The group is destroyed and the group handle becomes invalid. You should close a group when it is not in use, to release the associated memory. Plant SCADA closes all groups on shutdown.

## Syntax

**GrpClose(*hGrp*)**

*hGrp*:

The group handle, returned from the GrpOpen() function. The group handle identifies the table where all data on the associated group is stored.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[GrpOpen](#)

## Example

```
hGrp=GrpOpen( "MyGrp",1 );
..
GrpClose(hGrp);
```

## See Also

[Group Functions](#)

### GrpDelete

Deletes a single element or all elements from a group. You can also delete another group from within the group.

## Syntax

**GrpDelete(*hGrp*, *Value*)**

*hGrp*:

The group handle, returned from the GrpOpen() function. The group handle identifies the table where all data on the associated group is stored.

*Value*:

The element to delete from the group, from 0 to 16375.

- Set *Value* to -1 to delete all elements from the group.
- Set *Value* to a group handle to delete another group from this group.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[GrpInsert](#), [GrpOpen](#)

## Example

```
! Delete 10 and 14 from a group.  
GrpDelete(hGrp,10);  
GrpDelete(hGrp,14);
```

## See Also

[Group Functions](#)

### GrpFirst

Gets the value of the first element in a group. The first element in the group is the element with the lowest value. You can follow this function with a GrpNext() call, to get the value of all the elements in a group.

## Syntax

**GrpFirst(*hGrp*)**

*hGrp*:

The group handle, returned from the GrpOpen() function. The group handle identifies the table where all data on the associated group is stored.

## Return Value

The value of the first element in a group or -1 if the group is empty.

## Related Functions

[GrpOpen](#), [GrpNext](#)

## Example

```
Value=GrpFirst(hGrp);
IF Value<>-1 THEN
  Prompt("First value is "+Value:###);
END
```

## See Also

[Group Functions](#)

## GrpIn

Determines if an element is in a group.

## Syntax

**GrpIn(*hGrp, Value*)**

*hGrp*:

The group handle, returned from the GrpOpen() function. The group handle identifies the table where all data on the associated group is stored.

*Value*:

The element to locate, from 0 to 16375.

Set *Value* to a group handle to check if another group exists in the group.

## Return Value

1 if the element is in the group, otherwise 0 is returned.

## Related Functions

[GrpOpen](#), [GrpInsert](#), [GrpDelete](#)

## Example

```
IF GrpIn(hGrp,10) THEN
    Prompt("Area 10 in this group");
END
```

## See Also

[Group Functions](#)

## GrpInsert

Adds an element (or another group) to a group.

## Syntax

**GrpInsert(*hGrp*, *Value*)**

*hGrp*:

The group handle, returned from the GrpOpen() function. The group handle identifies the table where all data on the associated group is stored.

*Value*:

The element to add to the group, from 0 to 16375.

- Set *Value* to -1 to add all elements (0 to 16375) to the group.
- Set *Value* to a group handle to insert another group into the group.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[GrpOpen](#), [GrpDelete](#), [GrpIn](#)

## Example

```
! Add 10 and 14 to a group.
GrpInsert(hGrp,10);
GrpInsert(hGrp,14);
```

## See Also

[Group Functions](#)

## GrpMath

Performs mathematical operations on two groups, and stores the result in another group. You can add the two groups, subtract one from the other, or perform Boolean AND, NOT, and XOR operations on the two groups.

## Syntax

**GrpMath(*hResult*, *hOne*, *hTwo*, *Type*)**

*hResult*:

The group number where the result is placed.

*hOne*:

Number of first group used in the mathematical operation.

*hTwo*:

Number of the second group used in the mathematical operation.

*nType*:

Type of mathematical operation:

0 - Add groups one and two.

1 - Subtract group two from group one.

2 - AND groups one and two.

3 - NOT groups one and two.

4 - XOR groups one and two.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[GrpOpen](#)

## Example

```
hOne=GrpOpen("Plantwide",0);
hTwo=GrpOpen("Section1",0);
hResult=GrpOpen("Result",0);
! Subtract Section1 from Plantwide and place in Result.
GrpMath(hResult,hOne,hTwo,1);
```

## See Also

[Group Functions](#)

### GrpName

Gets the name of a group from a group handle.

## Syntax

**GrpName(*hGrp*)**

*hGrp*:

The group handle, returned from the GrpOpen() function. The group handle identifies the table where all data on the associated group is stored.

## Return Value

The name of the group (as a string).

## Related Functions

[GrpOpen](#)

## Example

```
! Get the current group name.  
sName=GrpName(hGrp);
```

## See Also

[Group Functions](#)

### GrpNext

Gets the value of the next element in a group. You can get the value of all the elements in a group. Call the GrpFirst() function to get the value of the first element, and then call this function in a loop.

## Syntax

**GrpNext(*hGrp, Value*)**

*hGrp*:

The group handle, returned from the GrpOpen() function. The group handle identifies the table where all data on the associated group is stored.

*Value:*

The value returned from GrpFirst() or the latest GrpNext() call.

## Return Value

The value of the next element in a group, or -1 if the end of the group has been found.

## Related Functions

[GrpFirst](#)

## Example

```
! Count all values in a group.  
Count=0;  
Value=GrpFirst(hGrp);  
WHILE Value<>-1 DO  
    Count=Count+1;  
    Value=GrpNext(hGrp,Value);  
END  
Prompt("Number of values in group is "+Count:###);
```

## See Also

[Group Functions](#)

## GrpOpen

Creates a group and returns a group handle, or gets the group handle of an existing group. After you open a group, you can use the group number in functions that use groups, for example, SetArea() and AlarmSetInfo(). You can open a group that is specified in the Groups database. You can also create groups at runtime.

When you open a group that is defined in the database, a copy of the group is made - the original group is not used. You can therefore change the values in the group without affecting other facilities that use this group.

## Syntax

**GrpOpen(sName, nMode)**

*sName*

The name of the group to open.

*Mode*

The mode of the open:

0 - Open an existing group

1 - Create a new group

2 - Attempts to open an existing group. If the group does not exist, it will create it.

## Return Value

The group handle , or -1 if the group cannot be created or opened. The group handle identifies the table where data on the associated group is stored.

## Related Functions

[GrpClose](#)

## Example

```
! Open Plantwide group defined in the database.  
hGrp=GrpOpen("Plantwide",0);  
! Set current user area to Plantwide.  
SetArea(hGrp);  
GrpClose(hGrp);  
! Set area to 1...10, 20 and 25 by creating a new group.  
hGrp=GrpOpen("MyGrp",1);  
StrToGrp(hGrp,"1..10,20,25");  
SetArea(hGrp);  
GrpClose(hGrp);
```

## See Also

[Group Functions](#)

## GrpToStr

Converts a group into a string of values separated by ", " and " .. ". You can then display the group on the screen or in a report.

## Syntax

**GrpToStr(*hGrp*)**

*hGrp*:

The group handle, returned from the GrpOpen() function. The group handle identifies the table where all data on the associated group is stored.

## Return Value

The group (as a string).

## Related Functions

[GrpOpen](#), [StrToGrp](#)

## Example

```
! Display current areas.  
hGrp=GetArea();  
Str=GrpToStr(hGrp);  
DspStr(21,"WhiteFont",Str);
```

## See Also

[Group Functions](#)

## I/O Device Functions

Following are functions relating to I/O devices:

<a href="#">DriverInfo</a>	Provides information about the driver for a particular I/O device.
<a href="#">IODeviceControl</a>	Provides control of individual I/O devices.
<a href="#">IODeviceInfo</a>	Gets information on an I/O device.
<a href="#">IODeviceStats</a>	Gets statistical information for I/O devices.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## DriverInfo

Provides information about the driver for a specified I/O device. Select the device using the *IODevice* argument, and the information to be returned using the *Type* argument.

This function can only be used if the I/O Server is on the current machine. When the I/O Server is not in the calling process, this function will become blocking and cannot be called from a foreground task. In this case, the return value will be undefined and a Cicode hardware alarm will be raised.

## Syntax

**DriverInfo**(*IODevice*, *nType* [, *sClusterName*] [, *ServerName*] )

*IODevice*:

The name of the I/O device.

*nType*:

The type of information returned about the driver. Specify one of the following:

0 - Driver Name

- 1 - Driver Title
- 2 - Block constant
- 3 - Max Retrys
- 4 - Transmit Delay
- 5 - Receive Timeout
- 6 - Polltime
- 7 - Watchtime (milliseconds)

---

**Note:** The DISKDRV driver name is returned as "Disk" instead of "DISKDRV". If the Polltime is set as "Interrupt", the function returns "0".

---

*sClusterName:*

Specifies the name of the cluster in which the I/O Server resides. This is optional if you have one cluster or are resolving the I/O server via the current cluster context. The argument is enclosed in quotation marks "".

*ServerName:*

Specifies the name of the the I/O Server. This parameter is only required if you are running more than one I/O server process from the same cluster on the same computer and need to instruct the system which process to redirect to. The argument is enclosed in quotation marks "".

## Return Value

The driver information as a string. In the case of an error the return value is an empty string.

## Example

```
// Using the IODevice Number
sName = DriverInfo(20, 0); ! Get the name of the driver used with I/O device 20
sName = DriverInfo(2, 1); ! Get the title of the driver used with I/O device 2
// Using the IODevice Name
sName = DriverInfo("IODev",3);
! Get the Max Retrys value of the driver used with IODev
sName = DriverInfo("IODev1",5);
! Get the Receive Timeout value of the driver used with IODev1
```

## See Also

[I/O Device Functions](#)

## IODeviceControl

Provides control of individual I/O devices. You might need to call this function several times. If you use incompatible values for the various options of this function, you might get unpredictable results.

This function can only be used if the I/O Server is on the current machine. When the I/O Server is not in the calling process, this function will become blocking and cannot be called from a foreground task. In this case, the return value will be undefined and a Cicode hardware alarm will be raised.

## Syntax

**IODeviceControl(*IODevice*, *nType*, *Data* [, *sClusterName*] [, *ServerName*] )**

*IODevice*:

The number or name of the I/O device. If you call this function from an I/O server, you can use the I/O device name. If you call this function from a client, you may use either the I/O device number or name. To specify all I/O devices, use "\*" (asterisk) or -1.

*nType*:

The type of control action:

0 - No longer supported.

1 - Enable/Disable the I/O device on the I/O server. If disabled, attempts to read and write from the I/O device are ignored. (If another I/O device is configured as a standby I/O server, Plant SCADA switches communications to that I/O device.) The I/O server does not attempt to communicate with the I/O device until it is re-enabled. When the I/O device is re-enabled, the I/O server attempts to re-establish communication immediately. Mode 1 can only be called by the I/O Server which is associated with this device.

2 - No longer supported. An invalid argument error is returned if this option is specified.

3 - No longer supported. An invalid argument error is returned if this option is specified.

4 - The data in the associated I/O device cache is flushed. This allows flushing of the data from the I/O device without waiting for the aging time. This is useful when you have long cache time and you want to force a read from the I/O device. The Data value is ignored with this mode.

5 - (For scheduled and remote I/O devices). The I/O device is added to the bottom of the list of I/O devices to be contacted. I/O devices already in the list (already waiting to be contacted) are given priority over this I/O device.

6 - (For scheduled and remote I/O devices). The I/O device is added to the top of the list of I/O devices to be contacted; it is given high priority. If there are already I/O devices at the top of the list with high priority, then this I/O device will be added to the list after them (that is it will be contacted after them). For dial-up remote I/O devices, if the modem is already in use - connected to another I/O device - this I/O device will not be dialled until that connection has been terminated.

7 - (For scheduled and remote I/O devices). The I/O device is added to the top of the list of I/O devices to be contacted, and it is given top priority. For dial-up remote I/O devices, if the modem is currently connected to another I/O device, the connection will be cancelled, and the top priority I/O device will be dialled. It will also stay connected until manually disconnected with another call to `IODeviceControl()`.

---

**Note:** This mode will not attempt to disconnect any other persistent connections. Persistent connections can only be disconnected using mode 8.

---

8 - (For scheduled and remote I/O devices). Disconnect an I/O device. Current requests will be completed before the I/O device is disconnected.

9 - (For scheduled I/O devices). The communication schedule for the I/O device is disabled. This is to minimize the likelihood that the I/O device will be contacted when its scheduled dial-time occurs.

10 - (For scheduled I/O devices). Puts the I/O device into Write On Request mode. That is, as soon as a write request is made, the I/O device will be added to the list of I/O devices to be contacted. It is given priority over existing read requests, but not over existing write requests. In this situation, there will be a delay while the I/O device is contacted. Do not mistake this pause for inactivity (for example, do not continually execute a command during this delay).

11 - Change the I/O device cache timeout. If the I/O Server is restarted, the cache timeout will return to its original value. (For scheduled I/O devices, this value can be checked using the Kernel Page Unit command. For all

other I/O devices, this value is configured in the Cache Time field at the I/O Devices Properties form.)

12 - The time of day at which to add the I/O device to the list of I/O devices to be contacted. Set the time in Data in seconds from midnight (for example, specify 6 p.m. as 18 \* 60 \* 60). Use Type 12 to specify a one-time communication.

13 - The communication period (the time between successive communication attempts). The value you specify represents different periods, depending on what type of schedule you are using (that is daily, weekly, monthly, or yearly. This is set using Type 15.). You can choose to specify the communication period either in seconds from midnight, day of week (0 to 6, Sunday = 0), month (1 to 12), or year. Enter the value in Data. For example, if your schedule is weekly, and you set Data = 3, you are specifying each Wednesday. If your schedule is monthly, Data = 3 indicates March. For daily communication, set the period in Data in seconds from midnight; for example, set Data to 6 \* 60 \* 60 to contact the I/O device every 6 hours.

14 - The time at which the I/O Server will first attempt to communicate with the I/O device. Set the time in Data in seconds from midnight, for example, to synchronize at 10a.m., set Data to 10 \* 60 \* 60.

15 - Type of schedule. Set Data to one of the following:

- 1 - Daily
- 2 - Weekly
- 3 - Monthly
- 4 - Yearly

16 - (For remote I/O devices) Read all tags from the I/O device. Data is unused - set it to 0 (zero).

18 - Set Control Inhibit (Control Mode) for all tags of the I/O device.

*Data:*

Data for the control operation\*:

1:

- Disable the I/O device (Disable Write On Request mode for Type 10)
- Set Control Inhibit to ON (mode for type 18)

0:

- Enable the I/O device (Enable Write On Request mode for Type 10) or the I/O device name (for Type 2 or 3).
- Set Control Inhibit to OFF (mode for type 18)

\* For Type 5-8, Data is ignored; enter 0 (zero).

*sClusterName:*

Specifies the name of the cluster in which the I/O Server resides. This is optional if you have one cluster or are resolving the I/O server via the current cluster context. The argument is enclosed in quotation marks "".

*ServerName:*

Specifies the name of the the I/O Server. This parameter is only required if you are running more than one I/O server process from the same cluster on the same computer and need to instruct the system which process to redirect to. The argument is enclosed in quotation marks "".

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[IODeviceInfo](#), [IODeviceStats](#), [TagReadEx](#), [TagWrite](#)

## Example

```
IODeviceControl(4, 1, 1); ! Disable I/O device 4
```

## See Also

[I/O Device Functions](#)

### IODeviceInfo

Gets information about a specified I/O device.

Apart from when *Type* is set to 3 or 17, this function can only be used if the I/O Server is on the current machine, otherwise the function will not succeed and will return empty string. When the I/O Server is not in the calling process, this function will become blocking and cannot be called from a foreground task. In this case, the return value will be undefined and a Cicode hardware alarm will be raised.

If both the primary and standby I/O devices are on the same server and they have the same I/O device name, you can get information about them individually by specifying the following:

```
IODeviceInfo("PLC1,P",1); // for the Primary  
IODeviceInfo("PLC1,S",1); // for the Standby
```

Where:

- P represents the primary I/O device
- S the standby I/O device.

If you have more than one standby device on the same server, there is currently no way of using this function for other than the first standby device.

---

**Note:** When the I/O server is not in the calling process, this function could become a blocking function if the information required by this function is on an I/O server (except types 3 and 17, which are normally non-blocking). If this is the case, this function cannot be called from a foreground task (such as a graphics page) or an expression. Otherwise the return value will be undefined and a Cicode hardware alarm raised.

---

## Syntax

**IODeviceInfo(*IODevice*, *Type* [, *sClusterName*] [, *ServerName*])**

*IODevice*:

The I/O device number, or the I/O device name enclosed in double quotes.

*nType*:

The type of information:

0 - Name of I/O device

1 - Protocol of I/O device

2 - Protocol address

3 - Client I/O device state

- 1 = Running - Client is either talking to an online I/O device or talking to a scheduled device that is not currently connected but has a valid cache

- 2 = Standby - Client is talking to an online standby I/O device

- 4 = Starting - Client is talking to an I/O device that is attempting to come online

- 8 = Stopping - Client is talking to an I/O device that is in the process of stopping

- 16 = Offline - Client is pointing to an I/O device that is currently offline

- 32 = Disabled - Client is pointing to a device that is disabled

- 66 = Standby write - client is talking to an I/O device configured as a standby write device

4 - Current generic error number (decimal)

5 - Current driver error number (decimal)

6 - Disabled flag

7 - Statistics, minimum read time

8 - Statistics, maximum read time

9 - Statistics, average read time

10 - I/O server I/O device state

- 1 = Running - I/O device for this I/O server is online or a scheduled device that is not currently connected but has a valid cache

- 2 = Standby - I/O device for this I/O server is online and a standby unit

- 4 = Starting - I/O device for this I/O server is attempting to come online. Starting may be combined with either Offline or Remote such as: 20 = Starting(4) + Offline(16) or 132 = Starting(4) + Remote(128).

- 8 = Stopping - I/O device for this I/O server is currently in the process of stopping

- 16 = Offline (only valid on an I/O server) - I/O device for this I/O server is currently offline

- 32 = Disabled - I/O device for this I/O server is disabled

- 66 = Standby write - I/O device for this I/O server is configured as a standby write device

- 128 = Remote (returned in combination with another value specified above - see Starting) - I/O device for this I/O server is a scheduled device but not currently connected

11 - Unit number

12 - Configured I/O server name

13 - Configured Port name

14 - Configured startup mode

15 - Configured comment

16 - The primary I/O server name the client uses to communicate to this device

17 - The I/O Server the client is using to communicate to this device. Will be Standby if the Primary is down.

18 - State of the remote unit:

- 0 = Remote unit is disconnected and OK

- 1 = Remote unit is connected and online

- 2 = Remote unit is in the dial queue

If you are trying to determine the state of a remote device that is configured for scheduled communications, you should use this type in conjunction with type 10. Firstly, use type 10 to determine if the device is in running state. If it is, use type 18 to determine its connection state.

This mode causes redirection to the I/O server if running in separate processes.

19 - Number of successful attempts to communicate with the scheduled I/O device.

20 - Number of unsuccessful attempts to communicate with the scheduled I/O device.

21 - Write mode: Write On Request, and normal (as per schedule defined in the Express Communications Wizard).

22 - Number of queued read requests for the scheduled I/O device. (This mode causes redirection to the I/O server if running in separate processes.)

23 - Number of queued write requests for the scheduled I/O device. (This mode causes redirection to the I/O server if running in separate processes.)

24 - The cache timeout (in milliseconds).

26 - The name of the line device (for example, modem) you are using to connect to the I/O device. (This mode causes redirection to the I/O server if running in separate processes.)

27 - The call\_status of a currently connected remote I/O device.

- DIALCALLSTATE\_UNAVAIL — 1
- DIALCALLSTATE\_IDLE — 2
- DIALCALLSTATE\_OFFERING — 3
- DIALCALLSTATE\_ACCEPTED — 4
- DIALCALLSTATE\_DIALTONE — 5
- DIALCALLSTATE\_DIALING — 6
- DIALCALLSTATE\_RINGBACK — 7
- DIALCALLSTATE\_BUSY — 8
- DIALCALLSTATE\_SPECIALINFO — 9
- DIALCALLSTATE\_CONNECTED — 10
- DIALCALLSTATE\_PROCEEDING — 11
- DIALCALLSTATE\_ONHOLD — 12
- DIALCALLSTATE\_CONFERENCE — 13
- DIALCALLSTATE\_ONHOLDPENDCONF — 14
- DIALCALLSTATE\_ONHOLDPENDTRANSFER — 15
- DIALCALLSTATE\_DISCONNECTED\_NORMAL — 16
- DIALCALLSTATE\_DISCONNECTED\_LINELOST — 17
- DIALCALLSTATE\_DISCONNECTED\_UNKNOWN — 18
- DIALCALLSTATE\_DISCONNECTED\_REJECT — 19
- DIALCALLSTATE\_DISCONNECTED\_PICKUP — 20
- DIALCALLSTATE\_DISCONNECTED\_FORWARDED — 21
- DIALCALLSTATE\_DISCONNECTED\_BUSY — 22
- DIALCALLSTATE\_DISCONNECTED\_NOANSWER — 23
- DIALCALLSTATE\_DISCONNECTED\_BADADDRESS — 24

- DIALCALLSTATE\_DISCONNECTED\_UNREACHABLE — 25
- DIALCALLSTATE\_DISCONNECTED\_CONGESTION — 26
- DIALCALLSTATE\_DISCONNECTED\_INCOMPATIBLE — 27
- DIALCALLSTATE\_DISCONNECTED\_UNAVAIL — 28
- DIALCALLSTATE\_DISCONNECTED\_NODIALTONE — 29
- DIALCALLSTATE\_DISCONNECTED\_NUMBERCHANGED — 30
- DIALCALLSTATE\_DISCONNECTED\_OUTOFORDER — 31
- DIALCALLSTATE\_DISCONNECTED\_TEMPFAILURE — 32
- DIALCALLSTATE\_DISCONNECTED\_QOSUNAVAIL — 33
- DIALCALLSTATE\_DISCONNECTED\_BLOCKED — 34
- DIALCALLSTATE\_DISCONNECTED\_DONOTDISTURB — 35
- DIALCALLSTATE\_DISCONNECTED\_CANCELLED — 36
- DIALCALLSTATE\_UNKNOWN — 48

(This mode causes redirection to the I/O server if running in separate processes.)

28 - The call rate (in bits per second) which may be the DTE or DCE connection speed depending on the server modem settings. (This mode causes redirection to the I/O server if running in separate processes.)

30 - The last time an I/O device from the remote I/O device redundant group was connected (primary or any standbys).

31 -The state of the remote I/O device redundant group:

- 0 = not connected (none of the redundant I/O devices connected)
- 1 =connected (one of the redundant I/O devices is connected)

32 - The next time the specified I/O device is scheduled to connect (unless a higher priority I/O device comes online).

#### *sClusterName:*

Specifies the name of the cluster in which the I/O Server resides. This is optional if you have one cluster or are resolving the I/O server via the current cluster context. The argument is enclosed in quotation marks "".

#### *ServerName:*

Specifies the name of the I/O Server. This parameter is only required if you are running more than one I/O server process from the same cluster on the same computer and need to instruct the system which process to redirect to. The argument is enclosed in quotation marks "".

## Return Value

The type of information (as a string).

## Related Functions

[IODeviceControl](#), [IODeviceStats](#), [TagReadEx](#), [TagWrite](#)

## Example

```
//Using the IODDevice Number
```

```
sName = IODeviceInfo(20, 0); ! Get the name of I/O device 20
sName = IODeviceInfo(2, 1); ! Get the protocol of I/O device 2
//Using the IODevice Name
sName = IODeviceInfo("IODev",10); ! Get the I/O Server State
sName = IODeviceInfo("IODev1",3); ! Get the State of IODev1
```

## See Also

[I/O Device Functions](#)

### IODeviceStats

Gets statistical information for all I/O devices, and displays the information in a dialog box.

**Note:** In a multi-process environment this function needs to be called from the I/O server process or redirected there using [MsgRPC](#). If this isn't done, some of the information on the IODeviceStats form will not be displayed correctly.

## Syntax

**IODeviceStats()**

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[IODeviceInfo](#)

## Example

```
IODeviceStats(); ! display all I/O device information
```

## See Also

[I/O Device Functions](#)

## Keyboard Functions

Following are functions relating to the keyboard:

<a href="#">KeyAllowCursor</a>	Allows the command cursor to move to any AN or only to ANs that have commands defined.
<a href="#">KeyBs</a>	Deletes the last character from the key command line.

<a href="#">KeyDown</a>	Moves the command cursor down.
<a href="#">KeyGet</a>	Gets the raw key code from the key command line.
<a href="#">KeyGetCursor</a>	Gets the AN where the cursor is positioned.
<a href="#">KeyLeft</a>	Moves the command cursor left.
<a href="#">KeyMove</a>	Moves the command cursor in the requested direction.
<a href="#">KeyPeek</a>	Gets a key from the key command line without removing the key.
<a href="#">KeyPut</a>	Puts a raw key code into the key command line.
<a href="#">KeyPutStr</a>	Puts a string into the key command line.
<a href="#">KeyReplay</a>	Replays the last key sequence.
<a href="#">KeyReplayAll</a>	Replays and executes the last key sequence.
<a href="#">KeyRight</a>	Moves the command cursor right.
<a href="#">KeySetCursor</a>	Moves the command cursor to a specified AN.
<a href="#">KeySetSeq</a>	Adds a keyboard sequence at runtime.
<a href="#">KeyUp</a>	Moves the command cursor up.
<a href="#">SendKeys</a>	Sends a keystroke (or string of keystrokes) to a window.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## KeyAllowCursor

Allows (or disallows) the command cursor to move to the specified AN or to all ANs. The command cursor normally moves only to ANs that have commands defined.

## Syntax

**KeyAllowCursor(*nAN*, *State*)**

*nAN*:

The AN where the command cursor can move. If 0, all ANs are implied.

*State:*

Allow state:

0 - Do not allow the cursor to move to this AN.

1 - Allow the cursor to move to this AN.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[KeyGetCursor](#)

## Example

```
KeyAllowCursor(20,1);
! Allows the command cursor to move to AN20.
KeyAllowCursor(0,1);
! Allows the command cursor to move to any AN.
```

## See Also

[Keyboard Functions](#)

## KeyBs

Removes the last key from the key command line. If the key command line is empty, this function will not perform any action.

You should call this function using a "Hot Key" command (as shown in the example below), otherwise the backspace character is placed into the key command line and the command does not execute. A "Hot Key" command is a command that executes as soon as it is placed into the key command line.

## Syntax

`KeyBs()`

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[KeyGet](#)

## Example

### System Keyboard

Key Sequence	*Bs
Command	KeyBs()
Comment	Define a backspace Hot Key

(\*) represents a HotKey command)

```
/* If "START A B C" is in the key command line and "START" is
the Key Name for the "F1" key: */
KeyBs();
! Removes ASCII "C".
KeyBs();
! Removes ASCII "B".
KeyBs();
! Removes ASCII "A".
KeyBs();
! Removes Key_F1.
KeyBs();
! Performs no action.
```

## See Also

[Keyboard Functions](#)

## KeyDown

Moves the command cursor down the page to the closest AN. If an AN-Down cursor override is specified (in the Page Keyboard database) for the graphics page, the command cursor moves to that AN instead.

## Syntax

`KeyDown()`

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[KeyUp](#), [KeyLeft](#), [KeyRight](#), [KeyMove](#)

## Example

See [KeyDown](#).

## See Also

[Keyboard Functions](#)

### KeyGet

Gets the last key code from the key command line. The key is removed from the command line. Use this function to process the operator key commands directly. You should call this function from the keyboard event function.

## Syntax

`KeyGet()`

## Return Value

The last key code from the key command line. If the key command line is empty, 0 (zero) is returned.

## Related Functions

[KeyPeek](#), [KeyPut](#)

## Example

```
/* If "START A B C" is in the key command line and "START" is
the Key Name for the "F1" key: */
Variable=KeyGet();
! Sets Variable to 67 (ASCII "C").
Variable=KeyGet();
! Sets Variable to 66 (ASCII "B").
Variable=KeyGet();
! Sets Variable to 65 (ASCII "A").
Variable=KeyGet();
! Sets Variable to 170 (the ASCII value of the F1 key (Key_F1)).
Variable=KeyGet();
! Sets Variable to 0.
```

## See Also

[Keyboard Functions](#)

### KeyGetCursor

Gets the AN at the position of the command cursor.

If this function is called from within a larger piece of code, the cursor may have moved away from where it was originally positioned when the larger piece of code was started.

If you are using groups, and there are currently two command cursors, the AN for the innermost will be returned. For example, if there is a cursor for a group as well as a cursor for one of its objects, the AN for the

object will be returned.

## Syntax

`KeyGetCursor()`

## Return Value

The AN at the position of the command cursor. If no cursor is visible, -1 is returned.

## Related Functions

[KeySetCursor](#)

## Example

```
! If the command cursor is on AN25:  
AN=KeyGetCursor();  
! Sets AN to 25.
```

## See Also

[Keyboard Functions](#)

## KeyLeft

Moves the command cursor left (across the page) to the closest AN. If an AN-Left cursor override is specified (in the Page Keyboard database) for the graphics page, the command cursor moves to that AN instead.

## Syntax

`KeyLeft()`

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[KeyRight](#), [KeyUp](#), [KeyDown](#), [KeyMove](#)

## Example

See [KeyRight](#).

## See Also

[Keyboard Functions](#)

## KeyMove

Moves the command cursor in a specified direction to the closest AN. If an AN cursor override is specified, the command cursor moves to that AN directly. This function is equivalent to the KeyUp(), KeyDown(), KeyLeft(), and KeyRight() functions.

## Syntax

**KeyMove(*Direction*)**

*Direction*:

Direction to move the cursor:

0 - Do not move

1 - Left

2 - Right

3 - Up

4 - Down

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[KeyUp](#), [KeyDown](#), [KeyLeft](#), [KeyRight](#)

## Example

```
KeyMove(1);
! Moves the cursor left.
```

## See Also

[Keyboard Functions](#)

## KeyPeek

Gets the ascii key code from the key command line (at a specified offset), without removing the key from the key command line. An offset of 0 returns the key code from the last position in the key command line.

## Syntax

**KeyPeek(*Offset*)**

*Offset*:

The offset from the end of the key command line

## Return Value

The ASCII key code.

## Related Functions

[KeyGet](#)

## Example

```
! If "A B C" is in the key command line:  
Variable=KeyPeek(0);  
! Sets Variable to 67 (ASCII "C")  
Variable=KeyPeek(2);  
! Sets Variable to 65 (ASCII "A")
```

## See Also

[Keyboard Functions](#)

## KeyPut

Puts an ASCII key code or Keyboard key code into the last position of the key command line. If this key completes any command, that command will execute.

## Syntax

**KeyPut(*KeyCode*)**

*KeyCode*:

The key code to put into the key command line.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[KeyGet](#)

## Example

```
KeyPut(Key_F1);
/* Puts "Key_F1" (the Key Code for the "F1" key) into the last
position of the key command line. If "START" is the Key Name for
the "F1" key, this would be equivalent to
KeyPutStr("START"). In either case, "START" will display on the
key command line. */
KeyPut(StrToChar("A"));
/* Puts the Key Code for the "A" key into the last position of the
key command line. */
```

## See Also

[Keyboard Functions](#)

### KeyPutStr

Puts a string at the end of the key command line. The string can contain either key names or data characters. If this string completes any command, that command will execute.

## Syntax

**KeyPutStr(*String*)**

*String*:

The string to put into the key command line.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[KeyPut](#)

## Example

```
KeyPutStr("START ABC");
! Places "START ABC" at the end of the key command line.
KeyPutStr("TURN PUMP 1 ON");
! Places "TURN PUMP 1 ON" at the end of the key command line.
```

## See Also

[Keyboard Functions](#)

## KeyReplay

Replays the last key sequence (except for the last key, which would execute the command). This function is useful when a command is to be repeated. To call this function from the keyboard, use a Hot Key "\*" (asterisk) command, otherwise the KeyReplay() function itself is replayed.

## Syntax

**KeyReplay(*Sub*)**

*Sub*:

Number of characters to subtract before replay. Default value is 1.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[KeyReplayAll](#)

## Example

If the previous contents of the key command line were:

```
"START ABC ENTER"  
KeyReplay();  
! Replays "START ABC".
```

## See Also

[Keyboard Functions](#)

## KeyReplayAll

Replays the last key sequence and executes the command. To call this function from the keyboard, use a Hot Key " \* " (asterisk) command, otherwise the KeyReplayAll() function itself is replayed.

## Syntax

**KeyReplayAll()**

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[KeyReplay](#)

## Example

If the previous contents of the key command line were:

```
"START ABC ENTER"  
KeyReplayAll();  
! Replays "START ABC ENTER" and executes the command.
```

## See Also

[Keyboard Functions](#)

## KeyRight

Moves the command cursor right (across the page) to the closest AN. If an AN-Right cursor override is specified (in the Page Keyboard database) for the graphics page, the command cursor moves to that AN instead.

## Syntax

**KeyRight()**

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[KeyUp](#), [KeyDown](#), [KeyLeft](#), [KeyMove](#)

## Example

See [KeyLeft](#).

## See Also

[Keyboard Functions](#)

## KeySetCursor

Displays the command cursor at a specified AN. A keyboard command needs to exist, or you need to first call the KeyAllowCursor() function (at the AN) to allow the cursor to move to the AN. If the AN does not exist, or if a command does not exist at that AN, or if KeyAllowCursor() has not been called, the command cursor does not

move.

## Syntax

**KeySetCursor(*nAN*)**

*nAN*:

The AN where the command cursor will be displayed.

## Return Value

If the AN does not exist, or if a command does not exist at that AN, or if KeyAllowCursor() has not been called, the return value is 1. Otherwise, the function will return 0.

## Related Functions

[KeyGetCursor](#), [KeyAllowCursor](#)

## Example

```
! Move the command cursor to AN20.  
KeySetCursor(20);
```

## See Also

[Keyboard Functions](#)

## KeySetSeq

Adds a keyboard sequence to the current page at runtime. The key sequence is only added to the current window. When the page is closed, the keyboard sequence is deleted.

## Syntax

**KeySetSeq(*sKeySeq*, *AN*, *Fn*)**

*sKeySeq*:

The keyboard sequence.

*nAN*:

The AN where the keyboard sequence will apply. If you set AN to 0 (zero), the keyboard sequence will apply to all ANs on the page.

*Fn*:

The function to call when the keyboard sequence matches. This function needs to be a callback function.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspButton](#), [DspButtonFn](#)

## Example

```
/* Set the key sequence and call the "Callback" function when the
sequence is found. */
KeySetSeq("F2 ### Enter", 0, Callback);
! This function is called when the key sequence is found.
INT
FUNCTION CallBack()
    INT Value;
    ! Get user data.
    Value=Arg1;
    ..
    RETURN 0;
END
```

## See Also

[Keyboard Functions](#)

## KeyUp

Moves the command cursor up the page to the closest AN. If an AN-Up cursor override is specified (in the Page Keyboard database) for the graphics page, the command cursor moves to that AN.

## Syntax

`KeyUp()`

## Return Value

0 (zero) if successful, otherwise an error is returned.

[KeyDown](#), [KeyLeft](#), [KeyRight](#), [KeyMove](#)

## See Also

[Keyboard Functions](#)

## SendKeys

Sends a keystroke (or string of keystrokes) to a window as if they were typed on the keyboard. The window receives input focus and is brought to the foreground.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**SendKeys(*sTitle*, *sKeys*)**

*sTitle*:

The title (caption) of the destination window.

*sKeys*:

The key (or keys) to send to *sTitle*.

- To send a single keyboard character, use the character itself. For example, to send the letter a, set *sKeys* to a. To send more than one character, append each additional character to the string. For example, to send the letters a, b, and c, set *sKeys* to abc.
- The plus (+), caret (^), and percent sign (%) have special meanings. To send one of these special characters, enclose the character with braces. For example, to send the plus sign, use {+}. To send a { character or a } character, use {{}} and {{}}, respectively.
- To specify characters that are not displayed when you press a key (such as **Enter** or **Tab**) and other keys that represent actions rather than characters, use the codes shown below:

Key	Code
Backspace	{backspace} or {bs} or{bksp}
Break	{break}
Caps Lock	{capslock}
Clear	{clear}
Del	{delete} or {del}
End	{end}
Enter	{enter} or ~
Esc	{escape} or {esc}
Help	{help}
Home	{home}
Insert	{insert}
Num Lock	{numlock}

Page Down	{pgdn}
Page Up	{pgup}
Print Screen	{prtsc}
Scroll Lock	{scrolllock}
Tab	{tab}
Up Arrow	{up}
Down Arrow	{down}
Right Arrow	{right}
Left Arrow	{left}
F1	{f1}
F2	{f2}
F3	{f3}
F4	{f4}
F5	{f5}
F6	{f6}
F7	{f7}
F8	{f8}
F9	{f9}
F10	{f10}
F11	{f11}
F12	{f12}

To specify keys combined with any combination of **Shift**, **Ctrl**, and **Alt**, precede the regular key code with one or more of these codes:

Key	Code
Shift	+
Ctrl	^
Alt	%

To specify that **Shift**, **Ctrl**, and/or **Alt** are held down while several keys are pressed, enclose the keys in

parentheses. For example, to hold down the **Shift** key while sending E then C, use +(EC). To hold down **Shift** while sending E, followed by C without the **Shift** key, use +EC. To specify repeating keys, use the form {key number}. For example, {left 42} means send the left arrow key 42 times. Be aware that you need to leave a space between the key and number.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[WndFind](#)

## Example

```
SendKeys("Untitled - Notepad", "abc");
// Send the key sequence "abc" to the Notepad application
```

## See Also

[Keyboard Functions](#)

## Mail Functions

Following are functions relating to sending or receiving mail:

<a href="#">MailError</a>	Gets the last mail error code
<a href="#">MailLogoff</a>	Logoff from the mail system
<a href="#">MailLogon</a>	Logon to the mail system
<a href="#">MailRead</a>	Reads a standard mail message
<a href="#">MailSend</a>	Sends a standard mail message

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## MailError

Gets the last mail error code. The error code is extracted from the MAPI mail system and explains what caused the MAPI error.

## Syntax

**MailError()**

## Return Value

0 (zero) if successful, otherwise an error is returned. Refer also to MAPI errors.

## Related Functions

[MailLogon](#), [MailLogoff](#), [MailSend](#), [MailRead](#)

## Example

```
! Logon to the mail system
IF MailLogon("RodgerG", "password", 0) THEN
    error = MailError();
    !do what is required
END
```

## See Also

[Mail Functions](#)

## MailLogoff

Logs off from the mail system. You should log off the mail system when all mail operations are complete. Plant SCADA automatically logs off the mail system on shutdown.

## Syntax

**MailLogoff()**

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[MailLogon](#), [MailLogoff](#), [MailRead](#)

## Example

```
! Send the report to Rodger
MailLogon("Andrew", "password", 0);
MailSend("Rodger Gaff", "Report", "This is the weekly report",
```

```
"[data]:weekly.txt", 0);  
MailLogoff();
```

## See Also

[Mail Functions](#)

## MailLogon

Logs on to the mail system. You need to call this function before any other mail function.

The mail system uses the MAPI standard interface, so you can use any mail system that supports this standard.

You should log on to the mail system when Plant SCADA starts, and log off only at shutdown. (The logon procedure can take a few seconds to complete.) You can only log on as one user at a time for each computer, so you can only read mail for this user name.

## Syntax

**MailLogon(*sName*, *sPassword*, *iMode*)**

*sName*:

The name of the mail user. This name is the user's mail box name (the unique shorthand name, not the full user's name).

*sPassword*:

The password of the mail user.

*iMode*:

The mode of the logon:

0 - Normal logon.

2 - Get unique logon, do not share existing mail client logon.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[MailLogoff](#), [MailSend](#), [MailRead](#), [MailError](#)

## Example

```
! Send the report to James  
MailLogon("RodgerG", "password", 0);  
MailSend("James Glover", "Report", "This is the weekly report",  
"[data]:weekly.txt", 0);  
MailLogoff();
```

## See Also

[Mail Functions](#)

### MailRead

Reads a standard mail message. The mail message can contain text, an attached file, or both.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

Before you can use this function, you need to use the MailLogon() function to log on to the mail system. You can only read mail sent to the user name specified in the MailLogon() function.

## Syntax

**MailRead(*sName*, *sSubject*, *sNote*, *sFileName*, *iMode*)**

*sName*:

The name of the mail user who sent the message. Must be a String type variable.

*sSubject*:

The subject text of the mail message. Must be a String type variable.

*sNote*:

The note section of the message. If the message is longer than 255 characters, Plant SCADA will save the message in a file and return the file name in *sNote*. Use the file functions to read the message. The name of the file will be in the form CTxxxxx where x is a unique number. You need to delete the file after you have finished with the mail message. Must be a String type variable.

*sFileName*:

The name of any attached file. If there is no attached file in the message, specify *sFileName* as an empty string "". Must be a String type variable.

*iMode*:

The mode of the read:

0 - Read a message. If no message is available, wait for a message.

1 - Read a message. If no message is available, return with an error code.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[MailLogon](#), [MailLogoff](#), [MailSend](#), [MailError](#)

## Example

```
! Logon to the mail system
MailLogon("RodgerG", "password", 0);
```

```
! Read a message. Don't wait if no message
IF MailRead(sName, sSubject, sNote, sFileName, 1) = 0 THEN
    ! got message now do something with it
END
WHILE TRUE DO
    ! wait for a mail message
    MailRead(sName, sSubject, sNote, sFileName, 0);
END;
MailLogoff();
```

## See Also

[Mail Functions](#)

## MailSend

Sends a standard mail message. The mail message can contain text, an attached file, or both.

Before you can use this function, you need to use the MailLogon() function to log on to the mail system. You can only send mail from the user name specified in the MailLogon() function. You can send mail to any mail user or to another Plant SCADA client.

## Syntax

**MailSend(*sName*, *sSubject*, *sNote*, *sFileName*, *iMode*)**

*sName*:

The name of the mail user who will receive the message. This name is the user's full name (not their mailbox name).

*sSubject*:

The subject text of the mail message (a short description of what the message is about).

*sNote*:

The note section of the message (the main section of the message text). You can enter up to 255 characters, or a file name for longer messages. If you enter a file name, set *iMode* to 1.

*sFileName*:

The name of any attached file. If there is no attached file in the message, set *sFileName* to an empty string "".

*iMode*:

The mode of the send:

0 - Normal mail message.

1 - The *sNote* argument is the name of a text file to send as the note.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[MailLogon](#), [MailLogoff](#), [MailRead](#), [MailError](#)

## Example

```
! Logon to the mail system
MailLogon("Wombat", "password", 0);
! send the report to Andrew
MailSend("Andrew Bennet", "Report", "Attached is the weekly report",
"[data]:weekly.txt", 0);
! send hello message to JR
MailSend("Jack Russell", "Hello", "You've only got yourself to blame!", "", 0);
! send a big note to Nigel
MailSend("Nigel Colless", "Big Message", "[data]:message.txt", "", 1);
MailLogoff();
```

## See Also

[Mail Functions](#)

## Map Functions

Map functions allow for values to be associated with keys within the map.

As the handle to a map (or map name) is a string, it can be stored either as a key or as a value in another map, therefore maps can be used to create arbitrarily complex data structures.

By treating a map as being equivalent to an "instance" of an "object" in an "object-oriented" language, it is possible to achieve a simplified object oriented programming style. This can be done by passing the map name as a parameter to the "object's" functions.

The keys in the map are case sensitive and are ordered alphabetically as can be observed by sequentially discovering each key in the map (using MapKeyFirst and MapKeyNext). The alphabetic ordering is not lexicographic ordering and only determines a repeatable order for a given set of keys.

Following are map functions:

<a href="#">MapOpen</a>	Create a new or open an existing map.
<a href="#">MapClear</a>	Clear all entries in a map.
<a href="#">MapClose</a>	Deletes a previously created map.
<a href="#">MapExists</a>	Check if the map exists using a returned error code.
<a href="#">MapKeyCount</a>	Retrieves the number of keys in a map.
<a href="#">MapKeyDelete</a>	Delete a key and its value from a map.
<a href="#">MapKeyExists</a>	Check if a key exists in a map.
<a href="#">MapKeyFirst</a>	Retrieves the first key in a maps so that all the keys in

	a map can be discovered.
MapKeyNext	Retrieves the next key after the supplied key in a map so that all the keys in a map can be discovered.
MapValueGet	Retrieves the value from a key in a map.
MapValueSet	Sets the value of a key in a map.
MapValueSetQuality	Sets the quality of a value in a map.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## MapClear

Clears all entries in a map and returns the error status.

## Syntax

**INT MapClear(STRING sMapName)**

*sMapName:*

Name of the map to be cleared.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

If the map name is empty an “Invalid argument” error (274) will be raised and returned.

If the map does not exist a “Record not found” error (536) will be returned.

## Example

```
! Demonstrate MapClear.
STRING sMapName = MapOpen();
! sMapName is a randomly generated text that uniquely identifies the map
INT iMapValueSetResult = MapValueSet(sMapName, "SomeKey", "SomeValue");
! iMapValueSetResult will be set to 0 (NO ERROR)
INT iMapKeyCountResultExpectOne = MapKeyCount(sMapName);
! iMapKeyCountResultExpectOne will be set to 1
INT iMapClearResult = MapClear(sMapName);
! iMapClearResult will be set to 0 (NO ERROR)
INT iMapKeyCountResultExpectZero = MapKeyCount(sMapName);
! iMapKeyCountResultExpectZero will be set to 0
INT iMapCloseResult = MapClose(sMapName);
! iMapCloseResult will be set to 0 (NO ERROR)
```

```
! The map that is identified by sMapName is now closed and unavailable
```

## Related Functions

[MapOpen](#), [MapExists](#), [MapKeyCount](#), [MapKeyDelete](#), [MapKeyExists](#), [MapKeyFirst](#), [MapViewGet](#), [MapKeyNext](#),  
[MapViewSet](#), [MapViewSetQuality](#)

## See Also

[Map Functions](#)

### MapClose

Closes a map and returns the error status.

---

**Note:** There is no limit on the number of maps or map items you can create. However, a large number of maps may require a lot of memory. It is recommended that maps created in custom Cicode are destroyed using MapClose when they are no longer needed.

---

**Note:** If the map was created with a close callback, that callback is run before the map is internally closed, so that the map name can be used in the closed callback.

---

## Syntax

```
INT MapClose(STRING sMapName)
```

*sMapName*:

Name of the map to be closed.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

If the map name is empty an "Invalid argument" error (274) will be raised and returned.

If the map does not exist a "Record not found" error (536) will be returned.

## Example

```
! Demonstrate MapClose.  
STRING sMapName = MapOpen();  
! sMapName is a randomly generated text that uniquely identifies the map  
INT iMapCloseResult = MapClose(sMapName);  
! iMapCloseResult will be set to 0 (NO ERROR)  
! The map that is identified by sMapName is now closed and unavailable
```

## Related Functions

[MapClear](#), [MapOpen](#), [MapKeyCount](#), [MapKeyDelete](#), [MapKeyExists](#), [MapKeyFirst](#), [MapViewGet](#), [MapKeyNext](#),  
[MapViewSet](#), [MapViewSet](#)

## See Also

[Map Functions](#)

### MapExists

Checks for the existence of the map.

## Syntax

**INT MapExists(STRING sMapName)**

*sMapName:*

Name of the map.

## Return Value

TRUE if the map exists, or FALSE if not.

If the map or key name is empty an "Invalid argument" error (274) will be raised.

## Example

```
! Demonstrate MapExists.  
STRING sMapName = MapOpen();  
! sMapName is a randomly generated text that uniquely identifies the map  
INT iMapExistsResultExpectTrue = MapExists(sMapName);  
! iMapExistsResultExpectTrue will be set to TRUE (non-zero)  
INT iMapCloseResult = MapClose(sMapName);  
! The map that is identified by sMapName is now closed and unavailable  
INT iMapExistsResultExpectFalse = MapExists(sMapName);  
! iMapExistsResultExpectFalse will be set to FALSE (zero)
```

## Related Functions

[MapClear](#), [MapClose](#), [MapOpen](#), [MapKeyCount](#), [MapKeyDelete](#), [MapKeyExists](#), [MapKeyFirst](#), [MapViewGet](#), [MapKeyNext](#), [MapViewSet](#), [MapViewSet](#)

## See Also

[Map Functions](#)

### MapKeyCount

Use this function to retrieve the number of keys in a map.

## Syntax

```
INT MapKeyCount(STRING sMapName)
```

*sMapName*:

Name of the map.

## Return Value

The number of keys in the map, or a value less than 0 if an error is encountered.

If the map name is empty an "Invalid argument" error (274) will be raised.

## Example

```
! Demonstrate MapKeyCount.  
STRING sMapName = MapOpen();  
! sMapName is a randomly generated text that uniquely identifies the map  
INT iMapKeyCountResultExpectZero = MapKeyCount(sMapName);  
! iMapKeyCountResultExpectZero will be set to 0  
INT iMapValueSetResult = MapValueSet(sMapName, "SomeKey", "SomeValue");  
INT iMapKeyCountResultExpectOne = MapKeyCount(sMapName);  
! iMapKeyCountResultExpectOne will be set to 1  
INT iMapCloseResult = MapClose(sMapName);  
! The map that is identified by sMapName is now closed and unavailable
```

## Related Functions

[MapClear](#), [MapClose](#), [MapOpen](#), [MapKeyDelete](#), [MapKeyExists](#), [MapKeyFirst](#), [MapViewGet](#), [MapKeyNext](#), [MapViewSet](#), [MapViewSet](#)

## See Also

[Map Functions](#)

## MapKeyDelete

Use this function to delete a key and value from a map.

## Syntax

```
INT MapKeyDelete(STRING sMapName, STRING sKeyName)
```

*sMapName*:

Name of map.

*sKeyName*:

Name of key and value to be deleted.

## Return Value

0 (zero) if successful, otherwise an error code is returned.  
If the map name is empty an "Invalid argument" error (274) will be raised and returned.  
If the map or the key does not exist a "Record not found" error (536) will be returned.

## Example

```
! Demonstrate MapKeyDelete.
STRING sMapName = MapOpen();
! sMapName is a randomly generated text that uniquely identifies the map
INT iMapValueSetResult = MapValueSet(sMapName, "SomeKey", "SomeValue");
INT iMapKeyCountResultExpectOne = MapKeyCount(sMapName);
! iMapKeyCountResultExpectOne will be set to 1
INT iMapKeyDeleteResult = MapKeyDelete(sMapName, "SomeKey");
INT iMapKeyCountResultExpectZero = MapKeyCount(sMapName);
! iMapKeyCountResultExpectZero will be set to 0
INT iMapCloseResult = MapClose(sMapName);
! The map that is identified by sMapName is now closed and unavailable
```

## Related Functions

[MapClear](#), [MapClose](#), [MapOpen](#), [MapKeyCount](#), [MapKeyExists](#), [MapKeyFirst](#), [MapValueGet](#), [MapKeyNext](#), [MapValueSet](#), [MapValueSet](#)

## See Also

[Map Functions](#)

## MapKeyExists

Use this function to check if a key exists in a map.

## Syntax

INT **MapKeyExists**(STRING *sMapName*, STRING *sKeyName*)

*sMapName*:

Name of map to get the value from.

*sKeyName*:

Name of the key to get the value from.

## Example

```
! Demonstrate MapKeyExists.
STRING sMapName = MapOpen();
! sMapName is a randomly generated text that uniquely identifies the map
```

```
INT iMapKeyExistsResultExpectFalse = MapKeyExists(sMapName, "SomeKey");
! iMapKeyExistsResultExpectFalse will be set to FALSE (zero)
INT iMapViewSetResult = MapValueSet(sMapName, "SomeKey", "SomeValue");
INT iMapKeyExistsResultExpectTrue = MapKeyExists(sMapName, "SomeKey");
! iMapKeyExistsResultExpectTrue will be set to TRUE (non-zero)
INT iMapCloseResult = MapClose(sMapName);
! The map that is identified by sMapName is now closed and unavailable
```

## Return Value

TRUE if the map exists, or FALSE if not.

If the map name is empty an "Invalid argument" error (274) will be raised.

## Related Functions

[MapClear](#), [MapClose](#), [MapOpen](#), [MapKeyCount](#), [MapKeyDelete](#), [MapKeyFirst](#), [MapViewGet](#), [MapKeyNext](#), [MapViewSet](#), [MapViewSet](#)

## See Also

[Map Functions](#)

## MapKeyFirst

Use this function to retrieve the first key in a map so that all the keys in a map can be discovered.

## Syntax

STRING **MapKeyFirst**(STRING *sMapName*)

*sMapName*:

Name of map to get the value from.

## Return Value

The first key in the map, if there are no keys in the map an empty string is returned.

If the map name is empty an "Invalid argument" error (274) will be raised.

**Note:** This key can then be used in [MapKeyNext](#) to discover all the property keys in the map.

## Example

```
! Demonstrate MapKeyFirst.
STRING sMapName = MapOpen();
! sMapName is a randomly generated text that uniquely identifies the map
INT iMapSetValueResult1 = MapSetValue(sMapName, "somekey", "somevalue");
INT iMapSetValueResult2 = MapSetValue(sMapName, "SomeKey", "SomeValue");
STRING sMapKeyFirstResult = MapKeyFirst(sMapName);
```

```
! sMapKeyFirstResult will be set to "SomeKey"
! This shows that the keys are visited in alphabetical order, and that
! the keys are case sensitive
INT iMapCloseResult = MapClose(sMapName);
! The map that is identified by sMapName is now closed and unavailable
```

## Related Functions

[MapClear](#), [MapClose](#), [MapOpen](#), [MapKeyCount](#), [MapKeyDelete](#), [MapKeyExists](#), [MapViewGet](#), [MapKeyNext](#), [MapViewSet](#), [MapViewSet](#)

## See Also

[Map Functions](#)

### MapKeyNext

Use this function to retrieve the next key after the supplied key in a map so that all the keys in a map can be discovered

## Syntax

```
STRING MapKeyNext(STRING sMapName, STRING sKeyName)
```

*sMapName*:

Name of map to retrieve the value from.

*sKeyName*:

Name of the key to retrieve the value for.

## Return Value

The next key if successful or, an empty string if no next key exists.

If the map name is empty an "Invalid argument" error (274) will be raised.

## Example

```
! Demonstrate MapKeyNext.
STRING sMapName = MapOpen();
! sMapName is a randomly generated text that uniquely identifies the map
INT iMapSetValueResult1 = MapSetValue(sMapName, "somekey", "somevalue");
INT iMapSetValueResult2 = MapSetValue(sMapName, "SomeKey", "SomeValue");
STRING sMapKeyFirstResult = MapKeyFirst(sMapName);
! sMapKeyFirstResult will be set to "SomeKey"
STRING sMapKeyNextResult = MapKeyNext(sMapName, sMapKeyFirstResult);
! sMapKeyNextResult will be set to "somekey"
! This shows that the keys are visited in ordinal order, and that
! the keys are case sensitive
INT iMapCloseResult = MapClose(sMapName);
```

! The map that is identified by sMapName is now closed and unavailable

## Related Functions

[MapClear](#), [MapClose](#), [MapOpen](#), [MapKeyCount](#), [MapKeyDelete](#), [MapKeyExists](#), [MapKeyFirst](#), [MapViewGet](#), [MapViewSet](#), [MapViewSet](#)

## See Also

[Map Functions](#)

### MapOpen

Use this function to create a new map or to open an existing map.

**Note:** There is no limit on the number of maps or map items you can create. However, a large number of maps may require a lot of memory. It is recommended that maps created in custom Cicode are destroyed using MapClose when they are no longer needed.

## Syntax

STRING **MapOpen**(STRING *sMapName*, INT *nOpenMode*, STRING *sCloseCallback*)

*sMapName*:

Name of the map to create or open. The default name is an empty string. If name is empty, it will generate an available random name.

*nOpenMode*:

Indicates the open or create mode of the map. Modes include:

- Mode 0 - Create a new map. If a map with the same name exists an "Out of handles" error (271) will be raised.
- Mode 1 - Opens an existing map. if there is no map with that name a "Record not found" error (536) will be raised.
- Mode 2 – Open an existing map , or creates a new map if the map name does not exist. The error codes from modes 0 and 1 can occur.

*sCloseCallback*:

The function to call when this instance is closed.

## Return Value

Name of map if successful, otherwise an empty string will be returned.

If the map name is empty an "Invalid argument" error (274) will be raised.

If the open mode is less than 0 or greater than 2, an "Invalid argument" error (274) will be raised.

If an empty map name is supplied along with open mode 1, an "Invalid argument" error (274) will be raised.

## Example

```
! Demonstrate MapOpen.
STRING sMapName = MapOpen();
! sMapName is a randomly generated text that uniquely identifies the map
INT iMapCloseResult = MapClose(sMapName);
! The map that is identified by sMapName is now closed and unavailable

! Demonstrate MapOpen with a close callback
STRING sMapName = MapOpen("", 0, "MyCloseCallback");
! sMapName is a randomly generated text that uniquely identifies the map
INT iMapCloseResult = MapClose(sMapName);
! The map that is identified by sMapName is now closed and unavailable
! The function MyCloseCallback is called, but it may or may not be
! completed before the MapClose function returns.

! This is a MapClose callback function
! The callback must take one string argument which is the name of the
! map that is being closed.
FUNCTION MyCloseCallback(STRING sMapName)
! sMapName is a valid map for the duration of the callback, at the
! exit of the callback the memory is released and any subsequent
! use of the map will result in an error.
END
```

## Related Functions

[MapClear](#), [MapClose](#), [MapKeyCount](#), [MapKeyDelete](#), [MapKeyExists](#), [MapKeyFirst](#), [MapViewGet](#), [MapViewNext](#),  
[MapViewSet](#), [MapViewSet](#)

## See Also

[Map Functions](#)

### MapViewGet

Use this function to retrieve the value from a key in a map.

## Syntax

VARIANT **MapViewGet**(STRING *sMapName*, STRING *sKeyName*)

*sMapName*:

Name of map.

*sKeyName*:

Name of the key to retrieve the value from.

## Return Value

Value returned is a VARIANT that also contains QUALITY and TIMESTAMP components, otherwise an error code is returned.

If the map or key does not exist an "Invalid argument" error (274) is raised.

## Example

```
! Demonstrate MapValueGet.  
STRING sMapName = MapOpen();  
! sMapName is a randomly generated text that uniquely identifies the map  
INT iMapValueSetResult = MapValueSet(sMapName, "SomeKey", "SomeValue");  
STRING sMapViewGetResult = MapValueGet(sMapName, "SomeKey");  
! sMapViewGetResult will be set to "SomeValue"  
INT iMapCloseResult = MapClose(sMapName);  
! The map that is identified by sMapName is now closed and unavailable
```

## Related Functions

[MapClear](#), [MapClose](#), [MapOpen](#), [MapKeyCount](#), [MapKeyDelete](#), [MapKeyExists](#), [MapKeyFirst](#), [MapKeyNext](#), [MapViewSet](#), [MapViewSet](#)

## See Also

[Map Functions](#)

## MapViewSet

Use this function to set a value of a map key.

## Syntax

INT **MapViewSet**(STRING *sMapName*, STRING *sKeyName*, VARIANT *Value* INT *nSetMode*)

*sMapName*:

Name of the map to create or open.

*sKeyName*:

Name of the key to set.

*Value*:

Name of the value to be set

*nSetMode*:

0 - Create a new property

1- Override an existing property

2 - Assign the value whether the key already exists or not. This is the default mode.

## Return Value

If the value is set correctly, the result is "No Error" (0), otherwise an error code is returned.  
If the map name or key name is empty, an "Invalid argument" error (274) will be raised and returned.  
If the map does not exist a "Record not found" error (536) will be returned.  
If the key does not exist and set mode is 1, a "Record not found" error (536) will be returned.  
If the key exists and set mode is 0, an "Out of handles" error (271) will be returned.

## Example

```
! Demonstrate MapValueSet.
STRING sMapName = MapOpen();
! sMapName is a randomly generated text that uniquely identifies the map
INT iMapValueSetResult = MapValueSet(sMapName, "SomeKey", "SomeValue");
STRING sMapValueGetResult = MapValueGet(sMapName, "SomeKey");
! sMapValueGetResult will be set to "SomeValue"
INT iMapCloseResult = MapClose(sMapName);
! The map that is identified by sMapName is now closed and unavailable
```

## Related Functions

[MapClear](#), [MapClose](#), [MapOpen](#), [MapKeyCount](#), [MapKeyDelete](#), [MapKeyExists](#), [MapKeyFirst](#), [MapValueGet](#), [MapKeyNext](#), [MapValueSet](#)

## See Also

[Map Functions](#)

## MapViewSetQuality

Use this function to set the quality of a property in a map.

## Syntax

**INT MapValueSetQuality(STRING *sMapName*, STRING *sKeyName*, QUALITY *Quality*)**

*sMapName*:

Name of the map.

*sKeyName*:

Name of the key to override the quality for

*Quality*:

Quality to override in an existing value. There is no default value. The QUALITY type is created using the QualityCreate function.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

If the map name or key name is empty, an "Invalid argument" error (274) will be raised and returned.

If the map does not exist or the key does not exist within the map a "Record not found" error (536) will be returned

## Example

```
! Demonstrate MapValueSetQuality.  
STRING sMapName = MapOpen();  
! sMapName is a randomly generated text that uniquely identifies the map  
INT iMapViewSetResult = MapValueSet(sMapName, "SomeKey", "SomeValue");  
QUALITY qVariableQualityResultExpectGood =  
VariableQuality(MapValueGet(sMapName, "SomeKey"));  
! qVariableQualityResultExpectGood is set to GOOD QUALITY (checked below)  
INT iQualityIsGoodResultExpectTrue =  
QualityIsGood(qVariableQualityResultExpectGood);  
! iQualityIsGoodResultExpectTrue is set to TRUE (non-zero)  
INT iMapViewSetQualityResult =  
MapViewSetQuality(sMapName, "SomeKey", QualityCreate(QUAL_BAD));  
! iMapViewSetQualityResult will be set to 0 (NO ERROR)  
QUALITY qVariableQualityResultExpectBad =  
VariableQuality(MapValueGet(sMapName, "SomeKey"));  
! qVariableQualityResultExpectBad is set to BAD QUALITY (checked below)  
INT iQualityIsGoodResultExpectFalse =  
QualityIsGood(qVariableQualityResultExpectBad);  
! iQualityIsGoodResultExpectFalse is set to FALSE (zero)  
INT iMapCloseResult = MapClose(sMapName);  
! The map that is identified by sMapName is now closed and unavailable
```

## Related Functions

[MapClear](#), [MapClose](#), [MapOpen](#), [MapKeyCount](#), [MapKeyDelete](#), [MapKeyExists](#), [MapKeyFirst](#), [MapViewGet](#), [MapKeyNext](#), [MapViewSet](#)

## See Also

[Map Functions](#)

## Math and Trigonometry Functions

Following are mathematical or trigonometrical functions:

Abs	Gets the absolute value of a number.
ArcCos	Gets the arccosine of an angle.
ArcSin	Gets the arcsine of an angle.

ArcTan	Gets the arctangent of an angle.
Cos	Gets the cosine of an angle.
DegToRad	Converts an angle from degrees to radians.
Exp	Raises e to the power of a number.
Fact	Gets the factorial of a number.
HighByte	Gets the high-order byte of a two-byte integer.
HighWord	Gets the high-order word of a four-byte integer.
Ln	Gets the natural logarithm of a number.
Log	Gets the base 10 logarithm of a number.
LowByte	Gets the low-order byte of a two-byte integer.
LowWord	Gets the low-order word of a four-byte integer.
Max	Gets the higher of two numbers.
Min	Gets the lower of two numbers.
Pi	Gets the value of pi.
Pow	Raises a number to the power of another number.
RadToDeg	Converts an angle from radians to degrees.
Rand	Gets a random number.
Round	Rounds a number.
Sign	Gets the sign of a number.
Sin	Gets the sine of an angle.
Sqrt	Gets the square root of a number.
Tan	Gets the tangent of an angle.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## Abs

Calculates the absolute (positive) value of a number. The absolute value of a number is the number without its sign.

## Syntax

**Abs(*Number*)**

*Number*:

Any number.

## Return Value

The absolute (positive) value of *Number*.

## Related Functions

[Sign](#)

## Example

```
Variable=Abs(-67);
! Sets Variable to 67.
Variable=Abs(67);
! Sets Variable to 67.
```

## See Also

[Math and Trigonometry Functions](#)

## ArcCos

Calculates the arccosine of an angle.

## Syntax

**ArcCos(*Number*)**

*Number*

The cosine of the angle.

## Return Value

The arccosine (the angle, in radians) of *Number*.

## Related Functions

[Cos](#)

## Example

```
Variable=ArcCos(0.4);  
! Sets Variable to 1.1592...
```

## See Also

[Math and Trigonometry Functions](#)

## ArcSin

Calculates the arcsine of an angle.

## Syntax

**ArcSin(*Number*)**

*Number:*

The sine of the angle.

## Return Value

The arcsine (the angle, in radians) of *Number*.

## Related Functions

[Sin](#)

## Example

```
Variable=ArcSin(1);  
! Sets Variable to 1.5707...
```

## See Also

[Math and Trigonometry Functions](#)

## ArcTan

Calculates the arctangent of an angle.

## Syntax

**ArcTan(*Number*)**

*Number*:

The tangent of the angle.

## Return Value

The arctangent (the angle, in radians) of *Number*.

## Related Functions

[Tan](#)

## Example

```
Variable=ArcTan(0.4);  
! Sets Variable to 0.3805...
```

## See Also

[Math and Trigonometry Functions](#)

## Cos

Calculates the trigonometric cosine of an angle.

## Syntax

**Cos(*Angle*)**

*Angle*:

Any angle (in radians).

## Return Value

The cosine of *Angle*.

## Related Functions

[ArcCos](#)

## Example

```
Variable=Cos(0.7854);
```

```
! Sets Variable to 0.7071...
```

## See Also

[Math and Trigonometry Functions](#)

### DegToRad

Converts an angle from degrees to radians.

## Syntax

**DegToRad(*Angle*)**

*Angle*:

Any angle (in degrees).

## Return Value

The angle in radians.

## Related Functions

[RadToDeg](#)

## Example

```
Variable=DegToRad(180);  
! Sets Variable to 3.1415... (pi).
```

## See Also

[Math and Trigonometry Functions](#)

### Exp

Calculates the exponential of a number (natural logarithm base e).

## Syntax

**Exp(*Number*)**

*Number*:

Any number.

## Return Value

The exponential of *Number* (to the base e).

## Related Functions

[Log](#)

## Example

```
Variable=Exp(1);  
! Sets Variable to 2.7182...
```

## See Also

[Math and Trigonometry Functions](#)

## Fact

Calculates the factorial of a number.

## Syntax

**Fact(*Number*)**

*Number*:

Any number.

## Return Value

The factorial of *Number*.

## Example

```
Variable=Fact(6);  
! Sets Variable to 720 (that is 720=1x2x3x4x5x6).
```

## See Also

[Math and Trigonometry Functions](#)

## HighByte

Gets the high-order byte of a two-byte integer.

## Syntax

**HighByte**(*TwoByteInteger*)

*TwoByteInteger*:

A two-byte integer.

## Return Value

The high-order byte (that is | X | - | )

## Related Functions

[LowByte](#), [HighWord](#), [LowWord](#)

## Example

```
Variable=HighByte(0x5678);  
! Sets Variable to 0x56.
```

## See Also

[Math and Trigonometry Functions](#)

## HighWord

Gets the high-order word of a four-byte integer.

## Syntax

**HighWord**(*FourByteInteger*)

*FourByteInteger*:

A four-byte integer.

## Return Value

The high-order word (that is | X | X | - | - | )

## Related Functions

[LowWord](#), [HighByte](#), [LowByte](#)

## Example

```
Variable=HighWord(0x12345678);
```

```
! Sets Variable to 0x1234.
```

## See Also

[Math and Trigonometry Functions](#)

### Ln

Calculates the natural (base e) logarithm of a number.

## Syntax

**Ln(Number)**

*Number:*

Any number.

## Return Value

The natural (base e) logarithm of *Number*.

## Related Functions

[Log](#)

## Example

```
Variable=Ln(2);  
! Sets Variable to 0.6931...
```

## See Also

[Math and Trigonometry Functions](#)

### Log

Calculates the base 10 logarithm of a number.

## Syntax

**Log(Number)**

*Number:*

Any number.

## Return Value

The base 10 logarithm of *Number*.

## Related Functions

[Ln](#)

## Example

```
Variable=Log(100);  
! Sets Variable to 2 (that is 100=10 to the power of 2).
```

## See Also

[Math and Trigonometry Functions](#)

## LowByte

Gets the low-order byte of a two-byte integer.

## Syntax

**LowByte**(*TwoByteInteger*)

*TwoByteInteger*:

A two-byte integer.

## Return Value

The low-order byte (that is | - | X |)

## Related Functions

[HighByte](#), [LowWord](#), [HighWord](#)

## Example

```
Variable=LowByte(0x5678);  
! Sets Variable to 0x78.
```

## See Also

[Math and Trigonometry Functions](#)

## LowWord

Gets the low-order word of a four-byte integer.

## Syntax

**LowWord(*FourByteInteger*)**

*FourByteInteger*:

A four-byte integer.

## Return Value

The low-order word (that is | - | - | X | X |)

## Related Functions

[HighByte](#), [LowByte](#), [HighWord](#)

## Example

```
Variable=LowWord(0x12345678);  
! Sets Variable to 0x5678
```

## See Also

[Math and Trigonometry Functions](#)

## Max

Gets the higher of two numbers.

## Syntax

**Max(*Number1*, *Number2*)**

*Number1*:

The first number.

*Number2*:

The second number.

## Return Value

The higher of numbers *Number1* and *Number2*.

## Related Functions

[Min](#)

## Example

```
Variable=Max(24,12);  
! Sets Variable to 24.
```

## See Also

[Math and Trigonometry Functions](#)

## Min

Returns the lower of two numbers.

## Syntax

**Min**(*Number1*, *Number2*)

*Number1*:

The first number.

*Number2*:

The second number.

## Return Value

The lower of numbers *Number1* and *Number2*.

## Related Functions

[Max](#)

## Example

```
Variable=Min(24,12);  
! Sets Variable to 12.
```

## See Also

[Math and Trigonometry Functions](#)

## Pi

Gets the value of pi (the ratio of the circumference of a circle to its diameter).

## Syntax

**Pi()**

## Return Value

The value of pi.

## Example

```
Variable=Pi();
! Sets Variable to 3.1415...
```

## See Also

[Math and Trigonometry Functions](#)

## Pow

Calculates  $x$  to the power of  $y$ .

## Syntax

**Pow( $X, Y$ )**

$X$ :

The base number.

$Y$ :

The exponent.

## Return Value

$X$  to the power of  $Y$ .

## Related Functions

[Exp](#)

## Example

```
Variable=Pow(5,3);
```

```
! Sets Variable to 125.
```

## See Also

[Math and Trigonometry Functions](#)

### RadToDeg

Converts an angle from radians to degrees.

## Syntax

**RadToDeg(*Angle*)**

*Angle*:

Any angle (in degrees).

## Return Value

The angle in degrees.

## Related Functions

[DegToRad](#)

## Example

```
Variable=RadToDeg(Pi());  
! Sets Variable to 180.
```

## See Also

[Math and Trigonometry Functions](#)

### Rand

Generates a random number between 0 and a specified maximum number less one.

The **Rand** function is zero-based, so the resultant number generated will range from zero to one less than the number provided in the *Maximum* argument.

## Syntax

**Rand(*Maximum*)**

*Maximum*:

The maximum number. This number needs to be between 2 and 32767 (inclusive).

## Return Value

A random number of integer type.

## Example

```
Variable=Rand(101);
! Sets Variable to a random number from 0 to 100.
// To create a random number between 0 and 1 with 2 decimal places,
divide the above variable by 100, as shown here: //
Variable = Variable/100;
```

## See Also

[Math and Trigonometry Functions](#)

## Round

Rounds a number to a specified number of decimal places.

## Syntax

**Round**(*Number*, *Places*)

*Number*:

The floating-point number to round.

*Places*:

The number of decimal places.

## Return Value

The number rounded to *Places* decimal places.

## Example

```
IF round(rTag,2) = 15.60 THEN
    tag2 = 5
END
```

## See Also

[Math and Trigonometry Functions](#)

## Sign

Gets the sign of a number.

## Syntax

**Sign(Number)**

*Number:*

Any number.

## Return Value

The sign of *Number*.

## Related Functions

[Abs](#)

## Example

```
Variable=Sign(100);
! Sets Variable to 1.
Variable=Sign(-300);
! Sets Variable to -1.
Variable=Sign(0);
! Sets Variable to 0.
```

## See Also

[Math and Trigonometry Functions](#)

## Sin

Calculates the trigonometric sine of an angle.

## Syntax

**Sin(Angle)**

*Angle:*

Any angle (in radians).

## Return Value

The sine of *Angle*.

## Related Functions

[ArcSin](#)

## Example

```
Variable=Sin(0.7854);  
! Sets Variable to 0.7071...
```

## See Also

[Math and Trigonometry Functions](#)

## Sqrt

Gets the square root of a number.

## Syntax

**Sqrt(*Number*)**

*Number*:

Any positive number.

## Return Value

The square root of *Number*.

## Related Functions

[Pow](#)

## Example

```
Variable=Sqrt(4);  
! Sets Variable to 2.
```

## See Also

[Math and Trigonometry Functions](#)

## Tan

Calculates the trigonometric tangent of an angle.

## Syntax

**Tan(*Angle*)**

*Angle*:

Any angle (in degrees).

## Return Value

The tan of *Angle*.

## Related Functions

[ArcTan](#)

## Example

```
Variable=Tan(1);  
! Sets Variable to 1.5574
```

## See Also

[Math and Trigonometry Functions](#)

## Menu Functions

The following functions allow you to access the contents of the menu configuration database in a tree-like format, and change the contents of the tree. Be reminded that changes made to the menu tree will not be persisted back to the menu configuration database.

<a href="#">MenuGetChild</a>	Returns the handle to the child node with the specified name.
<a href="#">MenuGetFirstChild</a>	Returns the handle to the first child of a menu node.
<a href="#">MenuGetGenericNode</a>	Returns the handle to the base node of the menu tree for the generic pages.
<a href="#">MenuGetNextChild</a>	Returns the next node that shares the same parent.
<a href="#">MenuGetNodeByPath</a>	Returns a menu handle corresponding to a menu item.
<a href="#">MenuGetPageNode</a>	Returns the handle to the base node of the menu tree of a specified page.
<a href="#">MenuGetParent</a>	Returns the parent node of the menu item.
<a href="#">MenuGetPrevChild</a>	Returns the previous node that shares the same parent.
<a href="#">MenuGetWindowNode</a>	Returns the handle to the root node for a given window.

<a href="#">MenuNodeAddChild</a>	Dynamically adds a new item to the menu at runtime.
<a href="#">MenuNodeGetCurr</a>	Gets the handle of a menu node when an associated command is called.
<a href="#">MenuNodeGetDepth</a>	Returns the depth of a specified menu node within a menu hierarchy.
<a href="#">MenuNodeGetProperty</a>	Return the item value of the specified menu node.
<a href="#">MenuNodeGetTargetPage</a>	Returns the target page for a specified menu node.
<a href="#">MenuNodeHasCommand</a>	Checks whether the menu node has a valid Cicode command associated with it.
<a href="#">MenuNodeIsDisabled</a>	Checks whether the menu node is disabled by evaluating its DisabledWhen Cicode expression.
<a href="#">MenuNodeGetExpanded</a>	Returns the expansion state value of the specified menu node.
<a href="#">MenuNodeIsHidden</a>	Checks whether the menu node is hidden by evaluating its HiddenWhen Cicode expression.
<a href="#">MenuNodeRemove</a>	Remove the menu node from the menu tree.
<a href="#">MenuNodeRunCommand</a>	Run the associated command for a menu node.
<a href="#">MenuNodeSetDisabledWhen</a>	Set the DisabledWhen expression for a newly added node.
<a href="#">MenuNodeSetExpanded</a>	Sets the expansion state value of the specified menu node.
<a href="#">MenuNodeSetHiddenWhen</a>	Set the HiddenWhen expression for a newly added node.
<a href="#">MenuNodeSetProperty</a>	Set the item value of the specified menu node.
<a href="#">MenuReload</a>	Reload base Menu Configuration from the compiled database.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

### MenuGetChild

Returns the handle to the child node with the specified name.

## Syntax

**MenuGetChild(*hParent*, *sName*)**

*hParent*:

Handle to the parent node in the menu tree.

*sName*:

The name of the child Menu node requested.

## Return Value

The handle of the child node with the requested name, or -1 if unsuccessful.

## Related Functions

[MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#), [MenuGetParent](#),  
[MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#), [MenuNodeGetDepth](#),  
[MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#), [MenuNodeHasCommand](#),  
[MenuNodeIsDisabled](#), [MenuNodeIsHidden](#), [MenuNodeRemove](#), [MenuNodeRunCommand](#),  
[MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## See Also

[Menu Functions](#)

## MenuGetFirstChild

Returns the handle to the first child of a menu node.

## Syntax

**MenuGetFirstChild(*hNode*)**

*hNode*:

The handle to the parent node in the menu tree.

## Return Value

The handle to the first child node of a menu node, or -1 if unsuccessful.

## Related Functions

[MenuGetChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#), [MenuGetParent](#),  
[MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#), [MenuNodeGetDepth](#),  
[MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#), [MenuNodeHasCommand](#),  
[MenuNodeIsDisabled](#), [MenuNodeIsHidden](#), [MenuNodeRemove](#), [MenuNodeRunCommand](#),

[MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## See Also

[Menu Functions](#)

### MenuGetGenericNode

Returns the handle to the base node of the menu tree for the generic pages. Its child nodes represent the menu items that do not have a page specified in the menu configuration database.

## Syntax

**MenuGetGenericNode([*bCreate*])**

*bCreate*:

Determines if the node should be created if it does not exist. Defaults to 0, do not create.

## Return Value

The handle to the base node of the menu tree, or -1 if it cannot find the node.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetNextChild](#), [MenuGetPageNode](#), [MenuGetParent](#),  
[MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#), [MenuNodeGetDepth](#),  
[MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#), [MenuNodeHasCommand](#),  
[MenuNodeIsDisabled](#), [MenuNodeIsHidden](#), [MenuNodeRemove](#), [MenuNodeRunCommand](#),  
[MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## See Also

[Menu Functions](#)

### MenuGetNextChild

Returns the next node that shares the same parent.

## Syntax

**MenuGetNextChild(*hChild*)**

*hChild*:

Handle to the current node in the menu tree

## Return Value

The handle to next node that shares the same parent, or -1 if unsuccessful.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetPageNode](#), [MenuGetParent](#),  
[MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#), [MenuNodeGetDepth](#),  
[MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#), [MenuNodeHasCommand](#),  
[MenuNodeIsDisabled](#), [MenuNodeIsHidden](#), [MenuNodeRemove](#), [MenuNodeRunCommand](#),  
[MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## See Also

[Menu Functions](#)

## MenugetNodeByPath

Returns a menu handle corresponding to a menu item expressed as a string path in the format  
<level>.<level>.<level>.<level>.

## Syntax

MenugetNodeByPath(INT *hMenuNode*, STRING *sMenuPath*)

*hMenuNode*

The menu to which the specified menu path belongs. Root menu node returned from [MenuGetPageNode\(\)](#).

*sMenuPath*

The menu path for which to get the handle.

## Return Value

Returns a menu handle for use with other Plant SCADA menu functions.

## Example

```
MenugetNodeByPath(hNavigationMenu, "MyPlant.MyArea.Page1")
```

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#),  
[MenuGetParent](#), [MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#),  
[MenuNodeGetDepth](#), [MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#),  
[MenuNodeHasCommand](#), [MenuNodeIsDisabled](#), [MenuNodeIsHidden](#), [MenuNodeRemove](#),  
[MenuNodeRunCommand](#), [MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#),  
[MenuReload](#)

## See Also

[Menu Functions](#)

### MenuGetPageNode

Returns the handle to the menu tree for a specified page. Its child nodes represent the menu items that have the particular page specified in the menu configuration database.

## Syntax

**MenuGetPageNode(*sPage* [, *nMode*])**

*sPage*:

The name of the page for which to return the menu tree handle.

*nMode*:

The mode to use for generating the menu handle for the page:

0 - Return the base node for the menu tree for the specified page if it exists, otherwise return -1 (do not create). Default value.

1 - Return the base node for the menu tree for the specified page. Create the menu tree if it does not already exist.

2 - Return a new duplicate of the menu tree for the specified page if it exists, otherwise return -1. Automatically dispose of the menu tree on page close.

3 - Return a new duplicate of the menu tree for the specified page if it exists, otherwise return -1. Do not automatically dispose of the menu tree. Call [MenuNodeRemove\(\)](#) to remove menu tree.

## Return Value

The handle of the menu tree, or -1 if no menu tree exists for a specified page.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetParent](#), [MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#), [MenuNodeGetDepth](#), [MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#), [MenuNodeHasCommand](#), [MenuNodeIsDisabled](#), [MenuNodeIsHidden](#), [MenuNodeRemove](#), [MenuNodeRunCommand](#), [MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## See Also

[Menu Functions](#)

### MenuGetParent

Returns the parent node of the menu item.

## Syntax

**MenuGetParent(*hNode*)**

*hNode*:

Handle to the current node in the menu tree.

## Return Value

The handle to parent menu node of the given menu item, or -1 if unsuccessful.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#),  
[MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#), [MenuNodeGetDepth](#),  
[MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#), [MenuNodeHasCommand](#),  
[MenuNodeIsDisabled](#), [MenuNodeIsHidden](#), [MenuNodeRemove](#), [MenuNodeRunCommand](#),  
[MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## See Also

[Menu Functions](#)

## MenuGetPrevChild

Returns the previous node that shares the same parent.

## Syntax

**MenuGetPrevChild(*hChild*)**

*hChild*:

Handle to the current node in the menu tree.

## Return Value

The handle to previous node that shares the same parent, or -1 if unsuccessful.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#),  
[MenuGetParent](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#), [MenuNodeGetDepth](#),  
[MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#), [MenuNodeHasCommand](#),  
[MenuNodeIsDisabled](#), [MenuNodeIsHidden](#), [MenuNodeRemove](#), [MenuNodeRunCommand](#),  
[MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## See Also

[Menu Functions](#)

### MenuGetWindowNode

Returns the handle to the root node for a given window. This menu node is dynamically created for each page instance. Its child nodes represent the menu items for the generic pages merged with the ones specific to this page.

## Syntax

**MenuGetWindowNode(*hWin*)**

*hWin*:

The window number of the desired window. You can call WinNumber() to get the window number of the page that calls this Cicode function.

## Return Value

The handle to the root node of a given window, or -1 if unsuccessful.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#), [MenuGetParent](#), [MenuGetPrevChild](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#), [MenuNodeGetDepth](#), [MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#), [MenuNodeHasCommand](#), [MenuNodeIsDisabled](#), [MenuNodeIsHidden](#), [MenuNodeRemove](#), [MenuNodeRunCommand](#), [MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## See Also

[Menu Functions](#)

### MenuNodeAddChild

Dynamically add a new item to the menu at runtime. Be reminded that the changes are for the current session only and will not be persisted to the \_Pagemen.RDB file.

Be reminded that changes made to the menu tree will not be persisted back to the menu configuration database.

## Syntax

**MenuNodeAddChild(*hParent*, *sName*, *sCommandName* [, *sCommandArgs*] [, *sSymbol*] [, *iOrder*])**

*hParent*:

Handle of the parent node to add the new menu item under.

*sName*:

The string label of the new menu item.

*sCommandName*:

Specifies the name of the Cicode function to run.

*sCommandArgs*:

Specifies the parameters of the Cicode function to run.

*sSymbol*:

The symbol to be associated with the menu item.

*iOrder*:

The relative position in the menu for new item.

## Return Value

The handle of the new node, or -1 if unsuccessful.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#), [MenuGetParent](#), [MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeGetCurr](#), [MenuNodeGetDepth](#), [MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#), [MenuNodeHasCommand](#), [MenuNodeIsDisabled](#), [MenuNodeIsHidden](#), [MenuNodeRemove](#), [MenuNodeRunCommand](#), [MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## See Also

[Menu Functions](#)

## MenuNodeGetCurr

Each menu node can have a Command associated with it when a menu is configured. This function allows you to get the handle of the menu node when this associated command is called.

**Note:** This function should only be used with expressions or Cicode functions invoked from the Command field of a menu configuration record.

## Syntax

**MenuNodeGetCurr( )**

## Return Value

The handle of the menu node.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#), [MenuGetParent](#), [MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetDepth](#), [MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#), [MenuNodeHasCommand](#), [MenuNodeIsDisabled](#), [MenuNodeIsHidden](#), [MenuNodeRemove](#), [MenuNodeRunCommand](#), [MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## Example

```
// Assume the following Cicode function is set to the Command field of a menu configuration record
FUNCTION MyMenuCommand()
    INT hNode = MenuNodeGetCurr(); // Get the current menu node that runs this command
    // Get more information from the menu record
    STRING sTargetPage = MenuNodeGetProperty(hNode, 13);
    // Display the page
    PageDisplay(sTargetPage);
END
```

## See Also

[Menu Functions](#)

## MenuNodeGetDepth

Returns the depth of a specified menu node within a menu hierarchy.

## Syntax

**MenuNodeGetDepth(*hNode*)**

*hNode*:

Handle of the menu node.

## Return Value

The depth of the specified node within the menu hierarchy. For example, 0 indicates it is on the root node of the menu, 1 indicates one level below the root node, and so on.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#), [MenuGetParent](#), [MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#), [MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#), [MenuNodeHasCommand](#), [MenuNodeIsDisabled](#), [MenuNodeIsHidden](#), [MenuNodeRemove](#), [MenuNodeRunCommand](#), [MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## See Also

[Menu Functions](#)

### MenuNodeGetProperty

Return the item value of the specified menu node.

## Syntax

**MenuNodeGetProperty(*hNode*, *iField*)**

*hNode*:

Handle to the current node in the menu tree.

*iField*:

Field for which you want the value:

0 - Name of menu item.

1 - Icon symbol to be associated with the menu item.

2 - Privilege level required to run the command, otherwise the menu item is disabled.

3 - Area level required to run the command, otherwise the menu item is disabled.

4 - Disabled Style. Allows different display style for a disabled menu item.

5 - Checked setting. Whether the menu item will display a check box next to the label.

6 - Width. Specifies the menu item width in pixels.

7 - Comment.

8 - Cluster.

9 - Equipment.

10 - Expansion state value stored by .

11 - Order.

12 - Page.

13 - Target page of the menu item.

101-108 - Custom 1 to Custom 8.

## Return Value

The value for the specified menu node field.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#),  
[MenuGetParent](#), [MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#),  
[MenuNodeGetDepth](#), [MenuNodeGetExpanded](#), [MenuNodeGetTargetPage](#), [MenuNodeHasCommand](#),  
[MenuNodeIsDisabled](#), [MenuNodeIsHidden](#), [MenuNodeRemove](#), [MenuNodeRunCommand](#),  
[MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## See Also

[Menu Functions](#)

### MenuNodeGetTargetPage

Returns the target page for a specified menu node.

## Syntax

**MenuNodeGetTargetPage(*hNode*)**

*hNode*:

Handle of the menu node. If not specified, the handle of the menu node will be determined through its association with the command that is currently being called. Each menu node can have a Command associated with it when a menu is configured.

## Return Value

The name of the target page for the specified node.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#),  
[MenuGetParent](#), [MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#),  
[MenuNodeGetDepth](#), [MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeHasCommand](#),  
[MenuNodeIsDisabled](#), [MenuNodeIsHidden](#), [MenuNodeRemove](#), [MenuNodeRunCommand](#),  
[MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## See Also

[Menu Functions](#)

### MenuNodeHasCommand

Checks whether the menu node has a valid Cicode command associated with it.

## Syntax

**MenuNodeHasCommand(*hNode*)**

*hNode*:

Handle of node to check.

## Return Value

1 if the menu node has a valid Cicode command, 0 if the menu node has no Cicode command.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#),  
[MenuGetParent](#), [MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#),  
[MenuNodeGetDepth](#), [MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#),  
[MenuNodeIsDisabled](#), [MenuNodeIsHidden](#), [MenuNodeRemove](#), [MenuNodeRunCommand](#),  
[MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## See Also

[Menu Functions](#)

## MenuNodeIsDisabled

Checks whether the menu node is disabled by evaluating its DisabledWhen Cicode expression.

## Syntax

**MenuNodeIsDisabled(*hNode*)**

*hNode*:

Handle of node to check.

## Return Value

1 if menu node DisabledWhen expression evaluates to true, 0 if menu node DisabledWhen expression evaluates to false.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#),  
[MenuGetParent](#), [MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#),  
[MenuNodeGetDepth](#), [MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#),  
[MenuNodeHasCommand](#), [MenuNodeIsHidden](#), [MenuNodeRemove](#), [MenuNodeRunCommand](#),  
[MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## Example

```
INT hNode = MenuNodeFirstChild(hParent);
IF (MenuNodeIsDisabled(hNode)) THEN
    ! set the menu item graphic state to disabled
END
```

## See Also

[Menu Functions](#)

### MenuNodeGetExpanded

Gets the expansion state value of the specified menu node in number. The expansion state value is user defined can be any integer value. The value is set by [MenuNodeSetExpanded](#) function and the initial value is zero(0).

## Syntax

**MenuNodeGetExpanded(*hNode*)**

*hNode*:

Handle to the current node in the menu tree.

## Return Value

The expansion state value to set for the Menu node or -1 if unsuccessful.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#), [MenuGetParent](#), [MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#), [MenuNodeGetDepth](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#), [MenuNodeHasCommand](#), [MenuNodeIsDisabled](#), [MenuNodeIsHidden](#), [MenuNodeRemove](#), [MenuNodeRunCommand](#), [MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## Example

```
MenuNodeSetExpanded(100, 99)
//Put 99 as an expansion state of the menu node handle 100
MenuNodeGetExpanded(100)
//This will return 99 that is stored by the previous call
```

## See Also

[Menu Functions](#)

### MenuNodeIsHidden

Checks whether the menu node is hidden by evaluating its HiddenWhen Cicode expression.

## Syntax

**MenuNodeIsHidden(*hNode*)**

*hNode:*

Handle of node to check.

## Return Value

1 if menu node HiddenWhen expression evaluates to true, 0 if menu node HiddenWhen expression evaluates to false.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#), [MenuGetParent](#), [MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#), [MenuNodeGetDepth](#), [MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#), [MenuNodeHasCommand](#), [MenuNodesDisabled](#), [MenuNodeRemove](#), [MenuNodeRunCommand](#), [MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## Example

```
INT hNode = MenuNodeFirstChild(hParent);
IF (MenuNodeIsHidden(hNode)) THEN
    ! set the menu item graphic state to hidden
END
```

## See Also

[Menu Functions](#)

## MenuNodeRemove

Remove the menu node from the menu tree.

Be reminded that changes made to the menu tree will not be persisted back to the menu configuration database.

## Syntax

**MenuNodeRemove(*hNode*)**

*hNode:*

Handle of node to remove.

## Return Value

Zero (0) if node successfully removed. -1 if *hNode* is an invalid menu handle.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#), [MenuGetParent](#), [MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#), [MenuNodeGetDepth](#), [MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#), [MenuNodeHasCommand](#), [MenuNodesDisabled](#), [MenuNodesHidden](#), [MenuNodeRunCommand](#), [MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## See Also

[Menu Functions](#)

### MenuNodeRunCommand

Run the associated command for a menu node.

## Syntax

**MenuNodeRunCommand(*hNode*)**

*hNode*:

Handle of node to run command.

## Return Value

No error(0) on success. Bad handle specified (269) if *hNode* does not refer to a valid node.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#), [MenuGetParent](#), [MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#), [MenuNodeGetDepth](#), [MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#), [MenuNodeHasCommand](#), [MenuNodesDisabled](#), [MenuNodesHidden](#), [MenuNodeRemove](#), [MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## See Also

[Menu Functions](#)

### MenuNodeSetDisabledWhen

Set the DisabledWhen expression for a newly added node. Be aware this function only works for menu nodes added with [MenuNodeAddChild\(\)](#). The DisabledWhen expression may only be set once for a node.

Be reminded that changes made to the menu tree will not be persisted back to the menu configuration database.

## Syntax

**MenuNodeSetDisabledWhen(*hNode*, *sDisabledWhenName* [, *sDisabledWhenArgs*] [, *iDisabledStyle*])**

*hNode*:

Handle of node to run command

*sDisabledWhenName*:

Cicode function for DisabledWhen expression. The function needs to return an INT.

*sDisabledWhenArgs*:

Cicode parameters for DisabledWhen expression. Only supports static arguments.

*iDisabledStyle*:

Disabled Style. Allows different display styles for a disabled menu item.

## Return Value

No Error(0) on success, Bad handle specified (269) if hNode does not refer to a valid node, Invalid argument passed (274) if DisabledWhen Cicode has already been set or is not a valid expression.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#),  
[MenuGetParent](#), [MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#),  
[MenuNodeGetDepth](#), [MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#),  
[MenuNodeHasCommand](#), [MenuNodesDisabled](#), [MenuNodesHidden](#), [MenuNodeRemove](#),  
[MenuNodeRunCommand](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## Example

```
INT hNode = MenuNodeAddChild((hParent, "LogIn", "LogIn");
INT Error = MenuNodeSetDisabledWhen(hNode, "UserInfo", "0", 1);
```

## See Also

[Menu Functions](#)

## MenuNodeSetExpanded

Set the expansion state value of the specified menu node in number. The expansion state value is defined by project designer and can be any integer value. The initial value is zero(0). It is not recommended to use -1 for the expansion state value though because this number indicates an error for [MenuNodeGetExpanded](#) function.

**Note:** Changes made to the menu tree will not be persisted back to the menu configuration database.

## Syntax

**MenuNodeSetExpanded(*hNode*, *iValue*)**

*hNode:*

Handle to the current node in the menu tree.

*iValue:*

The expansion state value to set for the Menu node.

## Return Value

Zero (0) if successful. -1 if hNode is invalid or an error code for any other error.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#), [MenuGetParent](#), [MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#), [MenuNodeGetDepth](#), [MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#), [MenuNodeHasCommand](#), [MenuNodesDisabled](#), [MenuNodesHidden](#), [MenuNodeRemove](#), [MenuNodeRunCommand](#), [MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## Example

```
MenuNodeSetExpanded(100, 99)
//Put 99 as an expansion state of the menu node handle 100
MenuNodeGetExpanded(100)
//This will return 99 that is stored by the previous call
```

## See Also

[Menu Functions](#)

## MenuNodeSetHiddenWhen

Set the HiddenWhen expression for a newly added node. Be aware this function only works for menu nodes added with [MenuNodeAddChild\(\)](#). The HiddenWhen expression may only be set once for a node.

Be reminded that changes made to the menu tree will not be persisted back to the menu configuration database.

## Syntax

**MenuNodeSetHiddenWhen**(*hNode*, *sHiddenWhenName* [, *sHiddenWhenArgs*])

*hNode:*

Handle of node to run command.

*sHiddenWhenName:*

Cicode function for HiddenWhen expression. The function needs to return an INT.

*sHiddenWhenArgs:*

Cicode parameters for HiddenWhen expression. Only supports static arguments.

## Return Value

No Error (0) on success, Bad handle specified (269) if hNode does not refer to a valid node, or Invalid argument passed (274) if HiddenWhen Cicode has already been set or is not a valid expression.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#),  
[MenuGetParent](#), [MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#),  
[MenuNodeGetDepth](#), [MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#),  
[MenuNodeHasCommand](#), [MenuNodesDisabled](#), [MenuNodesHidden](#), [MenuNodeRemove](#),  
[MenuNodeRunCommand](#), [MenuNodeSetDisabledWhen](#), [MenuNodeSetProperty](#), [MenuReload](#)

## Example

```
INT hNode = MenuNodeAddChild(hParent, "LogIn", "LogIn");
INT Error = MenuNodeSetHiddenWhen(hNode, "UserInfo", "0");
```

## See Also

[Menu Functions](#)

## MenuNodeSetProperty

Set the item value of the specified menu node.

Be reminded that changes made to the menu tree will not be persisted back to the menu configuration database.

## Syntax

**MenuNodeSetProperty**(*hNode*, *iField*, *sValue*)

*hNode*:

Handle to the current node in the menu tree.

*iField*:

Field for which you want to set the value:

0 - Name of menu item.

1 - Icon symbol to be associated with the menu item.

2 - Privilege level required to run the command, otherwise the menu item is disabled.

3 - Area level required to run the command, otherwise the menu item is disabled.

4 - Disabled Style. Allows different display style for a disabled menu item.

5 - Checked setting. Whether the menu item will display a check box next to the label.

6 - Width. Specifies the menu item width in pixels.

7 - Comment.

8 - Cluster.

9 - Equipment.

10 - Expansion state value.

13 - Target page for the menu item.

101 to108 - Custom 1 to Custom 8.

*sValue*:

The item value to set for the Menu node.

---

**Note:** The Cluster and Equipment fields are not validated at runtime.

---

## Return Value

Zero (0) if successful. -1 if *hNode* or *iField* is invalid. An error code for others.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#), [MenuGetParent](#), [MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#), [MenuNodeGetDepth](#), [MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#), [MenuNodeHasCommand](#), [MenuNodeIsDisabled](#), [MenuNodeIsHidden](#), [MenuNodeRemove](#), [MenuNodeRunCommand](#), [MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuReload](#)

## See Also

[Menu Functions](#)

## MenuReload

Reload base Menu Configuration from the compiled database.

Be reminded that the menu configuration loaded on the currently displayed page will not change until the page is refreshed. By default, menu configuration is automatically reloaded whenever a page is displayed.

## Syntax

**MenuReload()**

## Return Value

None.

## Related Functions

[MenuGetChild](#), [MenuGetFirstChild](#), [MenuGetGenericNode](#), [MenuGetNextChild](#), [MenuGetPageNode](#),  
[MenuGetParent](#), [MenuGetPrevChild](#), [MenuGetWindowNode](#), [MenuNodeAddChild](#), [MenuNodeGetCurr](#),  
[MenuNodeGetDepth](#), [MenuNodeGetExpanded](#), [MenuNodeGetProperty](#), [MenuNodeGetTargetPage](#),  
[MenuNodeHasCommand](#), [MenuNodesDisabled](#), [MenuNodesHidden](#), [MenuNodeRemove](#),  
[MenuNodeRunCommand](#), [MenuNodeSetDisabledWhen](#), [MenuNodeSetHiddenWhen](#), [MenuNodeSetProperty](#)

## See Also

[Menu Functions](#)

## Miscellaneous Functions

Following are miscellaneous functions.

<a href="#">AreaCheck</a>	Determines whether the current user has access to a specified area.
<a href="#">Assert</a>	Verifies a particular condition is true, or halts the task.
<a href="#">Beep</a>	Beeps the speaker or sound card in the computer.
<a href="#">CitectInfo</a>	Gets information about a Plant SCADA variable.
<a href="#">CodeTrace</a>	Traces Cicode into the Kernel and the SYSLOG.DAT file.
<a href="#">DebugBreak</a>	Causes a breakpoint error to start the Cicode Debugger.
<a href="#">DebugMsg</a>	In-line debug messages of user Cicode.
<a href="#">DebugMsgSet</a>	Enables/disables the DebugMsg function.
<a href="#">DelayShutdown</a>	Causes Plant SCADA to shut down after a specified period
<a href="#">DisplayRuntimeManager</a>	Starts and displays Plant SCADA Runtime Manager.
<a href="#">DumpKernel</a>	Dumps Kernel data to the Kernel.dat file.
<a href="#">EngToGeneric</a>	Converts a variable into generic scale format.
<a href="#">Exec</a>	Executes an application or PIF file.
<a href="#">GetArea</a>	Gets the current viewable areas.
<a href="#">GetEnv</a>	Gets an environment variable.
<a href="#">GetLogging</a>	Gets the current value for one or more logging

	parameters.
InfoForm	Displays graphics object help information for the AN closest to the cursor.
InfoFormAn	Displays graphics object help information for an AN.
Input	Gets text input from the operator.
IntToReal	Converts an integer variable into a real (floating point) number.
KerCmd	Executes a command in a kernel window.
KernelQueueLength	Obtains the number of rows in a queue.
KernelTableInfo	Provides a consistent method of accessing items within a Kernel Table.
KernelTableItemCount	Obtains the number of rows in a Kernel Table.
LanguageFileTranslate	Translates an ASCII file into the local language.
Message	Displays a message box on the screen.
ParameterGet	Gets the value of a system parameter.
ParameterPut	Updates a system parameter.
ProcessIsClient	Determines if the currently executing process contains a Client component.
ProcessIsServer	Determines if the currently executing process contains a particular server component.
ProcessRestart	Restarts the current process in which Cicode is running.
ProductInfo	Returns information about the Plant SCADA product.
ProjectInfo	Returns information about a particular project, which is identified by a project enumerated number.
ProjectRestartGet	Gets the path to the project to be run the next time Plant SCADA is restarted.
ProjectRestartSet	Sets the path to the project to be run the next time Plant SCADA is restarted.
ProjectSet	Sets the name or path of the current project.
Prompt	Displays a message in the prompt line.

<a href="#">Pulse</a>	Pulses (jogs) a variable tag every two seconds.
<a href="#">ServerDumpKernel</a>	Dumps Kernel data to the KERNEL.DAT file either on a local server or on a remote server.
<a href="#">ServiceGetList</a>	Gets information about services running in the component calling this function.
<a href="#">SetArea</a>	Sets the current viewable areas.
<a href="#">SetLanguage</a>	This function is now obsolete.
<a href="#">SetLogging</a>	Adjusts logging parameters while online.
<a href="#">Shutdown</a>	Ends Plant SCADA's operation.
<a href="#">ShutdownForm</a>	Displays a form that allows an operator to shut down the Plant SCADA system.
<a href="#">ShutdownMode</a>	Gets the mode of the shutdown/restart.
<a href="#">SwitchConfig</a>	Switches focus to the Plant SCADA configuration application.
<a href="#">TestRandomWave</a>	Generates a random wave.
<a href="#">TestSawWave</a>	Generates a saw wave.
<a href="#">TestSinWave</a>	Generates a sine wave.
<a href="#">TestSquareWave</a>	Generates a square wave.
<a href="#">TestTriangWave</a>	Generates a triangular wave.
<a href="#">Toggle</a>	Toggles a digital tag on or off.
<a href="#">TraceMsg</a>	Displays a message in the Kernel and Debugger debug windows.
<a href="#">Version</a>	Gets the version number of the Plant SCADA software.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## AreaCheck

Determines whether the current user has access to a specified area.

## Syntax

**AreaCheck(*Area*)**

*Area*:

The area number (0 - 255)

## Return Value

TRUE (1) if the user has access to the *Area* or FALSE (0) if not.

## Related Functions

[GetArea](#), [GetPriv](#)

## Example

```
IsArea = AreaCheck(5);
```

## See Also

[Miscellaneous Functions](#)

## Assert

Verifies that the specified expression is TRUE. If then expression is FALSE, the current task will be halted. This is useful to help prevent the execution of code you do not want to run in the event an error has been detected.

This function can be used in a debug mode, where the FALSE assertion will be logged to the Kernel and SysLog.DAT, with the time, date, Cicode file name, and line number. Additionally the operator will be prompted with a dialog. The debug mode can be set by using the [\[Code\]DebugMessage](#) parameter or DebugMsgSet() function.

---

**Note:** If you have the "Plant SCADA Runtime will start debugger on hardware errors" option set in the Cicode Editor, the Debugger will start with the position before the Halt() instruction. You need to 'step over' this command if you want to continue debugging the function that called the Assert().

---

## Syntax

**Assert(*bCondition*)**

*bCondition*:

The boolean expression. This expression needs to evaluate to TRUE (1) or FALSE (0).

## Return Value

None. However, if the assertion tests as FALSE, error 347 is generated.

## Related Functions

[Halt](#), [DebugMsg](#), [DebugMsgSet](#), [CodeTrace](#), [TraceMsg](#), [ErrLog](#)

## Example

```
INT
FUNCTION FileDisplayEx(STRING sFileName);
    INT hFile;
    hFile = FileOpen(sFileName, "r");
    Assert(hFile <> -1);
    ...
    FileClose(hFile);
    RETURN 0;
END
```

## See Also

[Miscellaneous Functions](#)

## Beep

Beeps the internal speaker or sound card (installed in the computer). If you use the internal speaker on your computer, the function does not return until the sound has completed. If you use a sound card, the function returns immediately and the sound plays in the background.

Use the Windows Control Panel to set up waveforms.

## Syntax

**Beep(*nSound*)**

*nSound*:

The type of sound:

-1 - Standard beep

0 - Default beep waveform

1 - Critical stop waveform

2 - Question waveform

3 - Exclamation waveform

4 - Asterisk waveform

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[DspPlaySound](#)

## Example

```
/* Beeps the speaker with the default waveform. */  
Beep(0);
```

## See Also

[Miscellaneous Functions](#)

## CitectInfo

Gets information about a Plant SCADA variable. This function returns internal statistics and other information about the Plant SCADA runtime system.

**Note:** This function is a non-blocking function and can only access data contained within the calling process; consequently it cannot return data contained in a different server process. This function is not redirected automatically by Plant SCADA runtime. If you want to make a call from one process to return data in another, use `MsgRPC()` to make a remote procedure call on the other process. Alternatively, include the server process that has the information that `CitectInfo` requires in the calling process.

## Syntax

`CitectInfo(sGroup, sName, sType)`

### **sGroup:**

The name of the group to which the variable belongs. Valid group names are: "General", "Port", "IODevice", "Network", "Stats", "Memory", "Disk", "Queue" or "Security".

### **sName:**

The name of the variable. This name depends on `sGroup`:

"General" - the name is ignored.

"Port" - the name of the I/O port configured in the database (with the Ports database). The port information is only valid for an I/O server. If the port name is "total", the total statistics for the I/O server are returned.

"IODevice" - the name of the I/O device configured in the I/O devices database.

"Network" - the name is ignored.

"Stats" - The name of the statistics buffer or Statistical Information Record (SIR):

- "Alarm Proc" - Alarm Processing (includes Digital, Analog, Advanced and High Resolution alarms).
- "Citect n" - The Plant SCADA window where n is the window number (returned from the `WinNumber()` function)
- "Code n" - The user Cicode task (thread) where n is the task handle (returned from the `TaskHnd()` function)
- "Reset" - Reset the Plant SCADA statistics.
- "Elapsed Time MS" - The elapsed time since statistics have been reset. Returns -1 if more than 20 days has

elapsed.

"Memory" - the measurement used

- 0 = bytes
- KB = kilobytes
- MB = megabytes
- GB = gigabytes

"Disk" - The disk drive to access:

- 0 = The current drive
- 3 = C:
- 4 = D: and so on.

"Queue" - The name of the queue.

**sType:**

The type of information to get, depending on *sGroup*:

"General" - General statistics:

- 0 - CPU usage
- 1 - Plant SCADA Kernel cycles per second
- 2 - Plant SCADA Kernel tasks per second
- 3 - Plant SCADA Kernel boot time
- 4 - Plant SCADA Kernel running time (in seconds)
- 5 - Plant SCADA startup time
- 6 - Plant SCADA running time in seconds
- 7 - Not supported in v7.10 or later
- 8 - Total read requests
- 9 - Total read requests per second
- 10 - Total write requests
- 11 - Total write requests per second
- 12 - Total Physical read requests
- 13 - Total Physical read requests per second
- 14 - Total Physical write requests
- 15 - Total Physical write requests per second
- 16 - Total Blocked read requests
- 17 - Total Blocked write requests
- 18 - Total Digital read requests
- 19 - Total Register read requests
- 20 - Total Digital read requests per second
- 21 - Total Register read requests per second
- 22 - Total Cache reads count
- 23 - Total Cache reads %

24 - Overall Average response time (ms)  
25 - Overall Minimum response time (ms)  
26 - Overall Maximum response time (ms)  
27 - Request sample for response times  
28 - Static point count is no longer supported. Calling the function with parameter 28 returns a value of 0 and a hardware alarm is raised.  
29 - Dynamic point count currently in use

30 - Number of pending read requests from the device  
31 - Number of pending write requests to the device  
32 - Determines if Plant SCADA Kernel window is open  
33 - Percentage of the CPU used by the current Plant SCADA process  
34 - Total CPU time spent by the current Plant SCADA process in milliseconds  
35 - Total number of handles opened by the current Plant SCADA process  
36 - Total number of threads owned by the current Plant SCADA process

"Port" - Port information for the I/O Server:

0 - Read requests  
1 - Write requests  
2 - Physical read requests  
3 - Physical write requests  
4 - Cached read requests  
5 - Cached write requests  
6 - Blocked read requests  
7 - Blocked write requests  
8 - Read requests per second  
9 - Write requests per second  
10 - Error count  
11 - Read bytes counter  
12 - Channel usage %  
13 - Read bytes per second  
14 - Statistics, minimum read time  
15 - Statistics, maximum read time  
16 - Statistics, average read time  
17 - Statistics, time of samples  
18 - Statistics, number of sample

100 - 119 - Driver specific counter values. Plant SCADA drivers can maintain up to 20 unique counters that can be accessed via this function. They are zero based, indexed from 100 to 119. If a value is not defined or maintained by the driver, 0 is returned for the value of the counter.

"IODevice" - I/O device information for the I/O device:

0 - Client side status:

- 1 = Running - Client is either talking to an online IO device or talking to a scheduled device that is not currently connected but has a valid cache
- 2 = Standby - Client is talking to an online standby IO device
- 4 = Starting - Client is talking to an IO device that is attempting to come online
- 8 = Stopping - Client is talking to an IO device that is in the process of stopping
- 16 = Offline - Client is pointing to an IO device that is currently offline
- 32 = Disabled - Client is pointing to a device that is disabled
- 66 = Standby write - Client is talking to an I/O device configured as a standby write device

1 - I/O Server status:

- 1 = Running - I/O Server is either talking to an online IO device or talking to a scheduled device that is not currently connected but has a valid cache
- 2 = Standby - I/O Server is talking to an online standby IO device
- 4 = Starting - I/O Server is talking to an IO device that is attempting to come online
- 8 = Stopping - I/O Server is talking to an IO device that is in the process of stopping
- 16 = Offline - I/O Server is pointing to an IO device that is currently offline
- 32 = Disabled - I/O Server is pointing to a device that is disabled
- 66 = Standby write - I/O Server is talking to an I/O device configured as a standby write device

2 - If this I/O device is a standby device

3 - Last generic error

4 - Last driver error

5 - Error count

6 - Initialization count

7 - Statistics, minimum read time

8 - Statistics, maximum read time

9 - Statistics, average read time

10 - Statistics, number of samples

"Network" - Network statistical information:

0 - Read Network Control Blocks (NCBs)

1 - Maximum pending read NCBs

2 - Minimum pending read NCBs

3 - Current pending read NCBs

4 - Number of short read NCBs

5 - Write NCBs

6 - Maximum pending write NCBs

7 - Minimum pending write NCBs

8 - Current pending write NCBs

9 - Number of short write NCBs

10 - Total NCBs

11 - Maximum pending total NCBs

12 - Minimum pending total NCBs

13 - Current pending total NCBs

14 - Number of short total NCBs

15 - Minimum send response time in milliseconds

16 - Maximum send response time in milliseconds

17 - Average send response time in milliseconds

18 - Send packet count

"Stats" - Statistical information:

0 - Minimum time between code executions (cycles)

1 - Maximum time between code executions (cycles)

2 - Average time between code executions (cycles)

3 - Total cycle time in milliseconds

4 - Minimum time to execute the code in milliseconds

5 - Maximum time to execute the code in milliseconds

6 - Average time to execute the code in milliseconds

7 - Total execute time in milliseconds

"Memory" - Memory information:

0 - Free virtual memory

1 - Free windows system resources as %

2 - Free Physical Memory

3 - Memory Paging File Size

4 - Total Physical Memory

5 - Total % of Physical Memory Used

6 - Total working set size counter for current Plant SCADA process

7 - Private bytes counter of current Plant SCADA process

"Disk" - Disk information:

0 - Free disk space in bytes

1 - Total disk space in bytes

2 - Free disk space in kilobytes

3 - Total disk space in kilobytes

4 - Free disk space in megabytes

5 - Total disk space in megabytes

"Queue" - Queue information:

0 - Returns the length of the queue from the process that this is being called on.

"Security" - Encryption information:

0 - Returns the encryption status:

- 0 = Insecure

- 1 = Mixed

- 2 = Encrypted.

## Return Value

The type of information (as an integer).

## Related Functions

[IODeviceInfo](#), [WinNumber](#), [TaskHnd](#)

## Example

```
! Get free memory
FreeMemory = CitectInfo("Memory", "", 0);
! Get free disk space on C:
FreeDisk = CitectInfo("Disk", 3, 0);
! Get max cycle time for digital alarms
MaxCycleTime = CitectInfo("Stats","Digital Alm","1");
```

## See Also

[Miscellaneous Functions](#)

## CodeTrace

Traces Cicode into the Kernel and the SYSLOG.DAT file. Use this function for finding bugs in your Cicode. It will trace the functions called, the arguments to those functions, and their return values. It will also trace any errors, writes to the I/O devices, and task state changes.

**Note:** Use this function only during system development; it can cause excessive loading on the CPU.



### UNINTENDED EQUIPMENT OPERATION

Only use the CodeTrace() function during system development and testing. This function is not intended for use in an operational system.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Syntax

**CodeTrace(*hTask*, *nMode*)**

*hTask*:

The Cicode task handle as returned from TaskHnd() function or any of the following special values:

0 - Foreground Cicode.

-2 - The next created task.

-3 - New created tasks.

-4 - All tasks.

*nMode:*

The mode of the trace:

0 - Tracing off

1 - Trace user Cicode functions calls

2 - Trace built-in function calls

4 - Trace errors

8 - Trace writes to the I/O devices

16 - Trace task state changes

-1 - All modes (except 0)

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[Assert](#), [TaskHnd](#), [DebugMsg](#), [DebugMsgSet](#), [TraceMsg](#), [ErrLog](#)

## Example

```
// Start tracing errors
CodeTrace(TaskHnd(), 4);
...
// Stop tracing
CodeTrace(TaskHnd(), 0);
// trace functions in new task
CodeTrace(-2, 1 + 2);
TaskNew("MyFunc", "data", 0);
```

## See Also

[Miscellaneous Functions](#)

## DebugBreak

Causes a breakpoint exception error to occur (error number 342). This allows programmers to trap invalid states in their Cicode. If the Cicode Editor is not running, and the **Plant SCADA Runtime will start debugger on hardware errors** option is set (**Debug** menu - **Options**), the Debugger will be started. When the debugger starts, the correct Cicode file, function, and line will be displayed.

## Syntax

**DebugBreak()**

## Return Value

None.

## Related Functions

[DspKernel](#), [KerCmd](#), [DumpKernel](#), [TraceMsg](#)

## Example

```
!Check to see that rSpan is greater than zero else cause a break.  
If rSpan equals 0 it would cause a Divide by Zero hardware error  
anyway.  
IF rSpan > 0 THEN  
    rCalcRate = iAmount/rSpan;  
ELSE  
    DebugBreak();  
END
```

## See Also

[Miscellaneous Functions](#)

## DebugMsg

Provides in-line debug messages of user Cicode, to the Kernel, Debugger Debug window, and the SysLog.DAT file. This function can be enabled or disabled with the [\[Code\]DebugMessage](#) parameter or [DebugMsgSet\(\)](#) function at runtime.

## Syntax

**DebugMsg(*sMessage*)**

*sMessage*:

The debugging message to log. Be sure to enclose this message in double quotes ("").

## Return Value

None.

## Related Functions

[Assert](#), [DebugMsgSet](#), [CodeTrace](#), [TraceMsg](#), [ErrLog](#)

## Example

INT

```
FUNCTION
FileDisplayEx(STRING sFileName);
    INT hFile;
    hFile = FileOpen(sFileName, "r");
    DebugMsg("When opening file " + sFileName + ", the handle was:
    " + IntToStr(hFile));
    ...
    FileClose(hFile);
    RETURN 0;
END
```

## See Also

[Miscellaneous Functions](#)

## DebugMsgSet

Enables/disables the DebugMsg() logging functionality. It also controls whether logging is enabled for the Assert() function. This function also sets the [\[Code\]DebugMessage](#) parameter appropriately.

## Syntax

**DebugMsgSet(*nMode*)**

*nMode*:

The logging mode:

0 - Disable logging.

1 - Enable logging.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[Assert](#), [DebugMsg](#)

## Example

Buttons

Text	Debug Enable
Command	DebugMsgSet(1)
Comment	Enable debug logging

## See Also

[Miscellaneous Functions](#)

### DelayShutdown

Terminates Plant SCADA's operation after the specified delay period (in milliseconds). This function is suitable to be called by the CTAPI. The delay period enables the user to close the connection between the CTAPI and third-party applications before Plant SCADA shuts down.

## Syntax

**DelayShutdown(*Delay*)**

*Delay*:

The period (in milliseconds) after which Plant SCADA will shut down.

## Return Value

No return value.

## Example

```
DelayShutdown(10 000)
!Terminates Plant SCADA's operation after 10 seconds
```

## See Also

[Miscellaneous Functions](#)

### DisplayRuntimeManager

This function will start the Plant SCADA Runtime Manager if it is not already running, otherwise it will just bring the Plant SCADA Runtime Manager to the foreground.

## Syntax

**DisplayRuntimeManager()**

## Return Value

0 (zero) if successful, otherwise an error is returned.

## See Also

[Miscellaneous Functions](#)

## DumpKernel

Dumps Kernel data to the KERNEL.DAT file. Instead of creating a new file subsequent calls to this function will append Kernel data to the KERNEL.DAT file.

## Syntax

**DumpKernel(*iMode*, *sName*)**

*iMode*:

The Kernel data to dump:

0x0001	Dump general statistics.
0x0002	Dump the task.
0x0004	Dump the I/O device.
0x0008	Dump the driver.
0x0010	Dump netstat. (This mode is deprecated and no longer active in version 7.10 or later.)
0x0020	Dump the table.
0x0040	Dump the queue.
0x0100	Dump list of current SCADA parameter settings
0x4000	Dump in verbose mode.
0x8000	Dump kernel data in non-verbose mode. To dump data in verbose mode, set <i>iMode</i> to 0xc000 (0x8000 + 0x4000).

You can select any one of the above modes or may add them together to get more than one type of information. For example, to dump driver and I/O device data in verbose mode, set *iMode* to 0x400c (0x0004 + 0x0008 + 0x4000). Using 0x4000 on its own will dump no data, it needs to be combined with another mode.

*sName*:

The queue or table name (if "", returns data for all queues or tables). Only valid if *iMode* is 0x0020 and 0x0040.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspKernel](#), [KerCmd](#), [TraceMsg](#)

## Example

```
DumpKernel(0x8000, "");  
!Dump the Kernel data
```

## See Also

[Miscellaneous Functions](#)

## EngToGeneric

Gets a variable in the Plant SCADA generic scale format. Plant SCADA uses this scale to display trends. It calls this function automatically for trends defined in the project. If you want to display a trend using Cicode you need to call this function.

## Syntax

**EngToGeneric**(*Value*, *EngLow*, *EngHigh*)

*Value*:

The value to convert to the Plant SCADA generic scale format.

*EngLow*:

The engineering units zero scale.

*EngHigh*:

The engineering units full scale.

## Return Value

The variable (in the range 0 - 32000).

## Related Functions

[DspBar](#), [DspTrend](#)

## Example

```
/* Using trend definition 5 at AN20, display the value of Tag1 on  
Pen1 of the trend. Tag1 has an engineering scale of 0 to 100. */  
DspBar(20,5,EngToGeneric(Tag1,0,100));
```

## See Also

[Miscellaneous Functions](#)

## Exec

Executes an application, LNK file or PIF file (32 bit system only). The application or command starts up and continues to run in parallel with Plant SCADA.

This function can return while the application is still starting up, so you should use the Sleep() function to allow the application enough time to start.

**Note:** Users can run this function only if their Allow Exec role is set to TRUE in the Security | Roles view **and** the [Security]BlockExec parameter is set to 0. For more information about the Allow Exec role, refer to [\[Security\]BlockExec](#).

## Syntax

**Exec**(*Command* [, *Mode*] )

*Command*:

The operating system command to execute (maximum of 255 characters).

*Mode*:

The mode of the window:

1 - Normal

3 - Maximized

6 - Minimized

If you do not enter a mode, the default mode is 1.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

Sleep

## Example

```
Exec("c:\winnt\system32\mspaint.exe");
! Starts up the Paint application with a normal window.
Exec("cmd /c mkdir c:\test");
! Uses the DOS shell to create a new directory
```

You need to quote paths and applications passed to the exec function (using ^" to embed quotes) so that the function can be executed correctly when long file names are used. For example:

```
Exec(PathToStr("^^" + "[BIN] :\Ctcicode") + "^" ^" +
PathToStr("[RUN] :\cicode.ci") + "^")
```

## See Also

[Miscellaneous Functions](#)

## GetArea

Gets the current logged-in areas.

## Syntax

**GetArea()**

## Return Value

The login area groups as an integer that represents a group handle. If this group is modified, the actual login areas do not change.

## Related Functions

[SetArea](#)

## Example

```
! If the current areas are 1, 5 and 20:  
hGrp=GetArea();  
Str=GrpToStr(hGrp);  
! sets Str to "1,5,20".
```

## See Also

[Miscellaneous Functions](#)

## GetEnv

Gets a DOS environment variable.

## Syntax

**GetEnv(*sName*)**

*sName*:

The name of the environment variable. The length of the name should not exceed 255 characters.

## Return Value

The DOS environment variable (as a string).The return value returns up to 255 characters and is truncated if exceeds.

## Example

```
/* Get the current DOS path. */
sPath = GetEnv("Path");
```

## See Also

[Miscellaneous Functions](#)

### GetLogging

Gets the current value for logging parameters.

The parameters you can query include the following:

- [Debug]DriverTrace
- [Debug]SysLogSize
- [Debug]Priority
- [Debug]CategoryFilterMode
- [Debug]CategoryFilter
- [Debug]SeverityFilterMode
- [Debug]SeverityFilter
- [Debug]LogShutDown
- [Debug]DebugAllTrans
- [Debug]EnableLogging
- [IOServer]RedundancyDebug
- [General]Verbose
- [General]VerboseToSysLog
- [CtAPI]Debug

## Syntax

**GetLogging(Section, Name)**

*Section*:

The INI section name.

*sName*:

The system parameter name.

## Return Value

If the function succeeds, the value is returned as a string. An empty string is returned if an error occurs. To get extended error information, call [IsError\(\)](#).

## Related Functions

[SetLogging](#)

## See Also

[Miscellaneous Functions](#)

## InfoForm

Displays graphics object information for the object under the mouse pointer. If there is no object directly under the mouse pointer, it displays information for the nearest object. Each tag associated with the object is displayed, along with its raw and engineering values and a button that displays the details from the Variable Tags settings. The function also displays the Cicode expression, with any result that the expression has generated.

This function only supports the display of simple Cicode expressions.

---

**Note:** [DsplInfoNew](#), [InfoForm](#) and [InfoFormAn](#) will not work if the parameter [General]SymbolicInfo = 0.

---

## Syntax

**InfoForm(*Mode*)**

*Mode*:

For security purposes, the Write button on the information form displayed by this function can be disabled.

0 - The **Write** button on the displayed information form will be available and will function normally.

1 - The **Write** button will not be shown.

If you omit the mode, the default mode is 0.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[InfoFormAn](#)

## Example

System Keyboard

Key Sequence	AnHelp
Command	InfoForm();
Comment	Display graphics object help for the AN closest to the cursor

## See Also

[Miscellaneous Functions](#)

### InfoFormAn

Displays graphics object information for a specified AN. This function displays each tag associated with the object, along with its raw and engineering values and a button that displays the details from the Variable Tags form. The function also displays the Cicode expression, with any result that the expression has generated.

**Note:** [DsplInfoNew](#), [InfoForm](#) and [InfoFormAn](#) will not work if the parameter [\[General\]SymbolicInfo = 0](#).

## Syntax

**InfoFormAn(***nAN* [, *Mode*] **)**

*nAN*:

The AN of the graphics object for which information is displayed.

*Mode*:

For security purposes, the Write button on the information form displayed by this function can be disabled.

0 - The **Write** button on the displayed information form will be available and will function normally.

1 - The **Write** button will not be shown.

If you omit the mode, the default mode is 0.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[InfoForm](#)

## Example

System Keyboard

Key Sequence	### AnHelp
Command	InfoFormAn(Arg1);
Comment	Display graphics object help for a specified AN

## See Also

[Miscellaneous Functions](#)

## Input

Displays a dialog box in which an operator can input a single value. The dialog box has a title, a prompt, and a single edit field. For multiple inputs, use the [Form Functions](#).

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

## Syntax

**Input**(*Title*, *Prompt*, *Default*)

*Title*:

The title of the input box.

*Prompt*:

The prompt text.

*Default*:

The default text that the operator can edit or replace.

## Return Value

The edit field entry (as a string). If the user presses the **Cancel** button, an empty string is returned and the [IsError\(\)](#) function returns the error code 299.

## Related Functions

[Message](#), [FormNew](#)

## Example

```
/* Shut down Plant SCADA if the user inputs "Yes". */
STRING sStr;
sStr=Input("Shutdown","Do you wish to shutdown?","Yes");
IF sStr="Yes" THEN
Shutdown();
END
```

## See Also

[Miscellaneous Functions](#)

## IntToReal

Converts an integer into a real (floating point) number.

When calling statements such as comparison or arithmetic operations that involve a mixture of variables of REAL and INT data types, there is no need to explicitly convert the variable of INT data type to REAL data type using the IntToReal function. Plant SCADA will automatically convert variables of INT data type to REAL data type before the operation is carried out.

However, if the expression only consists of variables of INT data type, the result of the expression will remain as INT data type and be subjected to its limitations such as integer overflow and wraparound. This does not change even if the expression is to be assigned to a variable of REAL data type.

If you want to increase the range of the expression, you will need to convert at least one of its operands to REAL data type using the IntToReal function (see Example 2 below).

## Syntax

**IntToReal(Number)**

*Number:*

The integer to convert.

## Return Value

The real number.

## Related Functions

[RealToStr](#), [StrToInt](#)

## Example 1

```
! Sets Variable to 45.0
Variable=IntToReal(45);
! Sets Variable to -45.0
Variable=IntToReal(-45);
```

## Example 2

```
INT nVar1 = 1000000001;
INT nVar2 = 2000000002;
REAL rVar1 = 1.00001;
REAL rVar2 = 0.0;

// rVar2 = 1000000002.00001, IntToReal is not needed when operands are mixture of INT and
REAL data types
rVar2 = nVar1 + rVar1;

// rVar2 = -1294967293.0, result is wrapped around due to range overflow of integer data
type
rVar2 = nVar1 + nVar2;

// rVar2 = 3000000003.0, result is promoted to REAL data type when one of the INT operands
is converted
rVar2 = nVar1 + IntToReal(nVar2);
```

## See Also

[Converting and Formatting Cicode Variables](#)  
[Miscellaneous Functions](#)

## KerCmd

Executes a command in a Kernel window.

## Syntax

**KerCmd(*Window*, *sCommand*)**

*Window*:

The name of the Kernel window.

*sCommand*:

The command to execute in the Kernel window.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspKernel](#), [TraceMsg](#), [DumpKernel](#)

## See Also

[Miscellaneous Functions](#)

## KernelQueueLength

Obtains the number of records in a queue.

## Syntax

**KernelQueueLength(*sName*)**

*sName*:

The name of the queue, which can be enumerated by the page queue kernel command.

## Return Value

This function returns the Length value of the page queue command.

## Related Functions

None

## See Also

[Miscellaneous Functions](#)

## KernelTableInfo

Provides a consistent method of accessing items within Kernel Table.

## Syntax

**KernelTableInfo(*sTable*, *sRecord*, *sField*)**

*sTable*:

The name of the table. The following tables are supported:

<b>sTable</b>	<b>sRecord</b>	<b>sField</b>	<b>Notes</b>
Bufferpool	The value of the 'Name' field	Mode, Size, Total, Free, InUse, Max, Peak, Short, Gets, TotErr, Grow, Shrink	These names are unique when each Bufferpool is created
CiCode	The value of the 'Name' field or the 'HND' field. If the user specifies a number as the sRecord we will assume it is a handle, otherwise we will look it up as a string name	Name, HND, State, CPU_Time, Poll_Time, Slice	The CiCode function TaskNew exposes the handle. That is why record access by handle is acceptable for this table. The name field may or may not be unique (if you run the same CiCode task twice with different parameters). Accessing by 'Name' will not succeed if there are duplicates
Task	The value of the 'Name' field	Mode, Hnd, State, Prty, Cpu, Min, Max, Avg, Count	None
Tran	The value of the TRAN 'Name' field	Name, Node, Type, Mode, Hnd, Cnt, Send, Rec, Wait, Stack, Service, State, Login, ReconnectCount, UpTime, TotalTxRxBytes	The counters ReconnectCount, UpTime and TotalTxRxBytes can be viewed in verbose mode on Table Tran page of the Kernel

sTable	sRecord	sField	Notes
Trendqueues	The value of the 'DriverName' field	FlushQueueLength, FlushQueuesMax , TotalFlushes, FileWrites	None

*sRecord:*

The key to a column in the table depending on the *sTable* parameter.

*sField:*

The key to a column in the table depending on the *sTable* parameter.

## Return Value

Returns the content of a field of the given table in the format of a Cicode STRING.

## Related Functions

[DspKernel](#), [DumpKernel](#)

## See Also

[Miscellaneous Functions](#)

## KernelTableItemCount

Obtains the number of rows in a Kernel Table.

## Syntax

**KernelTableItemCount(sName)**

*sName:*

The name of the table that can be enumerated by the page table kernel command.

## Return Value

Returns the number of active records in the table (not the length value displayed by the page table kernel command).

The following tables row counts are not returned by this command (it returns 0)

- IODevices.Tags
- IODevices.Subs
- IODevices.Blocks
- CSAToPSI.Subs

[KernelTableInfo](#), [DumpKernel](#)

## See Also

[Miscellaneous Functions](#)

### LanguageFileTranslate

Translates an ASCII file into a local language. Use this function to translate RTF reports for viewing on client screens.

The local language used by this function is specified by the [\[Language\]LocalLanguage](#) parameter. You need to set this parameter accordingly.

## Syntax

**LanguageFileTranslate(*sSourceFile*, *sDestFile*)**

*sSourceFile*:

The name of the ASCII file containing multi-language text, which will be copied and translated. You should specify the full path or use path substitution. The path and name should be contained within quotation marks.

*sDestFile*:

The name of the destination file which will be created/ replaced with the local/translated version of the source file. You should specify the full path or use path substitution. The path and name should be contained within quotation marks.

## Return Value

1 if successful, otherwise 0.

## Related Functions

[StrToLocalText](#)

## Example

```
/* Translates the text in Shift.RTF and saves it in LShift.RTF. */
LanguageFileTranslate("c:\MyApplication\data\Shift.RTF", "c:\MyApplication\data\
LShift.RTF");
/* Translates the text in [Data]:Shift.RTF and saves it in
[LocalData]:LShift.RTF. */
LanguageFileTranslate("[Data]:Shift.RTF", "[LocalData]:LShift.RTF"
);
```

## See Also

[Miscellaneous Functions](#)

## Message

Displays a message box on the screen and waits for the user to select the **OK** or **Cancel** button.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

This function will be blocked if called from a non-client process, as well as from kernel window of a client process if [\[Client\]DisableDisplay](#) is set to TRUE.

## Syntax

**Message**(*Title*, *Prompt*, *Mode*)

*Title*:

The title of the message box. The maximum length is 254 chars.

*Prompt*:

The prompt displayed in the message box.

*Mode*:

The mode of the message box:

0 - **OK** button

1 - **OK** and **Cancel** button

16 - Stop Icon

32 - Question Icon

48 - Exclamation Icon

64 - Information Icon

Select more than one mode by adding the modes. For example, set Mode to 33 to display the **OK** and **Cancel** buttons and the Question icon. You can only display one icon for the message box.

## Return Value

0 (zero) if successful, otherwise an error is returned. If the user presses the Cancel button the function returns an error code of 299.

## Related Functions

[Input](#)

## Example

```
/* Display an error message in a message box. */  
IF Total<>100 THEN  
    Message("Error","Total not 100%",48);  
END
```

## See Also

[Miscellaneous Functions](#)

### ParameterGet

Gets the value of a parameter. The parameter can exist in a local Citect.ini file, a profile INI file, or the parameters database.

- If a project is using a profile INI file, the value of the specified parameter will be retrieved from the profile INI.
- If the specified parameter exists in both Citect.ini and the parameters database, the value of the parameter is retrieved from the Citect.ini.
- If the parameter does not exist in any of these locations, the default value is returned.

## Syntax

**ParameterGet(Section, Name, Default)**

*Section:*

The section name.

*sName:*

The parameter name.

*Default:*

The default value of the parameter.

---

**Note:** If in your Cicode you perform a ParameterGet("alarm", "saveperiod", 1000) for the [Alarm]SavePeriod parameter, and no entry exists in the profile INI file, the Citect.ini or the parameters records, the returned value will be 1000. However Plant SCADA will internally be using the default value of 600.

---

## Return Value

The parameter (as a string).

## Related Functions

[ParameterPut](#), [WndGetFileProfile](#), [WndPutFileProfile](#)

## Example

```
Variable=ParameterGet("Page","Windows","4");
```

## See Also

[Miscellaneous Functions](#)

## ParameterPut

Updates a system parameter in the CITECT.INI file. If the system parameter does not exist, it is added to the CITECT.INI file.

## Syntax

**ParameterPut**(*Section*, *Name*, *Value*)

*Section*:

The section name.

*sName*:

The system parameter name.

*Value*:

The value to put in the system parameter.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[ParameterGet](#), [WndGetFileProfile](#), [WndPutFileProfile](#)

## Example

```
/* Changes the [Page] Windows system parameter in CITECT.INI to "4". */
ParameterPut("Page","Windows","4");
```

## See Also

[Miscellaneous Functions](#)

## ProcessIsClient

Determines if the currently executing process contains a Client component.

## Syntax

**ProcessIsClient()**

## Return Value

TRUE (1) if the process contains a Client component, otherwise FALSE (0).

## Related Functions

[ProcessIsServer](#), [ServerInfo](#), [ServerInfoEx](#), [ServiceGetList](#)

## Example

To execute local variable simulation code on the client only:

```
IF (ProcessIsClient()) THEN
    SimulateLocalVariables();
END
```

## See Also

[Miscellaneous Functions](#)

## ProcessIsServer

Determines if the currently executing process contains a particular server component.

## Syntax

**ProcessIsServer(*sServerType* [, *sClusterName*] [, *sServerName*])**

*sServerType*:

Case insensitive string specifying the type of server to check for. Supported values are "IOServer", "Trend", "Alarm" and "Report".

*sClusterName*:

Optional case insensitive string specifying the cluster name to combine with the server type specified in the first argument and the server name (if specified).

*sServerName*:

Optional case insensitive string specifying a server name to combine with the server type specified in the first argument and the cluster name (if specified).

## Return Value

TRUE (1) if the process contains the specified component, otherwise FALSE (0).

## Related Functions

[ProcessIsClient](#), [ServerInfo](#), [ServerInfoEx](#), [ServiceGetList](#)

## Example

To execute disk PLC tag simulation code only on the I/O server in Cluster1 with the name IOServer3:

```
IF (ProcessIsServer("IOServer", "Cluster1", "IOServer3")) THEN
```

```
SimulateDiskTags();  
END
```

## See Also

[Miscellaneous Functions](#)

## ProcessRestart

Restarts the current process in which Cicode is running.

## Syntax

```
INT error = ProcessRestart()
```

## Return Value

0 (zero) if successful, otherwise the following error is returned:

256 - General software error

See [Cicode and General Errors](#).

## Related Functions

[ServerRestart](#)

## Example

```
ProcessRestart()
```

## See Also

[Miscellaneous Functions](#)

## ProductInfo

Returns information about the Plant SCADA product.

## Syntax

**ProductInfo(*iType*)**

*iType*:

Type of information required:

0 - product name, default

1 - product company

- 2 - product major version
- 3 - product minor version
- 4 - product version string

## Return Value

The product information. An empty string will be returned if the type is invalid.

## Related Functions

[ProjectInfo](#)

## See Also

[Miscellaneous Functions](#)

## ProjectInfo

Returns information about a particular project, which is identified by a project enumerated number.

## Syntax

**ProjectInfo(*iProject*, *iType*)**

*iProject*:

Project number. This is a sequential number generated by the Compiler starting from 0,1,2,3,4,5... to 1026, so the maximum include projects are limited to 1027.

*iType*:

Type of information to return:

- 0 - Project name
- 1 - Project description
- 2 - Project major revision number
- 3 - Project minor revision number
- 4 - Project date
- 5 - Project time

## Return Value

The specified project information. An empty string will be returned if the project number or type is invalid

## Example

```
INT hFile;  
INT iProjectNumber = 0;
```

```
INT iMaxProjects = 1028;
STRING sProjectName;
//Iterate over all projects and write the version numbers to a file
hFile = FileOpen("[Data]:ProjectInfo.txt","a+");
//Iterate of the project numbers and retrieve the project name
WHILE iProjectNumber < iMaxProjects DO
    sProjectName = ProjectInfo(iProjectNumber,0);
    FileWriteLn(hFile,sProjectName);
    iProjectNumber = iProjectNumber + 1;
END
fileClose(hFile);
```

## Related Functions

[ProductInfo](#)

## See Also

[Miscellaneous Functions](#)

## ProjectRestartGet

Gets the path to the project to be run the next time Plant SCADA is restarted. You need to have a project already set using either [ProjectSet](#) or [ProjectRestartSet](#). Use this function with the [Shutdown\(\)](#) function to shut down and then restart the project that is currently running.

## Syntax

[ProjectRestartGet\(\)](#)

## Return Value

The path to the project to be run the next time Plant SCADA is restarted.

## Related Functions

[Shutdown](#), [ShutdownMode](#), [ProjectSet](#), [ProjectRestartSet](#)

## Example

See [Shutdown](#).

## See Also

[Miscellaneous Functions](#)

## ProjectRestartSet

Sets the path to the project to be run the next time Plant SCADA is restarted.

### Syntax

**ProjectRestartSet(*sPath*)**

*sPath*:

The path to the project. You need to use the full path.

### Return Value

0 (zero) if successful, otherwise an error code is returned.

### Related Functions

[Shutdown](#), [ShutdownMode](#), [ProjectSet](#), [ProjectRestartGet](#)

### See Also

[Miscellaneous Functions](#)

## ProjectSet

Sets either the name or the path of the project to be run next time Plant SCADA is restarted. The project path is written to the [\[CtEdit\]Run](#) parameter.

### Syntax

**ProjectSet(*sProject*)**

*sProject*:

The name of the project (for example "DEMO"), or the path to the project. If you specify the path to the project, you need to use the full path. If you do not specify a project, the current project will be used.

### Return Value

0 (zero) if successful, otherwise an error code is returned.

### Related Functions

[Shutdown](#), [ShutdownMode](#), [ProjectRestartGet](#)

## Example

```
/* Set the next project to "Demo". */
ProjectSet("Demo");
/* Set the next project to "MyPath". */
ProjectSet("I:\Plant SCADA\PROJECT1\MYPATH");
```

## See Also

[Miscellaneous Functions](#)

## Prompt

Displays a message in the prompt line (AN=2) on the operator's computer.

## Syntax

**Prompt(*String*)**

*String*:

The message to be displayed.

---

**Note:** This Cicode function only works on a Plant SCADA process with a display, such as a display client or a server running in single process mode and not as a service.

---

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[Message](#), [DspError](#)

## Example

```
/* Display "This is a prompt!" at the prompt AN. */
Prompt("This is a prompt!");
```

## See Also

[Miscellaneous Functions](#)

## Pulse

Pulses (jogs) a variable tag on, then off. The variable tag is switched ON (1) and two seconds later it is switched OFF (0). The exact period of the pulse is determined by the communication channel to the I/O device. If the

communication channel is busy, the pulse time may be longer than two seconds. The code in the I/O device should not be dependant on a pulse time of exactly 2 seconds. Use the pulse as a trigger only.

**Note:** The Pulse function cannot be used with Super Genie associations.

## Syntax

**Pulse(*sTag*)**

*sTag*:

The digital tag to pulse.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[Toggle](#)

## Example

Button

Text	Jog 145
Command	Pulse(M145)
Comment	Pulse the variable tag M145 every two seconds

## See Also

[Miscellaneous Functions](#)

## ServerDumpKernel

Dumps Kernel data to the KERNEL.DAT file (written to the Logs folder) either on a local server or on a remote server. The server is specified with the server and cluster parameters. Instead of creating a new file subsequent calls to this function will append Kernel data to the KERNEL.DAT file. This is a blocking function.

## Syntax

**ServerDumpKernel(*iMode*, *sName*, *sServer* [, *sCluster*])**

*iMode*:

The Kernel data to dump:

0x0001 - Dump general statistics.

0x0002 - Dump the task.  
0x0004 - Dump the I/O device.  
0x0008 - Dump the driver.  
0x0010 - Dump netstat. (This mode is deprecated and no longer active in version 7.10 or later.)  
0x0020 - Dump the table.  
0x0040 - Dump the queue.  
0x0100 - Dump list of current SCADA parameter settings  
0x4000 - Dump in verbose mode.  
0x8000 - Dump kernel data in non-verbose mode. To dump data in verbose mode, set *iMode* to 0xc000 (0x8000 + 0x4000).

You can select any one of the above modes or may add them together to get more than one type of information. For example, to dump netstat and I/O device data in verbose mode, set *iMode* to 0x4014 (0x0004 + 0x0010 + 0x4000). Using 0x4000 on its own will dump no data, it needs to be combined with another mode.

*sName*:

The queue or table name (empty for all queues or tables). Only valid if *iMode* is 0x0020 and 0x0040.

*sServer*:

Specifies the name of the server, as defined in the project, on which kernel data will be dumped.

*sCluster*:

Specifies the cluster of the server on which kernel data will be dumped. This parameter is optional in a single cluster environment.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[DumpKernel](#), [DspKernel](#), [KerCmd](#), [TraceMsg](#)

## Example

```
ServerDumpKernel(0x8000, "");  
!Dump the Kernel data
```

## See Also

[Miscellaneous Functions](#)

## ServiceGetList

Determines the service type(s), cluster name(s), and service name(s) of all services currently running in the component that called this function. It also determines if the client component is running.

If you call this function from a component that is running purely as a Control Client or View-only Client, the

function will return "Client".

If you call this function from a single-process component that includes:

- I/O server called IOServ1 on ClustA
- Trend server called Trend1 on ClustB
- Alarm server called Alarm1 on ClustA
- Report server called Report1 on ClustB
- Client

The function will return a string similar to:

"IOServer.ClustA.IOServ1,Trend.ClustB.Trend1,Alarm.ClustA.Alarm1,Report.ClustB.Report1,Client"

The order of components in the string is not fixed so the exact string may vary from the above. You should parse the returned string or alternatively use [ProcessIsClient](#) or [ProcessIsServer](#) to find the information you need.

## Syntax

**ServiceGetList()**

## Return Value

String in the form:

*serviceType1.clusterName1.serviceName1,serviceType2.clusterName2.serviceName2, ... ,Client*

## Related Functions

[ProcessIsClient](#), [ProcessIsServer](#), [ServerInfo](#), [ServerInfoEx](#)

## See Also

[Miscellaneous Functions](#)

## SetArea

Sets the current viewable areas. You can pass a single area number, or a group of areas to set multiple areas. You can only set areas that are flagged for the current user.

## Syntax

**SetArea(*Area*)**

*Area*:

The area to set (1 to 255).

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[GrpOpen](#)

## Example

```
/* Set current viewable area to Area 1. */  
SetArea(1);
```

## See Also

[Miscellaneous Functions](#)

## SetLogging

Adjusts logging parameters while online. The parameters you can modify include the following:

- [\[Debug\]DriverTrace](#)
- [\[Debug\]SysLogSize](#)
- [\[Debug\]Priority](#)
- [\[Debug\]CategoryFilterMode](#)
- [\[Debug\]CategoryFilter](#)
- [\[Debug\]SeverityFilterMode](#)
- [\[Debug\]SeverityFilter](#)
- [\[Debug\]LogShutDown](#)
- [\[Debug\]DebugAllTrans](#)
- [\[Debug\]EnableLogging](#)
- [\[IOServer\]RedundancyDebug](#)
- [\[General\]Verbose](#)
- [\[General\]VerboseToSysLog](#)
- [\[CtAPI\]Debug](#)

## Syntax

**SetLogging**(*Section*, *Name*, *Value*, *Persist*)

*Section*:

The INI section name.

*sName*:

The system parameter name.

*Value:*

The value you would like to set the parameter to.

*Persist:*

Makes the value persistent across restarts by writing the value to the INI file.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[GetLogging](#)

## See Also

[Miscellaneous Functions](#)

## Shutdown

Terminates Plant SCADA's operation. Use this function to shut down the Plant SCADA system, otherwise buffered data could be lost. This function impacts the operation of the computer as a whole, it is not process-based. If a computer is the target for shutdown, all Plant SCADA processes on the computer will be shutdown.

---

**Note:** With one exception, the Shutdown command will succeed only if there is an Alarm Server in your system. The exception to this is if you specify an empty string for the *sDest* parameter (shutdown this computer only). In this case the shutdown will succeed even if there is no Alarm Server.

---

The shutdown can affect only the computer that calls it, or all or part of a Plant SCADA network. If you are shutting down a network, specify the computers (Control Clients and servers) to be shut down in *sDest*, and the extent of the shutdown in *Mode*.

---

**Note:** If [\[Shutdown\]NetworkIgnore](#) parameter is set to 0 (zero) and a client receives a shutdown request message from a server, phase 2 clients only receive a shutdown request when the first phase 1 client has reconnected to the server. Set this parameter to 0 for all the computers that will send or accept a shutdown command.

---

You can allow selected computers to override the shutdown with the [\[Shutdown\]NetworkIgnore](#) parameter. (You might set this parameter for key servers, for example, I/O servers.)

---

Use the [ShutdownForm\(\)](#) function to prompt the user for verification before shutting Plant SCADA down.

---

**Note:** If the [\[Shutdown\]NetworkStart](#) parameter is set to 0 (zero), the [Shutdown\(\)](#) function will ignore the *sDest* argument. This will result in the shutting down and restarting of the machine the function is run on regardless of the machine specified.

## Syntax

**Shutdown( [*sDest*] [, *sProject*] [, *Mode*] [, *sClusterName*] [, *CallEvent*])**

*sDest:*

The destination computer(s) on which Plant SCADA will be shut down, as a string:

- "" (blank string) - this computer only (default value).
- ["Computer\_Name"] - a specified Plant SCADA computer. Use the name defined in the computer's CITECT.INI file. If this is not configured, you can use the Windows computer name.
- ["Server\_Name"] - a specified Plant SCADA server computer. Shutdown will only work if the specified server is the only one configured on the computer.
- "All Clients" - All Plant SCADA client computers on the network.
- "All Servers" - All Plant SCADA server computers on the network.
- "Everybody" - All Plant SCADA computers on the network.

*sProject:*

The full path of the project to run on restart as a string. The path is written to the configuration files and is used when the system restarts. The default value is "", which means that no changes are made to the configuration and the current project is restarted.

*Mode:*

The type of shutdown:

- 1 - Shutdown Plant SCADA only - Default value.
- 2 - Shutdown and restart Plant SCADA (without logging off Windows).
- 3 - Shutdown and restart Plant SCADA and log off Windows (an auto login to the Operating System and Plant SCADA needs to be configured to run on start up or log in).
- 4 - Shutdown Plant SCADA and re-boot the computer.
- 5 - Shutdown Plant SCADA only.
- 6 - Shutdown and restart Plant SCADA on remote computers, but not this computer.
- 7 - Shutdown Plant SCADA and shutdown the computer. If the computer supports power off feature the power will be turned off.

*sClusterName:*

The name of the cluster to which the machine(s) named in *Dest* belong. This is not required if:

- the function is called with *Dest* empty (the default); OR
- the client is connected to only one cluster containing an Alarm Server.

*CallEvent:*

Flag for initiating a user-specified shutdown event prior to shutting down. Refer to [OnEvent\(\)](#) type code for the value of shutdown event.

0 - no event

1 - raise event. The user defined shutdown event handler will be called before the shutdown occurs - Default value

---

**Note:** If the event handler is non-interactive with an instant return value, it can be called directly.

---

**Note:** If the event handler is interactive or with a long delay in processing the event, it needs to be called indirectly using the `NewTask("EventHandler")` function, and the actual handler, `EventHandler()`, needs to call `Shutdown()` with the `CallEvent` flag set to 0 from the handler if it decides the shutdown is permitted.

---

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[ProjectRestartGet](#), [ProjectRestartSet](#), [ProjectSet](#), [ShutdownMode](#), [ShutdownForm](#), [OnEvent](#)

## Example

```
/* Shut down Plant SCADA on this computer. */  
Shutdown();  
/* Shut down and restart Plant SCADA clients, but not this computer. */  
Shutdown("All Clients", ProjectRestartGet(), 6, "ClusterXYZ");
```

## See Also

[Miscellaneous Functions](#)

## ShutdownForm

Displays a dialog box to verify that the user really wants to shut down the Plant SCADA system. If the user selects [Yes], Plant SCADA is shut down.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**ShutdownForm()**

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[Shutdown](#)

## Example

System Keyboard

Key Sequence	Shutdown
Command	ShutdownForm();
Comment	Display the shutdown form

## Buttons

Text	Shutdown
Command	ShutdownForm();
Comment	Display the shutdown form

## See Also

[Miscellaneous Functions](#)

## ShutdownMode

Gets the mode of the last Shutdown function call. The mode is useful to identify the type of Shutdown that was performed.

## Syntax

**ShutdownMode()**

## Return Value

The shutdown mode set when shutdown was called.

## Related Functions

[Shutdown](#)

## Example

```
nMode = ShutdownMode()
If nMode = 4 Then
    Prompt ("Plant SCADA closed down and computer was rebooted")
END
```

## See Also

[Miscellaneous Functions](#)

## SwitchConfig

Switches focus to a specific Plant SCADA configuration application. If the specified application is not running, it will be started.

## Syntax

**SwitchConfig(*nApp*)**

*nApp*:

The configuration application:

- 1 - Graphics Builder (CTDRAW32.EXE)
- 2 - Not in Use
- 3 - Plant SCADA Studio (CITECTIDE.EXE)
- 4 - Cicode Editor (CTCICODE.EXE)

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Example

```
! Switch to the Graphics Builder.  
SwitchConfig(1);
```

## See Also

[Miscellaneous Functions](#)

## TestRandomWave

Generates a random wave. The height of the wave is restricted by minimum and maximum values. You can offset the starting point of the wave, for example, to display multiple waves at the same AN. You can use this function to generate test input.

## Syntax

**TestRandomWave( [*Period*] [, *Low*] [, *High*] [, *Offset*] )**

*Period*:

The number of seconds required to generate one cycle of the wave. If no period is entered, the period has a default of 60.

*Low*:

The lowest value of the wave. If no low value is entered, this value has a default of 0.

*High*:

The highest value of the wave. If no high value is entered, this value has a default of 100.

*Offset*:

The offset in seconds, to generate the wave. If no offset is entered, the offset has a default of 0.

## Return Value

The value of the random wave.

## Related Functions

[TestSawWave](#), [TestSinWave](#), [TestSquareWave](#), [TestTriangWave](#)

## Example

```
/* Specify a random wave of 60 seconds duration, with a minimum
value of 0 units and a maximum value of 100 units, with no offset.
*/
TestRandomWave(60,0,100,0);
```

## See Also

[Miscellaneous Functions](#)

## TestSawWave

Generates a saw wave. The height of the wave is restricted by minimum and maximum values. You can offset the starting point of the wave, for example, to display multiple waves at the same AN. You can use this function to generate test input.

## Syntax

**TestSawWave( [Period] [, Low] [, High] [, Offset] )**

*Period:*

The number of seconds required to generate one cycle of the wave. If no period is entered, the period has a default of 60.

*Low:*

The lowest value of the wave. If no low value is entered, this value has a default of 0.

*High:*

The highest value of the wave. If no high value is entered, this value has a default of 100.

*Offset:*

The offset in seconds, to generate the wave. If no offset is entered, the offset has a default of 0.

## Return Value

The value of the saw wave.

## Related Functions

[TestRandomWave](#), [TestSinWave](#), [TestSquareWave](#), [TestTriangWave](#)

## Example

```
/* Specifies a saw wave of 60 seconds duration, with a minimum
value of 0 units and a maximum value of 100 units, with no offset.
*/
TestSawWave();
```

## See Also

[Miscellaneous Functions](#)

## TestSinWave

Generates a sine wave. The height of the wave is restricted by minimum and maximum values. You can offset the starting point of the wave, for example, to display multiple waves at the same AN. You can use this function to generate test input.

## Syntax

**TestSinWave( [Period] [, Low] [, High] [, Offset] )**

*Period:*

The number of seconds required to generate one cycle of the wave. If no period is entered, the period has a default of 60.

*Low:*

The lowest value of the wave. If no low value is entered, this value has a default of 0.

*High:*

The highest value of the wave. If no high value is entered, this value has a default of 100.

*Offset:*

The offset in seconds, to generate the wave. If no offset is entered, the offset has a default of 0.

## Return Value

The value of the sine wave.

## Related Functions

[TestRandomWave](#), [TestSawWave](#), [TestSquareWave](#), [TestTriangWave](#)

## Example

```
/* Specifies a sine wave of 30 seconds duration, with a minimum
value of 0 units and a maximum value of 100 units, with no offset.
*/
TestSinWave(30);
```

## See Also

[Miscellaneous Functions](#)

### TestSquareWave

Generates a square wave. The height of the wave is restricted by minimum and maximum values. You can offset the starting point of the wave, for example, to display multiple waves at the same AN. You can use this function to generate test input.

## Syntax

**TestSquareWave( [Period] [, Low] [, High] [, Offset] )**

*Period:*

The number of seconds required to generate one cycle of the wave. If no period is entered, the period has a default of 60.

*Low:*

The lowest value of the wave. If no low value is entered, this value has a default of 0.

*High:*

The highest value of the wave. If no high value is entered, this value has a default of 100.

*Offset:*

The offset in seconds, to generate the wave. If no offset is entered, the offset has a default of 0.

## Return Value

The value of the square wave.

## Related Functions

[TestRandomWave](#), [TestSawWave](#), [TestSinWave](#), [TestTriangWave](#)

## Example

```
/* Specifies a square wave of 30 seconds duration, with a minimum
value of -1000 units and a maximum value of 1000 units, with an
offset of 10 seconds. */
TestSquareWave(30, -1000, 1000, 10);
```

## See Also

[Miscellaneous Functions](#)

### TestTriangWave

Generates a triangular wave. The height of the wave is restricted by minimum and maximum values. You can offset the starting point of the wave, for example, to display multiple waves at the same AN. You can use this function to generate test input.

## Syntax

**TestTriangWave( [Period] [, Low] [, High] [, Offset] )**

*Period:*

The number of seconds required to generate one cycle of the wave. If no period is entered, the period has a default of 60.

*Low:*

The lowest value of the wave. If no low value is entered, this value has a default of 0.

*High:*

The highest value of the wave. If no high value is entered, this value has a default of 100.

*Offset:*

The offset in seconds, to generate the wave. If no offset is entered, the offset has a default of 0.

## Return Value

The value of the triangular wave.

## Related Functions

[TestRandomWave](#), [TestSawWave](#), [TestSinWave](#), [TestSquareWave](#)

## Example

```
/* Specifies a triangular wave of 60 seconds duration, with a
minimum value of 0 units and a maximum value of 100 units, with no
offset. */
TestTriangWave();
```

## See Also

[Miscellaneous Functions](#)

## Toggle

Toggles a digital tag on or off. If the variable tag is ON (1), this function will turn it OFF. If the variable tag is OFF (0), this function will turn it ON.

## Syntax

**Toggle(*sTag*)**

*sTag*:

The digital tag to toggle.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[Pulse](#)

## Example

Buttons

Text	Motor 145
Command	Toggle(M145)
Comment	Toggle the variable tag M145 (Motor 145) on or off

To toggle the Paging Alarm property:

```
Toggle(MyCluster.Alarm_1.Paging);
```

## See Also

[Miscellaneous Functions](#)

## TraceMsg

Displays a message in the Kernel and Debugger debug windows.

## Syntax

**TraceMsg(*String*)**

*String*:

The message to display.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[DspKernel](#), [KerCmd](#), [DumpKernel](#), [DebugBreak](#), [Assert](#), [DebugMsg](#), [DebugMsgSet](#), [ErrLog](#)

## Example

```
TraceMsg("Error Found");
// Displays "Error Found" in the debug window
```

## See Also

[Miscellaneous Functions](#)

## Version

Gets the version number of the Plant SCADA software in use.

## Syntax

**Version(*nType*)**

*nType*:

The type of version:

- 0 - Major version number
- 1 - Minor version number
- 2 - Revision number
- 3 - Version text

## Return Value

The version number as a string.

## Example

```
! If the Plant SCADA version number is 1.2:
Version(0);
! Returns 1.
Version(1);
! Returns 2.
Version(3);
! Returns "1.2".
```

## See Also

[Miscellaneous Functions](#)

## .Net Functions

### CAUTION

#### CAUTION

.Net Functions do not support events and static methods. Do not use these functions to manipulate objects on graphics pages.

The following functions allow you to access web services via .net objects.

<a href="#">DIIClassDispose</a>	Use this function to clean up resources used by the .Net object and any other .Net objects created via the use of the object.
<a href="#">DIIClassCreate</a>	Use this function to instantiate a new .Net object by specifying the path, class and arguments required for the matching constructor of the class.
<a href="#">DIIClassGetProperty</a>	Use this function to get a property of the .Net object.
<a href="#">DIIClassIsValid</a>	Use this function to validate class. Uses the handle of the class returned from DIIClassCreate.
<a href="#">DIIClassCallMethod</a>	Use this function to call a method of a .Net object, passing in the method name and any arguments required for the matching prototype of the method.
<a href="#">DIIClass SetProperty</a>	Use this function to set a property of the .Net object. The property may be of any type or an object itself.

When returning data from a .net assembly to Cicode, use the [Conversion Table](#).

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

### **DIIClassDispose**

Use this function to clean up resources used by the .Net object and any other objects created via the use of the object.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT DllClassDispose(OBJECT Object)**

*Object:*

.Net object.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[DllClassCreate](#), [DllClassIsValid](#), [DllClassCallMethod](#), [DllClassSetProperty](#), [DllClassGetProperty](#)

## Example

See [DllClassCreate](#)

## See Also

[.Net Functions](#)

## DllClassCreate

Use this function to instantiate a new .Net object by specifying the path, class and arguments required for the matching constructor of the class.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

---

**Note:** If your .Net assembly was compiled for 32 bit, it will not be supported on an alarm server operating in Extended Memory mode. To allow this to work, you will need to recompile the assembly to target "Any CPU" or "x64".

---

## Syntax

**OBJECT DllClassCreate(STRING sPath, STRING sClass [, vParameters Args])**

*sPath:*

The full path string.

*sClass:*

The class of the object.

*Args:*

Args for the constructor.

## Return Value

.Net object if successful, otherwise an error code is returned.

## Related Functions

[DIIClassDispose](#), [DIIClassIsValid](#), [DIIClassCallMethod](#), [DIIClassSetProperty](#), [DIIClassGetProperty](#)

## Example

```
FUNCTION RunDllProxyTest()
    STRING sPath = "C:\Program Files (x86)\Reference Assemblies\Microsoft\
Framework\.NETFramework\v4.0\Profile\Client\System.dll";
    OBJECT hMailMessageProxy;
    OBJECT hSmtpClientProxy;
    OBJECT hEmailAddressProxy;
    INT result;
    OBJECT hEmailAddressCollection;
    hMailMessageProxy = DllClassCreate(sPath, "MailMessage");
    result = DllClassIsValid(hMailMessageProxy);
    hSmtpClientProxy = DllClassCreate(sPath, "SmtpClient", "mysmtpserver.mycompany.com");
    result = DllClassIsValid(hSmtpClientProxy);
    hEmailAddressProxy = DllClassCreate(sPath, "MailAddress", "myemail@hotmail.com");
    result = DllClassIsValid(hEmailAddressProxy);
    result = DllClassSetProperty(hMailMessageProxy, "From", hEmailAddressProxy);
    hEmailAddressCollection = DllClassGetProperty(hMailMessageProxy, "To");
    DllClassCallMethod(hEmailAddressCollection, "Add", "myemail@mycompany.com");
    result = DllClassSetProperty(hMailMessageProxy, "Subject", "Test");
    result = DllClassSetProperty(hMailMessageProxy, "Body", "Hello!");
    DllClassCallMethod(hSmtpClientProxy, "Send", hMailMessageProxy);

    result = DllClassDispose(hEmailAddressProxy);
    result = DllClassDispose(hSmtpClientProxy);
    result = DllClassDispose(hMailMessageProxy);
END
```

## See Also

[.Net Functions](#)

### DIIClassIsValid

Use this function to validate the handle for the class returned from [DIIClassCreate](#).

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT DIIClassIsValid(*OBJECT*)**

## Return Value

1 if handle is valid or 0

## Related Functions

[DIIClassCreate](#), [DIIClassDispose](#), [DIIClassCallMethod](#), [DIIClassSetProperty](#), [DIIClassGetProperty](#)

## Example

See [DIIClassCreate](#)

## See Also

[.Net Functions](#)

## DIIClassSetProperty

Use this function to set a property of the .Net object. The property may be of any type or an object itself. This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT DIIClassSetProperty(OBJECT *object*, STRING *sProperty*, VARIANT *Value*)

*object*:

.Net object.

*sProperty*

The name of the property of the .Net object.

*Value*

The value to which the property of the .Net object will be set to.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[DIIClassDispose](#), [DIIClassIsValid](#), [DIIClassCallMethod](#), [DIIClassCreate](#), [DIIClassGetProperty](#)

## Example

See [DIIClassCreate](#).

## See Also

[.Net Functions](#)

### DllClassGetProperty

Use this function to get a property of the .Net object.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

VAR DllClassGetProperty(OBJECT *object*, STRING *sProperty*)

*object*:

.Net object.

*sProperty*:

The property to read.

## Return Value

Var if successful, otherwise an error code is returned.

## Related Functions

[DllClassDispose](#), [DllClassIsValid](#), [DllClassCallMethod](#), [DllClassCreate](#), [DllClassSetProperty](#)

## Example

See [DllClassCreate](#).

## See Also

[.Net Functions](#)

### DllClassCallMethod

Use this function to call a method of a .Net object, passing in the method name and any arguments required for the matching prototype of the method.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

**Note:** This function is unable to call a static method as it requires an instance of the class to be created before a method can be called from it.

## Syntax

**VAR DllClassCallMethod(OBJECT *object*, STRING *sMethod*, VARARGS *args*)**

*object*:

.Net object.

*sMethod*:

The name of the method

*args*

Args for functions.

## Return Value

Var if successful, otherwise an error code is returned.

## Related Functions

[DllClassDispose](#), [DllClassIsValid](#), [DllClass SetProperty](#), [DllClassCreate](#), [DllClassGetProperty](#)

## Example

See [DllClassCreate](#)

## See Also

[.Net Functions](#)

## Conversion Table

 CAUTION	
<b>CAUTION</b>	
When returning data from a .net assembly to Cicode, use the following conversion table so that there is no loss of data. Failure to do so will result in a hardware error and the Cicode will stop execution.	
.Net Data Type	Cicode Data Type
Object	OBJECT
DateTime	OBJECT
Boolean	INT
Byte	INT

.Net Data Type	Cicode Data Type
SByte	INT
Int16	INT
UInt16	INT
Int32	INT
UInt32	REAL
Int64	REAL
UInt64	REAL
Single	REAL
Double	REAL
Decimal	REAL
Char	STRING
String	STRING

## See Also

[.Net Functions](#)

## Page Functions

Following are functions relating to graphics pages:

<a href="#">PageAlarm</a>	Displays a category of active alarms on the predefined alarms page.
<a href="#">PageBack</a>	Displays the previously displayed page in the Window.
<a href="#">PageDisabled</a>	Displays a category of disabled alarms on the predefined alarms page.
<a href="#">PageDisplay</a>	Displays a graphics page.
<a href="#">PageExists</a>	Checks if a particular page exists in your project.
<a href="#">PageFile</a>	Displays a file on the predefined file to screen page.
<a href="#">PageFileInfo</a>	Returns the width or height of an unopened page in pixels.

<a href="#">PageFileInfoEx</a>	Returns the width, height or content type of an unopened page.
<a href="#">PageForward</a>	PageForward() restores the previously displayed page in the window following a PageBack command.
<a href="#">PageGetInt</a>	Gets a local page-based integer.
<a href="#">PageGetStr</a>	Gets a local page-based string.
<a href="#">PageGoto</a>	Displays a graphics page without pushing the last page onto the page navigation history.
<a href="#">PageHardware</a>	Displays the active hardware alarms on the predefined alarms page.
<a href="#">PageHistoryDspMenu</a>	Displays a pop-up menu which lists the page history of current window.
<a href="#">PageHistoryEmpty</a>	Used to determine if the page history of the current window is empty.
<a href="#">PageHome</a>	Displays the predefined home page in the window.
<a href="#">PageInfo</a>	Gets page information.
<a href="#">PageLast</a>	Displays the last ten graphics pages.
<a href="#">PageListCount</a>	Gets number of pages in the page list of the current window.
<a href="#">PageListCurrent</a>	Gets index of the current page in the page list of the current window.
<a href="#">PageListInfo</a>	Gets information of a page at the specific index in the page list of current window.
<a href="#">PageListDisplay</a>	Recalls (displays) a page at the specific index in the page list of the current window, and moves the current index to the page. When a page is recalled, the original parameters (such as cluster context, super genie associations, PageTask arguments if applicable) used to display the page will be restored.
<a href="#">PageListDelete</a>	Removes a page at the specific index from the page list of the current window.
<a href="#">PageMenu</a>	Displays a menu page with page selection buttons.
<a href="#">PageNext</a>	Displays the next graphics page.
<a href="#">PagePeekCurrent</a>	Return the index in the page navigation history for the

	current page..
<a href="#">PagePeekLast</a>	Gets information about a Page at an offset in the page navigation history.
<a href="#">PagePopLast</a>	Gets the last page in the page navigation history.
<a href="#">PagePopUp</a>	Displays the pop up window at the mouse position.
<a href="#">PagePrev</a>	Displays the previous graphics page.
<a href="#">PageProcessAnalyst</a>	Displays a Process Analyst page (in the same window) preloaded with the pre-defined Process Analyst View (PAV) file.
<a href="#">PageProcessAnalystPens</a>	Display a page and add the specified pens to the first pane of the specified PA object on the page.
<a href="#">PagePushLast</a>	Puts a page on the end of the page navigation history.
<a href="#">PageRecall</a>	Displays the page at a specified depth in the page navigation history.
<a href="#">PageRichTextFile</a>	Used as a page entry function - creates a rich text object, and loads a rich text file into that object.
<a href="#">PageSelect</a>	Displays a dialog box with a list of graphics pages defined in the project.
<a href="#">PageSetInt</a>	Stores a local page-based integer.
<a href="#">PageSetStr</a>	Stores a local page-based string.
<a href="#">PageSOE</a>	Displays a category of sequence of events (SOE) entries on the SOE page.
<a href="#">PageSummary</a>	Displays a category of alarm summary entries on the predefined alarm page.
<a href="#">PageTask</a>	Used for running preliminary Cicode before displaying a page in a window.
<a href="#">PageTransformCoords</a>	Converts Page coordinates to absolute screen coordinates.
<a href="#">PageTrend</a>	Displays a standard trend page in a single cluster system.
<a href="#">PageTrendEx</a>	Displays a standard trend page in a multi-cluster system.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## PageAlarm

Displays a category of active alarms on the alarm page.

To use this function, you need to have a page in your project that was created using the Alarm template. By default, the name of the page is expected to be "Alarm". However, you can use an alarm page with a different name by adjusting the setting for the INI parameter [\[Page\]AlarmPage](#).

---

**Note:**

- The operation of this function has changed. In Version 2.xx this function displayed the built-in alarm page from the Include project.
  - This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.
- 

## Syntax

**PageAlarm( [Category] )**

*Category:*

The alarm category to display. Set to 0 (the default) to display all alarm categories.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[PageDisabled](#), [PageHardware](#)

## Example

System Keyboard

Key Sequence	AlarmPage
Command	PageAlarm(0)
Comment	Display active alarms on the alarm page

System Keyboard

Key Sequence	Alarm ### Enter
Command	PageAlarm(Arg1)

Comment	Display a specified category of active alarms on the alarm page
---------	---

## See Also

[Page Functions](#)

### PageBack

Displays the previously displayed page in the Window.

## Syntax

**PageBack([iCount])**

*iCount*:

Optional parameter to specify the number of pages to move back. Default value is 1.

## Return Value

No Error (0) on success. Bad handle specified (269) if current window handle does not correspond to a valid window. Invalid argument passed (274) if count is outside of allowable bounds.

## Related Functions

[PageForward](#), [PageHistoryDspMenu](#), [PageHistoryEmpty](#)

## See Also

[Page Functions](#)

### PageDisabled

Displays a category of disabled alarms on the disabled alarms page.

To use this function, you need to have a page in your project that was created using the Disabled template. By default, the name of the page is expected to be "Disabled". However, you can use a page with a different name by adjusting the setting for the INI parameter [\[Page\]DisabledPage](#).

---

#### Note:

- The operation of this function has changed. In Version 2.xx this function displayed the built-in alarm page from the Include project.
  - This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.
-

## Syntax

**PageDisabled( [Category] )**

*Category:*

The alarm category to display. Set to 0 (the default) to display all alarm categories.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[PageAlarm](#)

## Example

System Keyboard

Key Sequence	DisabledPage
Command	PageDisabled(0)
Comment	Display disabled alarms on the alarm page

System Keyboard

Key Sequence	Disabled ### Enter
Command	PageDisabled(Arg1)
Comment	Display a specified category of disabled alarms on the alarm page

## See Also

[Page Functions](#)

## PageDisplay

Displays a graphics page in the active window. The page needs to be in one of the operator's current areas. The page to be displayed is identified by its Page Name or the Page Number.

---

**Note:** If you just need to update the associations on a Super Genie window (rather than reloading the entire page), you can use the function [AssWinReplace](#).

You can specify if the page operates within the context of a particular cluster in a multiple cluster project. When the page launches during runtime, the ClusterName argument is used to resolve any tags that have a cluster omitted.

Plant SCADA stores the current page before it displays the required page. You can call PageLast() to re-display the pages in the page navigation history.

You cannot call this function from the Exit command field (see [Edit the Properties for a Page](#)) or a Cicode object.

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.

## Syntax

**PageDisplay(*Page*,[*ClusterName*])**

*Page*:

The name or page number of the page to display (in quotation marks ""). Can be prefixed by the name of a host cluster, that is "ClusterName.Page". This will take precedence over the use of the ClusterName parameter if the two differ.

*ClusterName*:

The name of the cluster that will accommodate the page at runtime (in quotation marks ""). The specified cluster is used to resolve any tags that have a cluster omitted. If the Page parameter is prefixed with the name of a cluster, this parameter will not be used.

This parameter is optional, however if you omit a cluster context in the Page properties, then any tags which omit an explicit Cluster. TagName will be ambiguous and become unresolved if you have multiple clusters defined in the project. Note that this ambiguity and resulting unresolved state will only occur if the parameter [\[General\]TagDB](#) is set to "0" which disables variable tag database loading.

## Return Value

0 (zero) if the page is successfully displayed, otherwise an error is returned.

## Related Functions

[PageGoto](#), [PageLast](#)

## Example

Buttons

Text	Mimic Page
Command	PageDisplay("Mimic")
Comment	Display the "Mimic" page

System Keyboard

Key Sequence	Page ##### Enter
Command	PageDisplay(Arg1)
Comment	Display a specified page

```
PageDisplay("MIMIC1");
! Displays page "MIMIC1".
PageDisplay("MIMIC2");
/* Displays page "MIMIC2" and places page "MIMIC1" onto the
page navigation history. */
PageDisplay("10");
/* Displays page "10" and places page "MIMIC2" onto the page navigation history. */
PageLast();
/* Displays the last page, that is page "MIMIC2" and
removes it from the page navigation history. */
```

**Note:** Before Plant SCADA version 5.0, page records could be edited in the Project Editor. One of the fields available for configuration was "Page Number". The value entered for a page could then be used in runtime with the Page Cicode functions such as PageDisplay(), PageGoto(), and PageInfo(1). For example, PageDisplay("1") can be used to display the page that has "1" (without the quotes) set in the **Page Number** field. PageInfo(1) returns the Page Number of the current page. From version 5.0 on, this feature is only backwards-supported. The "Alias" field in the project Pages.DBF file still contains the Page Numbers from upgraded projects; however, the Pages database records are no longer available for direct editing in Plant SCADA.

## See Also

[Page Functions](#)

## PageExists

Use this function to check if a particular page exists in your project.

## Syntax

**INT PageExists(STRING[sPageName])**

*sPageName*:

The name of the page to check for.

## Return Value

1 if the page does exist, 0 if the page does not.

## See Also

[Page Functions](#)

## PageFile

Displays a file on the page. After the file is displayed, you can scroll up and down through the file. To use this function, you need to use the Graphics Builder to create a page called "File" (using the file template).

**Note:**

- The operation of this function has changed. In Version 2.xx this function displayed the built-in alarm page from

---

the Include project.

- This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.
- 

## Syntax

**PageFile(*sName*)**

*sName*:

The name of the file to display.

## Return Value

0 (zero) if the file is successfully displayed, otherwise an error is returned.

## Related Functions

[DspFile](#)

## Example

System Keyboard

Key Sequence	File ##### Enter
Command	PageFile(Arg1)
Comment	Display the specified file on the page

## See Also

[Page Functions](#)

## PageFileInfo

Returns the width or height of an unopened page.

## Syntax

**PageFileInfo(*sPageName*, *nMode*)**

*sPageName*:

The name of the page you would like to retrieve size information for.

*nMode*:

Retrieves either the width or the height of the specified page in pixels.

0 - returns the page width

1 - returns the page height

## Return Value

The height or width of the specified page in pixels, depending on the value set for *nMode*.

## See Also

[Page Functions](#)

### PageFileInfoEx

Returns the width, height, content type, parent or title of an unopened page.

## Syntax

**PageFileInfoEx(*sPage*, *nMode*)**

*sPageName*:

The name of the page for which you would like to retrieve information.

*nMode*:

Determines if the width, height, content type, parent or title is returned for the specified page.

0 - returns the page width.

1 - returns the page height.

2 - returns the content type of the page.

3 - returns the parent page of the page.

4 - returns the configured page title of the page.

## Return Value

The height, width, content type, parent or title of the specified page, depending on the value set for *nMode*.

## See Also

[Page Functions](#)

### PageForward

If PageBack() is called, PageForward() will restore the previously displayed page in the window. PageForward requires PageBack to have been called and no other page display functions to have been called in between. Calling the display functions PageDisplay or PageGoto will remove the forward pages from the Page Navigation History.

## Syntax

**PageForward([*iCount*])**

*iCount*:

Optional parameter to specify the number of pages to move forward. Default value is 1.

## Return Value

No Error (0) on success. Bad handle specified (269) if current window handle does not correspond to a valid window. Invalid argument passed (274) if count is outside of allowable bounds.

## Related Functions

[PageBack](#), [PageHistoryDspMenu](#), [PageHistoryEmpty](#), [PageDisplay](#), [PageGoto](#)

## See Also

[Page Functions](#)

## PageGetInt

Returns the integer value associated with a variable name on a particular page. If two or more windows are displayed, each window has a unique copy of the variable. You can use these variables in Cicode to keep track of variables unique to each window.

---

**Note:**

- You can find out the current setting for [Page]ScanTime parameter, by calling this function as follows: PageGetInt(-2).
  - This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.
- 

## Syntax

**PageGetInt(*sLabel* [, *iWinNum*])**

*sLabel*:

String name of the variable to return

*iWinNum*:

Window number of the page. Default is current window.

## Return Value

Integer stored in variable *sLabel*.

## Related Functions

[PageSetInt](#), [PageGetStr](#), [PageSetStr](#)

## See Also

[Page Functions](#)

### PageGetStr

Gets the string associated with a variable name on a particular page. Page-based variables are local to each display page. If two or more windows are displayed, each window has a unique copy of the variable. You can use these variables in Cicode to keep track of variables unique to each window.

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.

## Syntax

**PageGetStr(*sLabel* [, *iWinNum*])**

*sLabel*:

String name of the variable to return

*iWinNum*:

Window number of the page. Default is current window.

## Return Value

Value stored in variable *sLabel*.

## Related Functions

[PageSetInt](#), [PageGetInt](#), [PageSetStr](#)

## See Also

[Page Functions](#)

### PageGoto

Displays a graphics page in the active window. The page needs to be in one of the operator's current areas. You can specify either the Page Name or the Page Number of the graphics page.

You can also specify if the page operates within the context of a particular cluster in a multiple cluster project. When the page launches during runtime, the ClusterName argument is used to resolve any tags that have the cluster name omitted.

This function is similar to PageDisplay(), however PageGoto() does not put the current page into the page

navigation history.

You cannot call this function from the Exit command field (see [Edit the Properties for a Page](#)) or a Cicode Object.

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.

## Syntax

**PageGoto(*Page*, *ClusterName*)**

*Page*:

The name or page number of the page to display (in quotation marks ""). Can be prefixed by the name of a host cluster, that is "ClusterName.Page". This will take precedence over the use of the ClusterName parameter if the two differ.

*sClusterName*:

The name of the cluster that will accommodate the page at runtime (in quotation marks ""). The specified cluster is used to resolve any tags that have the cluster name omitted. If the Page parameter is prefixed with the name of a cluster, this parameter will not be used.

## Return Value

0 (zero) if the page is successfully displayed, otherwise an error is returned.

## Related Functions

[PageDisplay](#)

## Example

```
PageDisplay("MIMIC1");
! Displays page "MIMIC1".
PageDisplay("MIMIC2");
/* Displays page "MIMIC2" and places page "MIMIC1" onto the
page navigation history. */
PageGoto("10");
/* Displays page "10". Page "MIMIC2" is not placed onto the
page navigation history. */
```

**Note:** Before Plant SCADA version 5.0, page records could be edited in the Project Editor. One of the fields available for configuration was "Page Number". The value entered for a page could then be used in runtime with the Page Cicode functions such as PageDisplay(), PageGoto(), and PageInfo(1).

For example, PageDisplay("1") can be used to display the page that has "1" (without the quotes) set in the **Page Number** field. PageInfo(1) returns the Page Number of the current page.

From version 5.0 on, this feature is only backwards-supported. The "Alias" field in the project Pages.DBF file still contains the Page Numbers from upgraded projects; however, the Pages database records are no longer available for direct editing in Plant SCADA.

## See Also

[Page Functions](#)

### PageHardware

Displays the active hardware alarms on the hardware alarms page.

To use this function, you need to have a page in your project that was created using the Hardware template. By default, the name of the page is expected to be "Hardware". However, you can use a page with a different name by adjusting the setting for the INI parameter [\[Page\]HardwarePage](#).

---

**Note:**

- This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.
  - When similar hardware alarms are triggered, for example "Tag not found" alarms, the hardware alarm page and hardware alarm queue show the last invalid entry. The previous entry of the same description is overwritten.
- 

## Syntax

`PageHardware()`

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[PageAlarm](#)

## Example

System Keyboard

Key Sequence	Hardware
Command	<code>PageHardware()</code>
Comment	Display active hardware alarms on the hardware alarms page

## See Also

[Page Functions](#)

## PageHistoryDspMenu

Displays a pop-up menu which lists the page history of current window. The user can select any page in the history to recall the page. When full page history is specified, the currently displayed page will also be listed and marked in the menu.

## Syntax

**PageHistoryDspMenu([*iType*])**

*iType*:

The type of page history to be listed:

0 - full history (default)

1 - back history

2 - forward history

## Return Value

Zero (0) if the function is executed successfully. Otherwise an error is returned.

## Related Functions

[PageBack](#), [PageForward](#), [PageHistoryEmpty](#)

## See Also

[Page Functions](#)

## PageHistoryEmpty

Used to determine if the page history of the current window is empty. The currently displayed page is not counted as history.

## Syntax

**PageHistoryEmpty([*iType*])**

*iType*:

The type of page history to be checked:

0 - full history (default)

1 - back history

2 - forward history

## Return Value

1 if page history of specified type is empty, or 0 if it is not empty.

## Related Functions

[PageBack](#), [PageForward](#), [PageHistoryDspMenu](#)

## See Also

[Page Functions](#)

### PageHome

Displays the predefined home page in the window.

## Syntax

**PageHome([*sCluster*])**

*sCluster*:

Optional parameter to the Cluster to associate the page being opened with. Default value "".

## Return Value

No error(0) on success. Bad handle specified (269) if current window handle does not correspond to a valid window. Bad handle specified (274) if INI parameter [\[Page\]HomePage](#) is not set.

## See Also

[Page Functions](#)

### PageInfo

Gets information about the current page.

---

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.

---

## Syntax

**PageInfo(*nType*)**

*nType*:

The type of page information required.

0 — Page name.

- 1 — Page number.
- 2 — Page title.
- 3 — Display file name.
- 4 — Symbol file name.
- 5 — Next page name.
- 6 — Previous page name
- 7 — Previous display count, incremented at each page scan. The page scan rate defaults to the value of the Citect.ini parameter [Page]ScanTime, and can be overridden per page by changing the scan time setting in the General tab of the page properties in Graphics Builder.
- 8 — Parent window number. Returns -1 if there is no parent.
- 9 — First child window number. Returns -1 if there are no children.
- 10 — Next child in child link. Returns -1 for the end of the list.
- 11 — Window mode (set by the WinNewAt() function).
- 12 — Width of window. If the target is a pinned window, the return value will represent the width of the window as it is on the unscaled version of the page that hosts it.
- 13 — Height of window. If the target is a pinned window, the return value will represent the height of the window as it is on the unscaled version of the page that hosts it.
- 14 — X position of window. If the target is a pinned window, the value will reflect the coordinates for the window based on its location on the unscaled version of the page that hosts it.
- 15 — Y position of window. If the target is a pinned window, the value will reflect the coordinates for the window based on its location on the unscaled version of the page that hosts it.
- 16 — Dynamic window horizontal scale.
- 17 — Dynamic window vertical scale.
- Types 16 and 17 return a real number between 0 and 1 (as a STRING) and will be identical, as the dynamic scaling does not allow a change in the aspect ratio.
- 18 — Flashing color state. Type 18 returns one of the following:
- "0" — the palette does not flash.
  - "1" — the palette is primary now.
  - "2" — the palette is secondary now.
- 19 — In animation cycle. Returns a 1 (true) or 0 (false).
- 20 — In communications cycle. Returns a 1 (true) or 0 (false).
- 21 — Width of background page.
- 22 — Height of background page.
- 23 — Returns the highest AN.
- 24 — Returns the number of ANs.
- 25 — Indicates when the page's "On Page Shown" event has been triggered. Returns 1 if triggered, 0 if it has not.
- 26 — The cluster that has been specified to host the page. Returns the cluster name, or an empty string if no cluster has been specified.
- 27 — Indicates whether the Cicode library used by the page is different from the currently loaded library. Returns 1 if different, 0 if the versions are the same.

- 28 — Return X Coordinate of Client rectangle origin.
- 29 — Return Y Coordinate of Client rectangle origin.
- 30 — Returns the name of the monitor the current page is displayed on at runtime. For example, in a project where three monitors "Screen 1", "Primary", and "Screen 2" have been configured as part of a multi-monitor screen profile, PageInfo(30) has been set on the monitor named "Left" which means "Left" would be displayed at runtime. For more information refer to the topic [Screen Profiles](#) in the main help.
- 31 — Returns the "Content Type" configured for that page.

## Return Value

The information (as a string).

## Related Functions

[PageDisplay](#), [PageTransformCoords](#)

## Example

```
! If currently on page "MIMIC1";
Variable=PageInfo(0);
! Sets Variable to "MIMIC1".
```

**Note:** Before Plant SCADA version 5.0, page records could be edited in the Project Editor. One of the fields available for configuration was "Page Number". The value entered for a page could then be used in runtime with the Page Cicode functions such as PageDisplay(), PageGoto(), and PageInfo(1).

For example, PageDisplay("1") can be used to display the page that has "1" (without the quotes) set in the **PageNumber** field. PageInfo(1) returns the Page Number of the current page.

From version 5.0 on, this feature is only backwards-supported. The "Alias" field in the project Pages.DBF file still contains the Page Numbers from upgraded projects; however, the Pages database records are no longer available for direct editing in Plant SCADA.

## See Also

[Page Functions](#)

## PageLast

Displays the graphics page that was last displayed. With this function, you can successively recall the last ten pages that were displayed.

Graphics pages displayed using this command cannot be subsequently recalled.

You cannot call this function from the Exit command field (see [Edit the Properties for a Page](#)) or a Cicode Object.

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.

## Syntax

**PageLast()**

## Return Value

0 (zero) if the page is successfully displayed, otherwise an error is returned.

## Related Functions

[PagePeekLast](#), [PagePopLast](#), [PagePushLast](#)

## Example

Buttons

Text	Last Page
Command	PageLast()
Comment	Display the graphics page that was last displayed

```
PageDisplay("MIMIC1");
! Displays page "MIMIC1".
PageDisplay("MIMIC2");
/* Displays page "MIMIC2" and places page "MIMIC1" onto the
page navigation history. */
PageDisplay("10");
! Displays page "10" and places page "MIMIC2" onto the page navigation history. */
PageLast();
/* Displays the last page, that is page "MIMIC2" and
removes it from the
page navigation history. */
PageLast();
/* Displays the last page, that is page "MIMIC1" and
removes it from the
page navigation history. */
PageLast();
/* Returns an "Out of range" error code as there are no more pages in the
page navigation history.*/
```

## See Also

[Page Functions](#)

## PageListCount

Gets number of pages in the page list of the current window.

See the Citect.ini parameter [\[Page\]MaxList](#).

## Syntax

```
PageListCount()
```

## Return Value

Number of pages in the list

## Related Functions

[PageListCurrent](#), [PageListInfo](#), [PageListDisplay](#), [PageListDelete](#)

## Example

```
INT count;
INT index;
INT error;
STRING PageName;

// While 4 entries will be added to page history, only 3 unique pages will be added to page
list
PageDisplay(Page1);
PageDisplay(Page2);
PageDisplay(Page1);
PageDisplay(Page3);

// iterate the pages in the list
count=PageListCount() //count=3
PageName=PageListInfo(0,0); //PageName=Page1
PageName=PageListInfo(1,0); //PageName=Page2
PageName=PageListInfo(2,0); //PageName=Page3
index=PageListCurrent(); //index=2, as Page3 is currently displayed
pageName=PageListInfo(index,0); //PageName=Page3
error=PageListDisplay(1); //recall Page2, error=0
index=PageListCurrent(); //index=1, as current index is moved as part of the recall
PageName=PageInfo(0); //PageName=Page2 as it becomes the current page
error=PageListDelete(index); //remove current page, i.e. Page2, error=0
PageName=PageInfo(0); //PageName=Page3, next page in the list is displayed
index=PageListCurrent(); //index=1, current index is not changed while the list is reduced
count=PageListCount(); //count=2, only 2 pages left in the list
```

## See Also

[Page Functions](#)

### PageListCurrent

Gets Index of the current page in the page list of current window.

## Syntax

```
PageListCurrent()
```

## Return Value

List index (0 to list size -1), or -1 if unable to get index

## Related Functions

[PageListCount](#), [PageListInfo](#), [PageListDisplay](#), [PageListDelete](#)

## Example

```
INT count;
INT index;
INT error;
STRING PageName;

// While 4 entries will be added to page history, only 3 unique pages will be added to page
list
PageDisplay(Page1);
PageDisplay(Page2);
PageDisplay(Page1);
PageDisplay(Page3);

// iterate the pages in the list
count=PageListCount() //count=3
PageName=PageListInfo(0,0); //PageName=Page1
PageName=PageListInfo(1,0); //PageName=Page2
PageName=PageListInfo(2,0); //PageName=Page3
index=PageListCurrent(); //index=2, as Page3 is currently displayed
pageName=PageListInfo(index,0); //PageName=Page3
error=PageListRecall(1); //recall Page2, error=0
index=PageListCurrent(); //index=1, as current index is moved as part of the recall
PageName=PageInfo(0); //PageName=Page2 as it becomes the current page
error=PageListDelete(index); //remove current page, i.e. Page2, error=0
PageName=PageInfo(0); //PageName=Page3, next page in the list is displayed
index=PageListCurrent(); //index=1, current index is not changed while the list is reduced
count=PageListCount(); //count=2, only 2 pages left in the list
```

## See Also

[Page Functions](#)

### PageListInfo

Gets information of a page at the specific index in the page list of the current window.

## Syntax

**PageListInfo(INT index,[ INT type])**

*Index*

Index to the page list (valid range 0 to list size minus 1)

*Type*

Information to return:

0- Page name (default)

1- Configured Page Title

2- Active Page Title

## Return Value

String value of the requested information, or empty string if no valid information can be found

## Related Functions

[PageListCount](#), [PageListCurrent](#), [PageListDisplay](#)

## Example

```
INT count;
INT index;
INT error;
STRING PageName;

// While 4 entries will be added to page history, only 3 unique pages will be added to page
list
PageDisplay(Page1);
PageDisplay(Page2);
PageDisplay(Page1);
PageDisplay(Page3);

// iterate the pages in the list
count=PageListCount() //count=3
PageName=PageListInfo(0,0); //PageName=Page1
PageName=PageListInfo(1,0); //PageName=Page2
PageName=PageListInfo(2,0); //PageName=Page3
index=PageListCurrent(); //index=2, as Page3 is currently displayed
pageName=PageListInfo(index,0); //PageName=Page3
error=PageListDisplay(1); //recall Page2, error=0
index=PageListCurrent(); //index=1, as current index is moved as part of the recall
PageName=PageInfo(0); //PageName=Page2 as it becomes the current page
error=PageListDelete(index); //remove current page, i.e. Page2, error=0
PageName=PageInfo(0); //PageName=Page3, next page in the list is displayed
index=PageListCurrent(); //index=1, current index is not changed while the list is reduced
count=PageListCount(); //count=2, only 2 pages left in the list
```

## See Also

[Page Functions](#)

### PageListDisplay

Recalls (displays) a page at the specific index in the page list of current window, and moves the current index to the page. When a page is recalled, the original parameters (such as cluster context, supergenie associations, PageTask arguments if applicable) used to display the page will be restored.

(See the Citect.ini parameter [\[Page\]MaxList](#).

## Syntax

**PageListDisplay(INT index)**

*Index*

Index to the page list (valid range 0 to list size minus 1)

## Return Value

0 (zero) if the page is successfully displayed, otherwise an error is returned.

## Related Functions

[PageListCount](#), [PageListCurrent](#), [PageListInfo](#), [PageListDelete](#)

## Example

```
INT count;
INT index;
INT error;
STRING PageName;

// While 4 entries will be added to page history, only 3 unique pages will be added to page
list
PageDisplay(Page1);
PageDisplay(Page2);
PageDisplay(Page1);
PageDisplay(Page3);

// iterate the pages in the list
count=PageListCount() //count=3
PageName=PageListInfo(0,0); //PageName=Page1
PageName=PageListInfo(1,0); //PageName=Page2
PageName=PageListInfo(2,0); //PageName=Page3
index=PageListCurrent(); //index=2, as Page3 is currently displayed
pageName=PageListInfo(index,0); //PageName=Page3
error=PageListDisplay(1); //recall Page2, error=0
index=PageListCurrent(); //index=1, as current index is moved as part of the recall
```

```
PageName=PageInfo(0); //PageName=Page2 as it becomes the current page
error=PageListDelete(index); //remove current page, i.e. Page2, error=0
PageName=PageInfo(0); //PageName=Page3, next page in the list is displayed
index=PageListCurrent(); //index=1, current index is not changed while the list is reduced
count=PageListCount(); //count=2, only 2 pages left in the list
```

## See Also

[Page Functions](#)

### PageListDelete

Removes a page at the specific index from the page list of the current window. This function will return an error if only one page is in the list. If the page is currently displayed, the next page will be displayed. If there is no next page, the previous page will be displayed.

## Syntax

```
PageListDelete(INT index)
Index
Index to the page list (valid range 0 to list size minus 1)
```

## Return Value

0 (zero) if the page is successfully removed, otherwise an error is returned.

## Related Functions

[PageListCount](#), [PageListCurrent](#), [PageListInfo](#), [PageListDisplay](#)

## Example

```
INT count;
INT index;
INT error;
STRING PageName;

// While 4 entries will be added to page history, only 3 unique pages will be added to page
list
PageDisplay(Page1);
PageDisplay(Page2);
PageDisplay(Page1);
PageDisplay(Page3);

// iterate the pages in the list
count=PageListCount() //count=3
PageName=PageListInfo(0,0); //PageName=Page1
PageName=PageListInfo(1,0); //PageName=Page2
```

```
PageName=PageListInfo(2,0); //PageName=Page3
index=PageListCurrent(); //index=2, as Page3 is currently displayed
pageName=PageListInfo(index,0); //PageName=Page3
error=PageListDisplay(1); //recall Page2, error=0
index=PageListCurrent(); //index=1, as current index is moved as part of the recall
PageName=PageInfo(0); //PageName=Page2 as it becomes the current page
error=PageListDelete(index); //remove current page, i.e. Page2, error=0
PageName=PageInfo(0); //PageName=Page3, next page in the list is displayed
index=PageListCurrent(); //index=1, current index is not changed while the list is reduced
count=PageListCount(); //count=2, only 2 pages left in the list
```

## See Also

[Page Functions](#)

## PageMenu

Displays a menu page with page selection buttons. A page goto button is displayed for each of the first 40 pages defined in the project.

## Syntax

**PageMenu()**

## Return Value

0 (zero) if the page is successfully displayed, otherwise an error is returned.

## Related Functions

[PageGoto](#), [PageLast](#), [PagePrev](#), [PageSelect](#)

## Example

### Buttons

Text	Menu
Command	PageMenu()
Comment	Display the menu page

### System Keyboard

Key Sequence	Menu
Command	PageMenu()
Comment	Display the menu page

## See Also

[Page Functions](#)

### PageNext

Displays the next page as specified in the project.

You cannot call this function from the Exit command field (see [Edit the Properties for a Page](#)) or a Cicode object.

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.

## Syntax

**PageNext()**

## Return Value

0 (zero) if the page is successfully displayed, otherwise an error is returned.

## Related Functions

[PagePrev](#)

## Example

### Buttons

Text	Page Next
Command	PageNext()
Comment	Display the next page in the browse sequence

### System Keyboard

Key Sequence	PageNext
Command	PageNext()
Comment	Display the next page in the browse sequence

```
/* If graphics page 1 is currently displayed, and the graphics
page 1 has Next Page Name=10. */
PageNext();
! Displays graphics page 10.
```

## See Also

[Page Functions](#)

### Page.PeekCurrent

Return the index in the page navigation history for the current page.

## Syntax

**Page.PeekCurrent()**

## Return Value

Index in the page navigation history for the current page. -1 indicates that current window handle does not correspond to a valid window.

## Related Functions

[Page.PeekLast](#), [Page.PopLast](#), [Page.PopUp](#), [Page.PushLast](#), [Page.Recall](#)

## See Also

[Page Functions](#)

### Page.PeekLast

Gets information about a page at an offset in the page navigation history.

---

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.

---

## Syntax

**Page.PeekLast(*iOffset* [, *iType*] )**

*iOffset*:

Pages are navigated using page navigation history. *iOffset* is used to indicate the position in that history. An *iOffset* value of "0" represents the currently viewed page. Positive increments of *iOffset* indicate previous pages visited in the navigation history. Negative values of *iOffset* are used to indicate pages in the future of the navigation history (negative *iOffsets* exist only if the user has used the back button to revisit earlier pages in the history).

For example:

If you navigate pages in the sequence A, B, C, D, E, and then remain on Page E. The *iOffset* value indicates the following pages:

*iOffset* value: 4 3 2 1 0

corresponding page: A B C D E

If you then navigate "Back" twice the iOffset value indicates the following pages:

iOffset value: 2 1 0 -1 -2

corresponding page: A B C D E

If you further navigate to pages F then G the iOffset value indicates the following pages:

iOffset value: 4 3 2 1 0

corresponding page: A B C F G

*iType*:

An enumeration representing the type of information required. The default value is 0.

0 - Page Name

1 - Configured Page Title

2 - Active Page Title

## Return Value

String value of the requested information, or empty string if no valid result for given arguments.

## Related Functions

[PagePeekCurrent](#), [PagePopLast](#), [PagePopUp](#), [PagePushLast](#)

## See Also

[Page Functions](#)

## PagePopLast

Gets the Page Name of the last item in the page navigation history and removes the page from the history.

---

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.

---

## Syntax

`PagePopLast()`

## Return Value

The page name or an empty string if there is no last page.

## Related Functions

[PageLast](#), [PagePeekCurrent](#), [PagePeekLast](#), [PagePopUp](#), [PagePushLast](#)

## Example

```
PageDisplay("MIMIC1");
! Displays page "MIMIC1".
PageDisplay("MIMIC2");
/* Displays page "MIMIC2" and places page "MIMIC1" onto the page navigation history. */
PageDisplay("10");
! Displays page "10" and places page "MIMIC2" onto the page navigation history.
Variable=PagePopLast();
/* Sets Variable to "MIMIC2" and removes "MIMIC2" from the page navigation history. */
PageLast();
! Displays page "MIMIC1".
```

## See Also

[Page Functions](#)

### PagePopUp

Display pop up window at the mouse position. If the mouse position is not known then the pop up will display in the centre of the screen. The window is displayed with no resize and will be closed if the page is changed.

## Syntax

**PagePopUp(*sPage*, [, *sClusterName*])**

*sPage*:

The name of the page (drawn with the Graphics Builder).

*sClusterName*:

The name of the cluster that will accommodate the page at runtime (in quotation marks ""). The specified cluster is used to resolve any tags that have a cluster omitted. If the Page parameter is prefixed with the name of a cluster, this parameter will not be used.

This parameter is optional, however if you omit a cluster context in the Page properties, then any tags which omit an explicit Cluster. TagName will be ambiguous and become unresolved if you have multiple clusters defined in the project. Note that this ambiguity and resulting unresolved state will only occur if the parameter [\[General\]TagDB](#) is set to "0" which disables variable tag database loading.

---

**Note:** Before Plant SCADA version 5.0, page records could be edited in the Project Editor. One of the fields available for configuration was "Page Number". The value entered for a page could then be used in runtime with the Page Cicode functions such as PageDisplay(), PageGoto(), and PageInfo(1).

For example, PageDisplay("1") can be used to display the page that has "1" (without the quotes) set in the **Page Number** field. PageInfo(1) returns the Page Number of the current page.

From version 5.0 on, this feature is only backwards-supported. The "Alias" field in the project Pages.DBF file still contains the Page Numbers from upgraded projects; however, the Pages database records are no longer available for direct editing in Plant SCADA.

---

## Related Functions

[PageLast](#), [PagePeekCurrent](#), [PagePeekLast](#), [PagePopLast](#), [PagePushLast](#)

## See Also

[Page Functions](#)

### PagePrev

Displays the previous page as specified in the project.

You cannot call this function from the Exit command field (see [Edit the Properties for a Page](#)) or a Cicode Object.

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.

## Syntax

**PagePrev()**

## Return Value

0 (zero) if the page is successfully displayed, otherwise an error is returned.

## Related Functions

[PageNext](#)

## Example

Buttons

Text	Page Previous
Command	PagePrev()
Comment	Display the previous page in the browse sequence

System Keyboard

Key Sequence	PagePrev
Command	PagePrev()
Comment	Display the previous page in the browse sequence

```
/* If graphics page 10 is currently displayed, and graphics page
10 has Prev Page Name=1. */
PagePrev();
! Displays graphics page 1.
```

## See Also

[Page Functions](#)

### PageProcessAnalyst

Displays a Process Analyst page (in the same window) preloaded with the pre-defined Process Analyst View (PAV) file.

## Syntax

```
PageProcessAnalyst(sPage, sPAVFile1 [, iFileLocation1 [, iButtonMask1 [, sObjName1 [, sPAVFile2 [, iFileLocation2  
[, iButtonMask2 [, sObjName2 ]]]]]]])
```

*sPage*:

The name of the page that contains Process Analyst object(s). For example, pages based on the Process Analyst templates found in the Tab\_Style\_Include project.

*sPAVFile1*:

Name of the 1st PAV file

*iFileLocation1*:

PAV file location code for the 1st PAV file, see PA doc LoadFromFile() for details.

*iButtonMask1*:

Bit mask for removing command buttons from the 1st PA, bit flags as shown below:

1 - Load View

2 - Save View

4 - Print

8 - Copy to Clipboard

16 - Copy to File

32 - Add Pens

64 - Remove Pens

128 - Show Properties

256 - Help

*sObjName1*:

Name of the PA object on the given Page where the 1st PAV file will be loaded. If this parameter is not specified or empty string, it is defaulted to the object name used in the tab style templates, that is "\_templatePA1".

*sPAVFile2*:

Name of the 2nd PAV file

*iFileLocation2*:

PAV file location code for the 2nd PAV file

*iButtonMask2*:

Bit mask for removing command buttons from the 2nd PA, refer *iButtonMask1* for details

*sObjName2*:

Name of the PA object on the given Page where the 2nd PAV file will be loaded. If this parameter is not specified or empty string, it is defaulted to the object name used in the tab style templates, that is "\_templatePA2".

## Return Value

Zero (0) if the page is successfully displayed. Otherwise an error is returned.

## Related Functions

[PageProcessAnalystPens](#), [ProcessAnalystLoadFile](#), [ProcessAnalystPopup](#), [ProcessAnalystSelect](#),  
[ProcessAnalystSetPen](#), [ProcessAnalystWin](#), [TrnSetPen](#), [WinNewAt](#)

## See Also

[Page Functions](#)

### PageProcessAnalystPens

Display a page and add the specified pens to the first pane of the specified PA object on the page. If a PAV file is also specified, it will be loaded first, and the pens in the first pane will be removed before the specified pens are created on the PA.

## Syntax

**PageProcessAnalystPens**(*sPage*, *sTag1* [, *sTag2..sTag8* [, *iButtonMask* [, *sObjName* [, *iPane* [, *sPAVFile* [, *iFileLocation* ]]]]])

*sPage*:

The name of the page that displays the PA.

*sTag1..sTag8*:

Up to 8 Trend tags can be added to the PA.

*iButtonMask*:

Mask to remove button(s) from the main tool bar of PA. The following values can be combined to remove multiple buttons:

1 - Load View

2 - Save View

4 - Print

8 - Copy to Clipboard

16 - Copy to File

32 - Add Pens

64 - Remove Pens

128 - Show Properties

256 - Help

***sObjName***

The name of the PA object. If not specified it is defaulted to "\_templatePA1" which is the name used by the built-in templates.

***iPane***

The pane in PA where the trend or variable tags are added. If this is not specified or less than 1, it is defaulted to 1 (the 1st pane). If the specified pane does not exist in the PA object, a new pane will be created.

***sPAVFile:***

Optional Process Analyst View file to be loaded, default ="".

***iFileLocation:***

Optional location of the PAV file. The allowed values are:

- 0 - Analyst Views subfolder in project folder (default)
- 1 - Folders specified in properties primary/standby server path
- 2 - My documents folder

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[PageProcessAnalyst](#), [ProcessAnalystLoadFile](#), [ProcessAnalystPopup](#), [ProcessAnalystSelect](#), [ProcessAnalystSetPen](#),  
[ProcessAnalystWin](#), [TrnSetPen](#), [WinNewAt](#)

## See Also

[Page Functions](#)

## PagePushLast

Places a page at the end of the page navigation history.

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.

## Syntax

**PagePushLast(*Page*)**

***Page:***

The Page Name or Page Number (of the page) to place at the end of the page navigation history.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[PageLast](#), [PagePeekCurrent](#), [PagePeekLast](#), [PagePopLast](#), [PagePopUp](#)

## Example

```
PageDisplay("MIMIC1");
! Displays page "MIMIC1".
PageDisplay("MIMIC2");
/* Displays page "MIMIC2" and places page "MIMIC1" onto the page navigation history. */
PageDisplay("10");
! Displays page "10" and places page "MIMIC2" onto the page navigation history.
PagePushLast("TREND1");
! Places page "TREND1" onto the page navigation history.
PageLast();
/* Displays the last page, that is page "TREND1" and removes it from the page navigation
history. */
PageLast();
/* Displays the last page, that is page "MIMIC2" and removes it from the page navigation
history. */
```

## See Also

[Page Functions](#)

## PageRecall

Displays the page at a specified depth in the page navigation history.

## Syntax

**PageRecall(*iIndex*)**

*iIndex*:

The index into the page navigation history of the Page to be displayed. To get the index for the currently displayed page, call [PagePeekCurrent\(\)](#). Then increment it to access pages in the forward history, or decrement it to access pages in the backward history. Be reminded that you cannot recall the page that is currently displayed.

## Return Value

No error(0) on success. Bad handle specified (269) if current window handle does not correspond to a valid window. Bad handle specified (274) if index is outside of allowable bounds.

## Related Function

[PagePeekCurrent](#)

## See Also

[Page Functions](#)

### PageRichTextFile

This function creates a rich edit object, and loads a copy of the rich text file *Filename* into that object. The rich text object will be rectangular in shape, with dimensions determined by *nHeight*, and *nWidth*. If you do not specify *nHeight* and *nWidth*, *nAN* will define the position of one corner, and (*AN* + 1) the position of the diagonally opposite corner. This function would often be used as a page entry function.

## Syntax

**PageRichTextFile(*nAN*, *Filename*, *nMode* [, *nHeight*] [, *nWidth*] )**

*nAN*:

The animation point at which to display the rich text object.

*Filename*:

The name of the file to be copied and loaded into the rich text object. The filename needs to be entered in quotation marks "".

If you are loading a copy of an RTF report, the report needs to have already been run and saved to a file.

Remember that the filename for the saved report comes from the File Name field in the Devices form. The location of the saved file needs to also be included as part of the filename. For example, if the filename in the Devices settings listed [Data];RepDev.rtf, then you would need to enter "[Data]\repdev.rtf" as your argument. Alternatively, you can manually enter the path, for example, "c:\MyApplication\data\repdev.rtf".

If you are keeping a number of history files for the report, instead of using the rtf extension, you need to change it to reflect the number of the desired history file, for example, 001.

*nMode*:

The display mode for the rich text object. The mode can be any combination of:

0 - Disabled - should be used if the rich text object is to be used for display purposes only.

1 - Enabled - allows you to select and copy the contents of the RTF object (for instance an RTF report), but you will not be able to make changes.

2 - Read/Write - allows you to edit the contents of the RTF object. Remember, however, that the object needs to be enabled before it can be edited. If it has already been enabled, you can just enter Mode 2 as your argument. If it is not already enabled, you will need to enable it. By combining Mode 1 and Mode 2 in your argument (3), you can enable the object, and make it read/write at the same time.

Because the content of the rich text object is just a copy of the original file, changes will not affect the actual file, until saved using the [DspRichTextSave](#) function.

*nHeight*:

The height of the rich text object in pixels. The height is established by measuring down from the animation point.

If you do not enter a height and width, the size of the object will be determined by the position of *nAN* and *AN+1*.

*nWidth*:

The width of the rich text object in pixels. The width is established by measuring across to the right of the animation point.

If you do not enter a height and width, the size of the object will be determined by the position of *nAN* and *AN+1*.

## Return Value

None

## Related Functions

[DspRichText](#), [DspRichTextLoad](#), [DspRichTextEdit](#), [DspRichTextEnable](#), [DspRichTextGetInfo](#), [DspRichTextPgScroll](#),  
[DspRichTextPrint](#), [DspRichTextSave](#), [DspRichTextScroll](#), [FileRichTextPrint](#)

## Example

```
PageRichTextFile(108,"f:\Plant SCADA\data\richtext.rtf",0);

// This function would produce a rich text object at animation
point 108. Into this object a copy of f:\Plant SCADA\data\richtext.rtf
would then be loaded. Remember, richtext.rtf is the name of the
output file for the report, as specified in the Devices form.
Because 0 was specified as the nMode for this example, the
contents of this object will be display only. //

PageRichTextFile(53,"[Data]\richtext.rtf",1);

//This function would produce a rich text object at animation
point 53. Into this object a copy of [Data]\richtext.rtf would
then be loaded. It will be possible to select and copy the
contents of the object, but not make changes. //
```

## See Also

[Page Functions](#)

## PageSelect

Displays a dialog box with a list of graphics pages defined in the project. AN operator can select a page name for display.

## Syntax

**PageSelect()**

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[PageGoto](#), [PageLast](#), [PagePrev](#), [PageMenu](#)

## Example

Buttons

Text	Page Select
Command	PageSelect()
Comment	Display the page selection dialog box

```
PageSelect();  
! Displays the page selection dialog box.
```

## See Also

[Page Functions](#)

## PageSetInt

Associates an integer variable with a particular page. Page-based variables are stored in an array, local to each display page. This function allows you to save integer variables in temporary storage.

---

**Note:**

- You can dynamically change the setting for [\[Page\]ScanTime](#) parameter, by calling this function as follows:  
PageSetInt(-2, <scantime>).
  - This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.
- 

## Syntax

**PageSetInt**(*sLabel*, *sVar* [, *iWinNum*])

*sLabel*:

String name of the variable which will contain *sValue*.

*sVar*:

The integer to store.

*iWinNum*:

Window number of the page. Default is current window.

## Return Value

No error(0) on success. 269 if *WinNum* handle does not correspond to a valid window. 274 if Label or Var is not a valid variable.

## Related Functions

[PageGetInt](#), [PageGetStr](#), [PageSetStr](#)

## See Also

[Page Functions](#)

### PageSetStr

Stores a local page-based string and associates the string with the page. Page-based variables are stored in an array, local to each display page. This function allows you to save string variables in temporary storage.

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.

## Syntax

**PageSetStr(*sLabel*, *sVar* [, *iWinNum*])**

*sLabel*:

String name of the variable which will contain *sVar*.

*sVar*:

The string to store. The string length is 128 characters.

*iWinNum*:

Window number of the page. Default is current window.

## Return Value

No error(0) on success. 269 if WinNum handle does not correspond to a valid window. 274 if Label or Var is not a valid variable.

## Related Functions

[PageGetInt](#), [PageGetStr](#), [PageSetInt](#)

## See Also

[Page Functions](#)

### PageSOE

Displays a category of sequence of events (SOE) entries on the SOE page.

To use this function, you need to have a page in your project that was created using the SOE template. By default, the name of the page is expected to be "SOE". However, you can use an alarm page with different name

by adjusting the setting for the INI parameter [\[Page\]SOEPage](#).

## Syntax

**PageSOE(INT Category, INT Fallback)**

*Category:*

The category number for the alarms you want to display their events

*Fallback:*

Whether to display the Summary page instead if the SOE page does not exist. If not specified it defaults to FALSE (0).

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[PageAlarm](#)

## Example

## See Also

[Page Functions](#)

## PageSummary

Displays a category of alarm summary entries on the alarms summary page.

To use this function, you need to have a page in your project that was created using the Summary template. By default, the name of the page is expected to be "Summary". However, you can use an alarm page with a different name by adjusting the setting for the INI parameter [\[Page\]SummaryPage](#).

---

**Note:**

- The operation of this function has changed. In Version 2.xx this function displayed the built-in summary alarm page from the Include project.
  - This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.
- 

## Syntax

**PageSummary(Category)**

*Category:*

The category number for the alarms you want to summarise.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[PageAlarm](#)

## Example

System Keyboard

Key Sequence	SummaryPage
Command	PageSummary(0)
Comment	Display alarm summary entries on the alarm page

System Keyboard

Key Sequence	Summary ### Enter
Command	PageSummary(Arg1)
Comment	Display a specified category of alarm summary entries on the alarm page

## See Also

[Page Functions](#)

## PageTask

PageTask() is used for running preliminary Cicode before displaying a page in a window. It makes it possible for the same Cicode to be run if the page is re-entered by navigating forward or back. PageTask() is similar to TaskNew().

PageTask() returns a handle to a code task the first time it is run. The custom Cicode of the *sFunctionName* parameter needs to call PageDisplay() in order to display the page. When the page changes, the function and its parameters will be pushed onto the Page Navigation History. The Cicode fnTask will be called again when the page is navigated to using the PageForward or PageBack functions.

## Syntax

**PageTask(*iWinNum*, *sFunctionName*, *sFunctionArg*)**

*iWinNum*:

The Window number of the window in which to display the page.

*sFunctionName*:

String representing the Cicode function to run each time the page is navigated to using the forward and backward navigation functions.

*sFunctionArg*:

String representing the parameters to use with function fnTask.

## Return Value

A handle to a code task the first time it is run. BAD\_HANDLE (-1) if the function did not complete.

## Related Functions

[PageDisplay](#), [PageBack](#), [PageForward](#)

## Example

```
FUNCTION DisplayAlarmLog()
    PageTask(WinNumber(), "_DisplayAlarmLog", "");
END

FUNCTION _DisplayAlarmLog()
    PageFile("[Data]:AlarmLog.Txt");
    DspFile(21,DspFont("Courier", -10, Black, Transparent), 24, 120);
    WinTitle("@(Alarm History)");
END
```

## See Also

[Page Functions](#)

## PageTransformCoords

Converts Page coordinates to absolute screen coordinates.

## Syntax

**PageTransformCoords(*hPage*, *iPageX*, *iPageY*, *iDisplayX*, *iDisplayY*, *iType*)**

*hPage*:

Page handle of the relevant Window.

*PageX*:

X coordinate of page coordinate.

*iPageY*:

Y coordinate of page coordinate.

*iDisplayX*:

Output parameter: Transformed X coordinate. Must be a Long type variable.

*iDisplayY*:

Output parameter: Transformed Y coordinate. Must be a Long type variable.

*iType:*

The value of output coordinate:

0 - Absolute screen coordinates

1 - Coordinates relative to Window origin

2 - Coordinates relative to Client rectangle origin

## Return Value

0 – Success

269 – ERROR\_BAD\_HANDLE – hPage is not a valid Page handle

274 – ERROR\_INVALID\_ARG – either DisplayX, DisplayY or Type is invalid.

## Related Functions

[PageInfo](#), [DspGetMouse](#), [DspAnGetPos](#), [DspGetAnExtent](#)

## See Also

[Page Functions](#)

## PageTrend

Displays a trend page with the specified trend pens. Use this function to display trends in a single cluster system with a single trend page. You need to create the trend page with the Graphics Builder and set the pen names to blank. Then display that page by calling this function and passing the required trend tags. Call this function from a menu of trend pages.

---

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised. For a multi-cluster system use [PageTrendEx](#).

## Syntax

**PageTrend**(*sPage*, *sTag1* [, *sTag2..sTag8*] )

*sPage:*

Name of the trend page (drawn with the Graphics Builder).

*sTag1:*

The first trend tag to display on the page.

*sTag2..sTag8:*

Optionally trend tags 2 to 8 to display on the page.

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** Before Plant SCADA version 5.0, page records could be edited in the Project Editor. One of the fields available for configuration was "Page Number". The value entered for a page could then be used in runtime with the Page Cicode functions such as PageDisplay(), PageGoto(), and PageInfo(1).

For example, PageDisplay("1") can be used to display the page that has "1" (without the quotes) set in the **Page Number** field. PageInfo(1) returns the Page Number of the current page.

From version 5.0 on, this feature is only backwards-supported. The "Alias" field in the project Pages.DBF file still contains the Page Numbers from upgraded projects; however, the Pages database records are no longer available for direct editing in Plant SCADA.

## Related Functions

[TrnNew](#), [TrnSelect](#), [TrendWin](#), [TrendPopUp](#), [PageTrendEx](#)

## Example

Buttons

Text	Process Trend
Command	PageTrend("MyTrend", "PV1", "PV2", "PV3")
Comment	Display the trend page with three trend pens

```
PageTrend("MyTrend", "PV1", "PV2", "PV3")
/* Display three trend tags on a single trend page. */
```

## See Also

[Page Functions](#)

## PageTrendEx

Displays a trend page of a specified cluster in a multi-cluster system with the specified trend pens. Use this function to display trends in a mult-cluster system with a single trend page. You need to create the trend page with the Graphics Builder and set the pen names to blank. Then display that page by calling this function and passing the required trend tags. Call this function from a menu of trend pages. This function can also be used in a single cluster system, the *sCluster* argument is optional in such a case.

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.

## Syntax

**PageTrendEx(*sPage*, *sCluster*, *sTag1* [*, sTag2..sTag8*] )**

*sPage*:

Name of the trend page (drawn with the Graphics Builder).

*sCluster*:

Name of the cluster in which the trend page is located.

*sTag1:*

The first trend tag to display on the page.

*sTag2..sTag8:*

Optionally trend tags 2 to 8 to display on the page.

## Return Value

0 (zero) if successful, otherwise an error is returned.

**Note:** Before Plant SCADA version 5.0, page records could be edited in the Project Editor. One of the fields available for configuration was "Page Number". The value entered for a page could then be used in runtime with the Page Cicode functions such as PageDisplay(), PageGoto(), and PageInfo(1).

For example, PageDisplay("1") can be used to display the page that has "1" (without the quotes) set in the **Page Number** field. PageInfo(1) returns the Page Number of the current page.

From version 5.0 on, this feature is only backwards-supported. The "Alias" field in the project Pages.DBF file still contains the Page Numbers from upgraded projects; however, the Pages database records are no longer available for direct editing in Plant SCADA.

## Related Functions

[TrnNew](#), [TrnSelect](#), [TrendWin](#), [TrendPopUp](#), [PageTrend](#)

## Example

Buttons

Text	Process Trend
Command	PageTrendEx("MyTrend", "MyCluster", "PV1", "PV2", "PV3")
Comment	Display the trend page on the specified cluster with three trend pens

```
PageTrendEx("MyTrend", "MyCluster", "PV1", "PV2", "PV3")
/* Display three trend tags on a single trend page on the
specified cluster. */
```

## See Also

[Page Functions](#)

## Plot Functions

Following are functions relating to plotting data:

<a href="#">PlotClose</a>	Displays and/or prints the plot, then closes the plot.
<a href="#">PlotDraw</a>	Draws a point, line, box, or circle on a plot.
<a href="#">PlotFile</a>	This function is now obsolete.
<a href="#">PlotGetMarker</a>	Gets the marker number of a symbol that is registered as a marker.
<a href="#">PlotGrid</a>	Draws gridlines to be used for plotted lines.
<a href="#">PlotInfo</a>	Gets information about a plot.
<a href="#">PlotLine</a>	Plots a line through a set of data points.
<a href="#">PlotMarker</a>	Draws markers on a plotted line or at a specified point.
<a href="#">PlotOpen</a>	Opens a new plot, sets its output device, and returns a plot handle for use by the other plot functions.
<a href="#">PlotScaleMarker</a>	Draws scale lines (with markers) beside the grid on your plot (if there is one).
<a href="#">PlotSetMarker</a>	Sets (registers) a symbol as a marker.
<a href="#">PlotText</a>	Draws text on a plot.
<a href="#">PlotXYLine</a>	Draws an XY line through a set of data points.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## PlotClose

Displays the plot on screen or sends it to the printer (depending on the output device you specified in the PlotOpen() function), then closes the plot. Once the plot is closed, it cannot be used.

## Syntax

**PlotClose(*hPlot*)**

*hPlot*:

The plot handle, returned from the PlotOpen() function. The plot handle identifies the table where all data on the plot is stored.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[PlotDraw](#), [PlotGetMarker](#), [PlotGrid](#), [PlotInfo](#), [PlotLine](#), [PlotMarker](#), [PlotOpen](#), [PlotScaleMarker](#), [PlotSetMarker](#),  
[PlotText](#), [PlotXYLine](#), [TrnPlot](#)

## Example

See [PlotOpen](#).

## See Also

[Plot Functions](#)

## PlotDraw

Constructs drawings on your plot. Use the coordinates (X1,Y1) and (X2,Y2) to define a point, line, rectangle, square, circle, or ellipse. You can specify the style, color, and width of the pen, and a fill color for a box or circular shape.

you need to call the PlotOpen() function first, to get the handle for the plot (*hPlot*) and to specify the output device.

## Syntax

**PlotDraw(*hPlot*, *Type*, *PenStyle*, *PenCol*, *PenWidth*, *nFill*, *X1*, *Y1*, *X2*, *Y2*)**

*hPlot*:

The plot handle, returned from the PlotOpen() function. The plot handle identifies the table where data on the plot is stored.

*nType*:

The type of drawing:

1 - Rectangle or square

2 - Circle or ellipse

3 - Line

4 - Point

*PenStyle*:

The style of the pen used to draw:

0 - Solid

1 - Dash ( - - - - )

2 - Dot (.....)

- 3 - Dash and dot ( - . - . - . - )
- 4 - Dash, dot, dot ( - . - . - . - - )

5 - Hollow

*PenCol:*

The color of the pen (flashing color is not supported). Select a color from the list of Predefined Color Names and Codes or create an RGB-based color using the function [MakeColour](#).

*PenWidth:*

Pen width in pixels. If the width is thicker than one pixel, you need to use a solid pen (PenStyle = 0). Maximum width is 32.

*nFill:*

The fill color of the rectangle, square, circle, or ellipse (flashing color is not supported). Select a color from the list of predefined color names and codes or create an RGB-based color using the function [MakeColour](#). For a point or line, nFill is ignored.

*X1, Y1:*

X and y coordinates (in pixels) of the upper-left corner of the drawing (the origin).

*X2, Y2:*

X and y coordinates (in pixels) of the lower-right corner of the drawing.

For a point, (X1,Y1) and (X2,Y2) are assumed to be the same, so (X2,Y2) is ignored. To draw a circle or ellipse, enter the coordinates for a square or rectangle; the circle or ellipse is automatically drawn within the box.

If the plot is for display on the screen, coordinates are relative to the AN specified in the PlotOpen() function. If the output device is a printer, coordinates are relative to the point (0,0).

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[PlotClose](#), [PlotGetMarker](#), [PlotGrid](#), [PlotInfo](#), [PlotLine](#), [PlotMarker](#), [PlotOpen](#), [PlotScaleMarker](#), [PlotSetMarker](#),  
[PlotText](#), [PlotXYLine](#), [TrnPlot](#)

## Example

See [PlotOpen](#).

## See Also

[Plot Functions](#)

## PlotGetMarker

Gets the marker number of a symbol. The symbol needs to be a symbol and registered with the PlotSetMarker() function.

## Syntax

**PlotGetMarker(*sSymbolName*)**

*sSymbolName*:

The library name and symbol name ("Library.Symbol") of the symbol that is registered as a marker.

## Return Value

The marker number if successful, otherwise -1 is returned.

## Related Functions

[PlotMarker](#), [PlotScaleMarker](#), [PlotSetMarker](#)

## Example

```
/*Assume that the symbol was registered by PlotSetMarker function */
PlotSetMarker(20,"Global.Hourglass");
/*Later on, this symbol can be used as shown below*/
hPlot = PlotOpen(36,"Display",1);
..
/* Display red hourglass as marker at point (100,200) on AN36. */
MarkerNo = PlotGetMarker("Global.Hourglass");
PlotMarker(hPlot,MarkerNo,red,1,1,100,200);
..
PlotClose(hPlot);
```

## See Also

[Plot Functions](#)

## PlotGrid

Defines a frame and draws horizontal and vertical grid lines within this frame. These grid lines can then be used by the PlotLine(), PlotXYLine(), and PlotScaleMarker() functions. You need to define the frame for a plot before you can plot points with the PlotLine() and PlotXYLine() functions. *nSamples* specifies the maximum number of samples that can be plotted for a single line. If you set *FrameWidth* to 0 (zero), the frame will be defined but not displayed (however, the plot will still be displayed).

You can specify the number of grid lines and their color, as well as the background color which will fill the frame. If *nHorGrid* and *nVerGrid* are set to 0 (zero), then the grid lines will not be drawn.

you need to call the PlotOpen() function, first, to get the handle for the plot (*hPlot*), and to specify the output device. Then call this function to set up the frame and grid. You can then call the PlotScaleMarker() function to draw scale lines beside the frame, and call the PlotLine() or PlotXYLine() to plot a set of data points.

## Syntax

**PlotGrid(*hPlot*, *nSamples*, *X1*, *Y1*, *X2*, *Y2*, *nHorGrid*, *HorGridCol*, *nVerGrid*, *VerGridCol*, *FrameWidth*, *FrameCol*, *nFill*, *nMode*)**

*hPlot*:

Plot handle, returned from the PlotOpen() function. The plot handle identifies the table where data on the plot is stored.

*nSamples*:

The maximum number of samples that can be plotted for a single line in this grid (valid values between 2 and 16000 inclusive). For example, if you set *nSamples* to 10, then plot 2 lines in this grid (using the PlotLine() function), each line will be plotted with a maximum of 10 samples. For this example, a line can possess less than 10 samples, but if it has more, it will be shortened to 10 samples.

*X1*, *Y1*:

The x and y coordinates of the upper-left corner of the frame containing the grid lines.

*X2*, *Y2*:

The x and y coordinates of the lower-right corner of the frame containing the grid lines.

If the plot is for display on the screen, you should set (*X1*,*Y1*) to (0,0). The origin of the frame is then positioned at the AN specified in the PlotOpen() function.

If the output device is a printer, both (*X1*,*Y1*) and (*X2*,*Y2*) are relative to the point (0,0).

*nHorGrid*:

The number of rows (formed by the horizontal grid lines) to draw within the frame. If there is no need of grid lines, set *nHorGrid* to 0 (zero) and *HorGridCol* to 0. *nHorGrid* cannot exceed the pixel width of the plot.

*HorGridCol*:

The color of the horizontal grid lines (flashing color is not supported). Select a color from the list of Predefined Color Names and Codes or create an RGB-based color using the function [MakeColour](#).

*nVerGrid*:

The number of columns (formed by the vertical grid lines) to draw within the frame. If there is no need of grid lines, set *nVerGrid* to 0 (zero) and *VerGridCol* to 0. *nVerGrid* cannot exceed the pixel height of the plot.

*VerGridCol*:

The color of the vertical grid lines (flashing color is not supported). Select a color from the list of Predefined Color Names and Codes or create an RGB-based color using the function [MakeColour](#).

*FrameWidth*:

The width (also called pen width) of the frame enclosing the grid, in pixels. To define the frame without drawing its boundaries, set *FrameWidth* to 0 (zero) and *FrameCol* to 0. The maximum is 32.

*FrameCol*:

The color of the frame enclosing the grid (flashing color is not supported). Select a color from the list of Predefined Color Names and Codes or create an RGB-based color using the function [MakeColour](#).

*nFill*:

The background color for the frame (flashing color is not supported). Select a color from the list of Predefined Color Names and Codes or create an RGB-based color using the function [MakeColour](#).

*nMode*:

The mode of the plot. For future use only - set it to 0 (zero).

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[PlotClose](#), [PlotDraw](#), [PlotGetMarker](#), [PlotInfo](#), [PlotLine](#), [PlotMarker](#), [PlotOpen](#), [PlotScaleMarker](#), [PlotSetMarker](#),  
[PlotText](#), [PlotXYLine](#), [TrnPlot](#)

## Example

See [PlotOpen](#)

## See Also

[Plot Functions](#)

## PlotInfo

Gets information about the plot. You can call this function to determine the number of pixels per page or inch, the resolution of a plot, and the size and spacing of characters for a specified text font. You can also check whether a printer can print rotated text. (See [PlotText](#).)

you need to first call the PlotOpen() function to get the handle for the plot (*hPlot*) and specify the output device.

## Syntax

**PlotInfo**(*hPlot*, *Type* [, *sInput*] )

*hPlot*:

The plot handle, returned from the PlotOpen() function. The plot handle identifies the table where all data on the plot is stored.

*nType*:

The type of plot information to get:

- 0 - Horizontal pixels on a printout page
- 1 - Vertical pixels on a printout page
- 2 - Horizontal pixels per inch
- 3 - Vertical pixels per inch
- 4 - Horizontal resolution
- 5 - Vertical resolution
- 6 - Height of the font used
- 7 - External leading of the font used
- 8 - Character width of the font used
- 9 - Rotatable text is allowed or not

10 - Indicates whether or not a font is supported

11 - Horizontal size of a page in millimeters

12 - Vertical size of a page in millimeters

*sInput:*

The font handle (hFont), returned from the [DspFont\(\)](#) function. Useful only for Type 6, 7, 8, or 10.

## Return Value

The attributes of the plot as a string.

## Related Functions

[PlotClose](#), [PlotDraw](#), [PlotGrid](#), [PlotLine](#), [PlotMarker](#), [PlotOpen](#), [PlotScaleMarker](#), [PlotText](#), [PlotXYLine](#), [TrnPlot](#)

## Example

```
hPlot = PlotOpen(36,"Display",1);
:
/* Print text in upward direction but first check if printer
supports text rotation. Set default text orientation to left to
right (just in case). */
Orient = 0;
IF PlotInfo(hPlot,9) THEN
Orient = 1;
END
PlotText(hPlot,hFont,Orient,100,100,"scale");
..
/* Print text "Plant SCADA Graph" centred horizontally at top of page.*/
PageWidth = PlotInfo(hPlot,0); ! Get width of page
hFont = DspFont("Courier",14,black,-1);
TextWidth = PlotInfo(hPlot,8,hFont); ! Get width of each character
TextPosn = (PageWidth - TextWidth * 12) / 2 ! Get start of 1st character
PlotText(hPlot,hFont,0,TextPosn,0,"Plant SCADA Graph");
..
PlotClose(hPlot);
```

## See Also

[Plot Functions](#)

## PlotLine

Draws a line (in the Plant SCADA plot system) for a set of data points. You specify the data points in the table *pTable*, and plot these points between the *LoScale* and *HiScale* values. The line is drawn inside the frame defined by the [PlotGrid\(\)](#) function.

For each line on a plot, you can specify a different pen style, color, and width, and a different marker style and color. You can draw lines either from left to right or from right to left.

You need to first call the [PlotOpen\(\)](#) function to get the handle for the plot (*hPlot*) and specify the output device.

You should then use the PlotGrid() function to set up the frame and grid, before you call this function to plot the line.

## Syntax

**PlotLine(*hPlot*, *PenStyle*, *PenCol*, *PenWidth*, *MarkerStyle*, *MarkerCol*, *nMarker*, *Length*, *pTable*, *LoScale*, *HiScale*, *Mode*)**

*hPlot*:

The plot handle, returned from the PlotOpen() function. The plot handle identifies the table where all data on the plot is stored.

*PenStyle*:

The style of the pen used to draw:

0 - Solid

1 - Dash ( - - - - )

2 - Dot (.....)

3 - Dash and dot ( - . - . - . - )

4 - Dash, dot, dot ( - . - . - . - )

5 - Hollow

*PenCol*:

The color of the pen (flashing color is not supported). Select a color from the list of Predefined Color Names and Codes or create an RGB-based color using the function MakeColour.

*PenWidth*:

The width of the pen, in pixels. If the width is thicker than one pixel, you need to use a solid pen (*PenStyle* = 0). The maximum width is 32.

*MarkerStyle*:

The style of the markers:

0 - No markers

1 - Triangle

2 - Square

3 - Circle

4 - Diamond

5 - Filled triangle

6 - Filled square

7 - Filled circle

8 - Filled diamond

20 - 32000 - User-defined markers. You can register any symbol as a marker with the PlotSetMarker() function. Call the PlotGetMarker() function if the number of markers you have previously registered are unknown.

*MarkerCol*:

The color of the markers (flashing color is not supported). Select a color from the list of predefined color names and codes or create an RGB-based color using the function MakeColour.

*nMarker*:

The number of samples between markers.

*Length:*

The length of the array, that is, the number of points in the table pTable for PlotLine(), or in tables xTable and yTable for PlotXYLine().

For every line you draw with the PlotLine() and PlotXYLine() functions within a plot, you need to add the Length arguments for each call, and pass the total to the PlotGrid() function (in the nSamples argument).

*pTable:*

The points to be plotted (as an array of floating-point values).

*LoScale:*

The lowest value that will be displayed on the plot (that is the value assigned to the origin of your grid). The LoScale and HiScale values determine the scale of your grid. This scale is used to plot values. for example, If LoScale = 0 (zero) and HiScale = 100, a value of 50 will be plotted half way up the Y-axis of your grid. LoScale needs to be in the same units as the values in pTable.

*HiScale:*

The highest value that will be displayed on the plot. The LoScale and HiScale values determine the scale of your grid. This scale is used to plot values. for example, If LoScale = 0 (zero) and HiScale = 100, a value of 50 will be plotted half way up the Y- axis of your grid. HiScale needs to be in the same units as the values in pTable.

*Mode:*

The origin of your grid, and the direction of the plotted line:

- 1 - Origin is bottom-left, x is left to right, y is upwards
- 2 - Origin is bottom-right, x is right to left, y is upwards
- 4 - Origin is top-left, x is left to right, y is downwards
- 8 - Origin is top-right, x is right to left, y is downwards

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[PlotClose](#), [PlotDraw](#), [PlotGetMarker](#), [PlotGrid](#), [PlotInfo](#), [PlotMarker](#), [PlotOpen](#), [PlotScaleMarker](#), [PlotSetMarker](#),  
[PlotText](#), [PlotXYLine](#), [TrnPlot](#)

## Example

See [PlotOpen](#).

## See Also

[Plot Functions](#)

## PlotMarker

Draws markers on a plotted line or at a specified point. You can plot any one of the standard markers, or use a symbol of your choice. (you need to first register your symbol as a marker, by using the PlotSetMarker() function.)

To draw a single marker at a specified point, set *X* and *Y* to the coordinates of the point, and set *Length* to 1.

You can draw markers on a plotted line when you draw the line, that is within the PlotLine() or PlotXYLine() function. You would use the PlotMarker() function only if you need to draw a second set of markers on the same line. Call PlotMarker() immediately after the line is drawn. Set *X* and *Y* to -1 and *Length* to the number of data points (specified in the *Length* argument of the PlotLine() or PlotXYLine() function).

you need to first call the PlotOpen() function to get the handle for the plot (*hPlot*) and specify the output device.

## Syntax

**PlotMarker(*hPlot*, *MarkerStyle*, *MarkerCol*, *nMarker*, *Length*, *X*, *Y*)**

*hPlot*:

The plot handle, returned from the PlotOpen() function. The plot handle identifies the table where data on the plot is stored.

*MarkerStyle*:

The style of the markers:

0 - No markers

1 - Triangle

2 - Square

3 - Circle

4 - Diamond

5 - Filled triangle

6 - Filled square

7 - Filled circle

8 - Filled diamond

20 - 32000: User-defined markers. You can register any symbol as a marker with the PlotSetMarker() function. Call the PlotGetMarker() function if the number of markers you have previously registered are unknown.

*MarkerCol*:

The color of the marker (flashing color is not supported). Select a color from the list of Predefined Color Names and Codes or create an RGB-based color using the function [MakeColour](#).

*nMarker*:

The number of samples between markers.

*Length*:

The length of the array (the number of line points in the table pTable) plotted in the PlotLine() or PlotXYLine() function. To draw only one marker at a specified point, set Length to 1.

*X*, *Y*:

The x and y coordinates, in pixels, of the point where the marker is to be drawn. If the plot is for display on the

screen, the coordinates are relative to the AN specified in the PlotOpen() function. If the output device is a printer, the coordinates are relative to the point (0,0).

To draw the markers on a plotted line, set both X and Y to -1, and set Length to the same value as the Length passed in the PlotLine() or PlotXYLine() function.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[PlotClose](#), [PlotDraw](#), [PlotGetMarker](#), [PlotGrid](#), [PlotInfo](#), [PlotLine](#), [PlotOpen](#), [PlotScaleMarker](#), [PlotSetMarker](#), [PlotText](#), [PlotXYLine](#), [TrnPlot](#)

## Example

```
hPlot=PlotOpen(36,"Display",1);
..
/* Draw a filled red square marker at the point (X=100,Y=200). */
PlotMarker(hPlot,6,red,1,1,100,200);
..
/* Draw 10 black triangles and 5 green cylinders along a plot
line. */
PlotLine(hPlot,0,black,3,5,black,10,100,Buf2[0],0,100,2);
PlotSetMarker(20,"Global.Cylinder");
PlotMarker(hPlot,20,green,5,100,-1,-1)
..
PlotClose(hPlot);
```

## See Also

[Plot Functions](#)

## PlotOpen

Opens a new plot, sets its output device, and returns its plot handle. You can send the plot to any one of your system printers, or display it on screen at the specified AN.

you need to call this function before you can call the other plot functions.

## Syntax

**PlotOpen(*nAN*, *sOutput*, *Mode*)**

*nAN*:

The animation point (AN) where the plot will display. Set the AN to 0 (zero) when *sOutput* is a printer.

Do not use an animation point number at which a graphic object exists as this will prevent the PlotOpen() function from succeeding.

**sOutput:**

The output device where the plot is sent, for example:

"Display" - Display on screen. The plot is recorded in a metafile and displayed (at the specified AN) when the plot system is closed.

- "LPT1:" - Send to printer LPT1.
- "LPT2:" - Send to printer LPT2.
- "\ABC\Printers\Color1" - Send to UNC port (and so on for any output device)

**Mode:**

When a plot is removed or updated, the portion of the background screen beneath it is blanked out. The mode determines how the background screen is restored. The mode of the plot system:

1 - Normal mode

2 - Use for compatibility with the old graph functions

17 - Soft (valid for normal mode). The background screen (a rectangular region beneath the plot) is restored with the original image. Any objects that are within the rectangular region are destroyed when the background is restored.

33 - Hard (valid for normal mode). The background screen (a rectangular region beneath the plot) is painted with the color at the AN.

65 - Persistent (valid for normal mode). The plot is not erased. As the plot is updated, it is re-displayed on top. This mode provides fast updates. Transparent color is supported in this mode.

129 - Opaque animation (valid for normal mode). The plot is not erased. As the plot is updated, it is re-displayed on top. This mode provides the fastest updates. Transparent color is not supported in this mode.

257 - Overlapped animation (valid for normal mode). The background screen (the rectangular region beneath the plot) is completely repainted.

## Return Value

The plot handle if the plot is opened successfully, otherwise -1 is returned. The plot handle identifies the table where all data on the associated plot is stored.

## Related Functions

[PlotClose](#), [PlotDraw](#), [PlotGetMarker](#), [PlotGrid](#), [PlotInfo](#), [PlotLine](#), [PlotMarker](#), [PlotScaleMarker](#), [PlotSetMarker](#),  
[PlotText](#), [PlotXYLine](#), [TrnPlot](#)

## Example

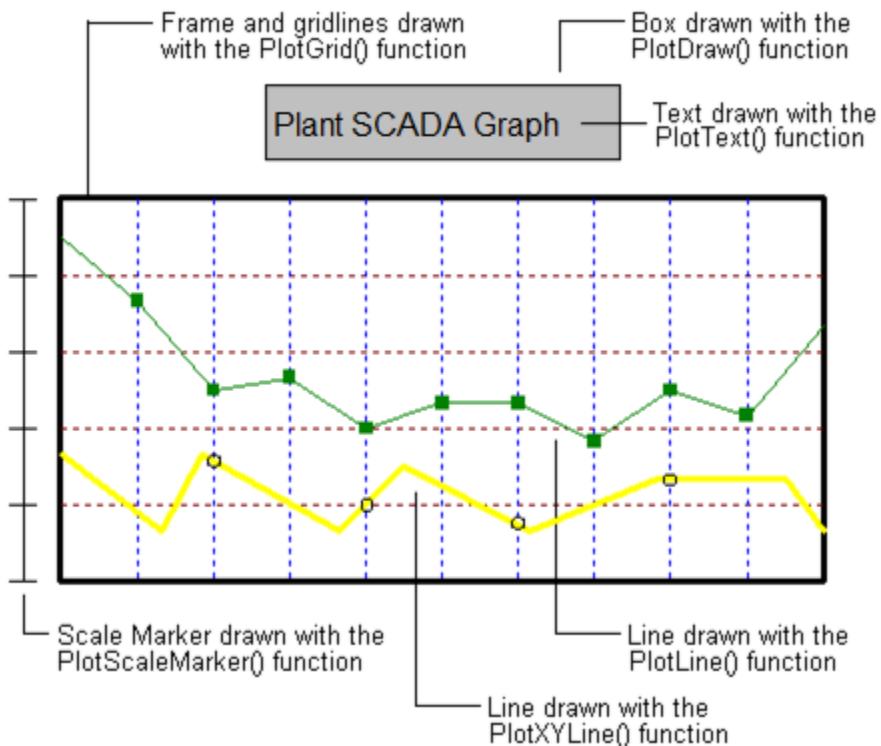
```
hPlot=PlotOpen(0,"LPT2:",1);
IF hPlot <> -1 THEN
    /* Set up a black frame with red & blue grid lines. */
    PlotGrid(hPlot,18,450,800,1850,1600,5,red,10,blue,4,black,white,0);
    /* Draw a scale line to the left of the frame. */
    PlotScaleMarker(hPlot,400,1600,6,1,black,0);
    /* Plot a simple line in green for a table of 10 values. */
    PlotLine(hPlot,0,green,3,6,green,2,10,Buf1,0,100,1);
    /* Plot a line in yellow (with black markers) for tables of 8 X and Y values. */
```

```

PlotXYLine(hPlot,0,yellow,4,3,black,2,8,Buf2,0,150,Buf3,0,100,1);
/* Draw a title box above the plot frame, with the heading "Plant SCADA Graph". */
PlotDraw(hPlot,1,0,black,1,grey,900,250,1400,400);
hFont = DspFont("Times",-60,black,grey);
PlotText(hPlot,hFont,0,950,350,"Plant SCADA Graph");
PlotClose(hPlot);
END

```

The above example prints the following (on the printer):



```

PlotOpen(0,"LPT1:",1) // opens a new plot to be sent to printer
PlotOpen(20,"DISPLAY",17) // normal plot with soft animation
PlotOpen(20,"DISPLAY",257) // normal plot with overlap animation
PlotOpen(20,"DISPLAY",1) // normal plot with overlap animation
(for default animation mode is overlap animation)
PlotOpen(20,"DISPLAY",16) // INVALID
(does not specify whether it is normal or Version 2.xx mode).
PlotOpen(20,"DISPLAY",2) // INVALID for Version 2.xx graph system
(does not support display as output).

```

## See Also

[Plot Functions](#)

### PlotScaleMarker

Draws scale lines beside the grid on your plot (if there is one) and places markers on them. The height of the scale line is automatically set to the height of the frame set in the PlotGrid() function.

You need to first call the PlotOpen() function to get the handle for the plot (*hPlot*) and specify the output device. You should then use the PlotGrid() function to set up the frame and grid, before you call this function to draw the

scale lines.

## Syntax

**PlotScaleMarker(*hPlot*, *X*, *Y*, *nMarker*, *PenWidth*, *PenCol*, *Mode*)**

*hPlot*:

The plot handle, returned from the PlotOpen() function. The plot handle identifies the table where data on the plot is stored.

*X*, *Y*:

The x and y coordinates of the point where the scale line starts. The end coordinates of the scale line are automatically defined by the size of the frame (set in the PlotGrid() function).

If the plot is for display on the screen, coordinates are relative to the AN specified in the PlotOpen() function. If the output device is a printer, coordinates are relative to the point (0,0).

*nMarker*:

The number of markers on the scale line.

*PenWidth*:

The width of the scale line, in pixels.

*PenCol*:

The color of the pen (flashing color is not supported). Select a color from the list of Predefined Color Names and Codes or create an RGB-based color using the function [MakeColour](#).

*Mode*:

The mode of the markers:

0 - Both sides of the scale line

1 - Left of the scale line

2 - Right of the scale line

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[PlotClose](#), [PlotDraw](#), [PlotGetMarker](#), [PlotGrid](#), [PlotInfo](#), [PlotLine](#), [PlotMarker](#), [PlotOpen](#), [PlotSetMarker](#), [PlotText](#), [PlotXYLine](#), [TrnPlot](#)

## Example

See [PlotOpen](#)

## See Also

[Plot Functions](#)

## PlotSetMarker

Registers a symbol as a marker. You can then draw the new marker at points and on plotted lines, by specifying the *MarkerNo* of the symbol as the *MarkerStyle* in the PlotMarker() function. Call the PlotGetMarker() function if you do not know the number of a marker.

## Syntax

**PlotSetMarker(*MarkerNo*, *sSymbolName*)**

*MarkerNo*:

The number of the marker, to be used as the MarkerStyle in the PlotMarker() function. Your marker numbers need to be greater than or equal to 20 (to a maximum of 32000).

*sSymbolName*:

The name and path of the symbol to be defined as a marker.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[PlotMarker](#), [PlotScaleMarker](#), [PlotGetMarker](#)

## Example

```
hPlot=PlotOpen(30,"Display",1);
..
/* Display red hourglass as marker at point (100,200). */
PlotSetMarker(20,"Global.Hourglass");
PlotMarker(hPlot,20,red,1,1,100,200);
..
PlotClose(hPlot);
```

## See Also

[Plot Functions](#)

## PlotText

Prints text on a plot. You can specify the font, position, and orientation of the text. If you specify an orientation other than 'left-to-right', you need to check that the font (and the printer) supports the orientation.

You need to first call the PlotOpen() function to get the handle for the plot (*hPlot*) and specify the output device. You also need to call the DspFont() function to get a handle for the font (*hFont*).

## Syntax

**PlotText(*hPlot*, *hFont*, *Orientation*, *X*, *Y*, *sText*)**

*hPlot*:

The plot handle, returned from the [PlotOpen\(\)](#) function. The plot handle identifies the table where all data on the plot is stored.

*hFont*:

The font handle, returned from the [DspFont\(\)](#) function. The font handle identifies the table where details of that font are stored.

*Orientation*:

The orientation of the text:

0 - Left-to-right

1 - Upwards

2 - Right-to-left

3 - Downwards

You should check that the font supports rotation (where Orientation = 1, 2, or 3). Most true type and vector fonts support rotation. If the [PlotInfo\(hPlot, 9\)](#) function returns false, you need to specify an Orientation of 0 (zero).

*X*, *Y*:

The x and y coordinates (in pixels) of the start of the text. If the plot is for display on the screen, the coordinates are relative to the AN specified in the [PlotOpen\(\)](#) function. If the output device is a printer, the coordinates are relative to the point (0,0).

*sText*:

The text string to be plotted.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[DspFont](#), [PlotClose](#), [PlotDraw](#), [PlotGrid](#), [PlotInfo](#), [PlotLine](#), [PlotMarker](#), [PlotScaleMarker](#), [PlotXYLine](#), [TrnPlot](#)

## Example

See [PlotOpen](#).

## See Also

[Plot Functions](#)

## PlotXYLine

Plots values from two different tables. Values from one table are considered X coordinates, and values from the other are considered Y coordinates. Points are plotted between the low and high scale values specified for x and y. The line is plotted inside the frame defined by the [PlotGrid\(\)](#) function.

For each line, you can specify a different pen style, color, and width, and a different marker style and color. You can draw lines either from left to right or from right to left. You need to first call the [PlotOpen\(\)](#) function to get the handle for the plot (*hPlot*) and specify the output device. You should then use the [PlotGrid\(\)](#) function to set up the frame and grid, before you call this function to plot the line.

## Syntax

**PlotXYLine(*hPlot*, *PenStyle*, *PenCol*, *PenWidth*, *MarkerStyle*, *MarkerCol*, *nMarker*, *Length*, *xTable*, *LoXScale*, *HiXScale*, *YTable*, *LoYScale*, *HiYScale*, *Mode*)**

*hPlot*:

The plot handle, returned from the [PlotOpen\(\)](#) function. The plot handle identifies the table where all data on the plot is stored.

*PenStyle*:

The style of the pen used to draw:

0 - Solid

1 - Dash ( - - - - )

2 - Dot (.....)

3 - Dash and dot ( - . . - . - . - )

4 - Dash, dot, dot ( - . . - - - - )

5 - Hollow

*PenCol*:

The color of the pen (flashing color is not supported). Select a color from the list of Predefined Color Names and Codes or create an RGB-based color using the function [MakeColour](#).

*PenWidth*:

The width of the pen, in pixels. If the width is thicker than one pixel, you need to use a solid pen (*PenStyle* = 0). The maximum width is 32.

*MarkerStyle*:

The style of the markers:

0 - No markers

1 - Triangle

2 - Square

3 - Circle

4 - Diamond

5 - Filled triangle

6 - Filled square

7 - Filled circle

8 - Filled diamond

20 - 32000 - User-defined markers. You can register any symbol as a marker with the PlotSetMarker() function. Call the PlotGetMarker() function to recall the number of a marker you have previously registered.

*MarkerCol:*

The color of the markers (flashing color is not supported). Select a color from the list of Predefined Color Names and Codes or create an RGB-based color using the function MakeColour.

*nMarker:*

The number of samples between markers.

*Length:*

The length of the array, that is the number of points in the table pTable for PlotLine(), or in tables *xTable* and *yTable* for PlotXYLine().

For every line you draw with the PlotLine() and PlotXYLine() functions within a plot, you need to add the Length arguments for each call, and pass the total to the PlotGrid() function (in the *nSamples* argument).

*xTable:*

The x coordinates for the points in the line, as an array of floating point values.

*LoXScale:*

The lowest X-axis value that will be displayed on the plot (that is the X-coordinate of the origin of your grid). The LoXScale and HiXScale values determine the scale of your grid. This scale is used to plot values. for example, If LoXScale = 0 (zero) and HiXScale = 100, a value of 50 will be plotted half way along the X-axis of your grid.

LoXScale needs to be in the same units as the values in *xTable*.

*HiXScale:*

The highest X-axis value that will be displayed on the plot. The LoXScale and HiXScale values determine the scale of your grid. This scale is used to plot values. for example, If LoXScale = 0 (zero) and HiXScale = 100, a value of 50 will be plotted half way along the X-axis of your grid.

HiXScale needs to be in the same units as the values in *xTable*.

*yTable:*

The y coordinates for the points in the line, as an array of floating point values.

*LoYScale:*

The lowest Y-axis value that will be displayed on the plot (that is the Y-coordinate of the origin of your grid). The LoYScale and HiYScale values determine the scale of your grid. This scale is used to plot values. for example, If LoYScale = 0 (zero) and HiYScale = 100, a value of 50 will be plotted half way up the Y-axis of your grid.

LoYScale needs to be in the same units as the values in *xTable*.

*HiYScale:*

The highest Y-axis value that will be displayed on the plot. The LoYScale and HiYScale values determine the scale of your grid. This scale is used to plot values. for example, If LoYScale = 0 (zero) and HiYScale = 100, a value of 50 will be plotted half way up the Y-axis of your grid.

HiYScale needs to be in the same units as the values in *xTable*.

*Mode:*

The origin of your grid, and the direction of the plotted line:

1 - Origin is bottom-left, x is left to right, y is upwards

2 - Origin is bottom-right, x is right to left, y is upwards

4 - Origin is top-left, x is left to right, y is downwards

8 - Origin is top-right, x is right to left, y is downwards

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[PlotClose](#), [PlotDraw](#), [PlotGrid](#), [PlotInfo](#), [PlotLine](#), [PlotMarker](#), [PlotScaleMarker](#), [PlotText](#), [TrnPlot](#)

## Example

See [PlotOpen](#)

## See Also

[Plot Functions](#)

## Process Analyst Functions

Following are functions relating to [Process Analyst](#):

<a href="#">ProcessAnalystLoadFile</a>	Loads the specified PAV file to a Process Analyst object.
<a href="#">ProcessAnalystPopup</a>	Displays a Process Analyst page (in a new page child window) at the current mouse position.
<a href="#">ProcessAnalystSelect</a>	Allows a set of pens to be selected before displaying the PA page.
<a href="#">ProcessAnalystSetPen</a>	Allows a new pen to be added to a PA display.
<a href="#">ProcessAnalystWin</a>	Displays a Process Analyst page (in a new window) preloaded with the pre-defined Process Analyst View (PAV) file.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## ProcessAnalystLoadFile

Loads the specified PAV file to a Process Analyst object, which is identified by parameter *sObjName*.

## Syntax

**ProcessAnalystLoadFile(*sPAVFile*, *iFileLocation*, *iButtonMask*, *sObjName* )**

*sPAVFile*:

Name of the PAV file

*iFileLocation*:

PAV file location code for the PAV file. Indicates which known location to load the file from.

Member Name	Description	Value
FileLocation_Local	Refers to the project folder	0
FileLocation_Server	Refers to the both the primary/ standby server paths	1
FileLocation_User	Refers to the My Documents folder	2

*iButtonMask*:

Bit mask for removing command buttons from the PA, bit flags as shown below:

1 - Load View

2 - Save View

4 - Print

8 - Copy to Clipboard

16 - Copy to File

32 - Add Pens

64 - Remove Pens

128 - Show Properties

256 - Help

*sObjName*:

Name of the PA object on the given Page where the PAV file will be loaded.

## Return Value

Zero (0) if the function is successfully run. otherwise an error code is returned.

## Related Functions

[PageProcessAnalyst](#), [PageProcessAnalystPens](#), [ProcessAnalystPopup](#), [ProcessAnalystSelect](#),  
[ProcessAnalystSetPen](#), [ProcessAnalystWin](#), [TrnSetPen](#), [WinNewAt](#)

## See Also

[Process Analyst Functions](#)

## ProcessAnalystPopup

Displays a Process Analyst page (in a new page child window) at the current mouse position preloaded with the pre-defined Process Analyst View (PAV) file.

## Syntax

**ProcessAnalystPopup(*sPage* [, *sPAVFile* [, *iFileLocation* [, *iButtonMask* [, *sObjName* [, *iMode* ]]]])**

*sPage*:

The name of the page that contains Process Analyst object(s). For example, pages based on the Process Analyst templates found in the Tab\_Style\_Include project.

*sPAVFile*:

Name of the PAV file

*iFileLocation*:

PAV file location code for the PAV file, see PA doc LoadFromFile() for details.

*iButtonMask*:

Bit mask for removing command buttons from the PA, bit flags as shown below:

1 - Load View

2 - Save View

4 - Print

8 - Copy to Clipboard

16 - Copy to File

32 - Add Pens

64 - Remove Pens

128 - Show Properties

256 - Help

*sObjName*:

Name of the PA object on the given Page where the PAV file will be loaded. If this parameter is not specified or empty string, it is defaulted to the object name used in the tab style templates, that is "\_templatePA1".

*iMode*:

The mode of the window (see WinNewAt() for details).

## Return Value

Window number if the window is successfully displayed. Otherwise -1 is returned.

## Related Functions

[PageProcessAnalyst](#), [PageProcessAnalystPens](#), [ProcessAnalystLoadFile](#), [ProcessAnalystSelect](#),  
[ProcessAnalystSetPen](#), [ProcessAnalystWin](#), [TrnSetPen](#), [WinNewAt](#)

## See Also

[Process Analyst Functions](#)

### ProcessAnalystSelect

Works like the existing Cicode Function `TrnSelect()`. It allows a set of pens to be selected before displaying the PA page. When `PageProcessAnalystPens()` is called after `ProcessAnalystSelect()`, the pens specified by both functions will be available in the final PA display. You can also repeat the call sequence of `ProcessAnalystSelect()` and `ProcessAnalystSetPen()` multiple times to set up multiple PA objects for the same page before displaying the page.

## Syntax

`ProcessAnalystSelect((iWindow, sPage [, sObjName [, sClusterName [, iButtonMask [, sPAVFile [, iFileLocation]]]]]))`

*iWindow*:

The window number (returned from the `WinNumber()` function):

-3 - for the current window

-2 - For the next window displayed

*sPage*:

The name of the page that displays the PA.

*sObjName*:

The name of the PA object. If this is not specified, it is defaulted to "`_TemplatePA1`" which is the name used by the built-in templates.

*sClusterName*:

The name of the cluster that is associated with any trend tag for this PA. This is optional if you have one cluster or are resolving the trend via the current cluster context. The argument is enclosed in quotation marks "".

*iButtonMask*:

Bit mask for removing command buttons from the PA, bit flags as shown below:

1 - Load View

2 - Save View

4 - Print

8 - Copy to Clipboard

16 - Copy to File

32 - Add Pens

64 - Remove Pens

128 - Show Properties

256 - Help

*sPAVFile*:

Name of the PAV file

*iFileLocation:*

PAV file location code for the PAV file, see PA doc LoadFromFile() for details.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[PageProcessAnalyst](#), [PageProcessAnalystPens](#), [ProcessAnalystLoadFile](#), [ProcessAnalystPopup](#),  
[ProcessAnalystSetPen](#), [ProcessAnalystWin](#), [TrnSetPen](#), [WinNewAt](#)

## See Also

[Process Analyst Functions](#)

### ProcessAnalystSetPen

Works like the existing function TrnSetPen(). Allows a new pen to be added to a PA display. The pane defaults to the first pane of the PA if it is not specified.

## Syntax

**ProcessAnalystSetPen((*iPen*, *sTag* [, *sObjName* [, *iPane* [, *iPenType*]]) )**

*iPen:*

Pen number. The allowed values are:

- <0 - new pen
- 0 - the currently selected pen
- existing pen number - change existing pen
- >existing pen number - new pen

Up to 8 pens can be added to the PA using the Cicode function if ObjName is set to "-2".

Be reminded that unlike trend objects, the pen numbers in Process Analyst are not fixed. They are dynamically reassigned when pens are added or deleted. When setting pens to the Process Analyst on the current display, pens are numbered within the scope of the pane they are in. On the other hand, when setting pens for the next display, pens are numbered in a flat scope regardless of pane number specified.

*sTag:*

The trend tag name to be assigned to the pen.

*sObjName:*

The name of the PA object. If this is set to "-2", the pen is set to the next displayed PA page set up by ProcessAnalystSelect(). If the specified ObjName is valid, the changes will be applied to the currently displayed PA. Otherwise, the function will try to set the pen to the specified object on the currently displayed page. If this parameter is not specified or is an empty string, it will default to the object name used in the tab style templates, that is "\_templatePA1".

*iPane*:

Optional number of the pane where the trend or variable tags are added. Please see the same parameter for function PageProcessAnalystPens() for details. Defaulted to 0, that is, the first pane.

*iPenType*:

Pen type for creation. The allowed values are:

0 - Analog Trend (Default)

1 - Digital Trend

2 - Alarm

---

**Note:** This optional parameter is ignored for existing pens. To change pen type, delete a pen and create a new one.

---

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[PageProcessAnalyst](#), [PageProcessAnalystPens](#), [ProcessAnalystLoadFile](#), [ProcessAnalystPopup](#),  
[ProcessAnalystSelect](#), [ProcessAnalystWin](#), [TrnSetPen](#), [WinNewAt](#)

## See Also

[Process Analyst Functions](#)

## ProcessAnalystWin

Displays a Process Analyst page (in a new window) preloaded with the pre-defined Process Analyst View (PAV) file.

## Syntax

**ProcessAnalystWin**(*sPage*, *iX*, *iY*, *iMode* [, *sPAVFile* [, *iFileLocation* [, *iButtonMask* [, *sObjName* ]]]])

*sPage*:

The name of the page that contains Process Analyst object(s). For example, pages based on the Process Analyst templates found in the Tab\_Style\_Include project.

*iX*:

The X pixel coordinate

*iY*:

The Y pixel coordinate

*iMode*:

The mode of the window (see [WinNewAt\(\)](#) for details).

*sPAVFile*:

Name of the PAV file

*iFileLocation*:

PAV file location code for the PAV file, see PA doc [LoadFromFile](#) for details.

*iButtonMask*:

Bit mask for removing command buttons from the PA, bit flags as shown below:

1 - Load View

2 - Save View

4 - Print

8 - Copy to Clipboard

16 - Copy to File

32 - Add Pens

64 - Remove Pens

128 - Show Properties

256 - Help

*sObjName*:

Name of the PA object on the given Page where the PAV file will be loaded. If this parameter is not specified or empty string, it is defaulted to the object name used in the tab style templates, that is "\_templatePA1".

## Return Value

Window number if the window is successfully displayed. Otherwise -1 is returned.

## Related Functions

[PageProcessAnalyst](#), [PageProcessAnalystPens](#), [ProcessAnalystLoadFile](#), [ProcessAnalystPopup](#),  
[ProcessAnalystSelect](#), [ProcessAnalystWin](#), [TrnSetPen](#), [WinNewAt](#)

## See Also

[Process Analyst Functions](#)

## Quality Functions

The following functions are used to interface with the QUALITY data type.

<a href="#">QualityCreate</a>	Creates a quality value based on the quality fields provided.
<a href="#">QualityGetPart</a>	Extracts a requested part of the Quality value from the QUALITY variable.
<a href="#">QualityIsBad</a>	Returns a value indicating whether the quality is bad.

<a href="#">QualityIsGood</a>	Returns a value indicating whether the quality is good.
<a href="#">QualityIsUncertain</a>	Returns a value indicating whether the quality is uncertain.
<a href="#">QualitySetPart</a>	Sets a Quality part's value to the QUALITY variable.
<a href="#">QualityIsOverride</a>	Returns a value indicating whether the tag is in Override Mode.
<a href="#">QualityIsControlInhibit</a>	Returns a value indicating whether the tag is in Control inhibit mode.
<a href="#">QualityToStr</a>	Returns a textual representation of the Plant SCADA quality.
<a href="#">VariableQuality</a>	Extracts the quality from a given variable.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## QualityCreate

Creates a quality value based on the quality fields provided. When the value of a particular quality field is out of range, the value of its corresponding part in the returned quality remains at 0, and hardware error is generated.

## Syntax

**QualityCreate(INT *generalQuality* [, INT *qualitySubstatus* [, INT *qualityLimit* [, INT *extendedSubstatus* [, INT *bOverride* [, *bControlInhibit* [, INT *datasourceErrorCode* ]]]]]])**

*generalQuality:*

Specifies the general quality.

*qualitySubstatus:*

Specifies the quality substatus.

*qualityLimit:*

The Quality Limit.

*extendedSubstatus:*

Specifies the extended quality substatus.

*bOverride:*

Specifies the Tag Status Override Flag.

*bControlInhibit:*

Specifies the Tag Status Control Inhibit Flag.

***datasourceErrorCode:***

Specifies the data source error code.

For further information on the quality arguments listed above, refer to the [Tag Extensions](#) documentation in the main help.

## Return Value

The Quality value of the element.

## Related Functions

[QualityGetPart](#), [QualityIsBad](#), [QualityIsControlInhibit](#), [QualityIsGood](#), [QualityIsOverride](#), [QualityIsUncertain](#),  
[QualitySetPart](#), [QualityToStr](#)

## Example

```
QUALITY q1;  
q1 = QualityCreate(QUAL_BAD, QUAL_BAD_NON_SPECIFIC, QUAL_LIMITED_HIGH,  
QUAL_EXT_NOT_REPLICATED);
```

## See Also

[Quality Functions](#)

## QualityGetPart

Extracts a requested part of the quality value from a variable tag's quality item.

## Syntax

**QualityGetPart(QUALITY *Quality*, INT *Part*)**

*Quality*:

Specifies the quality variable.

*Part*:

The part to extract:

0 – The General Quality value

1 – Quality Substatus value

2 - The Quality Limit value

3 - The Extended Quality Substatus value

4 – The Tag Status Override flag

5 – The Tag Status Control Inhibit flag

6 - The DataSource error code

7 – The OPC Quality (General + Substatus + Limit)

## Return Value

The value of the requested Quality part (see tables below), or -1 if error.

## Part 0 - General Quality Values

Cicode label name	Value	Description
QUAL_BAD	0x00	Value is not useful for reasons indicated by the Substatus Bit Field.
QUAL_UNCR	0x01	The Quality of the value is uncertain for reasons indicated by the Substatus Bit Field.
QUAL_GOOD	0x03	The Quality of the value is Good.

## Part 1 - Quality Substatus Values

This part returns a value from one of the following substatus groups:

- QUAL\_BAD
- QUAL\_UNCR (uncertain)
- QUAL\_GOOD

To determine which substatus group the return value is from, you need to initially call the General Quality Value (*part* = 0).

- **Quality Substatus Values - QUAL\_BAD**

Cicode label name	Substatus Value	Quality Value	Description
QUAL_BAD_NON_SPECIFIC	0x00	0x00000000	The value is bad but no specific reason is known.
QUAL_BAD_CONFIGURATION_ERROR	0x01	0x00000004	There is some server-specific incorrect configuration. For example, the item in question has been deleted from the configuration.
QUAL_BAD_NOT_CONNECTED	0x02	0x00000008	The input is necessary to be logically connected to something but is not.

Cicode label name	Substatus Value	Quality Value	Description
			This quality may reflect the fact that no value is available at this time, for reasons like the value may not have been provided by the data source.
QUAL_BAD_DEVICE_FAILURE	0x03	0x0000000c	An inoperative device has been detected.
QUAL_BAD_SENSOR_FAILURE	0x04	0x00000010	An inoperative sensor has been detected (the 'Limits' field can provide additional diagnostic information in some situations).
QUAL_BAD_LAST_KNOWN_VALUE	0x05	0x00000014	Communication has been lost, however, the last known value is available. The age of the value may be determined from the TIMESTAMP in the OPCITEMSTATE.
QUAL_BAD_COMM_FAILURE	0x06	0x00000018	Communication has been lost. There is no last known value available.
QUAL_BAD_OUT_OF_SERVICE	0x07	0x0000001C	The block is off scan or otherwise locked. This quality is also used when the active state of the item or the group containing the item is InActive.
QUAL_BAD_WAITING_FOR_INI_DATA	0x08	0x00000020	After items are added to a group, it may take some time for the server to actually obtain values for these items. In such cases, the client might perform a read (from cache), or establish a

Cicode label name	Substatus Value	Quality Value	Description
			ConnectionPoint based subscription and/or execute a refresh on such a subscription before the values are available. This substatus is only available from OPC DA 3.0 or newer servers.

- Quality Substatus Values - QUAL\_UNCR

Cicode label name	Substatus Value	Quality Value	Description
QUAL_UNCR_NON_SPECIFIC	0x00	0x00000040	The value is uncertain but no specific reason is known.
QUAL_UNCR_LAST_USABLE_VALUE	0x01	0x00000044	Whatever was writing this value has stopped doing so. Regard the returned value as 'stale'. This differs from a BAD value with Substatus 5 (0x14) (Last Known Value). This status is associated specifically with a detectable communication error on a 'fetched' value and indicates the failure of some external source to 'put' something into the value within an acceptable period of time. The age of the value can be determined from the TIMESTAMP in OPCITEMSTATE.
QUAL_UNCR_SENSOR_NOT_ACCURATE	0x04	0x00000050	Either the value has 'pegged' at one of the sensor limits (in which case, set the limit field to 1 or 2) or the sensor is otherwise known to be out of calibration via

Cicode label name	Substatus Value	Quality Value	Description
			some form of internal diagnostics (in which case, set the limit field to 0).
QUAL_UNCR_ENG_UNIT_EXCEEDED	0x05	0x00000054	The returned value is outside the limits defined for this parameter. In this case the Limits field indicates which limit has been exceeded but does NOT necessarily imply that the value cannot move farther out of range.
QUAL_UNCR_SUBNORM_AL	0x06	0x00000058	The value is derived from multiple sources and has less than the necessary number of Good sources.

- Quality Substatus Values - QUAL\_GOOD

Cicode label name	Substatus Value	Quality Value	Description
QUAL_GOOD_NON_SPECIFIC	0x00	0x000000c0	The value is good. There are no special conditions.
QUAL_GOOD_LOCAL_OVERRIDE	0x06	0x000000d8	The value has been Overridden. Typically this means the input has been disconnected and a manually entered value has been "forced".

## Part 2 - Quality Limit Values

Cicode label name	Value	Description
QUAL_LIMITED_NOT_LIMITED	0x0	The value is free to move up and down.
QUAL_LIMITED_LOW	0x1	The value has 'pegged' at some lower limit.
QUAL_LIMITED_HIGH	0x2	The value has 'pegged' at some

Cicode label name	Value	Description
		high limit.
QUAL_LIMITED_CONSTANT	0x3	The value is a constant and cannot move.

### Part 3 - Extended Quality Substatus Values

Cicode label name	Value	Description
QUAL_EXT_NON_SPECIFIC	0x00	There is no specific extended substatus value.
QUAL_EXT_SCHEDULED_OFFLINE	0x01	The device is a scheduled device that is offline and no cache value is available.
QUAL_EXT_INVALID_TAG	0x02	The tag configuration is invalid.
QUAL_EXT_INVALID_DATA	0x03	The value of the tag is invalid.
QUAL_EXT_SOFTWARE_ERROR	0x04	An internal software error occurred in the device driver.
QUAL_EXT_TOO_MANY_DEVICES	0x05	Too many devices are attached.
QUAL_EXT_COMM_NO_INIT	0x06	Communication is not initialised.
QUAL_EXT_COMM_BAD	0x07	Bad communication.
QUAL_EXT_TAG_OUT_OF_RANGE	0x08	Tag address is out of range.
QUAL_EXT_WRITE_ONLY	0x09	Tag is not readable.
QUAL_EXT_WRITE_PROTECTED	0x0A	Write operation is not authorised.
QUAL_EXT_NO_CLUSTER_SPECIFIED	0x0B	No cluster is specified within a system or for a given tag.
QUAL_EXT_CLUSTER_NOT_FOUND	0x0C	The requested cluster is not known or no clusters are available.

Cicode label name	Value	Description
QUAL_EXT_CLUSTER_DISABLED	0x0D	The requested cluster is disabled.
QUAL_EXT_SESSION_NOT_CONNECTED	0x0E	Cannot connect to the requested session.
QUAL_EXT_TAG_RESOLVE_TIMEOUT	0x0F	Tag could not be resolved.
QUAL_EXT_VALUE_OUT_OF_RANGE	0x10	Tag value is out of range.
QUAL_EXT_COMM_DEV_BAD	0x11	Communication loss - data source to PLC.
QUAL_EXT_COMM_IOSERVER_BAD	0x12	Communication loss - client to data source.
QUAL_EXT_STALE	0x13	Tag value is "stale".
QUAL_EXT_NOT_REPLICATED	0x14	Tag element value not replicated to every redundant data source.

#### Part 4 - Tag Status Values - Override Flag

Cicode label name	Value	Description
QTS_OVERRIDE	0x01	The tag is in Override mode.

#### Part 5 - Tag Status Values - Control Inhibit Flag

Cicode label name	Value	Description
QTS_CONTROL_INHIBIT	0x02	The tag is in Control Inhibit Mode.

#### Related Functions

[QualityIsBad](#), [QualityIsControlInhibit](#), [QualityIsGood](#), [QualityIsOverride](#), [QualityIsUncertain](#), [QualitySetPart](#), [QualityToStr](#), [QualityCreate](#)

## Example

```
INT qualityGeneral;  
qualityGeneral = QualityGetPart(Tag1.Field.Q, 0);
```

## See Also

[Quality Functions](#)

### QualityIsControlInhibit

Returns a value indicating whether the tag is in Control Inhibit Mode.

## Syntax

**QualityIsControlInhibit(QUALITY *quality*)**

*quality*:

Specifies the QUALITY variable.

## Return Value

0: the tag is not in Control inhibit Mode.

1: the tag is in Control inhibit Mode.

## Related Functions

[QualityGetPart](#), [QualityIsBad](#), [QualityIsGood](#), [QualityIsOverride](#), [QualityIsUncertain](#), [QualitySetPart](#),  
[QualityToStr](#), [QualityCreate](#)

## Example

```
INT controlInhibitEnabled;  
controlInhibitEnabled = QualityIsControlInhibit(Tag1.Field.Q);
```

## See Also

[Quality Functions](#)

### QualityIsBad

This function will return a value indicating whether the general part of quality is bad.

## Syntax

**QualityIsBad(QUALITY *quality*)**

*quality*:

Specifies the QUALITY variable.

## Return Value

0: the quality is not bad.

1: the quality is bad.

## Related Functions

[QualityGetPart](#), [QualityIsControllInhibit](#), [QualityIsGood](#), [QualityIsOverride](#), [QualityIsUncertain](#), [QualitySetPart](#),  
[QualityToStr](#), [QualityCreate](#)

## Example

```
INT bad;  
bad = QualityIsBad(Tag1.Field.Q);
```

## See Also

[Quality Functions](#)

## QualityIsGood

This function will return a value indicating whether the general part of quality is good.

## Syntax

**QualityIsGood(QUALITY *quality*)**

*quality*:

Specifies the QUALITY variable.

## Return Value

0: the quality is not good.

1: the quality is good.

## Related Functions

[QualityGetPart](#), [QualityIsBad](#), [QualityIsControllInhibit](#), [QualityIsOverride](#), [QualityIsUncertain](#), [QualitySetPart](#),  
[QualityToStr](#), [QualityCreate](#)

## Example

```
INT good;
good = QualityIsGood(Tag1.Field.Q);
```

## See Also

[Quality Functions](#)

### QualityIsOverride

Returns a value indicating whether the tag is in Override Mode.

## Syntax

**QualityIsOverride(QUALITY *quality*)**

*quality*:

Specifies the QUALITY variable.

## Return Value

0: the tag is not in Override Mode.

1: the tag is in Override Mode.

## Related Functions

[QualityGetPart](#), [QualityIsBad](#), [QualityIsControlInhibit](#), [QualityIsGood](#), [QualityIsUncertain](#), [QualitySetPart](#),  
[QualityToStr](#), [QualityCreate](#)

## Example

```
INT overrideEnabled;
overrideEnabled = QualityIsOverride(Tag1.Q);
```

## See Also

[Quality Functions](#)

### QualityIsUncertain

This function will return a value indicating whether the general part of quality is uncertain.

## Syntax

**QualityIsUncertain(QUALITY *quality*)**

*quality:*

Specifies the QUALITY variable.

## Return Value

0: the quality is not uncertain.

1: the quality is uncertain.

## Related Functions

[QualityGetPart](#), [QualityIsBad](#), [QualityIsControlInhibit](#), [QualityIsGood](#), [QualityIsOverride](#), [QualitySetPart](#),  
[QualityToStr](#), [QualityCreate](#)

## Example

```
INT uncertain;
uncertain = QualityIsUncertain(Tag1.Field.Q);
```

## See Also

[Quality Functions](#)

## QualitySetPart

Sets a Quality part's value to the QUALITY variable. This function can only be used to manipulate the quality of Cicode variables. The quality item of a variable tag element (for example, 'Tag1.Field.Q') can only be used by QualityGetPart() as it is read only.

## Syntax

**QualitySetPart**(QUALITY *quality*, INT *part*, INT *value*)

*quality:*

Specifies the quality variable.

*part:*

The part to extract:

0 – The General Quality value

1 – Quality Substatus value

2 – The Quality Limit value

3 – The Extended Quality Substatus value

4 – The Tag Status Override flag

5 – The Tag Status Control Inhibit flag

6 – The DataSource error code

7 – The OPC Quality (General + Substatus + Limit)

*value:*

The new value for the given part.

## Return Value

The modified Quality value, or the original value if the given part is not applicable.

## Related Functions

[QualityGetPart](#), [QualityIsBad](#), [QualityIsControlInhibit](#), [QualityIsGood](#), [QualityIsOverride](#), [QualityIsUncertain](#), [QualityToStr](#), [QualityCreate](#)

## Example

```
QUALITY q;  
INT qualityGeneral;  
// insert code here  
q = QualitySetPart(q, 0, qualityGeneral);
```

## See Also

[Quality Functions](#)

## QualityToStr

Returns a textual representation of the quality.

## Syntax

**QualityToStr(QUALITY *quality*, INT *part*, INT *localized*)**

*quality:*

Specifies the QUALITY variable.

*part:*

Specifies the part of quality to obtain the textual representation.

-2: Short representation in the format <General Quality> [- <Quality Substatus>]

-1: Full representation in the format <General Quality> [Override] [Control Inhibit] – <Quality Substatus>

0: <General Quality>

1: <Quality Substatus>

2: <Quality Limit>

3: <Extended Quality Substatus>

4: <Quality Override>

5: <Control Inhibit>

*localized:*

The flag indicating if the returned text should be in native language or in Runtime localized language.

## Return Value

A textual representation of the quality, or an empty string if the part given is not applicable.

## Related Functions

[QualityGetPart](#), [QualityIsBad](#), [QualityIsControlInhibit](#), [QualityIsGood](#), [QualityIsOverride](#), [QualityIsUncertain](#), [QualitySetPart](#), [QualityCreate](#)

## Example

```
QUALITY q;;
STRING str;
q = QualityCreate(QUAL_GOOD, 0, QUAL_LIMITED_NOT_LIMITED,
QUAL_EXT_NON_SPECIFIC, 1, 1, 0);
str = QualityToStr(q, -1, 0);
// The result is: Good [Override] [Control Inhibit]

q = QualityCreate(QUAL_GOOD, QUAL_GOOD_LOCAL_OVERRIDE,
QUAL_LIMITED_NOT_LIMITED, QUAL_EXT_NON_SPECIFIC);
str = QualityToStr(q, 1, 0);
// The result is: Overridden

q = QualityCreate(QUAL_GOOD, 0, QUAL_LIMITED_QL_LOW_LIMITED,
QUAL_EXT_NON_SPECIFIC);
str = QualityToStr(q, 2, 0);
// The result is: Below Low

Login("UserName", "Password", 0, "French(France)");

q = QualityCreate(QUAL_GOOD, 0, QUAL_LIMITED_NOT_LIMITED,
QUAL_EXT_NON_SPECIFIC, 1, 1, 0);
str = QualityToStr(q, -1, 1);
// The result is: Bon [Supplémenté] [Contrôle Inhibé]
// Entries for 'Bon', 'Supplémenté' and 'Contrôle Inhibé'
// needs to have been provided in FRENCH.dbf

Login("UserName", "Password", 0, "English");

q = QualityCreate(QUAL_BAD, QUAL_BAD_CONFIGURATION_ERROR,
QUAL_LIMITED_NOT_LIMITED, QUAL_EXT_NON_SPECIFIC);
str = QualityToStr(q, -1, 0);
// The result is: Bad - Configuration Error

Login("UserName", "Password", 0, "French(France)");

q = QualityCreate(QUAL_BAD, QUAL_BAD_CONFIGURATION_ERROR,
QUAL_LIMITED_NOT_LIMITED, QUAL_EXT_NON_SPECIFIC);
str = QualityToStr(q, -1, 1);
// The result is: Mauvais - Erreur de Configuration
```

```
>Login("UserName", "Password", 0, "English");

q = QualityCreate(QUAL_UNCR, QUAL_UNCR_NON_SPECIFIC,
QUAL_LIMITED_NOT_LIMITED, QUAL_EXT_TAG_OUT_OF_RANGE);
str = QualityToStr(q, 0, 0);
// The result is: Uncertain

q = QualityCreate(QUAL_UNCR, QUAL_UNCR_NON_SPECIFIC,
QUAL_LIMITED_NOT_LIMITED, QUAL_EXT_TAG_OUT_OF_RANGE);
str = QualityToStr(q, 3, 0);
// The result is: Tag Address Out Of Range

q = QualityCreate(QUAL_UNCR, QUAL_UNCR_SUBNORMAL,
QUAL_LIMITED_NOT_LIMITED, QUAL_EXT_NON_SPECIFIC);
str = QualityToStr(q, 1, 0);
// The result is: Subnormal
```

## See Also

[Quality Functions](#)

## VariableQuality

Extracts the quality from a given variable.

**Note:** This function is designed to be used within Cicode; using it on graphical pages may result in displaying an error message instead of an expected quality message when either its argument has not good quality or an execution error is set.

## Syntax

**VariableQuality(*Variable*)**

**Variable:**

The variable from which the quality will be extracted.

## Return Value

The QUALITY of the given variable. If Variable is NULL, it returns quality uncertain (0x40).

Timestamps of uninitialized stack variables, uninitialized code variables and constants are equal to 0 - invalid timestamp, while their qualities are GOOD

## Related Functions

[QualityCreate](#), [QualityGetPart](#), [QualityIsGood](#), [QualitySetPart](#), [QualityIsOverride](#), [QualityIsControlInhibit](#), [QualityToStr](#)

## Example

```
INT codeVariable = 1;
INT
FUNCTION
MyFunction(REAL arg1)
STRING str = "My string";
QUALITY q;
q = VariableQuality(codeVariable); //code variable
q = VariableQuality(arg1); //function argument
q = VariableQuality(str); //stack variable
q = VariableQuality(Tag1); //any tag/local variable

RETURN 1;
END
```

## See Also

[Quality Functions](#)

## Report Functions

Following are functions relating to Reports:

<a href="#">RepGetCluster</a>	Retrieves the name of the cluster the report is running on.
<a href="#">RepGetControl</a>	Gets report control information.
<a href="#">Report</a>	Runs a report.
<a href="#">RepSetControl</a>	Sets report control information.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

### RepGetCluster

This function retrieves the name of the cluster a report is running on. This function should only be called from a report file.

## Syntax

**RepGetCluster()**

## Return Value

The name of the cluster the report is running on.

## See Also

[Report Functions](#)

## RepGetControl

Gets report control information on a report. This function is a blocking function. It will block the calling Cicode task until the operation is complete.

## Syntax

**RepGetControl(ReportName, Type [, sClusterName])**

*ReportName*:

The name of the report (can be prefixed by the name of the cluster that is ClusterName.ReportName).

*nType*:

The type of report control information to get (send back in the return value):

0 - State of the report - returns one of:

- 0 -Idle
- 1 - Waiting for PLC data for trigger
- 2 - Waiting for PLC data
- 3 - Running

1 - Time of day that the report is due to run next.

2 - The report period, in seconds, or week day, month or year, for example, if the report is weekly, this is the day of the week, 0 (Sunday) to 6 (Saturday).

3 - Synchronisation time of day of the report, for example, 10:00:00 (In seconds from midnight).

4 - Type of report schedule - returns one of:

- 0 - Event triggered
- 1 - Daily
- 2 - Weekly
- 3 - Monthly
- 4 - Yearly

5 - Report state - returns one of:

- 0 - Enabled

- 1 - Disabled

*sClusterName:*

Name of the cluster in which the report resides. This is optional if you have one cluster or are resolving the report server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

The control information, as an integer.

## Related Functions

[RepSetControl](#), [Report](#)

## Example

```
Next=RepGetControl("SHIFT",1,"ClusterXYZ");
! Sets Next to the time that the report is due to run.
! Display a message at the prompt AN (AN2) if
! the report is running.
IF RepGetControl("SHIFT",0,"ClusterXYZ")=3 THEN
    Prompt("Shift report is running");
END
```

## See Also

[Report Functions](#)

## Report

Runs a report on the Report Server. This function only schedules the report for execution. The running of the report is controlled entirely by the Report Server.

This function will start the specified report on the Reports Server to which the Plant SCADA computer is communicating. If you are using the Reports Servers in Primary/Standy mode, the report can run on the Standby Server. If you call this function on the Standby Server then the report will definitely run on the Standby Server, even if the Primary Server is active.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

## Syntax

**Report(ReportName [, sClusterName] )**

*ReportName:*

The name of the report to run (can be prefixed by the name of the cluster that is ClusterName.ReportName).

*sClusterName:*

Name of the cluster in which the report resides. This is optional if you have one cluster or are resolving the report server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[RepSetControl](#), [RepGetControl](#)

## Example

### Buttons

Text: Shift Report

Command: Report("Shift", "ClusterXYZ")

Comment: Runs the Shift Report

### System Keyboard

Key Sequence: Report ##### Enter

Command: Report(Arg1)

Comment: Runs a specified Report

```
Report("SHIFT", "ClusterXYZ");
! Runs the report named "SHIFT".
Report("DAY", "ClusterXYZ");
! Runs the report named "DAY".
/* The "SHIFT" and "DAY" reports are started. The order in which
the reports are run cannot be determined. If you want the "DAY"
report to run after the "SHIFT" report, call Report("DAY") at the
end of the "SHIFT" report. */
```

## See Also

[Report Functions](#)

## RepSetControl

Sets report control information to temporarily override the normal settings for a specified report. You can change the report schedule for a periodic report, and run one-time or event-triggered reports. These new settings are set on both the primary and standby report servers, but are not saved to the database. When you restart your system, Plant SCADA uses the existing settings, defined in the Reports database.

You might need to call this function several times. For example, to change an event-triggered report to run at 6 hourly intervals, you need to change the schedule (Type 4), synchronization time (Type 3), and period (Type 2). If you use incompatible values for these options, you can get unpredictable results. To change more than one option, disable the report, set the options, and then re-enable the report.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

## Syntax

**RepSetControl(*ReportName*, *Type*, *Data* [, *sClusterName*] )**

*ReportName*:

The name of the report (can be prefixed by the name of the cluster that is ClusterName.ReportName).

*nType*:

The type of report control information to set:

1 - The time of day at which to run the next report in Cicode (date/time) variable type. Subsequent reports are run at the times calculated from the period (Type 2) and synchronisation time (Type 3). Use Type 1 to specify a one-time report. Set the time in Data in seconds from midnight (for example, specify 6 p.m. as TimeMidNight() + (18 \* 60 \* 60) ).

2 - The report period. Set the new period in Data according to the report schedule (Type 4), in seconds from midnight, day of week (0 to 6, Sunday = 0), month (1 to 2), or year.

For a daily report schedule, set the report frequency in Data in seconds from midnight; for example, set Data to 6 \* 60 \* 60 for a 6 hourly shift report. If the report is weekly, set Data to the day of the week, for example, when Data = 2, the day is Tuesday.

3 - Synchronisation time of day of the report. Set the time in Data in seconds from midnight, for example, to synchronize at 10a.m., set Data to 10 \* 60 \* 60.

4 - Type of report schedule. Set Data to one of the following:

- 0 - Event triggered
- 1 - Daily
- 2 - Weekly
- 3 - Monthly
- 4 - Yearly

5 - Report state. Set Data to either:

- 0 - Enabled
- 1 - Disabled

*Data*

The new data value, dependent on the Type.

*sClusterName*:

Name of the cluster in which the report resides. This is optional if you have one cluster or are resolving the report server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[RepGetControl](#), [Report](#)

## Examples

Run the "Shift" report in 1 minute.

```
RepSetControl("Shift",1,TimeCurrent() + 60,"ClusterXYZ");
```

Change weekly report to 8 hour shift starting at 7 am

```
RepSetControl("Weekly", 5, 1,"ClusterXYZ"); ! disable report  
RepSetControl("Weekly", 4, 1,"ClusterXYZ"); ! change mode to daily  
RepSetControl("Weekly", 3, 7 * 60 * 60,"ClusterXYZ"); ! sync at 7:00:00 am  
RepSetControl("Weekly", 2, 8 * 60 * 60,"ClusterXYZ"); ! run every 8 hours  
RepSetControl("Weekly", 5, 0,"ClusterXYZ"); ! enable report
```

Change yearly report to run on March 10 at 7 am

```
RepSetControl("Yearly", 5, 1,"ClusterXYZ"); ! disable report  
RepSetControl("Yearly", 4, 4,"ClusterXYZ"); ! change mode to yearly  
RepSetControl("Yearly", 3, 7 * 60 * 60,"ClusterXYZ"); ! sync at 7:00:00 am  
RepSetControl("Yearly", 2, 31 + 28 + 10,"ClusterXYZ"); ! run on March 10th  
RepSetControl("Yearly", 5, 0,"ClusterXYZ"); ! enable report
```

## See Also

[Report Functions](#)

## Scheduler Functions

The following functions relate to Scheduler.

<a href="#">SchdClose</a>	Terminates a browsing session and cleans up the resources used by the session.
<a href="#">SchdConfigClose</a>	Terminates a browsing session and cleans up the resources used by the session.
<a href="#">SchdConfigFirst</a>	Places the data browse cursor at the first record.
<a href="#">SchdConfigGetField</a>	Returns the value of the particular field in a record to which the data browse cursor is currently referencing.
<a href="#">SchdConfigNext</a>	Places the data browse cursor at the next available record.
<a href="#">SchdConfigNumRecords</a>	Returns the number of records that match the current filter criteria.
<a href="#">SchdConfigOpen</a>	Initiates a new session for browsing the schedules configured.
<a href="#">SchdConfigPrev</a>	Places the data browse cursor at the previous record.
<a href="#">SchdFirst</a>	Places the data browse cursor at the first record.

<a href="#">SchdGetField</a>	Returns the value of the particular field in a record to which the data browse cursor is currently referencing.
<a href="#">SchdNext</a>	Places the data browse cursor at the next available record.
<a href="#">SchdNumRecords</a>	Returns the number of records that match the current filter criteria.
<a href="#">SchdOpen</a>	Initiates a new session for browsing the runtime schedules.
<a href="#">SchdPrev</a>	Places the data browse cursor at the previous record.
<a href="#">ScheduleItemAdd</a>	Adds a new schedule to the scheduler engine.
<a href="#">ScheduleItemDelete</a>	Deletes an existing schedule.
<a href="#">ScheduleItemModify</a>	Modifies an existing schedule.
<a href="#">ScheduleItemSetRepeat</a>	Adds recurrence information for an existing schedule to the scheduler engine.

## Special Day Functions

<a href="#">SchdSpecialAdd</a>	Adds a new special day group to the scheduler engine.
<a href="#">SchdSpecialClose</a>	Terminates a browsing session and cleans up the resources used in the session.
<a href="#">SchdSpecialDelete</a>	Deletes an existing special day group.
<a href="#">SchdSpecialFirst</a>	Places the data browse cursor at the first record.
<a href="#">SchdSpecialGetField</a>	Returns the value of the particular field in a record to which the data browse cursor is currently referencing.
<a href="#">SchdSpecialItemAdd</a>	Adds a new special day to the scheduler engine.
<a href="#">SchdSpecialItemAddRange</a>	Adds a range of special days in a specified special day group.
<a href="#">SchdSpecialItemClose</a>	Terminates a browsing session and cleans up the resources used in the session.
<a href="#">SchdSpecialItemDelete</a>	Deletes an existing Special day.

SchdSpecialItemDeleteRange	Deletes a range of special days in a specified special day group.
SchdSpecialItemFirst	Places the data browse cursor at the first record.
SchdSpecialItemGetField	Returns the value of the particular field in a record to which the data browse cursor is currently referencing.
SchdSpecialItemModify	Modifies an existing special day.
SchdSpecialItemModifyRange	Modifies a range of special days in a specified special day group.
SchdSpecialItemNext	Places the data browse cursor at the next available record.
SchdSpecialItemNumRecords	Returns the number of records that match the current filter criteria.
SchdSpecialItemOpen	Initiates a new session for browsing the special days.
SchdSpecialItemPrev	Places the data browse cursor at the previous record.
SchdSpecialModify	Modifies an existing special day group.
SchdSpecialNext	Places the data browse cursor at the next available record.
SchdSpecialNumRecords	Returns the number of records that match the current filter criteria.
SchdSpecialOpen	Initiates a new session for browsing the special day groups.
SchdSpecialPrev	Places the data browse cursor at the previous record.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## SchdClose

The SchdClose function terminates a browsing session and cleans up the resources used by the session.

## Syntax

`INT SchdClose(LONG Session)`

*Session:*

The handle to a browse session previously returned by the SchdOpen call.

## Return Value

0 (zero) if the browsing session exists, otherwise an error code is returned.

## Related Functions

[SchdConfigOpen](#), [SchdConfigClose](#), [SchdConfigFirst](#), [SchdConfigNext](#), [SchdConfigPrev](#), [SchdConfigGetField](#),  
[SchdConfigNumRecords](#), [SchdOpen](#), [SchdFirst](#), [SchdNext](#), [SchdPrev](#), [SchdGetField](#), [SchdNumRecords](#),  
[ScheduleItemAdd](#), [ScheduleItemSetRepeat](#), [ScheduleItemModify](#), [ScheduleItemDelete](#)

## See Also

[Scheduler Functions](#)

## SchdConfigClose

The SchdConfigClose function terminates a browsing session and cleans up the resources used by the session.

## Syntax

`INT SchdConfigClose(LONG Session )`

*Session:*

The handle to a browse session previously returned by the SchdConfigOpen call.

## Return Value

0 (zero) if the browsing session exists, otherwise an error code is returned.

## Related Functions

[SchdClose](#), [SchdConfigOpen](#), [SchdConfigFirst](#), [SchdConfigNext](#), [SchdConfigPrev](#), [SchdConfigGetField](#),  
[SchdConfigNumRecords](#), [SchdOpen](#), [SchdFirst](#), [SchdNext](#), [SchdPrev](#), [SchdGetField](#), [SchdNumRecords](#),  
[ScheduleItemAdd](#), [ScheduleItemSetRepeat](#), [ScheduleItemModify](#), [ScheduleItemDelete](#)

## See Also

[Scheduler Functions](#)

### SchdConfigFirst

The SchdConfigFirst function places the data browse cursor at the first record.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT SchdConfigFirst(LONG Session)**

*Session:*

The handle to a browse session previously returned by the SchdConfigOpen call.

## Return Value

0 (zero) if the browsing session exists, otherwise an error code is returned.

## Related Functions

[SchdClose](#), [SchdConfigOpen](#), [SchdConfigClose](#), [SchdConfigNext](#), [SchdConfigPrev](#), [SchdConfigGetField](#),  
[SchdConfigNumRecords](#), [SchdOpen](#), [SchdFirst](#), [SchdNext](#), [SchdPrev](#), [SchdGetField](#), [SchdNumRecords](#),  
[ScheduleItemAdd](#), [ScheduleItemSetRepeat](#), [ScheduleItemModify](#), [ScheduleItemDelete](#)

## See Also

[Scheduler Functions](#)

### SchdConfigGetField

The SchdConfigGetField function returns the value of the particular field in a record to which the data browse cursor is currently referencing.

## Syntax

**STRING SchdConfigGetField(LONG Session, STRING Field )**

*Session:*

The handle to a browse session previously returned by the SchdConfigOpen call.

*Field*

A field to retrieve. Refer to [SchdConfigOpen](#) for detailed list of fields.

## Return Value

The value of the specified field as a string. An empty string may or may not be an indication that an error has been detected. The last error should be checked in this instance to determine if an error has actually occurred.

## Related Functions

[SchdClose](#), [SchdConfigOpen](#), [SchdConfigClose](#), [SchdConfigFirst](#), [SchdConfigNext](#), [SchdConfigPrev](#),  
[SchdConfigGetField](#), [SchdConfigNumRecords](#), [SchdOpen](#), [SchdFirst](#), [SchdNext](#), [SchdPrev](#), [SchdNumRecords](#),  
[ScheduleItemAdd](#), [ScheduleItemSetRepeat](#), [ScheduleItemModify](#), [ScheduleItemDelete](#)

## Example

```
TIMESTAMP tsStart = StrToTimestamp(SchdConfigGetFiel(hSession, "Start"),15,0);
TIMESTAMP tsEnd = StrToTimestamp(SchdConfigGetFiel(hSession, "End"),15,0);
```

## See Also

[Scheduler Functions](#)

## SchdConfigNext

The SchdConfigNext function places the data browse cursor at the next available record. This function will return an error if called when the data cursor is at end of the records.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **SchdConfigNext**(LONG *Session*)

*Session*:

The handle to a browse session previously returned by the SchdConfigOpen call.

## Return Value

0 (zero) if the browsing session exists, otherwise an error code is returned.

## Related Functions

[SchdClose](#), [SchdConfigOpen](#), [SchdConfigClose](#), [SchdConfigFirst](#), [SchdConfigPrev](#), [SchdConfigGetField](#),  
[SchdConfigNumRecords](#), [SchdOpen](#), [SchdFirst](#), [SchdNext](#), [SchdPrev](#), [SchdGetField](#), [SchdNumRecords](#),  
[ScheduleItemAdd](#), [ScheduleItemSetRepeat](#), [ScheduleItemModify](#), [ScheduleItemDelete](#)

## See Also

[Scheduler Functions](#)

## SchdConfigNumRecords

The SchdConfigNumRecords function returns the number of records that match the current filter criteria. This function uses iSession as an argument which is previously returned by the SchdConfigOpen function.

## Syntax

```
LONG SchdConfigNumRecords(LONG Session)
```

*Session:*

The handle to a browse session previously returned by the SchdConfigOpen call.

## Return Value

Returns number of records. 0 means no records were found.

## Related Functions

[SchdClose](#), [SchdConfigOpen](#), [SchdConfigClose](#), [SchdConfigFirst](#), [SchdConfigNext](#), [SchdConfigPrev](#),  
[SchdConfigGetField](#), [SchdOpen](#), [SchdFirst](#), [SchdNext](#), [SchdPrev](#), [SchdGetField](#), [SchdNumRecords](#),  
[ScheduleItemAdd](#), [ScheduleItemSetRepeat](#), [ScheduleItemModify](#), [ScheduleItemDelete](#)

## See Also

[Scheduler Functions](#)

## SchdConfigOpen

The SchdConfigOpen function initiates a new session for browsing the schedules configured. It returns a handle for the browsing session which can be used for further browsing operations.

## Syntax

```
LONG SchdConfigOpen([TIMESTAMP Start] [, LONG Duration] [, STRING Filter] [, STRING Fields] [, STRING Clusters] )
```

*Start:*

The start date of the schedules in UTC time to return during the browse. If not specified, today at midnight will be taken as start time. The types of this field is TIMESTAMP. Use [StrToTimestamp](#) or [TimestampCreate](#) Cicode functions to create a TIMESTAMP type.

*Duration:*

The duration of the browse in seconds. The default is 86400 seconds(24 hour).

*Filter:*

A filter expression specifying the records to return during the browse. An empty string indicates that every record will be returned.

All string fields can be filtered based on regular expressions. Using an operator other than = will cause strings to

not match the filter criteria. The following regular expressions are supported \*expr, expr\*, and \*expr\*.

**Fields:**

Specifies via a comma delimited string the columns to be returned during the browse. An empty string indicates that the server will return every available column.

Supported fields are:

**ID:** The unique ID of the schedule within the schedule engine. ID value is the same for all recurring schedules.

**EQUIPMENT:** The equipment name

**STATE:** The state of the schedule

**START:** The start time of the schedule.

**OCCURRENCESTART:** The start of this occurrence. For non-recurring entries will be the same as START.

**END:** The end time of the schedule.

**OCCURRENCEEND:** The end of this occurrence. For non-recurring entries will be the same as END.

**DESC:** The description of the schedule.

**MODIFIEDTIME:** The time when the entry was modified last time.

The following fields are used when a schedule is recurrent. The value of FREQ field -1 means the schedule is not recurrent:

**FREQ:** The type of the recurrence: 4 - daily, 5 - weekly, 6- monthly, 7 - yearly, 8 - special days. -1 means this schedule is non-recurring.

**INTERVAL:** The interval of the recurrence. 1 - every 1 day/week/month etc., 2 - every second day/week/month etc. and so on.

**WEEKDAY:** The first day of the week.

**WEEKDAYMASK:** The day of week mask. Defines day of week where recurrence happens.

- None = 0,
- Sunday = 1,
- Monday = 2,
- Tuesday = 4,
- Wednesday = 8,
- Thursday = 16,
- Friday = 32,
- Saturday = 64,
- Everyday = 127,
- Weekdays = 62,
- Weekendays = 65

Combination of days can be achieved by addition. For example, Tuesdays and Wednesdays would be 12 (4 + 8).

**MAXREC:** The maximum number recurrences (-1 means MAXREC is not specified. If MAXREC is -1 then the recurrence stops on the date specified by RECOUNTIL field. If RECOUNTIL is not specified the recurrence occurs forever).

**RECOUNTIL:** The time until recurrence occurs. -1 means never stops (or finishes after MAXREC occurrences if is specified).

**DAYORD:** Day ordinal. Applicable for monthly and yearly recurrence patterns. Using the values 1 to 4 you can set

the schedule to run on the first, second, third, or fourth occurrence of the DAY in each month (monthly recurrences), or the specified MONTH (yearly recurrences). Use the value -1 for the last week of the month.

**DAY:** The day of month

**MONTH:** The month of the year

**SPECIALINC:** Defined whether the special day is included in the pattern. 0 - none, 1 - all, 2 - selected.

**GROUPIDS** A list of group's ID included in the recurring pattern (used only when SPECIALINC is defined as 2 ("selected")).

*sClusters:*

An optional parameter that specifies via a comma delimited string the subset of the clusters to browse. An empty string indicates that the connected clusters will be browsed.

**Note:** Time values (START, OCCURRENCESTART, END, OCCURRENCEEND, MODIFIEDTIME) are TIMESTAMP type.  
Use [TimestampToStr\(\)](#) Cicode function to convert it to string.

## Return Value

Returns an integer handle to the browse session. Returns -1 when an error is detected.

## Related Functions

[SchdClose](#), [SchdConfigClose](#), [SchdConfigFirst](#), [SchdConfigNext](#), [SchdConfigPrev](#), [SchdConfigGetField](#),  
[SchdConfigNumRecords](#), [SchdOpen](#), [SchdFirst](#), [SchdNext](#), [SchdPrev](#), [SchdGetField](#), [SchdNumRecords](#),  
[ScheduleItemAdd](#), [ScheduleItemSetRepeat](#), [ScheduleItemModify](#), [ScheduleItemDelete](#)

## Example

```
tsConfigtimeBrowseStartTime = TimestampCreate(2011, 11, 20, 0, 0, 0, 0);
iConfigtimeSession = SchdConfigOpen(tsConfigtimeBrowseStartTime, 86400, "Id=1",
"", "Cluster1");
IF iConfigtimeSession = -1 THEN
    prompt("Could not open a schedule configtime browse session");
END
```

## See Also

[Scheduler Functions](#)

## SchdConfigPrev

The SchdConfigPrev function places the data browse cursor at the previous record. This function will return an error if called when the data cursor is at beginning of the records.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT SchdConfigPrev(LONG Session)**

*Session:*

The handle to a browse session previously returned by the SchdConfigOpen call.

## Return Value

0 (zero) if the browsing session exists, otherwise an error code is returned.

## Related Functions

[SchdClose](#), [SchdConfigOpen](#), [SchdConfigClose](#), [SchdConfigFirst](#), [SchdConfigNext](#), [SchdConfigGetField](#),  
[SchdConfigNumRecords](#), [SchdOpen](#), [SchdFirst](#), [SchdNext](#), [SchdPrev](#), [SchdGetField](#), [SchdNumRecords](#),  
[ScheduleItemAdd](#), [ScheduleItemSetRepeat](#), [ScheduleItemModify](#), [ScheduleItemDelete](#)

## See Also

[Scheduler Functions](#)

## SchdFirst

The SchdFirst function places the data browse cursor at the first record.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT SchdFirst(LONG Session)**

*Session:*

The handle to a browse session previously returned by the SchdOpen call.

## Return Value

0 (zero) if the browsing session exists, otherwise an error code is returned.

## Related Functions

[SchdClose](#), [SchdConfigOpen](#), [SchdConfigClose](#), [SchdConfigFirst](#), [SchdConfigNext](#), [SchdConfigPrev](#),  
[SchdConfigGetField](#), [SchdConfigNumRecords](#), [SchdOpen](#), [SchdNext](#), [SchdPrev](#), [SchdGetField](#), [SchdNumRecords](#),  
[ScheduleItemAdd](#), [ScheduleItemSetRepeat](#), [ScheduleItemModify](#), [ScheduleItemDelete](#)

## See Also

[Scheduler Functions](#)

## SchdGetField

The SchdGetField function returns the value of the particular field in a record to which the data browse cursor is currently referencing.

## Syntax

STRING **SchdGetField**(LONG *Session*, STRING *Field*)

*Session*:

The handle to a browse session previously returned by the SchdOpen call.

*Field*:

Specifies via a comma delimited string the columns to be returned during the browse. An empty string indicates that the server will return every available column. See [SchdOpen](#) for supported fields.

## Return Value

The value of the specified field as a string. An empty string may or may not be an indication that an error has been detected. The last error should be checked in this instance to determine if an error has actually occurred.

## Related Functions

[SchdClose](#), [SchdConfigOpen](#), [SchdConfigClose](#), [SchdConfigFirst](#), [SchdConfigNext](#), [SchdConfigPrev](#),  
[SchdConfigGetField](#), [SchdConfigNumRecords](#), [SchdOpen](#), [SchdFirst](#), [SchdNext](#), [SchdPrev](#), [SchdNumRecords](#),  
[ScheduleItemAdd](#), [ScheduleItemSetRepeat](#), [ScheduleItemModify](#), [ScheduleItemDelete](#)

### Example

```
TIMESTAMP tsStart = StrToTimestamp(SchdGetField(hSession, "Start"),15,0);
TIMESTAMP tsEnd = StrToTimestamp(SchdGetField(hSession, "End"),15,0);
```

## See Also

[Scheduler Functions](#)

## SchdNext

The SchdNext function places the data browse cursor at the next available record. This function will return an error if called when the data cursor is at end of the records.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **SchdNext**(LONG *Session*)

*Session*:

The handle to a browse session previously returned by the SchdOpen call.

## Return Value

0 (zero) if the browsing session exists, otherwise an error code is returned.

## Related Functions

[SchdClose](#), [SchdConfigOpen](#), [SchdConfigClose](#), [SchdConfigFirst](#), [SchdConfigNext](#), [SchdConfigPrev](#),  
[SchdConfigGetField](#), [SchdConfigNumRecords](#), [SchdOpen](#), [SchdFirst](#), [SchdPrev](#), [SchdGetField](#), [SchdNumRecords](#),  
[ScheduleItemAdd](#), [ScheduleItemSetRepeat](#), [ScheduleItemModify](#), [ScheduleItemDelete](#)

## See Also

[Scheduler Functions](#)

## SchdNumRecords

The SchdNumRecords function returns the number of records that match the current filter criteria.

## Syntax

LONG **SchdNumRecords**(LONG *Session*)

*Session*:

The handle to a browse session previously returned by the SchdOpen call.

## Return Value

The number of records that have matched the filter criteria. A value of 0 denotes that no records have matched. A value of -1 denotes that the browse session is unable to provide a fixed number. This may be the case if the data being browsed changed during the browse session.

## Related Functions

[SchdClose](#), [SchdConfigOpen](#), [SchdConfigClose](#), [SchdConfigFirst](#), [SchdConfigNext](#), [SchdConfigPrev](#),  
[SchdConfigGetField](#), [SchdConfigNumRecords](#), [SchdOpen](#), [SchdFirst](#), [SchdNext](#), [SchdPrev](#), [SchdGetField](#),  
[ScheduleItemAdd](#), [ScheduleItemSetRepeat](#), [ScheduleItemModify](#), [ScheduleItemDelete](#)

## See Also

[Scheduler Functions](#)

## SchdOpen

The SchdOpen function initiates a new session for browsing the runtime schedules. It returns a handle for the browsing session which can be used for further browsing operations.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

LONG **SchdOpen**(STRING *Equipment*[, TIMESTAMP *Start*] [,LONG *Duration*] [,STRING *Filter*] [,STRING *Fields*] [, STRING *Clusters*] )

*Equipment*:

The name of the equipment to browse.

*Start*:

The start date of the schedules. If not specified, today at midnight will be taken as the start time. The types of this field is TIMESTAMP. Use StrToTimestamp or TimestampCreate Cicode functions to create a TIMESTAMP type.

*Duration*:

The duration of the browse in seconds. The default is 86400 seconds(24 hour).

*Filter*:

A filter expression specifying the records to return during the browse. An empty string indicates that every record will be returned.

*Fields*:

Specifies via a comma delimited string the columns to be returned during the browse. An empty string indicates that the server will return every available column. Supported fields are:

**EQUIPMENT**: The equipment name

**STATE**: The state to be set by the schedule

**START**: The start time of the schedule. The time is returned as a Timestamp value. Use StrToTimestamp() to get a TIMESTAMP data value.

**END**: The end time of the schedule. The time is returned as a Timestamp value. Use StrToTimestamp() to get a TIMESTAMP data value.

**INHERITED**: Indicates if the schedule entry is inherited from a schedule defined in a parent equipment higher in the hierarchy.

**DESC**: The description of the schedule.

*Clusters*:

An optional parameter that specifies via a comma delimited string the subset of the clusters to browse. An empty string indicates that the connected clusters will be browsed.

## Return Value

Returns an integer handle to the browse session. Returns -1 when an error is detected.

For errors refer to the topic [Cicode and General Errors](#).

## Related Functions

[SchdClose](#), [SchdConfigOpen](#), [SchdConfigClose](#), [SchdConfigFirst](#), [SchdConfigNext](#), [SchdConfigPrev](#),  
[SchdConfigGetField](#), [SchdConfigNumRecords](#), [SchdFirst](#), [SchdNext](#), [SchdPrev](#), [SchdGetField](#), [SchdNumRecords](#),  
[ScheduleItemAdd](#), [ScheduleItemSetRepeat](#), [ScheduleItemModify](#), [ScheduleItemDelete](#)

## Example

```
tsRuntimeBwseStartTime = TimestampCreate(2011, 11, 20, 0, 0, 0, 0);
iRuntimeSession = SchdOpen("MyEquipment1", tsRuntimeBrowseStartTime, 86400,
"state=ON", "", "Cluster1");
IF iRuntimeSession = -1 THEN
    prompt("Could not open a schedule runtime browse session");
END
```

## See Also

[Scheduler Functions](#)

### SchdPrev

The SchdPrev function places the data browse cursor at the previous record. This function will return an error if called when the data cursor is at beginning of the records.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **SchdPrev**(LONG Session)

*Session:*

The handle to a browse session previously returned by the SchdOpen call.

## Return Value

0 (zero) if the browsing session exists, otherwise an error code is returned.

## Related Functions

[SchdClose](#), [SchdConfigOpen](#), [SchdConfigClose](#), [SchdConfigFirst](#), [SchdConfigNext](#), [SchdConfigPrev](#),  
[SchdConfigGetField](#), [SchdConfigNumRecords](#), [SchdOpen](#), [SchdFirst](#), [SchdNext](#), [SchdGetField](#), [SchdNumRecords](#),  
[ScheduleItemAdd](#), [ScheduleItemSetRepeat](#), [ScheduleItemModify](#), [ScheduleItemDelete](#)

## See Also

[Scheduler Functions](#)

### SchdSpecialAdd

The SchdSpecialAdd adds a new Special Day Group to the Scheduler engine.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

LONG **SchdSpecialAdd**(STRING *Cluster*, STRING *Name*)

*Cluster*:

The name of the cluster.

*Name*:

Name of the special day group.

## Return Value

Returns the ID of the Special Day Group which can be used for modifying and deleting this Special Day Group.

Returns the ID for an existing Special Day group. This function returns -1 if, unsuccessful. Trap the error to get the error returned by this function.

## Related Functions

[SchdSpecialDelete](#), [SchdSpecialModify](#)

## See Also

[Scheduler Functions](#)

## SchdSpecialClose

The SchdSpecialClose function terminates a browsing session and cleans up the resources used by the session. This function uses iSession as the argument which is previously returned by the SchdSpecialOpen function.

## Syntax

INT **SchdSpecialClose**(LONG *Session*)

*Session*:

Current special day group browsing session obtained by [SchdSpecialOpen](#).

## Return Value

0 (zero) if the browsing session exists, otherwise an error code is returned.

## Related Functions

[SchdSpecialOpen](#), [SchdSpecialFirst](#), [SchdSpecialNext](#), [SchdSpecialPrev](#), [SchdSpecialGetField](#),  
[SchdSpecialNumRecords](#)

## See Also

[Scheduler Functions](#)

### SchdSpecialDelete

The SchdSpecialDelete function deletes an existing special day group.

## Syntax

LONG **SchdSpecialDelete**(STRING *Cluster*, LONG *ID*)

*Cluster*:

Name of the Cluster.

*ID*

Special day group ID.

## Return Value

Returns 0 if successful otherwise it returns an error.

## Related Functions

[SchdSpecialAdd](#), [SchdSpecialModify](#)

## See Also

[Scheduler Functions](#)

### SchdSpecialFirst

The SchdSpecialFirst function places the data browse cursor at the first record. This function uses *Session* as the argument which is previously returned by the SchdSpecialOpen function.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **SchdSpecialFirst**(LONG *Session*)

*Session*:

Current special day group browsing session obtained by [SchdSpecialOpen](#).

## Return Value

0 (zero) if the browsing session exists, otherwise an error code is returned.

## Related Functions

[SchdSpecialOpen](#), [SchdSpecialClose](#), [SchdSpecialNext](#), [SchdSpecialPrev](#), [SchdSpecialGetField](#),  
[SchdSpecialNumRecords](#)

## See Also

[Scheduler Functions](#)

### SchdSpecialGetField

The SchdSpecialGetField function returns the value of a particular field from the record currently referenced by the data browse cursor. This function uses *Session* as an argument which is previously returned by the [SchdSpecialOpen](#) function and the field name of the value to be returned.

## Syntax

STRING **SchdSpecialItemGetField**(LONG *Session*, STRING *Field*)

*Session*

Current special day group browsing session obtained by [SchdSpecialOpen](#).

*Field*

Specifies via a comma delimited string the columns to be returned during the browse. An empty string indicates that the server will return every available column. See [SchdSpecialOpen](#) for supported fields.

## Return Value

The value of the specified field as a string. An empty string may or may not be an indication that an error has been detected. The last error should be checked in this instance to determine if an error has actually occurred.

## Related Functions

[SchdSpecialOpen](#), [SchdSpecialFirst](#), [SchdSpecialNext](#), [SchdSpecialPrev](#), [SchdSpecialClose](#)

## See Also

[Scheduler Functions](#)

### SchdSpecialItemAdd

The SchdSpecialItemAdd function adds a new Special Day to the scheduler engine. It returns the ID of the Special Day which can be used for modifying and deleting the Special Day.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

```
LONG SchdSpecialItemAdd(STRING Cluster, LONG GroupID, STRING Name, TIMESTAMP Day)
```

*Cluster:*

Name of the Cluster.

*GroupID:*

Special day group ID.

*Name:*

Name of special day.

*Day:*

The special day as a timestamp. Use [TimestampCreate](#) or [StrToTimestamp](#) Cicode functions to get a timestamp data value.

## Return Value

Returns the ID for an existing special day or -1 if unsuccessful. Trap the error to get the error returned by this function.

## Related Functions

[SchdSpecialItemModify](#), [SchdSpecialItemDelete](#)

## Example

```
TIMESTAMP tsDate = TimestampCreate(2012,1,2,0,0,0);
.
// Add the new special day
nCatID = SchdSpecialAdd(STRING Cluster, sCategory);
IF (nCatID > -1) THEN
    nSchdID = SchdSpecialItemAdd(STRING Cluster, nCatID, sLabel, tsDate);
    IF (nSchdID > -1) THEN
        // Use the new schedule special day item
    END
END
```

## See Also

[Scheduler Functions](#)

## SchdSpecialItemAddRange

The SchdSpecialItemAddRange function adds a range of special days in the specified special day group (category).

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

```
INT SchdSpecialItemAddRange(STRING Cluster, STRING GroupID, STRING Name, TIMESTAMP FirstDay,  
TIMESTAMP LastDay)
```

*Cluster:*

The name of the cluster.

*GroupID:*

The ID of the special day group.

*Name:*

The name of the special day.

*FirstDay:*

The first day of the special day range as a timestamp. Use TimestampCreate or StrToTimestamp Cicode functions to get a timestamp data value. When creating a timestamp, please use midnight of the required day.

*LastDay:*

Last day (inclusive) of the special day range as a timestamp. Use TimestampCreate or StrToTimestamp Cicode functions to get a timestamp data value. When creating a timestamp, please use midnight of the required day.

## Return Value

0 if successful, otherwise an error code is returned.

## Related Functions

[SchdSpecialItemDeleteRange](#), [SchdSpecialItemModifyRange](#)

## Example

```
// Add the special day range for group id 1  
// 2018-05-06 : 2018-05-10  
  
SchdSpecialItemAddRange("Cluster1", 1,  
TimestampCreate(2018,5,6,0,0,0,0), TimestampCreate(2018,5,10,0,0,0,0));
```

## See Also

[Scheduler Functions](#)

## SchdSpecialItemClose

This function terminates a browsing session and cleans up the resources used by the session. This function uses iSession as the argument which is previously returned by the [SchdSpecialItemOpen](#) function.

## Syntax

```
INT SchdSpecialItemClose(LONG Session)
```

*Session:*

Current schedule special item browsing session obtained by [SchdSpecialItemOpen](#).

## Return Value

0 (zero) if the browsing session exists, otherwise an error code is returned.

## Related Functions

[SchdSpecialItemOpen](#), [SchdSpecialItemFirst](#), [SchdSpecialItemNext](#), [SchdSpecialItemPrev](#),  
[SchdSpecialItemGetField](#), [SchdSpecialItemNumRecords](#)

## See Also

[Scheduler Functions](#)

## SchdSpecialItemDelete

This function deletes an existing Special Day.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

```
LONG SchdSpecialItemDelete(STRING Cluster, LONG ID)
```

*Cluster:*

Name of the cluster.

*ID:*

Special day ID.

## Return Value

This function returns 0 if successful otherwise it returns an error.

## Related Functions

[SchdSpecialItemAdd](#), [SchdSpecialItemModify](#)

## See Also

[Scheduler Functions](#)

## SchdSpecialItemDeleteRange

The SchdSpecialItemDeleteRange function removes a range of special days in a specified special day group (category).

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **SchdSpecialItemDeleteRange**(STRING *Cluster*, STRING *GroupID*, TIMESTAMP *FirstDay*, TIMESTAMP *LastDay*)

*Cluster*:

The name of the cluster.

*GroupID*:

The ID of the special day group.

*FirstDay*:

The first day of the special day range as a timestamp. Use TimestampCreate or StrToTimestamp Cicode functions to get a timestamp data value. When creating a timestamp, please use midnight of the required day.

*LastDay*:

Last day (inclusive) of the special day range as a timestamp. Use TimestampCreate or StrToTimestamp Cicode functions to get a timestamp data value. When creating a timestamp, please use midnight of the required day.

## Return Value

0 if successful, otherwise an error code is returned.

## Related Functions

[SchdSpecialItemAddRange](#), [SchdSpecialItemModifyRange](#)

## Example

```
// Delete the following special day range for group id 1
// 2018-05-06 : 2018-05-10

SchdSpecialItemDeleteRange("Cluster1", 1,
TimestampCreate(2018,5,6,0,0,0,0), TimestampCreate(2018,5,10,0,0,0,0));
```

## See Also

[Scheduler Functions](#)

## SchdSpecialItemFirst

The SchdSpecialItemFirst places the data browse cursor at the first record. This function uses iSession as the argument which is previously returned by the SchdSpecialItemOpen function.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT SchdSpecialItemFirst(LONG Session)**

*Session:*

Current schedule special item browsing session obtained by [SchdSpecialItemOpen](#).

## Return Value

0 (zero) if the browsing session exists, otherwise an error code is returned.

## Related Functions

[SchdSpecialItemOpen](#), [SchdSpecialItemClose](#), [SchdSpecialItemNext](#), [SchdSpecialItemPrev](#),  
[SchdSpecialItemGetField](#), [SchdSpecialItemNumRecords](#)

## See Also

[Scheduler Functions](#)

## SchdSpecialItemGetField

This function returns the value of the particular field in a record to which the data browse cursor is currently referencing. This function uses *Session* as an argument which is previously returned by the [SchdSpecialItemOpen](#) function and the field name of the value to be returned.

## Syntax

**STRING SchdSpecialItemGetField(LONG Session, STRING Field)**

*Session*

Current special day group browsing session obtained by [SchdSpecialItemOpen](#).

*Field*

Specifies via a comma delimited string the columns to be returned during the browse. An empty string indicates that the server will return every available column. Supported fields are:

**GROUPNAME:** The group name of the special day

**NAME:** The name of the special day

**DAY:** The special day. The time is returned as a Timestamp value. Use [StrToTimestamp\(\)](#) to get a TIMESTAMP data value

**ID:** The unique ID of the special day. Every special day is assigned a unique ID.

## Return Value

The value of the specified field as a string. An empty string may or may not be an indication that an error has been detected. The last error should be checked in this instance to determine if an error has actually occurred.

## Related Functions

[SchdSpecialItemOpen](#), [SchdSpecialItemClose](#), [SchdSpecialItemNext](#), [SchdSpecialItemPrev](#), [SchdSpecialItemFirst](#), [SchdSpecialItemNumRecords](#)

## Example

```
TIMESTAMP tsDay = StrToTimestamp(SchdSpecialItemGetField(hSession, "Day"),15,0);
```

## See Also

[Scheduler Functions](#)

## SchdSpecialItemModify

This function modifies an existing Special Day.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

```
LONG SchdSpecialItemModify(STRING Cluster, LONG ID, STRING Name, TIMESTAMP Day)
```

*Cluster:*

The name of the cluster.

*ID*

ID of the special day to be modified.

*Name*

The name of the special day.

*Day:*

The special day as a timestamp. Use TimestampCreate or StrToTimestamp Cicode functions to get a timestamp data value.

## Return Value

This function returns 0 if successful otherwise it returns an error code.

## Related Functions

[SchdSpecialItemAdd](#), [SchdSpecialItemDelete](#)

## See Also

[Scheduler Functions](#)

### SchdSpecialItemModifyRange

The SchdSpecialItemModifyRange function can be used to modify a range of special days in a specified special day group (category).

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

```
INT SchdSpecialItemAddRange(STRING Cluster, STRING GroupID, TIMESTAMP FirstDay, TIMESTAMP LastDay,  
STRING NewName, TIMESTAMP NewFirstDay, TIMESTAMP NewLastDay)
```

*Cluster*:

The name of the cluster.

*GroupID*:

The ID of the special day group.

*FirstDay*:

The first day of the special day range as a timestamp. Use TimestampCreate or StrToTimestamp Cicode functions to get a timestamp data value. When creating a timestamp, please use midnight of the required day.

*LastDay*:

Last day (inclusive) of the special day range as a timestamp. Use TimestampCreate or StrToTimestamp Cicode functions to get a timestamp data value. When creating a timestamp, please use midnight of the required day.

*NewName*:

A new name for the special days.

*NewFirstDay*:

A new first day for the special day range as a timestamp. Use TimestampCreate or StrToTimestamp Cicode functions to get a timestamp data value. When creating a timestamp, please use midnight of the required day.

*NewLastDay*:

A new last day (inclusive) of the special day range as a timestamp. Use TimestampCreate or StrToTimestamp Cicode functions to get a timestamp data value. When creating a timestamp, please use midnight of the required day.

## Return Value

0 if successful, otherwise an error code is returned.

## Related Functions

[SchdSpecialItemAddRange](#), [SchdSpecialItemDeleteRange](#)

## Example

```
// Modify the existing special day range for group id 1
// from 2018-05-06 : 2018-05-10
// to 2018-06-06 : 2018-06-10

SchdSpecialItemModifyRange("Cluster1", 1, "NewDays",
TimestampCreate(2018,5,6,0,0,0,0), TimestampCreate(2018,5,10,0,0,0,0),
TimestampCreate(2018,6,6,0,0,0,0), TimestampCreate(2018,6,10,0,0,0,0));
```

## See Also

[Scheduler Functions](#)

### SchdSpecialItemNext

This function places the data browse cursor at the next available record. This function uses iSession as the argument which is previously returned by the SchdSpecialItemOpen function. This function will return an EOF error code if called when the data cursor is at end of the records.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **SchdSpecialItemNext**(LONG Session)

*Session:*

Current special day group browsing session obtained by [SchdSpecialItemOpen](#).

## Return Value

0 (zero) if the browsing session exists, otherwise an error code is returned.

## Related Functions

[SchdSpecialItemOpen](#), [SchdSpecialItemClose](#), [SchdSpecialItemFirst](#), [SchdSpecialItemPrev](#),  
[SchdSpecialItemGetField](#)

## See Also

[Scheduler Functions](#)

### SchdSpecialItemNumRecords

The SchdSpecialItemNumRecords function returns the number of records that match the current filter criteria. This function uses iSession as an argument which is previously returned by the SchdSpecialItemOpen function.

## Syntax

LONG **SchdSpecialItemNumRecords**(LONG *Session*)

*Session*:

Current special day group browsing session obtained by [SchdSpecialItemOpen](#).

## Return Value

The number of records that have matched the filter criteria. A value of 0 denotes that no records have matched. A value of -1 denotes that the browse session is unable to provide a fixed number. This may be the case if the data being browsed changed during the browse session.

## Related Functions

[SchdSpecialItemOpen](#), [SchdSpecialItemClose](#), [SchdSpecialItemNext](#), [SchdSpecialItemPrev](#), [SchdSpecialItemFirst](#)

## See Also

[Scheduler Functions](#)

## SchdSpecialItemOpen

This function initiates a new session for browsing the special days. It returns a handle for the browsing session which can be used for further browsing operations.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

LONG **SchdSpecialItemOpen**(STRING *Filter*, STRING *Fields* [, STRING *Clusters*])

*Filter*:

A filter expression specifying the records to return during the browse. An empty string indicates that all records will be returned.

*Fields*:

Specifies via a comma delimited string the columns to be returned during the browse. An empty string indicates that the server will return every available column. See [SchdSpecialItemGetField](#) for the fields supported.

*Clusters*:

An optional parameter that specifies via a comma delimited string the subset of the clusters to browse. An empty string indicates that the connected clusters will be browsed.

## Return Value

Returns an integer handle to the browse session. Returns -1 when an error is detected.

## Related Functions

[SchdSpecialItemFirst](#), [SchdSpecialItemClose](#), [SchdSpecialItemNext](#), [SchdSpecialItemPrev](#),  
[SchdSpecialItemGetField](#), [SchdSpecialItemNumRecords](#)

## See Also

[Scheduler Functions](#)

### SchdSpecialItemPrev

This function places the data browse cursor at the previous record. This function uses iSession as the argument which is previously returned by the SchdSpecialItemOpen function. This function will return an error (EOF) if called when the data cursor is at beginning of the records.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT SchdSpecialItemPrev(LONG Session)**

*Session:*

Current special day group browsing session obtained by [SchdSpecialItemOpen](#).

## Return Value

0 (zero) if the browsing session exists, otherwise an error code is returned.

## Related Functions

[SchdSpecialItemOpen](#), [SchdSpecialItemClose](#), [SchdSpecialItemNext](#), [SchdSpecialItemFirst](#),  
[SchdSpecialItemGetField](#), [SchdSpecialItemNumRecords](#)

## See Also

[Scheduler Functions](#)

### SchdSpecialModify

This function modifies an existing special day group.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**LONG SchdSpecialModify(STRING Cluster, LONG ID, STRING NewName)**

*Cluster:*

The name of the cluster.

*ID*:

Existing ID for the current special day group.

*NewName*:

New name of the current special day group.

## Return Value

This function returns 0 if successful otherwise it returns an error.

For error code information codes refer to the topic [Cicode and General Errors](#).

## Related Functions

[SchdSpecialDelete](#), [SchdSpecialAdd](#)

## See Also

[Scheduler Functions](#)

## SchdSpecialNext

This function places the data browse cursor at the next available record. This function uses *iSession* as the argument which is previously returned by the [SchdSpecialOpen](#) function. This function will return an EOF error code if called when the data cursor is at end of the records.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **SchdSpecialNext**(LONG *Session*)

*Session*:

Current special day group browsing session obtained by [SchdSpecialOpen](#).

## Return Value

0 (zero) if the browsing session exists, otherwise an error code is returned.

## Related Functions

[SchdSpecialOpen](#), [SchdSpecialFirst](#), [SchdSpecialClose](#), [SchdSpecialPrev](#), [SchdSpecialGetField](#),  
[SchdSpecialNumRecords](#)

## See Also

[Scheduler Functions](#)

### SchdSpecialNumRecords

The SchdSpecialNumRecords function returns the number of records that match the current filter criteria. This function uses iSession as an argument which is previously returned by the SchdSpecialOpen function.

## Syntax

LONG **SchdSpecialNumRecords**(LONG *Session*)

*Session*:

Current special day group browsing session obtained by [SchdSpecialOpen](#).

## Return Value

The number of records that have matched the filter criteria. A value of 0 denotes that no records have matched. A value of -1 denotes that the browse session is unable to provide a fixed number. This may be the case if the data being browsed changed during the browse session.

## Related Functions

[SchdSpecialOpen](#), [SchdSpecialFirst](#), [SchdSpecialNext](#), [SchdSpecialPrev](#), [SchdSpecialGetField](#), [SchdSpecialClose](#)

## See Also

[Scheduler Functions](#)

### SchdSpecialOpen

The SchdSpecialOpen function initiates a new session for browsing the special day groups. It returns a handle for the browsing session which can be used for further browsing operations.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

LONG **SchdSpecialOpen**(STRING *Filter*, STRING *Fields*[, STRING *Clusters*])

*Filter*:

A filter expression specifying the records to return during the browse. An empty string indicates that every record will be returned. Multiple filters can be specified and separated with ';'. Where a fieldname is not specified in the filter, it is assumed to be tagname. For example, the filter "AAA" is equivalent to "name=AAA".

*Fields*:

Specifies via a comma delimited string the columns to be returned during the browse. An empty string indicates

that the server will return every available column. Supported fields are:

**NAME:** The name of the special day group.

**ID:** The unique ID of the special day group. Every special day group is assigned a unique ID.

*Clusters:*

An optional parameter that specifies via a comma delimited string the subset of the clusters to browse. An empty string indicates that the connected clusters will be browsed.

## Return Value

Returns an integer handle to the browse session. Returns -1 when an error is detected.

## Related Functions

[SchdSpecialClose](#), [SchdSpecialFirst](#), [SchdSpecialNext](#), [SchdSpecialPrev](#), [SchdSpecialGetField](#),  
[SchdSpecialNumRecords](#)

## See Also

[Scheduler Functions](#)

## SchdSpecialPrev

The SchdSpecialPrev places the data browse cursor at the previous record. This function uses iSession as the argument which is previously returned by the SchdSpecialOpen function. This function will return an error(EOF) if called when the data cursor is at beginning of the records.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **SchdSpecialPrev**(LONG *Session*)

*Session:*

Current special day group browsing session obtained by [SchdSpecialOpen](#).

## Return Value

0 (zero) if the browsing session exists, otherwise an error code is returned.

## Related Functions

[SchdSpecialOpen](#), [SchdSpecialFirst](#), [SchdSpecialNext](#), [SchdSpecialClose](#), [SchdSpecialGetField](#),  
[SchdSpecialNumRecords](#)

## See Also

[Scheduler Functions](#)

### ScheduleItemAdd

The ScheduleItemAdd function adds a new schedule to the scheduler engine. It returns the Id of the schedule which can be used for modifying, setting recurrence and deleting this schedule.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

```
LONG ScheduleItemAdd(STRING Cluster, STRING Equipment, STRING State, TIMESTAMP Start, TIMESTAMP End,  
STRING Desc)
```

*Cluster:*

The name of the cluster.

*Equipment:*

The name of the equipment to browse.

*State:*

The state of the schedule.

*Start:*

The start time of the schedule. The type of this parameter is TIMESTAMP. Use TimestampCreate or StrToTimestamp to get a TIMESTAMP value.

*End:*

The end time of the schedule. The type of this parameter is TIMESTAMP. Use TimestampCreate or StrToTimestamp to get a TIMESTAMP value.

*Desc*

The description of the schedule.

## Return Value

The id of the schedule which can be used for modifying, setting recurrence and deleting this schedule. This function returns -1 if unsuccessful. Trap the error to get the error returned by this function.

## Related Functions

[SchdClose](#), [SchdConfigOpen](#), [SchdConfigClose](#), [SchdConfigFirst](#), [SchdConfigNext](#), [SchdConfigPrev](#),  
[SchdConfigGetField](#), [SchdConfigNumRecords](#), [SchdOpen](#), [SchdFirst](#), [SchdNext](#), [SchdPrev](#), [SchdGetField](#),  
[SchdNumRecords](#), [ScheduleItemSetRepeat](#), [ScheduleItemModify](#), [ScheduleItemDelete](#)

## Example

```
//The following example creates a schedule item using state MyState1 for
MyEquipment1, that starts from 3am and ends at 4am on 20th of Nov 2011.
```

```
TIMESTAMP tsStartTime;
TIMESTAMP tsEndTime;
INT iScheduleID;
tsStartTime = TimestampCreate(2011, 11, 20, 3, 0, 0, 0);
tsEndTime = TimestampCreate(2011, 11, 20, 4, 0, 0, 0);
iScheduleID = ScheduleItemAdd("Cluster1",
"MyEquipment1","MyState1",tsStartTime,tsEndTime,"MyScheduleItem1");
```

## See Also

[Scheduler Functions](#)

### ScheduleItemDelete

The ScheduleItemDelete function deletes an existing schedule.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

**LONG ScheduleItemDelete(STRING Cluster, LONG ID)**

*Cluster:*

Name of cluster.

*ID:*

The unique ID of the schedule within the schedule engine.

## Return Value

Returns 0 if successful otherwise it returns an error.

## Related Functions

[SchdClose](#), [SchdConfigOpen](#), [SchdConfigClose](#), [SchdConfigFirst](#), [SchdConfigNext](#), [SchdConfigPrev](#),  
[SchdConfigGetField](#), [SchdConfigNumRecords](#), [SchdOpen](#), [SchdFirst](#), [SchdNext](#), [SchdPrev](#), [SchdGetField](#),  
[SchdNumRecords](#), [ScheduleItemAdd](#), [ScheduleItemSetRepeat](#), [ScheduleItemModify](#)

## See Also

[Scheduler Functions](#)

### ScheduleItemModify

The ScheduleItemModify function modifies an existing schedule.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

LONG **ScheduleItemModify**(STRING *Cluster*, LONG *ID*, STRING *State*, TIMESTAMP *Start*, TIMESTAMP *End*, STRING *Desc*)

*Cluster*:

Name of cluster.

*ID*:

The unique ID of the schedule within the schedule engine.

*State*:

The state of the schedule.

*Start*:

The start time of the schedule. The type of this parameter is TIMESTAMP. Use [TimestampCreate](#) or [StrToTimestamp](#) to get a TIMESTAMP value.

*End*:

The end time of the schedule. The type of this parameter is TIMESTAMP. Use [TimestampCreate](#) or [StrToTimestamp](#) to get a TIMESTAMP value.

*Desc*

The description of the schedule.

## Return Value

Returns 0 if successful otherwise it returns an error.

## Related Functions

[SchdClose](#), [SchdConfigOpen](#), [SchdConfigClose](#), [SchdConfigFirst](#), [SchdConfigNext](#), [SchdConfigPrev](#),  
[SchdConfigGetField](#), [SchdConfigNumRecords](#), [SchdOpen](#), [SchdFirst](#), [SchdNext](#), [SchdGetField](#), [SchdNumRecords](#),  
[ScheduleItemAdd](#), [ScheduleItemSetRepeat](#), [ScheduleItemDelete](#)

## See Also

[Scheduler Functions](#)

## ScheduleItemSetRepeat

The ScheduleItemSetRepeat function adds recurrence information to an existing schedule to the scheduler engine. The same function is used if the schedule already has recurrence information that need to be changed.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

LONG **ScheduleItemSetRepeat**(STRING *Cluster*, LONG *ID*, LONG *Freq*, LONG *Interval*, LONG *Weekday*, LONG *WeekDayMask*, LONG *MaxRec*, LONG *RecUntil*, LONG *DayOrd*, LONG *Day*, LONG *Month*, LONG *SpecialInc*, STRING

*GroupIds)*

*Cluster*

Name of cluster.

*ID*

The existing schedule id to apply recurrence.

*Freq:*

The type of the recurrence: 4 – daily, 5 – weekly, 6- monthly, 7 – yearly, 8 – special days.

*Interval:*

The interval of the recurrence. 1 – every 1 day/week/month etc., 2 – every second and so on.

*WeekDay:*

The first day of week – Sunday (0) or Monday (1).

*WeekDayMask:*

The day of week mask. Defines day of week where recurrence happens.

None = 0,

Sunday = 1,

Monday = 2,

Tuesday = 4,

Wednesday = 8,

Thursday = 16,

Friday = 32,

Saturday = 64,

Everyday = 127,

Weekdays = 62,

Weekendays = 65

Combination of days can be achieved by addition. For example, Tuesdays and Wednesdays would be 12 (4 + 8).

*MaxRec:*

The maximum number recurrences (-1 means does not stop).

*RecUntil:*

The time specifies the time until recurrence is occurring. The type of this parameter is TIMESTAMP. Use TimestampCreate or StrToTimestamp to get a TIMESTAMP value. To create a recurring schedule that does not end use TimestampCreate(1601, 1, 1, 0, 0, 0, 0, 1).

*DayOrd:*

Day ordinal. Applicable for monthly and yearly recurrence patterns. Can be 1-4 or -1 (every 1, second, third, fourth or last week of a month).

*Day:*

The day of month.

*Month:*

The month of the year.

*SpecialInc:*

Defined whether the special day is included in the pattern. 0 – none, 1 – every, 2 – selected.

*GroupIds* :

A list of group's ID included in the recurring pattern (used only when SpecialIncl is defined as 2 ("selected")).

## Return Value

Returns 0 if successful otherwise it returns an error.

## Related Functions

[SchdClose](#), [SchdConfigOpen](#), [SchdConfigClose](#), [SchdConfigFirst](#), [SchdConfigNext](#), [SchdConfigPrev](#),  
[SchdConfigGetField](#), [SchdConfigNumRecords](#), [SchdOpen](#), [SchdFirst](#), [SchdNext](#), [SchdPrev](#), [SchdGetField](#),  
[SchdNumRecords](#), [ScheduleItemAdd](#), [ScheduleItemModify](#), [ScheduleItemDelete](#)

## See Also

[Scheduler Functions](#)

## Screen Profile Functions

Following are functions relating to Screen Profiles:

<a href="#">ResetScreenProfile</a>	Use this function to move all top level windows back to their original screen starting position as defined by the screen profile.
------------------------------------	---

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## ResetScreenProfile

Returns all top level windows back to their original screen starting position as defined by the screen profile in a multi-monitor setup.

## Syntax

**ResetScreenProfile()**

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

### Example

```
ResetScreenProfile();
```

### See Also

[Screen Profile Functions](#)

## Security Functions

Following are functions relating to Security:

<a href="#">FullName</a>	Gets the full name of the current operator.
<a href="#">GetLanguage</a>	Gets the language currently used on the display client.
<a href="#">GetPriv</a>	Checks the privilege and area of the current operator.
<a href="#">Login</a>	Logs an operator into the Plant SCADA system. Not available when logged in as Windows user.
<a href="#">LoginForm</a>	Displays a form that allows an operator to log in to the Plant SCADA system.
<a href="#">Logout</a>	Logs an operator out of the Plant SCADA system.
<a href="#">LogoutIdle</a>	Sets an idle time logout for the current operator.
<a href="#">MultiSignatureForm</a>	Displays a form that allows up for 4 users to have their credentials verified in order to approve an operation.
<a href="#">MultiSignatureTagWrite</a>	Displays a form that allows up for 4 users to have their credentials verified in order to approve a write of a specific value to a specific tag.
<a href="#">Name</a>	Gets the user name of the current operator.
<a href="#">UserCreate</a>	Creates a new user record during run time. This function is now deprecated and will be removed in the next release. Use the <a href="#">UserCreateByRole</a> function instead.
<a href="#">UserCreateByRole</a>	Creates a record for a new Plant SCADA user. The user can be assigned to a specified role. Not available when logged in as Windows user.
<a href="#">UserCreateForm</a>	Displays a form to create a record for a new user. Not

	available when logged in as Windows user.
UserDelete	Deletes a new user record during run time. Not available when logged in as Windows user.
UserEditForm	Displays a form for a selected user to create or delete user records during run time. Not available when logged in as Windows user.
UserInfo	Gets information about the operator who is currently logged-in to the system.
UserLogin	Logs an operator into the Plant SCADA system using a secure password string.
UserPassword	Changes a user's password during run time. Not available when logged in as Windows user.
UserPasswordExpiryDays	Returns the number of days left before the user's password is due to expire. Not available when logged in as Windows user.
UserPasswordForm	Displays a form for the operator to change their own password during run time. Not available when logged in as Windows user.
UserSetStr	Sets the value of the given field for the given user record in the project configuration (users.dbf ) on the local machine.
UserUpdateRecord	Triggers a recompile of the local project configuration, then notifies the running system that user configuration has been modified and needs to be reloaded.
UserVerify	Uses the authentication functionality in the user login system.
VerifyPrivilegeForm	Displays a form that allows a single user to enter their credentials.
VerifyPrivilegeTagWrite	Displays a form that allows any single user to enter their credentials in order to approve a write of a specific value to a specific tag.
WhoAmI	Displays the name of the operator who is currently logged-in to the system.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## FullName

Gets the full name of the user who is currently logged on to the system. The user can be a Plant SCADA or a Windows user. For a Plant SCADA user the full name is the one defined in the user settings. For a Windows user the full name is in the format of <DomainName>\<UserName>. When there is no one logged in or the logged in user is a "system user" this function returns an empty string.

## Syntax

**FullName()**

If the user is logged on as a Domain user the name should be the Windows domain name and user account name in the format of <DomainName\UserName>.

If the user is logged on as a local user the name should be the local machine name and user account name in the format of <MachineName\UserName>.

## Return Value

The user name (as a string).

## Related Functions

[NameUserInfo](#)

## Example

```
/* Display the full name of the current user at AN20. */  
DspText(20, 0, FullName());
```

## See Also

[Security Functions](#)

## GetLanguage

Gets the language currently used on the display client.

## Syntax

STRING **GetLanguage(INT iType)**

*iType*

The type of language information to be returned:

0 (default) — the name of the current language as defined in the project. For example, "English (United States)" or "French (France)".

1 — the country language code of the current language. Includes the language code (ISO-639) and country code (ISO-3166). For example:

- "en-US" for English (United States)
- "fr-FR" for French (France).

## Return Value

The specified information describing the language that is currently in use.

## Related Functions

[Login](#), [LoginForm](#), [UserLogin](#)

## Example

```
// login a user -> Login("ConfigUsers","SCADA",1,"German(Germany)")  
STRING sLanguage;  
sLanguage = GetLanguage(0) // will return German(Germany)  
sLanguage = GetLanguage(1) // will return de-DE
```

## See Also

[Security Functions](#)

## GetPriv

Checks if the current user has a privilege for a specified area. With this function, you can write your own Cicode functions to control user access to the system.

## Syntax

**GetPriv(*Priv*, *Area*)**

*Priv*

The privilege level (1..8).

*Area*

The area of privilege (0..255).

## Return Value

Returns 1 if the user has the specified privilege in the area, or 0 (zero) if the user does not have the privilege.

## Related Functions

[SetArea](#)

## Example

```
/* User needs to have privilege 2, or cannot do operation. */
IF GetPriv(2, 0) THEN
    ! Do operation here
ELSE
    Prompt("No privilege for command");
END
```

## See Also

[Security Functions](#)

## Login

Not available when logged in as Windows user.

Logs a user into the Plant SCADA system, using Plant SCADA security and gives users access to the areas and privileges assigned to them in the Users database, and uses the locale language defined for that user. Only one user can be logged into a computer at any one time. If a user is already logged in when a second user logs in, the first user is automatically logged out.

When using Windows security use [UserLogin](#) to limit Windows credentials being exposed as plain text.

At startup, or when the user logs out, a default user is active, with access to area 0 (zero) and privilege 0 (zero) only. Use the [LoginForm\(\)](#) function to display a form for logging in to the system.

## Syntax

**Login**(*sUserName*, *sPassword* [, *bSync*] [, *sLanguage*])

*sUserName*:

The user's name, as defined in the Users database.

*sPassword*:

The user's password, as defined in the Users database.

*bSync*:

Optionally specifies whether the function operates in blocking or non-blocking mode. If set to 1 blocks caller until login is complete. If set to 0 (default) does not block caller.

*sLanguage*:

The specified language needs to be one of the languages defined in the [Languages](#) view in Plant SCADA Studio's [Setup](#) activity. If the specified language is undefined, the default language is used by the login user, and a message "Undefined language" is shown in the prompt line.

An empty string (i.e. "") can be specified to indicate that the default language is used by the login user. The default language defined by [\[Language\]LocalLanguage](#) INI parameter.

The default value of this parameter is "".

## Return Value

0 (zero) if successful.

## Related Functions

[LoginForm](#), [Logout](#), [LogoutIdle](#), [Message](#), [Input](#), [UserLogin](#)

## Example

```
/*
** Log in a user whose user name is "FRED" and whose password
** is "ABC".
**
** The language of the logged in user session will be set to
** the default language.
*/
Login("FRED", "ABC");

/*
** Log in a user whose user name is "FRED" and whose password
** is "ABC".
**
** The language of the logged in user session will be set to
** English.
*/
Login("FRED", "ABC", 0, "English");

/*
** Log in a user whose user name is "FRED" and whose password
** is "ABC".
**
** The language of the logged in user session will be set to
** French.
*/
Login("FRED", "ABC", 0, "French(France)");

/*
** Log in a user whose user name is "FRED" and whose password
** is "ABC".
**
** The language of the logged in user session will be set to
** Chinese.
*/
Login("FRED", "ABC", 0, "Chinese(Simplified, PRC)");
```

## See Also

[Security Functions](#)

### LoginForm

Displays a form in which a user can log in to the Plant SCADA system by entering their name and password and local language. If the login is correct, the user is logged into the Plant SCADA system with the area(s) and privilege(s) assigned to them in the Users database.

---

**Note:** You can only call this function on a client process.

From version 7.10 this form can be pre-filled by the caller. Both Plant SCADA and Windows users are supported.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

## Syntax

**LoginForm([*sUserName* [, *sPassword* [, *sLanguage*]])**

*sUserName*:

Optionally, the user's name, as defined in the Users database.

*sPassword*:

Optionally, the user's password, as defined in the Users database.

*sLanguage*:

The specified language must be one of the languages defined in the **Languages** view in Plant SCADA Studio's **Setup** activity. If the specified language is undefined, the default language is used by the login user, and a message "Undefined language" is shown in the prompt line.

An empty string (i.e. "") can be specified to indicate that the default language is used by the login user. The default language defined by [\[Language\]LocalLanguage](#) INI parameter.

The default value of this parameter is "".

---

**Note:** If the language is changed a page reload will occur.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[Login](#), [UserLogin](#)

## Example

System Keyboard

Key Sequence	Login
LoginForm	Display the Login form
Comment	Allow user login

#### Buttons

Text	Operator Login
LoginForm	Display the Login form
Comment	Allow user login

## See Also

[Security Functions](#)

## Logout

Logs the current user out of the Plant SCADA system. Plant SCADA continues to run, but with access to area 0 (zero) and privilege 0 (zero) only. If the current page requires access for a specified area (as defined by the page's area property), the system returns to the home page as specified by the parameter [\[Page\]HomePage](#), and if unsuccessful that returns to the startup page. When multiple pages are currently displayed, this occurs for each open window.

Calling this function to logout the logged on user will cause an automatically logged in user to be logged back on. If there is no user logged in, calling this function will return an error. When the logged on user is an automatically logged in user, calling this function will return an error.

## Syntax

**Logout()**

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[Login](#), [LoginForm](#), [LogoutIdle](#), [UserInfo](#), [Message](#), [Input](#)

## Example

```
/* Log the current user out of the system. */  
Logout();
```

## See Also

[Security Functions](#)

### LogoutIdle

Sets an idle time for logging out the current user. If the current user does not execute a command within the specified idle time, a prompt is displayed. If the prompt is ignored, then the user is logged out. For every user to have the same idle time, you would call this function at startup. Otherwise, you can call the function from the Users database to specify an idle time for each user. This function will not log out an automatically logged on user.

Until reset LogoutIdle remains active. To reset call LogoutIdle (-1) from the Exit command of the Users database record.

## Syntax

**LogoutIdle(*Idle*)**

*Idle*:

The number of minutes the user needs to be idle before logout will occur. Set Idle to -1 to disable the current logout timeout.

## Return Value

No return value.

## Related Functions

[Logout](#), [Login](#), [LoginForm](#)

## Example

Users

User Name	Operator1
LoginForm	LogoutIdle(5)
Comment	Logs the user out after five minutes

## See Also

[Security Functions](#)

## MultiSignatureForm

Displays a form that allows up to four users to have their credentials verified in order to approve an operation. The usernames can be Plant SCADA or Windows users.

## Syntax

**MultiSignatureForm**(*sOperationDescription*, *sLogDevice*, *sUser1*, *sUser2*, *sUser3*, *sUser4*)

*sOperationDescription*:

A description of the operation that requires approval. This string will be displayed on the signature form and logged to the log device if the operation is approved.

*sLogDevice*:

The name of a log device if logging is required, otherwise pass an empty string.

*sUser1..4*:

Each *sUser* argument needs to be either a Plant SCADA user name, a Windows user name (including domain\ prefix) or a blank string. Even though the *sUser* arguments are numbered 1 through 4, this only controls the order in which users are displayed on the multi-signature form. You can pass empty strings for any of these arguments, but at least one user needs to be specified.

## Return Value

TRUE (1) if the operation approved (that is all users' credentials were verified and the operator clicked the "Approve" button, otherwise FALSE (0).

## Related Functions

[FormSecurePassword](#), [MultiSignatureTagWrite](#), [VerifyPrivilegeForm](#), [VerifyPrivilegeTagWrite](#)

## Example

```
// This example sets the page integer to indicate the approval status, but
// it can be used to perform any logic necessary to trigger the operation
// that was approved.
PageSetInt(1, MultiSignatureForm("Shut down plant", "ApprovalLog", "shiftsupervisor",
"DOMAIN\mike.manager", "", ""));
```

## See Also

[Security Functions](#)

[Form Functions](#)

## MultiSignatureTagWrite

Displays a form that allows up to four users to have their credentials verified in order to approve a write of a specific value to a specific tag. If all users are verified successfully, the write to the tag is performed by this

function before it returns. The usernames can be Plant SCADA or Windows users.

## Syntax

**MultiSignatureTagWrite(*sTagName*, *sValueToWrite*, *sLogDevice*, *sUser1*, *sUser2*, *sUser3*, *sUser4*)**

*sTagName*:

The name of the tag to which a write needs to be approved.

*sValueToWrite*:

The value to write to the tag if approval succeeds.

*sLogDevice*:

The name of a log device if logging is required, otherwise pass an empty string.

*sUser*:

Each *sUser* argument needs to be either a Plant SCADA user name, a Windows user name (including domain\prefix) or an empty string. Even though the *sUser* arguments are numbered 1 through 4, this only controls the order in which users are displayed on the multi-signature form. You can pass empty strings for any of these arguments, but at least one user needs to be specified.

## Return Value

TRUE (1) if the operation was approved (that is all users' credentials were verified and the operator clicked the "Approve" button, otherwise FALSE (0).

## Related Functions

[FormSecurePassword](#), [MultiSignatureForm](#), [VerifyPrivilegeForm](#), [VerifyPrivilegeTagWrite](#)

## Example

```
// This example generates a form to request two users to approve the tag write operation.  
// When approved, the PLC_VAR1 tag is written with the value 123 and a page string  
// is set to indicate the approval status.  
IF  
  MultiSignatureTagWrite("PLC_VAR1", "123", "", "John Smith", "Angela Huth", "", "")  
THEN  
  PageSetStr(1, "TagWrite Successful");  
ELSE  
  PageSetStr(1, "TagWrite Not Successful");  
END
```

## See Also

[Security Functions](#)

[Form Functions](#)

## Name

Gets the name of the operator who is currently logged on to the display system. The user can be a Plant SCADA or a windows user. If this function is called on a server, it returns the name of the local operator. If there is no one logged on, or the logged on user is a "system user" this function returns an empty string.

## Syntax

**Name**([*bIncludeDomain*])

*bIncludeDomain*:

- *bIncludeDomain* = 0: Return Windows user name only
- *bIncludeDomain* = 1: (Default) Return Windows user name with domain name

---

**Note:** This parameter is ignored when a normal Plant SCADA account.

---

## Return Value

The name of the user as a string. If the user is logged on as a Windows user the name will be the Windows user account name.

## Related Functions

[FullName](#), [Login](#), [LoginForm](#)

## Example

```
/* Display the user name of the current user at AN20. */  
DspText(20,0,Name());
```

## See Also

[Security Functions](#)

## UserCreate

This function is now deprecated and will be removed in the next release. Use the [UserCreateByRole](#) function instead.

Not available for a Windows user.

Creates a record for a new Plant SCADA user. The name of the user needs to be unique.

The user can be of a specified **Type** or assigned to a specified **Role**, depending on the current setting for the parameter [\[Security\]CreateUserByRole](#).

---

**Note:** Only users assigned an administrator role can create a user. Other users attempting to run this will encounter an error.

An alarm server needs to be configured and running in the project, so that all clients are notified that an online change to the users configuration has been completed.

In order to perform user configuration changes online in a system with multiple computers running SCADA nodes using these functions, you will need to use the RUN and COPY parameters to check the updates are distributed throughout the system, and that the functions are called from the computer which uses the COPY path as its RUN path.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**UserCreate(*sName*, *sFullName*, *sPassword*, *sType/sRole*)**

*sName*:

The name of the user.

*sFullName*:

The full name of the user.

*sPassword*:

The password of the user.

The *sPassword* argument is optional. If not passed, this argument defaults to an empty string which is subsequently ignored. It is included for the purposes of handling duplicate user names and separate password identification compatibility.

*sType*:

The generic type of user. The type needs to be defined in the general properties for a user in Plant SCADA Studio's Security activity (see [Add a Plant SCADA User](#)).

*sRole*:

The role to which the user will be assigned. The role needs to be defined in Plant SCADA Studio's Security activity (see [Roles](#)).

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[UserDelete](#), [UserEditForm](#), [UserPassword](#), [UserPasswordForm](#), [UsercreateForm](#)

## Example

```
/* Create a new user */
UserCreate("Fred", "Fred Jones", "secret", "Operator");
```

## See Also

[Security Functions](#)

## UserCreateByRole

This function is not available for a Windows user.

Creates a record for a new Plant SCADA user. The user can be assigned to a specified role (see [Roles](#)). The name of the user needs to be unique.

---

**Note:** Only users assigned an administrator role can create a user. Other users attempting to run this will encounter an error.

---

An alarm server needs to be configured and running in the project, so that all clients are notified that an online change to the users configuration has been completed.

In order to perform user configuration changes online in a system with multiple computers running SCADA nodes using these functions, you will need to use the RUN and COPY parameters to check the updates are distributed throughout the system, and that the functions are called from the computer which uses the COPY path as its RUN path.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**UserCreateByRole(*sName*, *sFullName*, *sPassword*, *sRole*)**

*sName*:

The name of the user.

*sFullName*:

The full name of the user.

*sPassword*:

The password of the user.

The *sPassword* argument is optional. If not passed, this argument defaults to an empty string which is subsequently ignored. It is included for the purposes of handling duplicate user names and separate password identification compatibility.

*sRole*:

The role(s) to which the user will be assigned. If required, you can enter multiple roles as a comma separated list. The role(s) you specify need to be defined in Plant SCADA Studio's **Security** activity (see [Roles](#)).

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[UserDelete](#), [UserEditForm](#), [UserPassword](#), [UserPasswordForm](#), [UsercreateForm](#)

## Example

```
/* Create a new user */  
UserCreateByRole("Fred", "Fred Jones", "secret", "Operator");
```

## See Also

[Security Functions](#)

### UserCreateForm

Not available for a Windows user.

Displays a form to create a record for a new Plant SCADA user. The name of the user needs to be unique. The form allows the user to be assigned to a specified .

An alarm server needs to be configured and running in the project, so that all clients are notified that an online change to the users configuration has been completed.

In order to perform user configuration changes online in a system with multiple computers running SCADA nodes using these functions, you will need to use the RUN and COPY parameters to check the updates are distributed throughout the system, and that the functions are called from the computer which uses the COPY path as its RUN path.

## Syntax

`UserCreateForm()`

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[UserDelete](#), [UserEditForm](#), [UserPassword](#), [UserPasswordForm](#), [UserCreate](#)

## Example

```
UserCreateForm()
```

## See Also

[Security Functions](#)

### UserDelete

Not available for a Windows user.

Deletes the record for a user. Changes are written to both the Users database and the runtime database in memory.

---

**Note:** Only those users assigned a role with "Manage User" privilege set to True can delete a user. Other users attempting to run will encounter an error.

---

An alarm server needs to be configured and running in the project, so that all clients are notified that an online

change to the users configuration has been completed.

In order to perform user configuration changes online in a system with multiple computers running SCADA nodes using these functions, you will need to use the RUN and COPY parameters to check the updates are distributed throughout the system, and that the functions are called from the computer which uses the COPY path as its RUN path.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**UserDelete(*sName*)**

*sName*:

The name of the user.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[UserCreate](#), [UserEditForm](#)

## Example

```
/* Delete Fred from the database */  
UserDelete("Fred");
```

## See Also

[Security Functions](#)

## UserEditForm

Not available for a Windows user.

Displays a form to allow the user to create or delete any user record in the database. This function should have restricted access. Changes are written to both the Users database and the runtime database in memory.

An alarm server needs to be configured and running in the project, so that all clients are notified that an online change to the users configuration has been completed.

In order to perform user configuration changes online in a system with multiple computers running SCADA nodes using these functions, you will need to use the RUN and COPY parameters to check the updates are distributed throughout the system, and that the functions are called from the computer which uses the COPY path as its RUN path.

## Syntax

**UserEditForm()**

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[UserCreate](#), [UserDelete](#)

## Example

```
/* Display a form for the user to create or delete user records. */  
UserEditForm();
```

## See Also

[Security Functions](#)

## UserInfo

Gets information about the operator who is currently logged-in to the system.

## Syntax

**UserInfo(*nType*)**

*nType*:

The type of user information:

0 - Flag to indicate whether any user other than a view-only user is logged in

1 - The login name of the user

2 - The full name of the user

3 - The time the user logged in

4 - The time the user entered the last command

5 - The number of commands entered by the user

6 - The type of login:

- "0" indicates that the current user is a view-only user.

- "1" indicates there is Plant SCADA or Windows non-default user explicitly logged in.

- "2" indicates the logged on user is a Plant SCADA default user (control client auto login Plant SCADA user).

- "3" indicates the logged on user is a Windows default user (control client auto login windows user).

## Return Value

The information (as a string). If an error is detected, an empty string is returned.

## Related Functions

[Login](#)

## Example

```
/* Check if a user has logged on. If so, get the user's full name and the number of
commands they have performed. */
String sName;
String sCount;
IF UserInfo(0) = "1" THEN
    sName = UserInfo(2);
    sCount = UserInfo(5);
END
```

## See Also

[Security Functions](#)

## UserLogin

Logs a user into the Plant SCADA system, using either Windows integrated security or Plant SCADA security and gives users access to the areas and privileges assigned to them in the Users database, and uses the locale language defined for that user. Only one user can be logged into a computer at any one time. If a user is already logged in when a second user logs in, the first user is replaced by the newly logged on user. When a newly logged in user does not have access to view the current page (as defined by the page's area), the system returns to the home page as specified by the parameter[Page]HomePage, and if unsuccessful that returns to the startup page. When multiple pages are currently displayed, this occurs for each open window.

To call this function at user login, the password argument passed needs to be in secure string format.

At startup, or when the user logs out, a default user is active, with access to area 0 (zero) and privilege 0 (zero) only. Use the [LoginForm\(\)](#) function to display a form for logging in to the system.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**UserLogin(sUserName, sPassword, [sLanguage])**

**sUserName:**

The user's name as defined in the Users database, or the Windows User account name, in plain text.

**sPassword:**

The user's password, as defined in the Users database or Windows account formatted as a secure string.

To improve the user credentials protection provides a system built-in user login function that takes the user

name and secure password as the arguments. This reduces the chance that the user's password can be exposed in plain text from the runtime system

*sLanguage:*

The specified language needs to be one of the languages defined in the Languages view in Plant SCADA Studio's Setup activity. If the specified language is undefined, the default language is used by the login user, and a message "Undefined language" is shown in the prompt line.

An empty string (i.e. "") can be specified to indicate that the default language is used by the login user. The default language defined by [\[Language\]LocalLanguage](#) INI parameter.

The default value of this parameter is "".

## Return Value

0 (zero) if successful.

## Related Functions

[LoginForm](#), [Logout](#), [LogoutIdle](#), [Message](#), [Input](#)

## Example

```
/*
** FUNCTION NAME:LoginForm
**
** This function displays the login form, get the user name, password and
** the language of the user session and then trys to log the user in.
** If the login does not succeed it will retry until login is ok or user
** presses the cancel button.
*/
INT
FUNCTION
LoginForm(STRING sName="", STRING sPassword="", STRING sLanguage="English")
INT bDone;
INT nStatus;
INT hForm;
bDone = FALSE;
WHILE bDone = FALSE DO
    FormNew("@(Login Form)", 35, 5, 5);
    FormPrompt(1, 0, "@(Name)");
    FormInput(16, 0, "", sName, 16);
    FormPrompt(1, 2, "@(Password)");
    FormSecurePassword(18, 2, "", sPassword, 16);
    FormPrompt(1, 4, "@(Language)");
    FormInput(16, 4, sLanguage, 16);
    FormButton(6, 6, " " + "@(OK)" + " ", 0, 1);
    FormButton(20, 6, "@(Cancel)", 0, 2);
IF FormRead(0) = 0 THEN
    hForm = FormNew("@(User Login)", 36, 1, 8 + 16 + 128 + 256);
    FormPrompt(1, 0, "@(Authentication in progress ...)");
    FormRead(1);
    SleepMs(200);
```

```
IF UserLogin(sName, sPassword, sLanguage) = 0 THEN
    bDone = TRUE
    nStatus = 0;
ELSE
    sPassword = "";
END
IF FormActive(hForm) THEN
    FormDestroy(hForm);
END
ELSE
    bDone = TRUE;
    nStatus = 298;
END
END
RETURN nStatus;
END
```

## See Also

[Security Functions](#)

### UserPassword

Not available for a Windows user.

Changes the password for the user. Changes are written to both the Users database and the runtime database in memory.

An alarm server needs to be configured and running in the project, so that all clients are notified that an online change to the users configuration has been completed.

If you use this function to perform user configuration changes across an online system with multiple computers running SCADA nodes, you will need to use the [\[CtEdit\]Run](#) and [\[CtEdit\]Copy](#) parameters to distribute the updated files throughout the system.

You will also need to execute the UserPassword function on:

- The computer where the master/source Users.dbf and \_Users.rdb files are located.  
Or:
- The computer to which [CtEdit]Copy is pointing.  
Or:
- The computer from which [CtEdit]Copy is copying files.

UserPassword can only work for users that are contained in the main project.

---

**Note:** Only those users assigned a role with "Manage User" privilege set to True can set a user password without knowing the existing password. Other users attempting to run will encounter an error.

---

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**UserPassword(*sName* [, *sPassword*] [, *sOldPassword*] )**

*sName:*

The name of the user.

*sPassword:*

The password of the user.

The *sPassword* argument is optional. If not passed, this argument defaults to an empty string which is subsequently ignored. It is included for the purposes of handling duplicate user names and separate password identification compatibility.

*sOldPassword:*

The password assigned to the user before the `UserPassword()` function is run.

The *sOldPassword* argument is optional. If passed, Plant SCADA will only permit the password change (and consequent re-setting of the expiry period) when the old password is correctly entered. If the *sOldPassword* parameter is not passed, the password change will proceed without restriction.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[UserPasswordForm](#), [UserCreate](#), [UserEditForm](#)

## Example

```
/* Change Fred's password */  
UserPassword("Fred", "secret");
```

## See Also

[Security Functions](#)

## UserPasswordExpiryDays

Not available for a Windows user.

Returns the number of days left before the user's password is due to expire.

To use this function, you can build a form page by using Cicode that takes the user name and password as inputs and output the number of days that return by `UserPasswordExpiryDays`.

## Syntax

**UserPasswordExpiryDays(*sUserName* [, *sPassword*] )**

*sUserName:*

The name of the user.

*sPassword:*

The password of the user.

The *sPassword* argument is optional. If not passed, this argument defaults to an empty string which is subsequently ignored. It is included for the purposes of handling duplicate user names and separate password identification compatibility.

## Return Value

The return value contains either the number of days before password expiry, or one of two exception conditions:

- 0 to 365 - number of days
- -1 - no expiry
- -2 - user not found or password wrong

## Related Functions

[UserPassword](#)

## See Also

[Security Functions](#)

## UserPasswordForm

Not available for a Windows user.

Display a form to allow users to change their own passwords. Changes are written to both the Users database and the runtime database in memory.

An alarm server needs to be configured and running in the project, so that all clients are notified that an online change to the users configuration has been completed.

In order to perform user configuration changes online in a system with multiple computers running SCADA nodes using these functions, you will need to use the RUN and COPY parameters to check the updates are distributed throughout the system, and that the functions are called from the computer which uses the COPY path as its RUN path.

UserPassword can only work for users that are contained in the main project.

## Syntax

**UserPasswordForm()**

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[UserPassword](#), [UserCreate](#), [UserEditForm](#)

## Example

```
/* Allow users to change their own passwords */  
UserPasswordForm();
```

## See Also

[Security Functions](#)

### UserSetStr

Sets the value of the given field for the given user record in the project configuration (users.dbf ) on the local machine.

After this function has been called, use the function [UserUpdateRecord](#) to update the user record on the running system.

An alarm server needs to be configured and running in the project, so that all clients are notified that an online change to the users configuration has been completed.

In order to perform user configuration changes online in a system with multiple computers running SCADA nodes using these functions, you will need to use the RUN and COPY parameters to check the updates are distributed throughout the system, and that the functions are called from the computer which uses the COPY path as its RUN path.

---

**Note:** Only those users assigned the role with "Manage User" privilege set to True can update other users details. The current logged in user can only update their own details. Other users attempting to run will encounter an error.

---

## Syntax

**UserSetStr**(*sName*, *sField*, *sData*)

*sName*:

The name of the user who's configuration record we wish to modify.

*sField*:

The name of the field in users.dbf to modify.

*sData*:

The new value of the field.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[UserDelete](#), [UserEditForm](#), [UserPassword](#), [UserPasswordForm](#), [UserCreateForm](#)

## Example

```
UserSetStr("Fred", "Comment", "Fred is an engineer");
UserUpdateRecord();
```

## See Also

[Security Functions](#)

## UserUpdateRecord

Triggers a recompile of the local project configuration, then notifies the running system that user configuration has been modified and needs to be reloaded.

Use this function in conjunction with [UserSetStr](#) to modify user configuration online.

An alarm server needs to be configured and running in the project, so that all clients are notified that an online change to the users configuration has been completed.

In order to perform user configuration changes online in a system with multiple computers running SCADA nodes using these functions, you will need to use the RUN and COPY parameters to check the updates are distributed throughout the system, and that the functions are called from the computer which uses the COPY path as it's RUN path.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

```
UserUpdateRecord()
```

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[UserDelete](#), [UserEditForm](#), [UserPassword](#), [UserPasswordForm](#), [UserCreateForm](#)

## Example

```
UserSetStr("Fred", "Comment", "Fred is an engineer");
UserUpdateRecord();
```

## See Also

[Security Functions](#)

### UserVerify

Verifies a given user by authenticating the user's credential, verifies the user privileges and areas against those specified in the functions parameters. Successful verification however does not log the user into the Plant SCADA runtime system.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**UserVerify**(*sName*, *sPassword* [, *sAccess*] [, *sPrivGlobal*] [, *sPriv1..sPriv8*] )

*sName*:

The name of the user.

*sPassword*:

The password of the user. The *sPassword* argument needs to be passed as a secure string.

*sAccess*:

Specifies the required user's viewable areas.

*sPrivGlobal*:

Specifies the required user's global privilege.

*sPriv1-8*:

Specifies the required areas for privileges 1 - 8. That is, *sPriv1* contains the areas (1,2,3,4,...,255) where the user has Privilege 1.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

The successful verification has to meet the following conditions:

- The selected user credentials can be authenticated
- The required user privileges are included in the authenticated user's total privileges.

## Related Functions

[UserDelete](#), [UserEditForm](#), [UserPassword](#), [UserPasswordForm](#), [UserCreateForm](#)

## Example

```
INT FUNCTION UserVerifyTest()
  STRING sName;
  STRING sPassword;
  INT bDone;
```

```
INT nStatus;
bDone = FALSE;
WHILE bDone = FALSE DO
    FormNew("@(Login Form)", 30, 5, 5);
    FormInput(1, 0, "@(Name)" + " ", sName, 16);
    FormSecurePassword(1, 2, "@(Password)" + " ", sPassword, 16);
    FormButton( 1, 4, " " + "@(OK)" + " ", 0, 1);
    FormButton(17, 4, "@(Cancel)", 0, 2);
    IF FormRead(0) = 0 THEN
        IF UserVerify(sName, sPassword) = 0 THEN
            bDone = TRUE;
            nStatus = 0;
            Message("Info", "Verification successful", 0)
        ELSE
            sPassword = "";
            Message("Info", "Verification not successful", 0)
        END
    ELSE
        bDone = TRUE;
        nStatus = 298;
    END
END
RETURN nStatus;
END
```

## See Also

[Security Functions](#)

### VerifyPrivilegeForm

Displays a form that allows a single user to enter their credentials. These credentials are checked against a specified set to ensure the user has the required privileges before allowing the operation to proceed.

The user can be a Plant SCADA or Windows user.

## Syntax

**VerifyPrivilegeForm(*sOperationDescription*,*sLogDevice*, *sAccess*, *sGlobalPriv*, *sPriv1*, *sPriv2*, *sPriv3*, *sPriv4*, *sPriv5*,  
*sPriv6*, *sPriv7*, *sPriv8*)**

*sOperationDescription*:

A description of the operation that requires approval.

*sLogDevice*:

The name of a log device if logging is required, otherwise pass an empty string.

*sAccess*:

The required user viewable areas, or pass an empty string for none.

*sGlobalPriv*:

The required global privilege, otherwise pass an empty string.

*sPriv1..8*:

Specifies the required areas for privileges 1 - 8. That is, *sPriv1* contains the areas (1,2,3,4,...,255) where the user has Privilege 1. Each argument needs to be either specified or an empty string for none.

## Return Value

The name of the user that met the required privileges, otherwise ""

## Related Functions

[FormSecurePassword](#), [MultiSignatureForm](#), [MultiSignatureTagWrite](#), [VerifyPrivilegeTagWrite](#)

## Example

```
// This example generates a form to request a user to approve an operation.  
// This user needs the global privilege level of 8.  
// When approved, the operation is completed and a page string  
// is set to indicate the approval status.  
IF (VerifyPrivilegeForm("Shut Down Plant", "ApprovalLog",  
"PlantWide", "8", "", "", "", "", "", "", "")<>"") THEN  
    // Do operation  
    PageSetStr(1, "Operation approved");  
ELSE  
    PageSetStr(1, "Operation not approved");  
END
```

## See Also

[Security Functions](#)

[Form Functions](#)

## VerifyPrivilegeTagWrite

Displays a form that allows any single user to enter their credentials in order to approve a write of a specific value to a specific tag. These credentials are checked against a specified set to ensure the user has the required privileges before allowing the operation to proceed.

The usernames can be Plant SCADA or Windows users.

## Syntax

**VerifyPrivilegeTagWrite**(*sTagName*, *sValueToWrite*, *sLogDevice*, *sAccess*, *sGlobalPriv*, *sPriv1*, *sPriv2*, *sPriv3*,  
*sPriv4*, *sPriv5*, *sPriv6*, *sPriv7*, *sPriv8*)

*sTagName*:

The name of the tag to which a write needs to be approved.

*sValuetowrite*:

The value to write to the tag if approval succeeds.

*sLogDevice*:

The name of a log device if logging is required, otherwise pass an empty string.

*sAccess:*

The required user viewable areas, or pass an empty string for none.

*sGlobalPriv:*

The required global privilege, otherwise pass an empty string.

*sPriv1..8:*

Specifies the required areas for privileges 1 - 8. That is, *sPriv1* contains the areas (1,2,3,4,...,255) where the user has Privilege 1. Each argument needs to be either specified or an empty string for none.

## Return Value

Name of user that met the required privileges (and therefore the value was written to the specified tag), otherwise ""

## Related Functions

[FormSecurePassword](#), [MultiSignatureForm](#), [MultiSignatureTagWrite](#), [VerifyPrivilegeForm](#)

## Example

```
// This example generates a form to request a user to approve the tag write operation.  
// This user needs privilege levels of 6 and 3.  
// When approved, the PLC_VAR1 tag is written with the value 123 and a page string  
// is set to indicate the approval status.  
IF (VerifyPrivilegeTagWrite("PLC_VAR1", "123", "ApprovalLog",  
"PlantWide", "", "6", "3", "", "", "", "", "")<>"") THEN  
    PageSetStr(1, "TagWrite Successful");  
ELSE  
    PageSetStr(1, "TagWrite Not Successful");  
END
```

## See Also

[Security Functions](#)

[Form Functions](#)

## WhoAmI

Displays the user name and full name of the user currently logged-in to the system. When the current logged on user is Windows user this function returns the user's full name in the format of <DomainName>\<UserName>. The names are displayed at the prompt AN.

## Syntax

**WhoAmI()**

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[Name](#), [FullName](#), [UserInfo](#)

## Example

```
/* Display the user's full name and user name at the prompt AN. */  
WhoAmI();
```

## See Also

[Security Functions](#)

## Sequence of Events (SOE) Functions

Following are functions relating to sequence of events:

<a href="#">SOEventAdd</a>	Inserts a new event into the event journal
<a href="#">SOEArchive</a>	Archives event journals
<a href="#">SOEMount</a>	Mounts archive volume
<a href="#">SOEDismount</a>	Dismount archive volume

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## SOEArchive

Use this function to archive event journals. To use this function you need to log in with a user assigned to a role that has full privileges (privilege 1..8).

**Note:** When you call this function, archiving will commence according to the values defined for the Plant SCADA parameters "ArchiveAfter" and "KeepOnlineFor". For more information, see the topic [Configure the Archiving Parameters](#) in the main help.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT SOEArchive(STRING Path[, STRING ClusterName])**

*Path: (optional)*

The path set for SOEArchive (for example, "C:\temp\SOEArchive").

If a path is not entered, the default archive path for the alarm server configuration will be used.

*sClusterName:*

Optional cluster on which to perform mounting operation. If not specified, the operation will be performed on all clusters.

---

**Note:** The volume where data is to be saved should be labeled. This includes any removable storage devices. If this is not the case, archiving will not succeed and a message will display on the SOE page.

---

## Return Value

0 (zero) if successful, otherwise an error code is returned.

### Examples

SOEArchive(<path>, <ClusterName>) <-- Archiving specific Cluster of SOE and Summary Data to a specific location

SOEArchive("", <ClusterName>) <-- Archiving specific Cluster of SOE and Summary Data to the default Archive Path defined in Alarm Server configuration (form)

SOEArchive(<path>) <-- Archiving all Clusters of SOE and Summary Data to a specific location

SOEArchive() <-- Archiving all Clusters of SOE and Summary Data to the default Archive Path defined in Alarm Server configuration (form)

## See Also

[Sequence of Events \(SOE\) Functions](#)

## SOEDismount

Use this function to dismount archive volume. To use this function you will need to log in with a role (e.g.Engineer) who has full privilege (privilege 1..8).

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT SOEDisMount([, STRING ClusterName])**

*sClusterName:*

Optional cluster on which to perform mounting operation. If not specified, the operation will be performed on all clusters.

## Example

```
SOEDismount() <-- Dismounting all Clusters of SOE and Summary Data  
SOEDismount(<ClusterName>) <-- Dismounting specific Cluster of SOE and Summary Data
```

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## See Also

[Sequence of Events \(SOE\) Functions](#)

## SOEventAdd

Use this function to insert an event into the Event Journal.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

```
INT SOEventAdd(TIMESTAMP TimeStamp, STRING Message[,STRING Tag, STRING Cluster])
```

*TimeStamp*:

The time of the inserted event.

*Message*:

The message for the event.

*Tag*:

Alarm tag associated with the event. This can be Cluster.Tag or Tag if running on a single cluster system. If not specified the event is classed as a 'User' event.

*Cluster*:

Specify if running on a multi-cluster system and Tag has been specified. If Tag has not been specified and the Cluster is blank, the User event will be broadcast to all of the clusters to which the client is currently connected.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

---

**Note:** If you have installed Plant SCADA on an English operating system and want to add events messages at runtime using a Unicode language (such as Korean, Russian or Chinese), you will need to change the Windows™ Region setting for the runtime computer. To do this, go to **Control Panel** and open the **Region** dialog box. On the **Administrative** tab, use the **Change system locale** button to select the required system locale. Be aware that you will have to restart your computer. When you launch runtime, select the matching **Language** on the login form. Your event messages will be recorded using the specified language characters.

---

## See Also

[Sequence of Events \(SOE\) Functions](#)

## SOEMount

Use this function to mount an archive volume. Archived data from the volume can be displayed on the Alarm Summary and Sequence of events page. To use this function you will need to log in with a role (e.g.Engineer) who has full privilege (privilege 1..8).

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

`INT SOEMount(STRING sPath [, STRING ClusterName])`

*sClusterName:*

Optional cluster on which to perform mounting operation. If not specified, the operation will be performed on all clusters.

**Note:** Check the archive output is mounted on the correct cluster. If the archive output is mounted on the wrong cluster, the incorrect data may be presented for user.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Examples

```
SOEMount(<path>, <ClusterName>) <-- Mounting specific Cluster of SOE and Summary Data from a specific location
```

```
SOEMount(<path>) <-- Mounting all Clusters of SOE and Summary Data from a specific location
```

## See Also

[Sequence of Events \(SOE\) Functions](#)

## Server Functions

Following are functions relating to servers.

<a href="#">ServerBrowseClose</a>	The ServerBrowseClose function terminates an active data browse session and cleans up resources associated with the session.
<a href="#">ServerBrowseFirst</a>	The ServerBrowseFirst function places the data browse cursor at the first record.

<a href="#">ServerBrowseGetField</a>	The ServerBrowseGetField function retrieves the value of the specified field from the record the data browse cursor is currently referencing.
<a href="#">ServerBrowseNext</a>	The ServerBrowseNext function moves the data browse cursor forward one record.
<a href="#">ServerBrowseNumRecords</a>	The ServerBrowseNumRecords function returns the number of records that match the filter criteria.
<a href="#">ServerBrowseOpen</a>	The ServerBrowseOpen function initiates a new browse session and returns a handle to the new session that can be used in subsequent data browse function calls.
<a href="#">ServerBrowsePrev</a>	The ServerBrowsePrev function moves the data browse cursor back one record.
<a href="#">ServerGetProperty</a>	Returns information about a specified server and can be called from any client.
<a href="#">ServerInfo</a>	Gets client and server information.
<a href="#">ServerInfoEx</a>	Gets client and server information from a specified process in a multiprocessor environment.
<a href="#">ServerIsOnline</a>	The ServerIsOnline function checks if the given server can be contacted by the client for giving the online/offline status of the server.
<a href="#">ServerReload</a>	Reloads the server specified by cluster and server name.
<a href="#">ServerRestart</a>	Restart any specific alarm, report, trend or I/O server from any Cicode node in system, without affecting other server processes running on the same machine.
<a href="#">ServerRPC</a>	Calls a remote procedure on the Plant SCADA server specified by the <i>ServerName</i> argument.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

### ServerBrowseClose

The ServerBrowseClose function terminates an active data browse session and cleans up resources associated

with the session.

## Syntax

**ServerBrowseClose(*iSession*)**

*iSession*:

Handle to a browse session previously opened by a [ServerBrowseOpen](#) call.

## Return Value

0 (zero) if the server browse session exists, otherwise an error code is returned.

## Related Functions

[ServerBrowseFirst](#), [ServerBrowseGetField](#), [ServerBrowseNext](#), [ServerBrowseNumRecords](#), [ServerBrowseOpen](#), [ServerBrowsePrev](#)

## Example

```
INT iRetVal = ServerBrowseClose(hServers);
```

## See Also

[Server Functions](#)

## ServerBrowseOpen

The [ServerBrowseOpen](#) function initiates a new browse session and returns a handle to the new session that can be used in subsequent data browse function calls.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**ServerBrowseOpen(STRING *Filter* , STRING *Fields* , STRING *Clusters*)**

*Filter*:

A semicolon delimited list of field name=field value pairs, specifying the records to return during the browse. An empty string indicates that all records will be returned.

The following regular expressions are supported: \*expr, expr\*, and \*expr\*. To specify an exclusion filtering condition, use the NOT keyword after the = operator.

*Fields*:

Specifies via a comma-delimited string the columns to be returned during the browse. An empty string indicates that the server will return available columns. Supported fields are:

NAME, TYPE, COMMENT, CLUSTER, MODE, NETADDR, PORT.

See [Browse Function Field Reference](#) for information about fields.

*Clusters:*

An optional parameter that specifies via a comma delimited string the subset of clusters to browse. An empty string indicates that connected clusters will be browsed.

## Return Value

An integer handle to the browse session. Returns -1 on error.

The returned entries will be ordered alphabetically by name.

## Related Functions

[ServerBrowseClose](#), [ServerBrowseFirst](#), [ServerBrowseGetField](#), [ServerBrowseNext](#), [ServerBrowseNumRecords](#),  
[ServerBrowsePrev](#)

## Example

```
INT hServers = ServerBrowseOpen("Type=alarm;Mode=1", "Name", "");
```

## See Also

[Server Functions](#)

## ServerBrowseFirst

The ServerBrowseFirst function places the data browse cursor at the first record.

## Syntax

**ServerBrowseFirst(*iSession*)**

*iSession:*

Handle to a browse session previously opened by a [ServerBrowseOpen](#) call.

## Return Value

0 (zero) if the server browse session exists, otherwise an error code is returned.

## Related Functions

[ServerBrowseClose](#), [ServerBrowseGetField](#), [ServerBrowseNext](#), [ServerBrowseNumRecords](#), [ServerBrowseOpen](#),  
[ServerBrowsePrev](#)

## Example

```
INT iReturn = ServerBrowseFirst(hServers);
```

## See Also

[Server Functions](#)

### ServerBrowseNext

The ServerBrowseNext function moves the data browse cursor forward one record. If you call this function after you have reached the end of the records, error 412 is returned (Databrowse session EOF).

## Syntax

**ServerBrowseNext(*iSession*)**

*iSession*:

Handle to a browse session previously opened by a [ServerBrowseOpen](#) call.

## Return Value

0 (zero) if the server browse session exists, otherwise an error code is returned.

## Related Functions

[ServerBrowseClose](#), [ServerBrowseFirst](#), [ServerBrowseGetField](#), [ServerBrowseNumRecords](#), [ServerBrowseOpen](#),  
[ServerBrowsePrev](#)

## See Also

[Server Functions](#)

### ServerBrowsePrev

The ServerBrowsePrev function moves the data browse cursor back one record. If you call this function after you have reached the beginning of the records, error 412 is returned (Databrowse session EOF).

## Syntax

**ServerBrowsePrev(*iSession*)**

*iSession*:

Handle to a browse session previously opened by a [ServerBrowseOpen](#) call.

## Return Value

0 (zero) if the server browse session exists, otherwise an error code is returned.

## Related Functions

[ServerBrowseClose](#), [ServerBrowseFirst](#), [ServerBrowseGetField](#), [ServerBrowseNext](#), [ServerBrowseNumRecords](#),  
[ServerBrowseOpen](#)

## See Also

[Server Functions](#)

## ServerBrowseGetField

The ServerBrowseGetField function retrieves the value of the specified field from the record the data browse cursor is currently referencing.

## Syntax

**ServerBrowseGetField(*iSession*, *sFieldName*)**

*iSession*:

Handle to a browse session previously opened by a [ServerBrowseOpen](#) call.

*sFieldName*:

The name of the field that references the value to be returned. Supported fields are:

NAME, TYPE, COMMENT, CLUSTER, MODE, NETADDR, PORT.

See [Browse Function Field Reference](#) for information about fields.

## Return Value

The value of the specified field as a string. An empty string may or may not be an indication that an error has been detected. The last error should be checked in this instance to determine if an error has actually occurred.

## Related Functions

[ServerBrowseClose](#), [ServerBrowseFirst](#), [ServerBrowseNext](#), [ServerBrowseNumRecords](#), [ServerBrowseOpen](#),  
[ServerBrowsePrev](#)

## Example

```
STRING ServName = ServerBrowseGetField(hServers, "NAME");
```

## See Also

[Server Functions](#)

### ServerBrowseNumRecords

The ServerBrowseNumRecords function returns the number of records that match the filter criteria.

## Syntax

**ServerBrowseNumRecords(*iSession*)**

*iSession*:

Handle to a browse session previously opened by a [ServerBrowseOpen](#) call.

## Return Value

The number of records that matched the filter criteria. A value of 0 denotes that no records matched. A value of -1 denotes that the browse session is unable to provide a fixed number. This could be the case if the data being browsed changed during the browse session.

## Related Functions

[ServerBrowseClose](#), [ServerBrowseFirst](#), [ServerBrowseGetField](#), [ServerBrowseNext](#), [ServerBrowseOpen](#),  
[ServerBrowsePrev](#)

## Example

```
INT iNumRecords = ServerBrowseNumRecords(hServers);
```

## See Also

[Server Functions](#)

### ServerGetProperty

This function returns information about a specified server and can be called from any client.

---

**Note:** This function can only be called for Alarm, Report and Trend Servers.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**STRING ServerGetProperty(STRING *Server*, STRING *Property* [, STRING *Cluster*]).**

*sServer*:

The name of the server to be queried in quotations marks "". Can be prefixed by the name of the host cluster,

that is "ClusterName.ServerName".

*sProperty:*

The name of the requested property. Can be one of the following:

"LastReloadError" - Error Code from the latest reload

"LibRDBMemTime" - Date and time of currently loaded cicode library (\_library.RDB)

"LibRDBDiskTime" - Date and time of cicode library on disk (\_library.RDB)

"RDBDiskTime" - Returns the date and time of RDB on disk (compiled)

"RDBMemTime" - Returns the date and time of currently loaded RDB (in-memory)

"ReloadStatus" - Returns 1 if the server is reloading, 0 if not

"ReloadProgress" - Reload Progress (in percent) Range: 0 – 100

"ServerStartState" - Returns the start up state of the server. Returns 0 if the server is starting, 1 if it is running.

"SyncStatus" - Returns the startup synchronization status:

- 0 – the server has been synchronized with its redundant peer.
- 1 – the server is not connected to peer server.
- 2 – the server is synchronizing with its redundant peer.

"SyncProgress" - The percentage of the server synchronization progress: Range: 0-100

*sCluster:*

The cluster of the server to be queried in quotation marks "". This parameter is optional. However, if the Server Name is not local or not specified in ClusterName.ServerName format an error code is returned.

**Note:** The "ServerStartState" property is only supported for alarm servers. The "SyncStatus" and "SyncProgress" properties are only supported for trend and alarm servers.

## Return Value

The value of the server property requested.

## Related Functions

[ServerInfo](#), [ServerInfoEx](#), [ServerIsOnline](#), [ServerReload](#), [ServerRestart](#)

## Example

```
ServerGetProperty("AlarmServer1", "ReloadStatus", "Cluster12");
```

## See Also

[Server Functions](#)

## ServerInfo

Gets status information on clients and servers.

**Note:** This function is a non-blocking function and can only access data contained within the calling process;

---

consequently it cannot return data contained in a different server process. This function is not redirected automatically by Plant SCADA runtime. If you want to make a call from one process to return data in another, use `MsgRPC()` to make a remote procedure call on the other process. Alternatively, use the [ServerInfoEx](#) function that allows you to specify the name of the component from which you want to retrieve data.

---

## Syntax

`STRING ServerInfo(STRING Name, INT Type [, STRING ClusterName] )`

**Name:**

The name of the client or server, either "Client", "Server", "Alarm", "Trend", or "Report".

You can also pass a number instead of the name (but it still needs to be enclosed in quotes). The number represents the target client. For example, if there are 12 clients, passing "3" will get information on the 3rd client.

If this server is an Alarm, Trend, Report, and I/O server then each client will be attached 4 times. So 12 clients would mean there are 3 Plant SCADA computers using this server - one of which is itself.

**Type:**

The type of information required (depends on the **Name** you specify):

**"Alarm", "Trend", or "Report" name:**

0 - Active flag (returns 1 if this is the active server, 0 if an inactive server).

1 - Number of clients attached to this server.

2 - If this client is attached to the primary or standby server for the specified server name. If Name is "Alarm" and if this client is attached to the primary alarm server, the return value is 0. If this client is attached to the standby, the return value is 1.

3 - The status of the client connection to the specified server name. If Name is "Report" and the client is talking to a report server (either primary or standby), the return value is 1. If not, the return value is 0.

4 - Alarm DB Status for the alarm server. Returns 1 if initializing, 2 if main, 3 if standby and 4 if invalid.

**"Client" name:**

- 0 - The computer name, as specified by [LAN]Node.
- 1 - Not supported.
- 2 - Not supported.
- 3 - Not supported.

For modes 1,2 and 3, use [ServerInfoEx](#) instead.

**"Server" name:**

- 0 - Not supported.
- 1 - The number of clients attached to this server. This is the total number of Alarm, Trend Report, and I/O server clients.

**"<number>":**

- 0 - The name of the server this client is talking to. For example, "Alarm", "Trend", "Report", or "IOServer".
- 1 - The login name of the client. This may be an empty string if the client has not logged in.
- 2 - The Plant SCADA computer name of the client computer.
- 3 - The time the client logged in.

- 4 - The number of messages received from this client.
- 5 - The number of messages sent to this client.
- 6 - If this client has a licence (1) from this server or not (0).
- 7 - The type of the licence; full licence (0), View-only Client (1), or Control Client (2).
- 8 - If the client is remote (1) or local (0).

*ClusterName:*

The name of the cluster that the server belongs to. This is only relevant if:

- The Name is "alarm", "report", or "trend"; AND
- The type of information required, *nType*, is 2 or 3.

**Note:**

**Alarm Server:** When an alarm server is active, it writes entries to the alarm log device(s). However, this alarm server may not be the 'Main' alarm server. The 'Main' alarm server stores data in the primary archive used for displaying the 'Sequence of Events' and 'Summary' pages. Currently an alarm server may be 'active' for the log devices, but 'standby' for the primary archive.

**Trend Server:** Trend server redundancy does not have the concept of "active". For trend servers, Type=0 will return 1 if the process is running on the computer defined as "Primary", and zero (0) if the process is running on the machine defined as "Secondary".

## Return Value

Status information specified by *nnType*.

## Related Functions

[ServerGetProperty](#), [ServerInfoEx](#), [ServerIsOnline](#), [ServerReload](#), [ServerRestart](#)

## Example

```
sSrvInfo=ServerInfo("Report",0);
IF sSrvInfo THEN
    ! This is a primary report server.
ELSE
    ! This is a stand-by report server.
END
/* Get and store the names of clients attached to this server */
iCount = 0;
iClients = ServerInfo("Server", 1);
WHILE iCount < iClients DO
    sName[iCount] = ServerInfo(IntToStr(iCount), 2);
    iCount = iCount + 1;
END
```

## See Also

[Server Functions](#)

## ServerInfoEx

Gets status information on clients and servers from a specified component in a multiprocess runtime environment.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

When this function is called, the system redirects the call to the process that contains the specified component. If the specified component is in the calling process, the call is not redirected. If the specified component is not one of the servers listed in the *sComponent* argument description (see below), or if the system cannot find the component from the connected local processes, a hardware alarm is raised.

## Syntax

```
STRING ServerInfoEx(STRING Name, INT Type, STRING Component [, STRING ClusterName] [, STRING ServerName] )
```

*Name*:

The name of the client or server, either "Client", "Server", "Alarm", "Trend", or "Report".

You can also pass a number instead of the name (but it still needs to be enclosed in quotes). The number represents the target client. For example, if there are 12 clients, passing "3" will get information on the 4th client.

If this server is an Alarm, Trend, Report, and I/O server then each client will be attached 4 times. So 12 clients would mean there are 3 Plant SCADA computers using this server - one of which is itself.

*Type*:

The type of information required (depends on the *Name* you specify):

When *Name* is set to "Alarm", "Trend", or "Report":

0 - Active flag (returns 1 if this is the active server, 0 if an inactive server).

When an alarm server is active, it writes entries to the alarm log device(s). However, this alarm server may not be the 'Main' alarm server. The 'Main' alarm server stores data in the primary archive used for displaying the 'Sequence of Events' and 'Summary' pages. Currently an alarm server may be 'active' for the log devices, but 'standby' for the primary archive.

Trend server redundancy does not have the concept of "active". For trend servers, *Type*=0 will return 1 if the process is running on the computer defined as "Primary", and zero (0) if the process is running on the machine defined as "Secondary".

1 - Number of clients attached to this server.

2 - If this client is attached to the primary or standby server for the specified server name. If *Name* is "Alarm" and if this client is attached to the primary alarm server, the return value is 0. If this client is attached to the standby, the return value is 1.

If the *Component* is "IOServer" and the *Name* is "Client", Types 1 and 2 return an empty string, as the concept of primary and standby servers does not apply to I/O servers.

3 - The status of the client connection to the specified server name. If *Name* is "Report" and the client is talking to a report server (either primary or standby), the return value is 1. If not, the return value is 0.

Since v7.30, the Client connection to the alarm server is not permanently maintained, and is reopened for each new user. The client connection to the trend servers is permanent, and is not affected by the new user login process.

4 - Alarm DB Status for the alarm server. Returns 1 if initializing, 2 if main, 3 if standby and 4 if invalid.

If *Name* is set to "**Client**"

- 0 - The computer name, as specified by [LAN]Node.
- 1 - The primary server name, as specified in the server configuration settings.
- 2 - The secondary server name, as specified , as specified in the server configuration settings.
- 3 - The name of the INI file being used, for example, Citect.INI.

If *Name* is set to "**Server**":

- 0 - The server name, as specified in the server configuration settings.
- 1 - The number of clients attached to this server. This is the total number of Alarm, Trend Report, and I/O server clients.

If *Name* uses an ordinal option "**<number>**":

- 0 - The name of the server this client is talking to. For example, "Alarm", "Trend", "Report", or "IOServer".
- 1 - The login name of the client. This may be an empty string if the client has not logged in.
- 2 - The Plant SCADA computer name of the client computer.
- 3 - The time the client logged in.
- 4 - The number of messages received from this client.
- 5 - The number of messages sent to this client.
- 6 - Determines the licensing relationship between the target client (specified by "<number>") and the server (specified by *Component*). '0' indicates there is no licensing relationship, '1' indicates the server has provided the target client with a license, '2' indicates the server received a license from the target client (acting as the license server). If the server is running in multi-process mode, you will need to iterate through all the server processes on the computer using mode 6 to detect if a license is being issued from that computer.
- 7 - The type of the licence; full licence (0), View-only Client (1), or Control Client (2). If the client has not obtained a license from the listed server component (i.e. ServerInfoEx mode 6 has returned 0), then this will not be defined and always return 0.
- 8 - If the client is remote (1) or local (0). "Remote" means the client is not in the same process as the server. For example, the client could be in another process on the same computer as the server, or even on another computer. "Local" means the client is in the same process as the server.

Be aware that ServerInfoEx has been enhanced so that when using the ordinal option (*Name* is "<number>") the number representing *Type* can be supplied in a 100-based format so that all connections between server and client will be considered in evaluating that type. Not using 100-based format may mean only a subset of connections will be evaluated.

For example, using *Type* 106 will search all connections to determine whether the client has a licence from the server, whereas using 6 will ignore some connections.

Please note the enhanced 100-based format should be used with v8.30 or above to avoid missing connections. Do not mix the enhanced 100-based mode with the normal mode. For example, if your Cicode uses mode 106 rather than 6 to query whether the client has a licence from the server, then it should use 107 rather than 7 to enquire about the type of licence.

*Component*:

Specifies the component name from which to retrieve the status information:

- " "- An empty string causes the function to run on the calling process.
- "Alarm" - Redirects the function to the alarm server process.

- "Trend" - Redirects the function to the trend server process.
- "Report" - Redirects the function to the report server process.
- "IOMServer" - Redirects the function to the I/O server process.

If *Name* is "Client", the function will not be redirected to the component specified by *Component*. Instead, the function gets information of type *Type* from the current client connection to the component specified by *Component*.

#### ClusterName:

The name of the cluster that the server belongs to. This is only relevant if:

The Name is "alarm", "report", or "trend"; AND

The type of information required, *nType*, is 2 or 3.

#### ServerName:

Specifies the name of the the I/O Server. This parameter is only required if you are running more than one I/O server process from the same cluster on the same computer and need to instruct the system which process to redirect to. The argument is enclosed in quotation marks "".

## Return Value

Status information specified by *nType*.

## Related Functions

[ServerGetProperty](#), [ServerInfo](#), [ServerIsOnline](#), [ServerReload](#), [ServerRestart](#)

## Example

This example gets the server information from the report process.

```
sSrvInfo=ServerInfoEx("Report",0, "Report");
IF sSrvInfo THEN
    ! This is a primary report server.
ELSE
    ! This is a stand-by report server.
END
/* Get and store the names of clients attached to the report server */
iCount = 1;
iClients = ServerInfoEx("Server", 1, "Report");
WHILE iCount <= iClients DO
    sName[iCount] = ServerInfoEx(IntToStr(iCount), 2, "Report");
    iCount = iCount + 1;
END
```

## See Also

[Server Functions](#)

## ServerIsOnline

The ServerIsOnline function checks if the given server can be contacted by the client for giving the online/offline status of the server.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

```
INT ServerIsOnline(STRING ServerName[, STRING Clusters][, INT bLocal])
```

*ServerName*:

The name of the server to be queried in quotations marks "".

*Cluster*:

The cluster of the server to be queried in quotation marks "". An empty string indicates that the default cluster will be used.

*bLocal*:

The mode of the server status check.

A value of "1" results in a local check of the server status, checking if the current computer can connect to the server directly. In local mode, this function will return a value immediately, without blocking the Cicode.

A value of "0" (default) indicates that a remote check of the server status is required to confirm if the server can be reached through its peer. In remote mode, this function will block the Cicode while waiting for a response to the query.

## Return Value

An integer value of online/offline status. Returns 1 for online, 0 for offline.

## Related Functions

[ServerGetProperty](#), [ServerInfo](#), [ServerInfoEx](#), [ServerReload](#), [ServerRestart](#)

## Example

```
ServerIsOnline("AlarmServer1", "Cluster12");
```

## See Also

[Server Functions](#)

## ServerReload

This function can only be called for Alarm, Report and Trend Servers and reloads the server specified by cluster and server name. If the server is not found an error code is returned. ServerReload can be used in blocking or non-blocking modes using the bSync parameter. When used in non-blocking mode, the server status can be returned using the [ServerGetProperty](#) function.

To call this function successfully a valid user needs to be logged in as the client and you need to set the [\[LAN\]AllowRemoteReload](#) parameter in the Citect.ini file to "1" on the computer where the receiving server is running.

It is recommended that the ServerGetProperty Cicode function be used with the *LibRDBMemTime* and *LibRDBDiskTime* properties to check if there is a change to the Cicode library before attempting a reload. Following a reload please check the corresponding server's syslog.dat file for any reload messages. The Cicode changes will not be reloaded, therefore a restart may be more appropriate.

## WARNING

### UNINTENDED EQUIPMENT OPERATION

Restart the server process if a "Cicode library timestamp differs" error is detected. A library mismatch is indicated on the server in either the hardware alarm or the server's syslog.dat file.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** A message in the Syslog.dat file and hardware alarm of "Cicode library timestamp differs" (error code 454) will be raised if the Cicode library used by one or more server runtime databases is different from the one in memory. The timestamps will be different if the project has been fully recompiled (with or without Cicode modification), or if the project has been incrementally recompiled after any Cicode has been modified.

## Syntax

**INT ServerReload(STIRNG Server [, STRING Cluster] [, INT Sync] )**

*sServer:*

The name of the server to be reloaded in quotations marks "". Can be prefixed by the name of the host cluster, that is "ClusterName.ServerName".

*sCluster:*

The cluster of the server to be reloaded in quotation marks "". This parameter is optional. However, if the server name is not local or not specified in ClusterName.ServerName format, an error code is returned.

*bSync:*

Specifies whether the function operates in blocking or non-blocking mode. If bSync is set to 1, the function will not return until the server reload is complete. The reload is complete when all the records of all rdb files have been processed and updated. Blocking mode cannot be used from a foreground task (for example on graphic pages). When bSync is set to 0, the function operates in non-blocking mode. You can get the latest status of the reload using the ServerGetProperty function. Default value is 0.

## Return Value

0 (zero) if the function was successful. Returns an error if unsuccessful. Outline of errors:

- 418 - Server is not found or if the client is not allowed to visit the cluster described in "sCluster".
- 281 - Server is sitting on a remote machine to the client and the connection towards the server is not available.
- 519 - Server connection was available but interrupted.
- 451 - Server is busy with previous load request.

## Related Functions

[ServerGetProperty](#), [ServerInfo](#), [ServerInfoEx](#), [ServerIsOnline](#), [ServerRestart](#)

## Example

```
ServerReload("AlarmServer1", "Cluster1", 0);
```

## See Also

[Server Functions](#)

### ServerRestart

Allows you to restart any specific alarm, report, trend or I/O server from any Cicode node in a system, without affecting other server processes running on the same machine. To call this function, a valid user needs to be logged in.

For this function to operate successfully, you need to make the following Citect.ini parameter settings:

- [\[Shutdown\]NetworkStart](#) = "1" on the computer where the issuing client is running.
- [\[Shutdown\]NetworkIgnore](#) = "0" (zero) on the computer where the server is running.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

```
INT error = ServerRestart (STRING sServerName, STRING sCluster = "")
```

*sServerName*:

The name of the server to restart

*sCluster*:

The cluster the server belongs to. This parameter is optional. If *sCluster* is not specified the current system cluster is used.

## Return Value

0 (zero) if successful, otherwise one of the following error codes is returned:

- 256 - General software error
- 292 - Invalid function
- 403 - Cluster not found
- 418 - No server of type on cluster
- 512 - Time out error
- 513 - Access denied error

## See Also

[Cicode and General Errors.](#)

## Related Functions

[ServerGetProperty](#), [ServerInfo](#), [ServerInfoEx](#), [ServerIsOnline](#), [ServerReload](#)

## Example

```
ServerRestart("AlarmServer1", "Cluster1");
```

## See Also

[Server Functions](#)

## SQL Functions

Following are functions related to SQL operations.

## Miscellaneous functions

<a href="#">ExecuteDTSPkg</a>	Loads and executes a Data Transformation Services package which initiates data transfer between OLE DB data sources.
-------------------------------	--

## Connection functions

<a href="#">SQLClose</a>	Closes a SQL connection between the DB connection object and a database.
<a href="#">SQLConnect</a>	Makes a connection to a database system for execution of SQL statements.
<a href="#">SQLCreate</a>	Creates an internal DB connection object.
<a href="#">SQLDisconnect</a>	Closes a database connection.
<a href="#">SQLDispose</a>	Disposes the DB connection object.
<a href="#">SQLInfo</a>	Gets information about a database connection.
<a href="#">SQLOpen</a>	Opens an SQL connection between the DB connection object and the database.

## Transaction functions

<a href="#">SQLBeginTran</a>	Starts a database transaction.
<a href="#">SQLCommit</a>	Commits a transaction to the database.
<a href="#">SQLRollBack</a>	Rolls back (or cancels) the last database transaction.

## Statement functions

<a href="#">SQLAppend</a>	Appends a text string to the SQL buffer.
<a href="#">SQLCall</a>	Executes an SQL query on a database.
<a href="#">SQLEnd</a>	Terminates an SQL query.
<a href="#">SQLExec</a>	Executes an SQL query on a database.
<a href="#">SQLGetRecordset</a>	Executes an SQL query on a database.
<a href="#">SQLGetScalar</a>	Executes an SQL query on a database.
<a href="#">SQLSet</a>	Sets a statement string in the SQL buffer.

## Multiple queries functions

<a href="#">SQLQueryCreate</a>	Creates a new query and returns its handle.
<a href="#">SQLQueryDispose</a>	Disposes a query.

## Recordset functions

<a href="#">SQLCancel</a>	Cancels both the current operation on the given connection and all other pending operations on the given connection.
<a href="#">SQLFieldInfo</a>	Gets information about the fields or columns selected in an SQL query.
<a href="#">SQLGetField</a>	Gets field or column data from a database record.
<a href="#">SQLIsNullField</a>	Checks presence of null value in field from a recordset.
<a href="#">SQLNext</a>	Gets the next database record from a SQL query.
<a href="#">SQLNoFields</a>	Gets the number of fields or columns that were returned by the last SQL statement.

<a href="#">SQLNumChange</a>	Gets the number of records that were modified in the last insert, update, or delete SQL statement.
<a href="#">SQLNumFields</a>	Gets the number of fields or columns that were returned by the last SQL statement.
<a href="#">SQLPrev</a>	Gets the previous database record from an SQL query.
<a href="#">SQLRowCount</a>	Gets the number of rows in the recordset.

## Error and tracing functions

<a href="#">SQLErrMsg</a>	Returns an error message from the SQL system.
<a href="#">SQLTraceOff</a>	Turns off the debug trace.
<a href="#">SQLTraceOn</a>	Turns on the debug trace.

## Parametrized queries functions

<a href="#">SQLParamsClearAll</a>	Remove all parameters associated with a particular connection object.
<a href="#">SQLParamsSetAsInt</a>	Adds or replaces a parameterized query's parameter and its value in the specified connection. The value of the parameter is given as an int.
<a href="#">SQLParamsSetAsReal</a>	Adds or replaces a parameterized query's parameter and its value in the specified connection. The value of the parameter is given as a real.
<a href="#">SQLParamsSetAsString</a>	Adds or replaces a parameterized query's parameter and its value in the specified connection. The value of the parameter is given as a string.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## ExecuteDTSPkg

Loads and executes a DTS (Data Transformation Services) package which initiates data transfer and transformations between OLE DB data sources.

A DTS package is created using the DTS utility provided in Microsoft SQL Server 7.0. It can be saved in a COM structured file, a Microsoft Repository, or in an SQL Server Database.

All except the first of this function's parameters are optional, and their use will depend on your needs.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**ExecuteDTSPkg(*sFileOrSQLSvrName* [, *sPkgName*] [, *sParam1*, ..., *sParam5*] [, *sPkgPwd*] [, *sPkgVer*] [, *sLogFile*] [, *sSQLSvrUsr*] [, *sSQLSvrPwd*])**

*sFileOrSQLSvrName*:

The path and name of the file containing the package (for file-based packages), or the SQL Server name (for SQL Server stored packages).

*sPkgName*:

The package name.

For file-based packages where only one package is stored in a file, you can ignore this parameter, as the package name defaults to the name of the file.

If the package has been named differently to the file, or a file contains more than one package, you need to specify the package name. You need to also specify the package name for SQL Server stored packages.

*sParam1*, *sParam5*:

Five optional variables which may be used as global variables within the DTS package. The variables need to be named Param1, Param2, Param3, Param4, and Param5.

*sPkgPwd*:

The package password.

The creator of the DTS package may have implemented a password so that unauthorized users cannot access it. In this case, you need to specify the package password. If no password has been implemented, you can omit this parameter.

*sPkgVer*:

The package version. If you don't specify a version, the most recent version is used.

*sLogFile*:

An optional path and name for a log file. The log file can track activity such as:

File DTS package detected

SQL DTS package detected

Package initialized successfully

Package executed successfully

Package execution was not successful

*sSQLSvrUsr*:

The user name providing access to the SQL Server where the DTS package is stored. A user's account on the SQL Server consists of this user name and, in most cases, a password.

This parameter also determines which method is used to load the package.

If *sSQLSvrUsr* is specified, the package is assumed to be an SQL Server stored package. In this case, the package is loaded using the *LoadFromSQLServer()* method. Otherwise, the package is file-based and *LoadFromStorageFile()* is called.

*sSQLSvrPwd*:

The password providing access to the SQL Server, if the user's account on the server requires a password.

## Return Value

0 (zero) if the package was executed successfully, otherwise a DTS error number is returned.

## Example

```
/* File-based package with one package per file, where the package
name is the same as the file name.*/
iResult = ExecuteDTSPkg("c:\dtspackages\package.dts");
/*SQL Server stored package with additional parameters */
iResult = ExecuteDTSPkg("Server1", "TestPackage", "Param1", "Param2", "Param3",
"Param4", "Param5", "Fred", "1", "c:\packages\PkgLog.txt", "jsmith", "secret");
```

## See Also

[SQL Functions](#)

## SQLAppend

Appends a query string to the SQL buffer. Cicode cannot send an SQL query that is longer than 255 characters. If you have an SQL query that is longer than the 255 character limit, you can split the query into smaller strings, and use this function to append the query in the SQL buffer.

This function can be called in the foreground or background.

Queries which are built on the basis of user data, for example inputed by users via graphics pages or forms, may be prone to SQL Injection attacks. In such case, try to limit the risk by using CiCode functions from parameterized queries group and refer to a professional advice in this matter.

### NOTICE

#### SECURITY BREACH VIA SQL INJECTION

- Validate all textbox entries using validation controls, regular expressions and code
- Use parameterized SQL or stored procedures
- Use a limited access account to connect to the database

**Failure to follow these instructions can result in equipment damage.**

Building queries from pieces (SQLSet, SQLAppend) or adding parameters to either queries or connections (SQLParam functions) requires a few calls to respective CiCode functions. If a few functions try to manipulate the same connection in the same time some conflicts and unintended operations may occur. It is a typical multithreading problem.

To avoid this, instead of manipulating connections, consider using locally created and locally disposed queries. For example:

```
int function SAFE_SQL_CICODE_MULTITHREAD_USE()
//locally created query
int hStmt = SQLQueryCreate(hConnection);

//Set the query
```

```
SQLSet(hStmt, "select * from TAB where NAME=@Name");

//Add parameters to the query
SQLParamsSetAsString(hStmt, "Name", "Aaa");

//Execute the query
SQLGetRecordset(hStmt, "");

//the locally created query is disposed
SQLQueryDispose(hStmt);
End
```

## Syntax

**SQLAppend(*hGeneral*, *String*)**

*hGeneral*:

The handle either to the DB connection object (returned from either SQLCreate() or SQLConnect() function) or to the query handle (returned from SQLQueryCreate()). When it is the connection handle, the operation is performed on the default query in that DB connection object. When it is the query handle, the operation is performed on that query through the DB object which is associated to it.

*String*:

The query string to append to the SQL buffer.

## Return Value

0 (zero) if successful, otherwise an error number is returned. (For details of the 307 error code, call the [SQLErrMsg](#) function).

## Related Functions

[SQLSet](#), [SQLBeginTran](#), [SQLCommit](#), [SQLConnect](#), [SQLDisconnect](#), [SQLEnd](#), [SQLErrMsg](#)

## Example

See [SQLSet](#)

## See Also

[SQL Functions](#)

## SQLBeginTran

Starts a database transaction. When you make a transaction, your changes are not written to the database until you call the SQLCommit() function. Alternatively, you can use the SQLRollBack() function to discard all changes made during the transaction.

After you begin a transaction, you need to call either SQLCommit() to save the changes or SQLRollBack() to discard the changes. You need to use one of these functions to complete the transaction and release all database

locks.

A single database connection can only handle one transaction at a time. After you call SQLBeginTran(), you need to complete that transaction before you can call SQLBeginTran() again.

If you disconnect from a database while a transaction is active (not completed), the transaction is automatically "rolled back" any changes you made to the database in that transaction are discarded.

You do not need to begin a transaction to modify a database. Any changes you make to a database before you call the SQLBeginTran() are automatically committed, and no database locks are held.

The SQLBeginTran() function is not supported by all databases. If the function is not performing as you expect, check that both your database and/or DB provider support transactions. Refer to the documentation for your database for more information on transactions.

This function is a blocking function and should not be called from a foreground task.

## Syntax

**SQLBeginTran(*hSQL*)**

*hSQL*:

The handle to the DB connection object, returned from either SQLCreate() or SQLConnect() function. The handle identifies the DB connection object where details of the associated SQL connection are stored.

## Return Value

0 (zero) if successful, otherwise an error number is returned. (For details of the 307 error code, call the [SQLErrMsg](#) function).

## Related Functions

[SQLCommit](#), [SQLConnect](#), [SQLDisconnect](#), [SQLEnd](#), [SQLErrMsg](#), [SQLExec](#), [SQLFieldInfo](#), [SQLGetField](#), [SQLInfo](#), [SQLNext](#), [SQLNoFields](#), [SQLNumChange](#), [SQLRollBack](#), [SQLTraceOff](#), [SQLTraceOn](#)

## Example

```
/* Increase each employee's salary and superannuation by a
specified amount. If any errors occur, the changes are aborted */
INT
FUNCTION
PayIncrease(STRING sIncrease)
    INT hSQL;
    INT Count1;
    INT Count2;
    hSQL = SQLConnect("DRV=QEDBF");
    SQLBeginTran(hSQL);
    SQLExec(hSQL, "UPDATE C:\DATA\EMPLOYEE SET Salary = Salary + " +sIncrease);
    Count1 = SQLNumChange(hSQL);
    SQLExec(hSQL, "UPDATE C:\DATA\EMPLOYEE SET Super = Super + " +sIncrease);
    Count2 = SQLNumChange(hSQL);
    IF Count1 = Count2 THEN
        SQLCommit(hSQL);
    ELSE
```

```
    SQLRollBack(hSQL);
END
SQLEnd(hSQL);
SQLDisconnect(hSQL);
END
```

## See Also

[SQL Functions](#)

### SQLCall

Executes an SQL query on a database. The function returns the number of rows affected by the executed query. With this function, you can execute any SQL query or command supported by the SQL database. If it returns some data (as for SELECT query), the data is ignored.

This function is a blocking function and should not be called from a foreground task.

Queries which are built on the basis of user data, for example inputed by users via graphics pages or forms, may be prone to SQL Injection attacks. In such case, try to limit the risk by using CiCode functions from parameterized queries group and refer to a professional advice in this matter.

#### NOTICE

#### SECURITY BREACH VIA SQL INJECTION

- Validate all textbox entries using validation controls, regular expressions and code
- Use parameterized SQL or stored procedures
- Use a limited access account to connect to the database

**Failure to follow these instructions can result in equipment damage.**

## Syntax

**SQLCall(*hGeneral*, *sSelect*)**

*hGeneral*:

The handle either to the DB connection object (returned from either SQLCreate() or SQLConnect() function) or to the query handle (returned from SQLQueryCreate()). When it is the connection handle and *sSelect* is an empty string, the operation is performed on the first query in that DB connection object. When it is the query handle, the operation is performed on that query through the DB object which is associated to it.

*sSelect*:

The SQL query to be sent to the SQL database.

## Return Value

The number of affected records or -1 if an error is detected. (For details call the SQLErrMsg() function). The presence of error code can be tested by calling the [IsError](#) CiCode function.

## Related Functions

[SQLCreate](#), [SQLOpen](#), [SQLClose](#), [SQLDispose](#), [SQLConnect](#), [SQLDisconnect](#), [SQLInfo](#), [SQLSet](#), [SQLAppend](#), [SQLExec](#), [SQLGetRecordset](#), [SQLCall](#), [SQLGetScalar](#), [SQLEnd](#)

## See Also

[SQL Functions](#)

### SQLCancel

This function cancels both the current operation on the given connection and all other pending operations on the given connection. The cancellation means that the current operation on the SCADA side is immediately finished. The cancelled operation returns error code 299 and its state on the database side should be considered as undefined, that is, it is not known that it either succeeded, partially succeeded, or was unsuccessful.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete. It can be called in the foreground or background.

## Syntax

**SQLCancel(*hConnection*)**

*hConnection*:

The handle to the connection.

## Return Value

0 (zero) if successful, otherwise an error code is returned. (For details of the 307 error code, call the [SQLErrMsg\(\)](#) function).

## Related Functions

[SQLCreate](#), [SQLOpen](#), [SQLClose](#), [SQLDispose](#), [SQLConnect](#), [SQLDisconnect](#), [SQLInfo](#)

## See Also

[SQL Functions](#)

### SQLClose

Closes a SQL connection between the DB connection object specified by the function's parameter and a database. The DB connection object is not deleted from the memory until calling [SQLDispose](#) function.

This function is a blocking function and should not be called from a foreground task.

## Syntax

**SQLClose(*hSQL*)**

*hSQL*:

The handle to the DB connection object, returned from the SQLCreate() function. The handle identifies the DB connection object where details of the associated SQL connection are stored.

## Return Value

0 if success, otherwise an error code (for details call the [SQLErrMsg\(\)](#) function).

## Related Functions

[SQLCreate](#), [SQLOpen](#), [SQLClose](#), [SQLDispose](#), [SQLConnect](#), [SQLDisconnect](#), [SQLInfo](#)

## Example

See [SQLCreate](#).

## See Also

[SQL Functions](#)

## SQLCommit

Commits (to the database) all changes made within a transaction. If you call the [SQLBeginTran\(\)](#) function to begin a transaction, you need to call the SQLCommit() function to save the changes you make to the database during that transaction (with the Insert, Delete, and Update SQL commands).

The SQLCommit() and [SQLRollBack\(\)](#) functions both complete a transaction and release all database locks. But while the SQLCommit() function saves all changes made during the transaction, the SQLRollBack() function discards these changes. Unless you call the SQLCommit() function before you disconnect the database, the transaction is automatically rolled back any changes you made to the database in that transaction are discarded.

The SQLCommit() function could affect different databases in different ways. If the function is not performing as you expect, check that your database is able to service transactions. Refer to the documentation for your database for information on committing transactions.

This function is a blocking function and should not be called from a foreground task.

## Syntax

**SQLCommit(*hSQL*)**

*hSQL*:

The handle to the DB connection object, returned from either [SQLCreate\(\)](#) or [SQLConnect\(\)](#) function. The handle identifies the DB connection object where details of the associated SQL connection are stored.

## Return Value

0 (zero) if successful, otherwise an error number is returned. (For details of the 307 error code, call the SQLErrMsg() function).

## Related Functions

[SQLBeginTran](#), [SQLConnect](#), [SQLDisconnect](#), [SQLEnd](#), [SQLErrMsg](#), [SQLExec](#), [SQLFieldInfo](#), [SQLGetField](#), [SQLInfo](#), [SQLNext](#), [SQLNoFields](#), [SQLNumChange](#), [SQLRollBack](#), [SQLTraceOff](#), [SQLTraceOn](#)

## Example

See [SQLBeginTran](#).

## See Also

[SQL Functions](#)

## SQLConnect

Creates an internal database connection object and tries to connect it to a database specified by the connection string. Returns a handle to the database connection object for use by the other database functions.

You only require one database connection object for each database system to be accessed (for example, Oracle, dBASE, Excel, etc.).

It is recommended not to use an SQL database for storage of real-time data (such as alarms), because SQL databases do not provide real-time performance when accessing database data. Only use an SQL database where data transfer is not a priority (for example, recipes or reports). If you try to use SQL to store real time data, Plant SCADA's performance could be greatly decreased.

Each database connection object created by SQLConnect should be released by calling SQLDisconnect with handle to the object. The releasing operation should be performed even when the SQL connection to database is no longer active; for example automatically dropped by a remote database or manually closed by SQLClose. Memory leaks can occur if the handles are not properly released.

---

### Note:

Currently there are certain configurations of providers, connection strings and database systems which can automatically close connections in order to save the database systems' resources; for example MS SQL Server dedicated provider with activated pooling closes the connection to MS SQL Server when there is no data exchange between the client and the database server for some predefined time. If such behaviour is observed and is not desired from the system design point of view, it can be either:

- Switched off by finding and using respective attributes in the connection string (if existing and supported by database) or
- By actively checking error codes returned from SQL Cicode functions and reconnecting by using SQLOpen if the connection is dropped.

The actual state of connections can be checked by [SQLInfo](#) with type 5.

---

This function is a blocking function and should not be called from a foreground task.

## Syntax

**SQLConnect(*sConnect*)**

*sConnect*:

The connection string, in the format:

`<attribute>=<value>[;<attribute>=<value>...]`

Acceptable attributes and their values vary accordingly to the provider and/or the database system, so please refer to your database documentation. For example, connecting to a SQL Server via ODBC usually requires giving the computer name and the database name which can be done by defining "Server" and "Database" attributes. The same connection via OleDB requires defining the computer as "Data Source" and the database as "Initial Catalog".

Providing username and password as a plain text in the connection string may lead to a security breach on the database side. Please consider using other forms of authentication instead of username/password login, for example Windows Authentication. If this is not possible, try to limit the database account rights and not use the same username and password for other vital parts of the system such as for SCADA.

### NOTICE

#### SECURITY BREACH VIA SQL INJECTION

- Validate all textbox entries using validation controls, regular expressions and code
- Use parameterized SQL or stored procedures
- Use a limited access account to connect to the database

**Failure to follow these instructions can result in equipment damage.**

Some simple examples are shown below:

ODBC provider

```
"Driver={SQL Server};Server=(local);Trusted_Connection=Yes; Database=MyDatabase;"  
"Driver={Microsoft ODBC for Oracle};Server=ORACLE8i7;Persist Security  
Info=False;Trusted_Connection=Yes"  
"Driver={Microsoft Access Driver (*.mdb)};DBQ=c:\doc\MyDatabase.mdb"  
"Driver={Microsoft Excel Driver (*.xls)};DBQ=c:\doc\MySheet.xls"  
"SCADA Data Provider=Odbc;Driver={Microsoft Text Driver (*.txt; *.csv)};DBQ=c:\doc"  
"DSN=MyDSNname"
```

OleDB provider

```
"SCADA Data Provider=OleDb;Provider=MSDAORA; Data Source=ORACLE8i7;Persist Security  
Info=False;Integrated Security=Yes"  
"SCADA Data Provider=OleDb;Provider=Microsoft.Jet.OLEDB.4.0; Data Source=c:\bin\  
LocalAccess40.mdb"  
"SCADA Data Provider=OleDb;Provider=SQLOLEDB;Data Source=(local);Integrated Security=SSPI"
```

SQLClient Provider

```
"SCADA Data Provider=SQLClient;Persist Security Info=False;Integrated Security=true;Initial  
Catalog=MyCatalog;server=(local)"
```

SCADA Data Provider

The provider to be used to communicate to a database. Allowed values are as follows:

	ODBC – ODBC provider, default one if no provider specified, OLEDB – OLEDB provider, SQLClient – MS SQL Server dedicated provider
SCADA Connection Timeout	The timeout in seconds for establishing connections.
SCADA Query Timeout	The timeout in seconds for executing queries. This attribute overwrites the [SQL]QueryTimeout INI parameter.

## Return Value

The handle to the database connection object if the connection is successful, otherwise -1 is returned. (For details call the SQLErrMsg() function). The handle identifies the database connection object where details of the associated SQL connection to a DB are stored.

## Related Functions

[SQLCreate](#), [SQLOpen](#), [SQLClose](#), [SQLDispose](#), [SQLConnect](#), [SQLDisconnect](#), [SQLInfo](#)

## Example

```
/* Make a connection to an SQL server and select the name field from each record in the
employee database. */
UNCTION
ListNames()
    INT hSQL;
    STRING sName;
    INT Status;
    hSQL = SQLConnect("DSN=MyDatabase;UID=billw;SRVR=CI1");
    IF hSQL <> -1 THEN
        Status = SQLExec(hSQL, "SELECT NAME FROM EMPLOYEE");
        IF Status = 0 THEN
            WHILE SQLNext(hSQL) = 0 DO
                sName = SQLGetField(hSQL, "NAME");
                ..
            END
            SQLEnd(hSQL);
        ELSE
            Message("Information", SQLErrMsg(), 48);
        END
        SQLDisconnect(hSQL);
    ELSE
        Message("Information", SQLErrMsg(), 48);
    END
END
```

## See Also

[SQL Functions](#)

### SQLCreate

Creates an internal DB connection object and returns a handle to the object for use by the other DB functions. The object is in disconnected state and can be connected to a database by executing the [SQLOpen](#) function.

You only require one DB connection object for each database system to be accessed (for example, Oracle, dBASE, Excel, etc.).

Each DB connection object created by SQLCreate should be released by calling [SQLDispose](#) with the handle to the object. The releasing operation should be performed even when the SQL connection to DB is no longer active; for example, automatically dropped by a remote DB. Memory leaks can occur if the handles are not properly released.

This function can be called in the foreground or background.

## Syntax

**SQLCreate(sConnect)**

*sConnect:*

The connection string, in the format:

<attribute>=<value>[ ;<attribute>=<value> . . . ]

Acceptable attributes and their values vary accordingly to the provider and/or the database system, so please refer to your database documentation. For example, connecting to a SQL Server via ODBC usually requires giving the computer name and the database name which can be done by defining "Server" and "Database" attributes. The same connection via OleDB requires defining the computer as "Data Source" and the database as "Initial Catalog".

Providing username and password as a plain text in the connection string may lead to a security breach on the database side. Please consider use of other forms of authentication instead of username/password login as for example Windows Authentication. If not possible, try to limit the database account rights and not use the same username and password as for other vital part of the system as for example for SCADA.

#### NOTICE

#### SECURITY BREACH VIA SQL INJECTION

- Validate all textbox entries using validation controls, regular expressions and code
- Use parameterized SQL or stored procedures
- Use a limited access account to connect to the database

**Failure to follow these instructions can result in equipment damage.**

Some simple examples are shown below:

#### ODBC provider

```
"Driver={SQL Server};Server=(local);Trusted_Connection=Yes; Database=MyDatabase;"  
"Driver={Microsoft ODBC for Oracle};Server=ORACLE8i7;Persist Security"
```

```
Info=False;Trusted_Connection=Yes"
"Driver={Microsoft Access Driver (*.mdb)};DBQ=c:\doc\MyDatabase.mdb"
"Driver={Microsoft Excel Driver (*.xls)};DBQ=c:\doc\MySheet.xls"
"SCADA Data Provider=Odbc;Driver={Microsoft Text Driver (*.txt; *.csv)};DBQ=c:\doc\
"DSN=MyDSNname"
```

**OleDb provider**

```
"SCADA Data Provider=OleDb;Provider=MSDAORA; Data Source=ORACLE8i7;Persist Security
Info=False;Integrated Security=Yes"
"SCADA Data Provider=OleDb;Provider=Microsoft.Jet.OLEDB.4.0; Data Source=c:\bin\
LocalAccess40.mdb"
"SCADA Data Provider=OleDb;Provider=SQLOLEDB;Data Source=(local);Integrated Security=SSPI"
```

**SQLClient Provider**

```
"SCADA Data Provider=SQLClient;Persist Security Info=False;Integrated Security=true;Initial
Catalog=MyCatalog;server=(local)"
```

SCADA Data Provider	The provider to be used to communicate to a DB. Allowed values are as follows: ODBC - ODBC provider, default one if no provider specified, OLEDB - OLEDB provider, SQLClient - MS SQL Server dedicated provider
SCADA Connection Timeout	The timeout for establishing connections.
SCADA Query Timeout	The timeout for executing queries. This attribute overwrites the [SQL]QueryTimeout INI parameter.

**Return Value**

The handle to the DB connection object if the connection is successful, otherwise -1 is returned. (For details call the SQLErrMsg() function). The handle identifies the DB connection object where details of the associated SQL connection to a DB are stored.

**Related Functions**

[SQLOpen](#), [SQLClose](#), [SQLDispose](#), [SQLConnect](#), [SQLDisconnect](#), [SQLInfo](#)

**Example**

```
/* Make a connection to an SQL server and select the name field
from each record in the employee database. */
FUNCTION
ListNames()
    INT hSQL;
    STRING sName;
    INT Status;
    INT hRec;
    INT nColumns;
    INT nRows;
    INT i;
```

```
hSQL = SQLCreate("DSN=MyDatabase;UID=billw;SRVR=CI1");
IF hSQL <> -1 THEN
    Status = SQLOpen(hSQL);
    IF Status = 0 THEN
        hRec = SQLGetRecordset(hSQL, "SELECT NAME FROM EMPLOYEE");
        IF hRec <> -1 THEN
            nRows = SQLRowCount(hRec);
            FOR i=0 TO nRows - 1 DO
                sName = SQLGetField(hRec, "NAME", i);
                ..
            END
            SQLEnd(hRec);
        ELSE
            Message("Information", SQLErrMsg(), 48);
        END
        SQLClose(hSQL);
    ELSE
        Message("Information", SQLErrMsg(), 48);
    END
    SQLDispose(hSQL);
END
END
```

## See Also

[SQL Functions](#)

## SQLDisconnect

Closes the SQL connection to a database and disposes the DB connection object specified by the function parameter.

This function is a blocking function and should not be called from a foreground task.

## Syntax

**SQLDisconnect(*hSQL*)**

*hSQL*:

The handle to the DB connection object, returned from the [SQLConnect\(\)](#) function. The handle identifies the DB connection object where details of the associated SQL connection are stored.

## Return Value

0 if success, otherwise an error code (for details call the [SQLErrMsg\(\)](#) function).

## Related Functions

[SQLCreate](#), [SQLOpen](#), [SQLClose](#), [SQLDispose](#), [SQLConnect](#), [SQLDisconnect](#), [SQLInfo](#)

## Example

See [SQLConnect](#).

## See Also

[SQL Functions](#)

## SQLDispose

Disposes the DB connection object specified by its parameter. If there is an active SQL connection to a database, the SQL connection has to be closed before disposal.

This function can be called in the foreground or background.

## Syntax

**SQLDispose(*hSQL*)**

*hSQL*:

The handle to the DB connection object, returned from either the [SQLCreate\(\)](#) or [SQLConnect\(\)](#) function. The handle identifies the DB connection object where details of the associated SQL connection are stored.

## Return Value

0 – if success, otherwise an error code (For details call the [SQLErrMsg\(\)](#) function).

## Related Functions

[SQLCreate](#), [SQLOpen](#), [SQLClose](#), [SQLDispose](#), [SQLConnect](#), [SQLDisconnect](#), [SQLInfo](#)

## Example

See [SQLCreate](#).

## See Also

[SQL Functions](#)

## SQLEnd

The function can be called with:

- The connection handle, in which case the function releases resources allocated for the default recordset for this connection. Generally, SCADA does the operation automatically each time when a new query is executed through [SQLExec](#) on the connection, but the operation can be also triggered manually. However, using the [SQLEnd](#) function aids efficiency and creates good programming standards. [SQLEnd](#) releases the memory that

was allocated when the last query was executed via SQLExec.

- The recordset handle, in which case the function then removes the disconnected recordset associated with the handle from the system. This operation has to be done each time when the recordset is no longer necessary. Not removing recordsets from the system can result in consuming all available memory and halting SCADA and other applications on a computer.

This function is a blocking function and should not be called from a foreground task.

## Syntax

**SQLEnd(*hGeneral*)**

*hGeneral*:

The handle either to:

The DB connection object, returned from either [SQLCreate](#) or [SQLConnect](#) function. The handle identifies the DB connection object where details of the associated SQL connection are stored.

Or:

The recordset.

## Return Value

0 (zero) if successful, otherwise an error code is returned. (For details on the 307 error code, call the [SQLErrMsg\(\)](#) function).

## Related Functions

[SQLCreate](#), [SQLOpen](#), [SQLClose](#), [SQLDispose](#), [SQLConnect](#), [SQLDisconnect](#), [SQLInfo](#), [SQLSet](#), [SQLAppend](#),  
[SQLExec](#), [SQLGetRecordset](#), [SQLCall](#), [SQLGetScalar](#), [SQLEnd](#)

## Example

See [SQLConnect](#).

## See Also

[SQL Functions](#)

## SQLErrMsg

Returns an error message from either a particular data object or entire data system. If a 307 error code occurs when one of the SQL functions is called, an SQL error message is generated. Call this function to get that error message.

This function can be called in the foreground or background.

## Syntax

**SQLErrMsg(*hGeneral*)**

*hGeneral*:

The handle to any data object either to the database connection object, recordset or query. The default value is -1 and then the function returns the error message for entire SQL system.

## Return Value

The error message (as a string).

## Related Functions

[SQLTraceOff](#).

## Example

See [SQLConnect](#).

## See Also

[SQL Functions](#)

## SQLExec

Executes an SQL query on a database. With this function, you can execute any SQL query or command supported by the SQL database.

---

**Note:** All types of fields can be requested in statements, but SCADA has to convert values of the fields to MBCS 8-bit strings which is not always possible. For example either single byte database strings or numbers can be converted to MBCS 8-bit strings, multi-byte strings can be converted to MBCS (their proper presentation depends on correct setup of SCADA and OS), while blobs cannot be encoded at all.

---

Data obtained by this function is stored in the default recordset. The default recordset is always a connected recordset. Connected recordsets fetch only small portion of data when the query is executed, but later they have to fetch further portions when recordset functions are used. This kind of recordset needs open connections to DB, but they need little memory.

The SQLNext() function needs to be called after the SQLExec() function before you can access data in the first record.

Only one query can be active at a time, so there is no need to end one query before you execute another query; each time you call SQLExec(), the previous query (through a previous SQLExec() call) is automatically ended. It means that each query executed from SQLExec cleans the default recordset.

Queries are queued for execution and a result from one query overwrites the result from preceding one. Similarly, Plant SCADA automatically ends the latest query when it disconnects the database, even if you have not called SQLEnd(). However, the SQLEnd() function aids efficiency; SQLEnd() releases the memory that was allocated when the latest query was executed.

This function is a blocking function and should not be called from a foreground task.

Queries which are built on the basis of user data, for example inputed by users via graphics pages or forms, may be prone to SQL Injection attacks. In such case, try to limit the risk by using CiCode functions from parameterized queries group and refer to a professional advice in this matter.

---

**Note:****SECURITY BREACH VIA SQL INJECTION**

- Validate all textbox entries using validation controls, regular expressions, code
  - Use parameterized SQL or stored procedures
  - Use a limited access account to connect to the database.
- 

## Syntax

**SQLExec(*hGeneral*, *sSelect*)**

*hGeneral*:

The handle either to the DB connection object (returned from either SQLCreate() or SQLConnect() function) or to the query handle (returned from SQLQueryCreate()). When it is the connection handle and *sSelect* is an empty string, the operation is performed on the first query in that DB connection object. When it is the query handle, the operation is performed on that query through the DB object which is associated to it.

*sSelect*:

The SQL query to be sent to the SQL database.

## Return Value

0 (zero) if successful, otherwise an error number is returned. (For details of the 307 error code, call the SQLErrMsg() function).

## Related Functions

[SQLCreate](#), [SQLOpen](#), [SQLClose](#), [SQLDispose](#), [SQLConnect](#), [SQLDisconnect](#), [SQLInfo](#), [SQLSet](#), [SQLAppend](#), [SQLExec](#), [SQLGetRecordset](#), [SQLCall](#), [SQLGetScalar](#), [SQLEnd](#)

## Example

These examples assume that the following tables are setup in a SQL server (with the name configured in Windows Control Panel) and opened with the SQLConnect() function:

### PEOPLE

SURNAME	FIRSTNAME	OCCUPATION	DEPARTMENT
MARTIAN	MARVIN	ENGINEER	MANAGEMENT
CASE	CARRIE	SUPPORT	SCADA
LIGHT	LARRY	PROGRAMMER	SCADA
BOLT	BETTY	ENGINEER	SYSTEMS

**PHONE**

SURNAME	NUMBER
MARTIAN	5551000
CASE	5551010
BOLT	5551020
LIGHT	5551030

Each SQL string (sSQL) should be encased within the SQLExec function, for example:

```
SQLExec(hSQL, sSQL);
To add a record to a table:
sSQL = "INSERT INTO PEOPLE (SURNAME, FIRSTNAME, OCCUPATION, DEPARTMENT)
        VALUES('ALLEN','MATTHEW','PROGRAMMER','SCADA')";
```

This SQL command changes the PEOPLE table to:

**PEOPLE**

SURNAME	FIRSTNAME	OCCUPATION	DEPARTMENT
MARTIAN	MARVIN	ENGINEER	MANAGEMENT
CASE	CARRIE	SUPPORT	SCADA
LIGHT	LARRY	PROGRAMMER	SCADA
BOLT	BETTY	ENGINEER	SYSTEMS
ALLEN	MATTHEW	PROGRAMMER	SCADA

To remove records from a table:

```
sSQL = "DELETE FROM (PEOPLE, PHONE) WHERE SURNAME='MARTIAN'";
SQLBeginTran(hSQL);
SQLExec(hSQL,sSQL);
IF (Message("Alert", "Do you really want to DELETE MARTIAN", 33) = 0) THEN
    SQLCommit(hSQL);
ELSE
    SQLRollback(hSQL);
END
```

Assuming that OK was clicked on the Message Box, the tables change to:

**PEOPLE**

SURNAME	FIRSTNAME	OCCUPATION	DEPARTMENT
CASE	CARRIE	SUPPORT	SCADA
LIGHT	LARRY	PROGRAMMER	SCADA
BOLT	BETTY	ENGINEER	SYSTEMS

## PHONE

SURNAME	NUMBER
CASE	5551010
BOLT	5551020
LIGHT	5551030

**To change a record:**

```
sSQL = "UPDATE PEOPLE SET OCCUPATION='SUPPORT' WHERE FIRSTNAME='LARRY'" ;
```

This SQL command changes the PEOPLE table to:

**PEOPLE**

SURNAME	FIRSTNAME	OCCUPATION	DEPARTMENT
MARTIAN	MARVIN	ENGINEER	MANAGEMENT
CASE	CARRIE	SUPPORT	SCADA
LIGHT	LARRY	SUPPORT	SCADA
BOLT	BETTY	ENGINEER	SYSTEMS

**To select a group of records from a table:**

```
sSQL = "SELECT SURNAME FROM PEOPLE WHERE OCCUPATION='ENGINEER'" ;
```

This SQL command will return the following table back to Plant SCADA. The table can then be accessed by the SQLNext() function and the SQLGetField() functions.

Plant SCADA table for hSQL

SURNAME
MARTIAN
BOLT

You can also select data using a much more complete SQL string, for example:

```
sSQL = "SELECT (SURNAME, OCCUPATION, NUMBER) FROM (PEOPLE, PHONE)  
WHERE DEPARTMENT='SCADA' AND PEOPLE.SURNAME = PHONE.SURNAME" ;
```

This SQL command retrieves the following table:

SURNAME	OCCUPATION	NUMBER
CASE	SUPPORT	5551010
LIGHT	PROGRAMMER	5551030

**To extract information from a table:**

```
STRING sInfo[3][10]  
int i = 0;  
WHILE ((SQLNext(hSQL) = 0) and (i < 10)) DO
```

```

sInfo[0][i] = SQLGetField(hSQL, "SURNAME");
sInfo[1][i] = SQLGetField(hSQL, "OCCUPATION");
sInfo[2][i] = SQLGetField(hSQL, "NUMBER");
END

```

This code example leaves the information in the sInfo two dimensional array as follows:

### SInfo

	<b>0</b>	<b>1</b>	<b>2</b>
0	CASE	SUPPORT	5551010
1	LIGHT	PROGRAMMER	5551030
2			
3			
4			
...			

## See Also

[SQL Functions](#)

### SQLFieldInfo

Gets information about the fields or columns selected by a SQL query. The function returns the name and width of the specified field. If you call the function within a loop, you can return the names and sizes of all the fields in the database.

When the *hGeneral* is the connection handle, the function returns information about the default recordset.

When the *hGeneral* is the recordset handle, the function returns information about the recordset itself.

Keywords such as "DATE", "TIME", and "DESC" cannot be used as field names by some database systems. To use fields with these names, you need to append underscores to the names (for example, "TIME\_", "DATE\_", "DESC\_").

This function can be called in the foreground or background.

Some combinations of database drivers and databases can return no name for certain classes of fields/columns, for example for aggregation queries such as "select SUM". In those cases such fields can be either explicitly named in the query itself, or accessed via a unified name in the format of "Field{ColumnNumber}" where {ColumnNumber} is the number of the requested field/column in the recordset.

## Syntax

**SQLFieldInfo(*hGeneral*, *hField*, *sName*, *Width*)**

*hGeneral*:

The handle either to:

The DB connection object, returned from either [SQLCreate](#) or [SQLConnect](#) function. The handle identifies the DB

connection object where details of the associated SQL connection are stored.

Or:

The recordset.

*hField*:

The field (or column) handle, indicating the position of the field in the database.

*sName*:

Output Parameter: A string in which the function stores the field name. The argument is returned by the function. Must be a String type variable.

*Width*:

Output Parameter: An integer in which the function stores the maximum number of characters in the field. The argument is returned by the function. Must be an Integer type variable.

## Return Value

0 (zero) if successful, otherwise an error number is returned. (For details of the 307 error code, call the [SQLErrMsg](#) function).

## Related Functions

[SQLSet](#), [SQLAppend](#), [SQLExec](#), [SQLGetRecordset](#), [SQLCall](#), [SQLGetScalar](#), [SQLEnd](#), [SQLNumChange](#), [SQLNoFields](#), [SQLNumFields](#), [SQLGetField](#), [SQLIsNullField](#), [SQLRowCount](#), [SQLNext](#), [SQLPrev](#)

## Example

```
! Lists all fields in the Employee database
FUNCTION
ListFields()
    INT hSQL;
    STRING sField;
    INT Count;
    INT Width;
    INT Index;
    SQLTraceOn("C:\DATA\TRACE.LOG");
    hSQL = SQLConnect("DRV=QEDBF");
    SQLExec(hSQL, "SELECT * FROM C:\DATA\EMPLOYEE");
    Count = SQLNoFields(hSQL);
    Index = 0;
    WHILE Index < Count DO
        SQLFieldInfo(hSQL,Index,sField,Width);
        ..
    END
    SQLEnd(hSQL);
    SQLDisconnect(hSQL);
    SQLTraceOff();
END
```

## See Also

[SQL Functions](#)

### SQLGetField

Gets field or column data from a database field. To get to a specific record in the recordset, use either the [SQLNext\(\)](#) or [SQLPrev\(\)](#) functions or, in case of disconnected recordsets, the *nRowIndex* argument.

When the *hGeneral* is the connection handle, the function returns information from the default recordset. When the *hGeneral* is the recordset handle, the function returns information from the recordset itself.

---

**Note:** All types of fields can be requested in statements, but SCADA has to convert values of the fields to MBCS 8-bit strings which is not always possible. For example either single byte database strings or numbers can be converted to MBCS 8-bit strings, multi-byte strings can be converted to MBCS (their proper presentation depends on correct setup of SCADA and OS), while blobs cannot be encoded at all.

---

Keywords such as "DATE", "TIME", and "DESC" cannot be used as field names by some database systems. To use fields with these names, you need to append underscores to the names (for example, "TIME\_", "DATE\_", "DESC\_").

This function can be called in the foreground or background.

Some combinations of database drivers and databases can return no name for certain classes of fields/columns, for example for aggregation queries such as "select SUM". In those cases such fields can be either explicitly named in the query itself, or accessed via a unified name in the format of "Field{ColumnNumber}" where {ColumnNumber} is the number of the requested field/column in the recordset.

## Syntax

**SQLGetField(*hGeneral*, *sField*, *nRowIndex*)**

*hGeneral*:

The handle either to:

The DB connection object, returned from either [SQLCreate\(\)](#) or [SQLConnect\(\)](#) function. The handle identifies the DB connection object where details of the associated SQL connection are stored.

Or:

The recordset.

*sField*:

The name of the field or column.

*nRowIndex*:

If *hGeneral* is a connection handle and *nRowIndex* is not equal -1, then the function returns an empty string.

## Return Value

The field or column data (as a string).

The maximum length of the return data is 255 characters. If the returned data is longer than this, the function will return error 306 (can be checked using the [IsError\(\)](#) Cicode function).

## Related Functions

[SQLSet](#), [SQLAppend](#), [SQLExec](#), [SQLGetRecordset](#), [SQLCall](#), [SQLGetScalar](#), [SQLEnd](#), [SQLNumChange](#), [SQLNoFields](#), [SQLNumFields](#), [SQLFieldInfo](#), [SQLGetField](#), [SQLIsNullField](#), [SQLRowCount](#), [SQLNext](#), [SQLPrev](#)

## Example

See [SQLConnect](#).

## See Also

[SQL Functions](#)

### SQLGetRecordset

Executes an SQL query on a database and returns a handle to any resulting disconnected recordset. All recordsets created in this way should be closed by the [SQLEnd\(\)](#) function. If unsuccessful, the function returns an invalid handle and an error code is returned (can be tested by [IsError\(\)](#)). In case of no data returned, the function returns an invalid handle and no error. With this function, you can execute any SQL query or command supported by the SQL database, but it is designed mainly to work with queries returning data like for example `SELECT`. Queries which don't return database data, like `INSERT` or `UPDATE`, should be executed via [SQLCall](#).

Recordsets produced by this function are disconnected. The disconnected recordsets fetch the data from the DB when the query is executed and thus they don't need an open connection when recordset functions are used. This kind of recordset improves speed of access to data after executing the query, but may consume significant amounts of memory.

Creating disconnected recordsets without releasing them may lead to utilizing all available memory, which can result in degraded performance.

#### NOTICE

#### EXCESSIVE MEMORY USAGE

Limit the number of disconnected recordsets using the [\[SQL\]MaxConnections](#) parameter. Release unused disconnected recordsets. Test projects using disconnected recordsets prior to deployment in production.

**Failure to follow these instructions can result in equipment damage.**

**Note:** All types of fields can be requested in statements, but SCADA has to convert values of the fields to MBCS 8-bit strings which is not always possible. For example either single byte database strings or numbers can be converted to MBCS 8-bit strings, multi-byte strings can be converted to MBCS (their proper presentation depends on correct setup of SCADA and OS), while blobs cannot be encoded at all.

This function is a blocking function and should not be called from a foreground task.

Queries which are built on the basis of user data, for example inputed by users via graphics pages or forms, may be prone to SQL Injection attacks. In such case, try to limit the risk by using CiCode functions from parameterized queries group and refer to a professional advice in this matter.

#### NOTICE

#### SECURITY BREACH VIA SQL INJECTION

- Validate all textbox entries using validation controls, regular expressions and code
- Use parameterized SQL or stored procedures
- Use a limited access account to connect to the database

**Failure to follow these instructions can result in equipment damage.**

## Syntax

**SQLGetRecordset(*hGeneral*, *sSelect*)**

*hGeneral*:

The handle either to the DB connection object (returned from either [SQLCreate\(\)](#) or [SQLConnect\(\)](#) function) or to the query handle (returned from [SQLQueryCreate\(\)](#)). When it is the connection handle and *sSelect* is an empty string, the operation is performed on the first query in that DB connection object. When it is the query handle, the operation is performed on that query through the DB object which is associated to it.

*sSelect*:

The SQL query to be sent to the SQL database.

## Return Value

The handle to a recordset holding the result of the query. If unsuccessful an invalid handle and an error code are returned (for details call the [SQLErrMsg\(\)](#) function). If no data is returned, an invalid handle and no error message are returned. The presence of an error can be tested by calling the [IsError\(\)](#) Cicode function

## Related Functions

[SQLCreate](#), [SQLOpen](#), [SQLClose](#), [SQLDispose](#), [SQLConnect](#), [SQLDisconnect](#), [SQLInfo](#), [SQLSet](#), [SQLAppend](#), [SQLExec](#), [SQLCall](#), [SQLGetScalar](#), [SQLEnd](#)

## Example

See [SQLCreate](#).

## See Also

[SQL Functions](#)

## SQLGetScalar

Executes an SQL query on a database. The value from the first column of the first row is returned. With this function, you can execute any SQL query or command supported by the SQL database. If it doesn't return any data (like INSERT or UPDATE), a respective error code is set which can be tested by calling [IsError\(\)](#).

**Note:** All types of fields can be requested in statements, but SCADA has to convert values of the fields to MBCS 8-bit strings which is not always possible. For example either single byte database strings or numbers can be

---

converted to MBCS 8-bit strings, multi-byte strings can be converted to MBCS (their proper presentation depends on correct setup of SCADA and OS), while blobs cannot be encoded at all.

---

This function is a blocking function and should not be called from a foreground task.

Queries which are built on the basis of user data, for example inputed by users via graphics pages or forms, may be prone to SQL Injection attacks. In such case, try to limit the risk by using CiCode functions from parameterized queries group and refer to a professional advice in this matter.

#### NOTICE

#### SECURITY BREACH VIA SQL INJECTION

- Validate all textbox entries using validation controls, regular expressions and code
- Use parameterized SQL or stored procedures
- Use a limited access account to connect to the database

**Failure to follow these instructions can result in equipment damage.**

## Syntax

**SQLGetScalar(*hGeneral*, *sSelect*, *isNull*)**

*hGeneral*:

The handle either to the DB connection object (returned from either [SQLCreate](#) or [SQLConnect](#) function) or to the query handle (returned from [SQLQueryCreate](#)). When it is the connection handle and *sSelect* is an empty string, the operation is performed on the first query in that DB connection object. When it is the query handle, the operation is performed on that query through the DB object which is associated to it.

*sSelect*:

The SQL query to be sent to the SQL database.

*isNull*:

Output Parameter: Indicated whether the returned variable is NULL. The argument is returned by the function.

## Return Value

String representing a value from the first column and the first row of the result of executing the SQL query. If the value is NULL, the string is empty and *isNull* parameter is set to TRUE. If there are no records in the result, the string is empty and the error code is set to 294. For details of the 307 error code, call the [SQLErrMsg\(\)](#) function. The presence of error code can be tested by calling the [IsError\(\)](#) CiCode function.

## Related Functions

[SQLCreate](#), [SQLOpen](#), [SQLClose](#), [SQLDispose](#), [SQLConnect](#), [SQLDisconnect](#), [SQLInfo](#), [SQLSet](#), [SQLAppend](#), [SQLExec](#), [SQLGetRecordset](#), [SQLCall](#), [SQLEnd](#)

## See Also

[SQL Functions](#)

## SQLInfo

Gets information about a database connection, recordset or query properties.

This function can be called in the foreground or background.

## Syntax

**SQLInfo(*hGeneral*, *Type*)**

*hGeneral*:

The handle either to:

The DB connection object, returned from either [SQLCreate](#) or [SQLConnect](#) function. The handle identifies the DB connection object where details of the associated SQL connection are stored.

Or:

The recordset.

Or:

The query.

*nType*:

The type of information to get:

0 - The connection string.

1 - The current SQL query for connection handles and query handles or the source SQL query for recordset handles.

2 - The current database filename (only works with SQL device).

3 - No longer supported.

4 - No longer supported. Now returns an empty string and sets the SQL error to 'Invalid Type'.

5 - The DB connection object's state: Closed (The connection is closed), Open (The connection is open), Connecting (The connection object is connecting to the data source. For future release), Executing (The connection object is executing a command. For future release), Fetching (The connection object is retrieving data. For future release), Broken (The connection to the data source is broken. This can occur only after the connection has been opened. A connection in this state may be closed and then re-opened. For future release). An empty string means either error occurred or the handle is invalid.

6 - The handle to the connection if used on either recordset or query handle.

7 - Checks whether a new transaction can be opened. The function perform a check how many transaction levels have been already opened and returns TRUE if a new one can be opened, otherwise FALSE. The function doesn't check ability of the external DB to open transactions. The functions returns FALSE for recordset handles.

---

**Note:** There is no guarantee that the objects behind the connection handles still exist in the system.

---

## Return Value

The information (as a string).

## Related Functions

[SQLCreate](#), [SQLOpen](#), [SQLClose](#), [SQLDispose](#), [SQLConnect](#), [SQLDisconnect](#)

## Example

```
SQLInfo(1,2);
```

## See Also

[SQL Functions](#)

## SQLIsNullField

Checks presence of null value in field from a recordset. To get to a specific record in the recordset, use either the [SQLNext\(\)](#)/[SQLPrev\(\)](#) functions or the *nRowIndex* argument.

When the *hGeneral* is the connection handle, the function returns information from the default recordset. When the *hGeneral* is the recordset handle, the function returns information from the recordset itself.

This function can be called in the foreground or background.

## Syntax

**SQLIsNullField**(*hGeneral*, *sField*, *nRowIndex*=-1)

*hGeneral*:

The handle either to:

The DB connection object, returned from either [SQLCreate](#) or [SQLConnect](#) function. The handle identifies the DB connection object where details of the associated SQL connection are stored.

Or:

The recordset.

*sField*:

The name of the field or column.

*nRowIndex*:

The number of the row which data is requested. The first row has index 0. -1 means that an internal pointer is used instead. The pointer can be moved forward and backward by executing [SQLNext\(\)](#) or [SQLPrev\(\)](#) functions.

## Return Value

TRUE if the value is NULL, FALSE otherwise.

## Related Functions

[SQLSet](#), [SQLAppend](#), [SQLExec](#), [SQLGetRecordset](#), [SQLCall](#), [SQLGetScalar](#), [SQLEnd](#), [SQLNumChange](#), [SQLNoFields](#), [SQLNumFields](#), [SQLFieldInfo](#), [SQLGetField](#), [SQLIsNullField](#), [SQLRowCount](#), [SQLNext](#), [SQLPrev](#)

## See Also

[SQL Functions](#)

### SQLNext

Gets the next database record from an SQL query. Use the [SQLExec/SQLGetRecordset](#) function to select a number of records or rows from the SQL database, and then use the [SQLNext](#) function to step through each record separately.

When the *hGeneral* is the connection handle, the function moves in the default recordset. When the *hGeneral* is the recordset handle, the function moves in the recordset itself.

This function can be called in the foreground or background.

## Syntax

**SQLNext(*hGeneral*)**

*hGeneral*:

The handle either to:

The DB connection object, returned from either [SQLCreate](#) or [SQLConnect](#) function. The handle identifies the DB connection object where details of the associated SQL connection are stored.

Or:

The recordset.

## Return Value

0 (zero) if successful, otherwise an error number is returned. (For details of the 307 error code, call the [SQLErrMsg](#) function).

## Related Functions

[SQLSet](#), [SQLAppend](#), [SQLExec](#), [SQLGetRecordset](#), [SQLGetScalar](#), [SQLEnd](#), [SQLNumChange](#), [SQLNoFields](#), [SQLNumFields](#), [SQLFieldInfo](#), [SQLGetField](#), [SQLIsNullField](#), [SQLRowCount](#), [SQLPrev](#)

## Example

See [SQLConnect](#).

## See Also

[SQL Functions](#)

### SQLNoFields

When the *hGeneral* is the connection handle, the function returns the number of fields or columns that were

returned by the last SQL statement. When the *hGeneral* is the recordset handle, the function returns the number of fields or columns in the recordset itself.

---

**Note:** This function is deprecated and may be removed in future releases.

This function can be called in the foreground or background.

## Syntax

**SQLNoFields(*hGeneral*)**

*hGeneral*:

The handle either to:

The DB connection object, returned from either [SQLCreate\(\)](#) or [SQLConnect\(\)](#) function. The handle identifies the DB connection object where details of the associated SQL connection are stored.

Or:

The recordset.

## Return Value

The number of fields. A value of 0 is returned if no fields were returned or if an error has been detected. (For details of an error code, call the [SQLErrMsg](#) function).

## Related Functions

[SQLSet](#), [SQLAppend](#), [SQLExec](#), [SQLGetRecordset](#), [SQLCall](#), [SQLGetScalar](#), [SQLEnd](#), [SQLNumChange](#), [SQLNoFields](#), [SQLNumFields](#), [SQLFieldInfo](#), [SQLGetField](#), [SQLIsNullField](#), [SQLRowCount](#), [SQLNext](#), [SQLPrev](#)

## Example

See [SQLFieldInfo](#).

## See Also

[SQL Functions](#)

## SQLNumChange

Gets the number of records that were modified in the last SQL Insert, Update, or Delete statement.

This function can be called in the foreground or background.

## Syntax

**SQLNumChange(*hSQL*)**

*hSQL*:

The handle to the DB connection object, returned from either [SQLCreate\(\)](#) or [SQLConnect\(\)](#) function. The handle

identifies the DB connection object where details of the associated SQL connection are stored.

## Return Value

The number of records that were modified. A value of 0 is returned if no fields were returned or if an error has occurred. (For details of an error code, call the [SQLErrMsg](#) function).

## Related Functions

[SQLSet](#), [SQLAppend](#), [SQLExec](#), [SQLGetRecordset](#), [SQLCall](#), [SQLGetScalar](#), [SQLEnd](#), [SQLNoFields](#), [SQLNumFields](#), [SQLFieldInfo](#), [SQLGetField](#), [SQLIsNullField](#), [SQLRowCount](#), [SQLNext](#), [SQLPrev](#)

## Example

See [SQLBeginTran](#).

## See Also

[SQL Functions](#)

## SQLNumFields

When *hGeneral* is the connection handle, the function returns the number of fields or columns that were returned by the last SQL statement. When *hGeneral* is the recordset handle, the function returns the number of fields or columns in the recordset itself.

This function can be called in the foreground or background.

## Syntax

**SQLNumFields(*hGeneral*)**

*hGeneral*:

The handle either to:

The DB connection object, returned from either [SQLCreate\(\)](#) or [SQLConnect\(\)](#) function. The handle identifies the DB connection object where details of the associated SQL connection are stored.

Or:

The recordset.

## Return Value

The number of fields. A value of 0 is returned if no fields were returned or if an error has been detected. (For details of an error code, call the [SQLErrMsg](#) function).

## Related Functions

[SQLSet](#), [SQLAppend](#), [SQLExec](#), [SQLGetRecordset](#), [SQLCall](#), [SQLGetScalar](#), [SQLEnd](#), [SQLNumChange](#), [SQLNoFields](#), [SQLNumFields](#), [SQLFieldInfo](#), [SQLGetField](#), [SQLIsNullField](#), [SQLRowCount](#), [SQLNext](#), [SQLPrev](#)

## Example

See [SQLFieldInfo](#).

## See Also

[SQL Functions](#)

## SQLOpen

Opens an SQL connection between the DB connection object specified by the function's parameter and the database defined by the connection string given before as the parameter to either [SQLCreate](#) or [SQLConnect](#) function.

---

### Note:

Currently there are certain configurations of providers, connection strings and database systems which can automatically close connections for the sake of saving the database systems' resources; for example MS SQL Server dedicated provider with activated pooling closes the connection to MS SQL Server when there is no data exchange between the client and the DB server for some predefined time. If such behaviour is observed and is not desired from the system design point of view, it can be either:

- Switched off by finding and using respective attributes in the connection string (if existing and supported by DB) or
- By actively checking error codes returned from SQL CiCode functions and reconnecting by using SQLOpen if the connection is dropped.

The actual state of connections can be checked by SQLInfo with type 5.

---

This function is a blocking function and should not be called from a foreground task.

## Syntax

**SQLOpen(*hSQL*)**

*hSQL*:

The handle to the DB connection object, returned from either the [SQLCreate\(\)](#) or [SQLConnect\(\)](#) function. The handle identifies the DB connection object where details of the associated SQL connection are stored.

## Return Value

0 if success, otherwise an error code (For details call the [SQLErrMsg\(\)](#) function)

## Related Functions

[SQLCreate](#), [SQLOpen](#), [SQLClose](#), [SQLDispose](#), [SQLConnect](#), [SQLDisconnect](#), [SQLInfo](#)

## Example

See [SQLCreate](#).

## See Also

[SQL Functions](#)

### SQLParamsClearAll

Remove all parameters associated with a particular connection object.

Each database provider (Odbc, OleDb, SQL Server) uses parameterized queries in a different way. It is recommended that you look at documentation and examples included with your database.

This function is a blocking function and should not be called from a foreground task.

Building queries from pieces (SQLSet, SQLAppend) or adding parameters to either queries or connections (SQLParam functions) requires a few calls to respective CiCode functions. If a few functions try to manipulate the same connection in the same time some conflicts and unintended operations may occur. It is a typical multithreading problem.

To avoid this, instead of manipulating connections, consider using locally created and locally disposed queries.  
For example:

```
int function SAFE_SQL_CICODE_MULTITHREAD_USE()
//locally created query
int hStmt = SQLQueryCreate(hConnection);

//Set the query
SQLSet(hStmt, "select * from TAB where NAME=@Name");

//Add parameters to the query
SQLParamsSetAsString(hStmt, "Name", "Aaa");

//Execute the query
SQLGetRecordset(hStmt, "");

//the locally created query is disposed
SQLQueryDispose(hStmt);
End
```

## Syntax

**SQLParamsClearAll(*hSQL*)**

*hSQL*:

The handle to the DB connection object, returned from either [SQLCreate\(\)](#) or [SQLConnect\(\)](#) function. The handle identifies the DB connection object where details of the associated SQL connection are stored.

## Return Value

0 (zero) if successful, otherwise an error number is returned. (For details of the 307 error code, call the

[SQLErrMsg](#) function.)

## Related Functions

[SQLSet](#), [SQLAppend](#), [SQLExec](#), [SQLGetRecordset](#), [SQLCall](#), [SQLGetScalar](#), [SQLEnd](#), [SQLParamsSetAsInt](#),  
[SQLParamsSetAsReal](#), [SQLParamsSetAsString](#)

## Example

See [SQLParamsSetAsInt](#).

## See Also

[SQL Functions](#)

### SQLParamsSetAsInt

Adds or replace a parameterized query parameter and its value in the specified connection. The value of the parameter is given as integer.

Each database provider (Odbc, OleDb, SQL Server) uses parameterized queries in a different way. It is recommended that you look at documentation and examples included with your database.

Building queries from pieces (SQLSet, SQLAppend) or adding parameters to either queries or connections (SQLParam functions) requires a few calls to respective CiCode functions. If a few functions try to manipulate the same connection in the same time some conflicts and unintended operations may occur. It is a typical multithreading problem.

To avoid this, instead of manipulating connections, consider using locally created and locally disposed queries. For example:

```
int function SAFE_SQL_CICODE_MULTITHREAD_USE()
    //locally created query
    int hStmt = SQLQueryCreate(hConnection);
    //Set the query
    SQLSet(hStmt, "select * from TAB where NAME=@Name");
    //Add parameters to the query
    SQLParamsSetAsString(hStmt, "Name", "Aaa");
    //Execute the query
    SQLGetRecordset(hStmt, "");
    //the locally created query is disposed
    SQLQueryDispose(hStmt);
End
```

This function is a blocking function and should not be called from a foreground task.

## Syntax

**SQLParamsSetAsInt**(*hSQL*, *ParamName*, *ParamValue*)

*hSQL*:

The handle to the DB connection object, returned from either [SQLCreate\(\)](#) or [SQLConnect\(\)](#) function. The handle identifies the DB connection object where details of the associated SQL connection are stored.

**ParamName:**

The name of the parameter to add or change.

**ParamValue:**

The value of the parameter as an integer.

## Return Value

0 (zero) if successful, otherwise an error number is returned. (For details of the 307 error code, call the [SQLErrMsg](#) function).

## Related Functions

[SQLSet](#), [SQLAppend](#), [SQLExec](#), [SQLGetRecordset](#), [SQLCall](#), [SQLGetScalar](#), [SQLEnd](#), [SQLParamsSetAsReal](#), [SQLParamsSetAsString](#), [SQLParamsClearAll](#)

## Examples

The following examples assume that the following table is setup and opened for the three data providers, ODBC, OleDb and SQLClient, respectively:

```
PEOPLE
SURNAME FIRSTNAME AGE HEIGHT
MARTIAN MARVIN 27 1.78
CASE CARRIE 18 1.73
```

### ODBC

A parameter is identified by a character "?" in the SQL query. Since the protocol uses a sequential approach for statement parameterization, the parameters order matters. In that case, to confirm that the correct parameters are picked up by the query, it is suggested to clear all the parameters after a query is executed and configure new ones in a correct order for another query unless the parameters are exactly the same in terms of value and position in the sequence between the two queries:

```
INT nError = 0;
STRING sValue0 = "";
STRING sValue1 = "";
INT nIsNull = 0;
INT hQueryDelete = SQLQueryCreate(hSQL);
SQLSet(hQueryDelete, "delete from PEOPLE where SURNAME=? and FIRSTNAME=?");
SQLParamsClearAll(hSQL);
//the name of the parameter does not matter
SQLParamsSetAsString(hSQL, "sName", "CASE");
SQLParamsSetAsString(hSQL, "fName", "CARRIE");
SQLCall(hQueryDelete, "");
SQLParamsClearAll(hSQL);
SQLParamsSetAsString(hSQL, "sName", "JACKSON");
SQLParamsSetAsString(hSQL, "fName", "DAVID");
SQLParamsSetAsInt(hSQL, "nAge", 28);
SQLParamsSetAsReal(hSQL, "nHeight", 1.85);
SQLCall(hSQL, "insert into PEOPLE (SURNAME, FIRSTNAME, AGE, HEIGHT) values (?,?,?,?)");
SQLParamsClearAll(hSQL);
SQLParamsSetAsString(hSQL, "fName", "DAVID");
SQLParamsSetAsString(hSQL, "sName", "JACKSON");
```

```
sValue0 = SQLGetScalar(hSQL, "select AGE from PEOPLE where FIRSTNAME=? and SURNAME=?",
nIsNull);
sValue1 = SQLGetScalar(hSQL, "select HEIGHT from PEOPLE where FIRSTNAME=? And SURNAME=?",
nIsNull);
```

**OleDb**

Same to ODBC, the association and later substitution is based on order of the parameters in a query statement and "?" is used as the mark. Thus the last example for ODBC also works for general cases of OleDb. Additionally, for some databases, i.e. Microsoft Access, user may have another option: parameter name with "@" prefix. Following examples are specific for communicating with Microsoft Access through OleDb data protocol.

```
INT nError = 0;
STRING sValue0 = "";
STRING sValue1 = "";
INT nIsNull = 0;
INT hQueryDelete = SQLQueryCreate(hSQL);
SQLSet(hQueryDelete, "delete from PEOPLE where SURNAME=@sName and FIRSTNAME=@fName");
SQLParamsClearAll(hSQL);
SQLParamsSetAsString(hSQL, "sName", "CASE");
SQLParamsSetAsString(hSQL, "fName", "CARRIE");
SQLCall(hQueryDelete, "");
SQLParamsClearAll(hSQL);
SQLParamsSetAsString(hSQL, "sName", "JACKSON");
SQLParamsSetAsString(hSQL, "fName", "DAVID");
SQLParamsSetAsInt(hSQL, "nAge", 28);
SQLParamsSetAsReal(hSQL, "nHeight", 1.85);
SQLCall(hSQL, "insert into PEOPLE (SURNAME, FIRSTNAME, AGE, HEIGHT) values (@sName, @fName,
@sName, @nHeight)");
SQLParamsClearAll(hSQL);
SQLParamsSetAsString(hSQL, "fName", "DAVID");
SQLParamsSetAsString(hSQL, "sName", "JACKSON");
sValue0 = SQLGetScalar(hSQL, "select AGE from PEOPLE where FIRSTNAME=@fName and
SURNAME=@sName", nIsNull);
sValue1 = SQLGetScalar(hSQL, "select HEIGHT from PEOPLE where FIRSTNAME=@fName And
SURNAME=@sName", nIsNull);
```

**Note:** If you want to use a parameter more than once in the same query, there is no need to define it multiple times. However, the parameters have to be prepared in proper order based on their occurrence order in the query statement.

**Example**

```
INT nError = 0;
STRING sValue0 = "";
STRING sValue1 = "";
INT nIsNull = 0;
INT hQueryInsert = SQLQueryCreate(hSQL);
SQLSet(hQueryInsert, "insert into TABLE (COLUMN0, COLUMN1, COLUMN2, COLUMN3, COLUMN4)
values (@param0, @param0, @param1, @param1, @param0");
SQLParamsClearAll(hSQL);
SQLParamsSetAsString(hSQL, "param0", "Value0");
SQLParamsSetAsString(hSQL, "param1", "Value1");
SQLCall(hQueryInsert, "");
The result will be:
TABLE
COLUMN0 COLUMN1 COLUMN2 COLUMN3 COLUMN4
Value0 Value0 Value1 Value1 Value0
```

**SQLClient**

SQL server uses a named parameter approach for parameterization. Parameters in queries are preceded by the '@' character. The order of the parameters does not matter.

```
INT nError = 0;
STRING sValue0 = "";
STRING sValue1 = "";
INT nIsNull = 0;
INT hQueryDelete = SQLQueryCreate(hSQL);
SQLSet(hQueryDelete, "delete from PEOPLE where SURNAME=@sName and FIRSTNAME=@fName");
SQLParamsSetAsString(hSQL, "sName", "CASE");
SQLParamsSetAsString(hSQL, "fName", "CARRIE");
SQLCall(hQueryDelete, "");
//Order does not matter
SQLParamsSetAsInt(hSQL, "nAge", 28);
SQLParamsSetAsReal(hSQL, "nHeight", 1.85);
//If a parameter has been defined already, setting the value again //will replace the old
value.
SQLParamsSetAsString(hSQL, "sName", "JACKSON");
SQLParamsSetAsString(hSQL, "fName", "DAVID");
SQLCall(hSQL, "insert into PEOPLE (SURNAME, FIRSTNAME, AGE, HEIGHT) values (@sName, @fName,
@nAge, @nHeight)");
sValue0 = SQLGetScalar(hSQL, "select AGE from PEOPLE where FIRSTNAME=@fName and
SURNAME=@sName", nIsNull);
sValue1 = SQLGetScalar(hSQL, "select HEIGHT from PEOPLE where FIRSTNAME=@fName And
SURNAME=@sName", nIsNull);
```

By the given Cicode examples, the table will be updated to:

```
PEOPLE
SURNAME FIRSTNAME AGE HEIGHT
MARTIAN MARVIN 27 1.78
JACKSON DAVID 28 1.85
```

## See Also

[SQL Functions](#)

### SQLParamsSetAsReal

Adds or replaces a parameterized query's parameter and its value in the specified connection. The value of the parameter is given as a real.

Each database provider (Odbc, OleDb, SQL Server) uses parameterized queries in a different way. It is recommended that you look at documentation and examples included with your database.

This function is a blocking function and should not be called from a foreground task.

Building queries from pieces (SQLSet, SQLAppend) or adding parameters to either queries or connections (SQLParam functions) requires a few calls to respective CiCode functions. If a few functions try to manipulate the same connection in the same time some conflicts and unintended operations may occur. It is a typical multithreading problem.

To avoid this, instead of manipulating connections, consider using locally created and locally disposed queries. For example:

```
int function SAFE_SQL_CICODE_MULTITHREAD_USE()
//locally created query
int hStmt = SQLQueryCreate(hConnection);
```

```
//Set the query
SQLSet(hStmt, "select * from TAB where NAME=@Name");

//Add parameters to the query
SQLParamsSetAsString(hStmt, "Name", "Aaa");

//Execute the query
SQLGetRecordset(hStmt, "");

//the locally created query is disposed
SQLQueryDispose(hStmt);
End
```

## Syntax

**SQLParamsSetAsReal(*hSQL*, *ParamName*, *ParamValue*)**

*hSQL*:

The handle to the DB connection object, returned from either SQLCreate() or SQLConnect() function. The handle identifies the DB connection object where details of the associated SQL connection are stored.

*ParamName*:

The name of the parameter to add or change.

*ParamValue*:

The value of the parameter as a real.

## Return Value

0 (zero) if successful, otherwise an error number is returned. (For details of the 307 error code, call the [SQLErrMsg](#) function).

## Related Functions

[SQLSet](#), [SQLAppend](#), [SQLExec](#), [SQLGetRecordset](#), [SQLCall](#), [SQLGetScalar](#), [SQLEnd](#), [SQLParamsSetAsInt](#), [SQLParamsSetAsString](#), [SQLParamsClearAll](#)

## Example

See [SQLParamsSetAsInt](#).

## See Also

[SQL Functions](#)

## SQLParamsSetAsString

Adds or replaces a parameterized query's parameter and its value in the specified connection. The value of the

parameter is given as a string.

Each database provider (Odbc, OleDb, SQL Server) uses parameterized queries in a different way. It is recommended that you look at documentation and examples included with your database.

Building queries from pieces (SQLSet, SQLAppend) or adding parameters to either queries or connections (SQLParam functions) requires a few calls to respective CiCode functions. If a few functions try to manipulate the same connection in the same time some conflicts and unintended operations may occur. It is a typical multithreading problem.

To avoid this, instead of manipulating connections, consider using locally created and locally disposed queries. For example:

```
int function SAFE_SQL_CICODE_MULTITHREAD_USE()
//locally created query
int hStmt = SQLQueryCreate(hConnection);

//Set the query
SQLSet(hStmt, "select * from TAB where NAME=@Name");

//Add parameters to the query
SQLParamsSetAsString(hStmt, "Name", "Aaa");

//Execute the query
SQLGetRecordset(hStmt, "");

//the locally created query is disposed
SQLQueryDispose(hStmt);
End
```

This function is a blocking function and should not be called from a foreground task.

## Syntax

**SQLParamsSetAsString(*hSQL*, *ParamName*, *ParamValue*, *nStrType*)**

*hSQL*:

The handle to the DB connection object, returned from either SQLCreate() or SQLConnect() function. The handle identifies the DB connection object where details of the associated SQL connection are stored.

*ParamName*:

The name of the parameter to add or change.

*ParamValue*:

The value of the parameter as a string.

*nStrType*:

The index specifying the type of the string:

0 - Allowing ADO engine to use the default string type of the protocol. For SQLClient, OleDb and ODBC, the default string type is NVarChar (A variable-length stream of Unicode characters ranging between 1 and 4,000 characters);

1 - Forcing the string type to be NVarChar;

2 - Forcing the string type to be NChar (A fixed-length stream of Unicode characters ranging between 1 and 4,000 characters);

3 - Forcing the string type to be VarChar (A variable-length stream of non-Unicode characters ranging between 1

and 8,000 characters. VarChar is used when the database column is varchar(max));  
4 - Forcing the string type to be Char (A fixed-length stream of non-Unicode characters ranging between 1 and 8,000 characters).  
The default value of nStrType is 0.

## Return Value

0 (zero) if successful, otherwise an error number is returned. (For details of the 307 error code, call the [SQLErrMsg](#) function).

## Related Functions

[SQLSet](#), [SQLAppend](#), [SQLExec](#), [SQLGetRecordset](#), [SQLCall](#), [SQLGetScalar](#), [SQLEnd](#), [SQLParamsSetAsInt](#), [SQLParamsSetAsReal](#), [SQLParamsClearAll](#)

## Example

See [SQLParamsSetAsInt](#)

## See Also

[SQL Functions](#)

## SQLPrev

Gets the previous database record from an SQL query. The function works only with disconnected recordsets. Use the [SQLGetRecordset\(\)](#) function to select a number of records or rows from the SQL database, and then use the [SQLNext\(\)](#)/[SQLPrev\(\)](#) function to step through each record separately.

This function can be called in the foreground (only for disconnected recordsets) or background.

## Syntax

**SQLPrev(*hGeneral*)**

*hGeneral*:

The handle to the disconnected recordset.

## Return Value

0 (zero) if successful, otherwise an error number is returned. (For details of the 307 error code, call the [SQLErrMsg](#) function).

## Related Functions

[SQLSet](#), [SQLAppend](#), [SQLExec](#), [SQLGetRecordset](#), [SQLCall](#), [SQLGetScalar](#), [SQLEnd](#), [SQLNumChange](#), [SQLNoFields](#),

[SQLNumFields](#), [SQLFieldInfo](#), [SQLGetField](#), [SQLIsNullField](#), [SQLRowCount](#), [SQLNext](#)

## Example

See [SQLConnect](#)

## See Also

[SQL Functions](#)

### SQLQueryCreate

The function creates a new query and returns its handle. The query is empty and has to be set by using [SQLSet\(\)](#) and/or [SQLAppend\(\)](#) functions. The query handle can be used with the statement functions as [SQLExec\(\)](#) or [SQLGetRecordset\(\)](#).

Queries allocated by [SQLQueryCreate](#) should be disposed by [SQLQueryDispose](#) when no longer necessary. Each DB connection object has one default query which is created and disposed automatically. The default query need not be disposed by the function [SQLQueryDispose](#).

This function can be called in the foreground or background.

## Syntax

**SQLQueryCreate(*hSQL*)**

*hSQL*:

The handle to the DB connection object, returned from either [SQLCreate\(\)](#) or [SQLConnect\(\)](#) function. The handle identifies the DB connection object where details of the associated SQL connection are stored.

## Return Value

The handle to the query if successful, otherwise the invalid handle.

## Related Functions

[SQLSet](#), [SQLAppend](#), [SQLExec](#), [SQLGetRecordset](#), [SQLCall](#), [SQLGetScalar](#), [SQLEnd](#), [SQLQueryDispose](#),  
[SQLNumChange](#), [SQLNumFields](#), [SQLFieldInfo](#), [SQLGetField](#), [SQLIsNullField](#), [SQLRowCount](#), [SQLNext](#), [SQLPrev](#)

## See Also

[SQL Functions](#)

### SQLQueryDispose

The function disposes the query which handle is given as the argument.

Queries allocated by [SQLQueryCreate](#) should be disposed by [SQLQueryDispose](#) when no longer necessary. Each

DB connection object has one default query which is created and disposed automatically. The default query need not be disposed by the function SQLQueryDispose.

This function can be called in the foreground or background.

## Syntax

**SQLQueryDispose(*hQuery*)**

*hQuery*:

The handle to the query, returned from SQLQueryCreate() function.

## Return Value

0 (zero) if successful, otherwise an error code is returned. (For details of the 307 error code, call the [SQLErrMsg\(\)](#) function).

## Related Functions

[SQLSet](#), [SQLAppend](#), [SQLExec](#), [SQLGetRecordset](#), [SQLCall](#), [SQLGetScalar](#), [SQLEnd](#), [SQLQueryCreate](#),  
[SQLQueryDispose](#), [SQLNumChange](#), [SQLNoFields](#), [SQLNumFields](#), [SQLFieldInfo](#), [SQLGetField](#), [SQLIsNullField](#),  
[SQLRowCount](#), [SQLNext](#), [SQLPrev](#)

## See Also

[SQL Functions](#)

## SQLRollBack

Rolls back (discards) all changes made to the database within the current transaction. If you call the [SQLBeginTran\(\)](#) function to begin a transaction, you are not committed to changes to the database made by the Insert, Delete, and Update commands until you call the [SQLCommit\(\)](#) function. You can discard these changes by calling the [SQLRollBack\(\)](#) function.

You can only call the [SQLRollBack\(\)](#) function if you have called [SQLBeginTran\(\)](#) to begin a transaction. You do not need to begin a transaction to modify a database, but any changes you make to a database outside of a transaction are automatically committed.

The [SQLRollBack\(\)](#) function could affect different databases in different ways. If the function is not performing as you expect, check that your database is able to service transactions. Refer to the documentation for your database for more information on rolling back transactions.

This function is a blocking function and should not be called from a foreground task.

## Syntax

**SQLRollBack(*hSQL*)**

*hSQL*:

The handle to the DB connection object, returned from either [SQLCreate\(\)](#) or [SQLConnect\(\)](#) function. The handle

identifies the DB connection object where details of the associated SQL connection are stored.

## Return Value

0 (zero) if successful, otherwise an error number is returned. (For details of the 307 error code, call the [SQLErrMsg](#) function).

## Related Functions

[SQLCreate](#), [SQLOpen](#), [SQLClose](#), [SQLDispose](#), [SQLConnect](#), [SQLDisconnect](#), [SQLInfo](#), [SQLBeginTran](#), [SQLCommit](#)

## Example

See [SQLBeginTran](#).

## See Also

[SQL Functions](#)

## SQLRowCount

Gets the number of rows in the recordset.

This function can be called in the foreground or background.

## Syntax

**SQLRowCount(*hGeneral*)**

*hGeneral*:

The handle either to:

The DB connection object, returned from either [SQLCreate\(\)](#) or [SQLConnect\(\)](#) function. The handle identifies the DB connection object where details of the associated SQL connection are stored.

Or:

The recordset.

## Return Value

The number of rows in the given recordset for disconnected recordsets. For the connected recordset, the function returns always -1.

## Related Functions

[SQLSet](#), [SQLAppend](#), [SQLExec](#), [SQLGetRecordset](#), [SQLCall](#), [SQLGetScalar](#), [SQLEnd](#), [SQLNumChange](#), [SQLNoFields](#), [SQLNumFields](#), [SQLFieldInfo](#), [SQLGetField](#), [SQLIsNullField](#), [SQLRowCount](#), [SQLNext](#), [SQLPrev](#)

## Example

See [SQLCreate](#).

## See Also

[SQL Functions](#)

## SQLSet

Sets a query string in the SQL buffer. Cicode cannot send an SQL query that is longer than 255 characters. If you have an SQL query that is longer than the 255 character limit, you can split the query into smaller strings, and use this function and the [SQLAppend\(\)](#) function to append the query in the SQL buffer.

This function can be called in the foreground or background.

Queries which are built on the basis of user data, for example inputed by users via graphics pages or forms, may be prone to SQL Injection attacks. In such case, try to limit the risk by using Cicode functions from parameterized queries group and refer to a professional advice in this matter.

### NOTICE

#### SECURITY BREACH VIA SQL INJECTION

- Validate all textbox entries using validation controls, regular expressions and code
- Use parameterized SQL or stored procedures
- Use a limited access account to connect to the database

**Failure to follow these instructions can result in equipment damage.**

Building queries from pieces (SQLSet, SQLAppend) or adding parameters to either queries or connections (SQLParam functions) requires a few calls to respective CiCode functions. If a few functions try to manipulate the same connection in the same time some conflicts and unintended operations may occur. It is a typical multithreading problem.

To avoid this, instead of manipulating connections, consider using locally created and locally disposed queries. For example:

```
int function SAFE_SQL_CICODE_MULTITHREAD_USE()
//locally created query
int hStmt = SQLQueryCreate(hConnection);

//Set the query
SQLSet(hStmt, "select * from TAB where NAME=@Name");

//Add parameters to the query
SQLParamsSetAsString(hStmt, "Name", "Aaa");

//Execute the query
SQLGetRecordset(hStmt, "");

//the locally created query is disposed
SQLQueryDispose(hStmt);
```

End

## Syntax

**SQLSet(*hGeneral*, *sString*)**

*hGeneral*:

The handle either to the DB connection object (returned from either [SQLCreate\(\)](#) or [SQLConnect\(\)](#) function) or to the query handle (returned from [SQLQueryCreate\(\)](#)). When it is the connection handle, the operation is performed on the first query in that DB connection object. When it is the query handle, the operation is performed on that query through the DB object which is associated to it.

*sString*:

The query string to set in the SQL buffer. The string needs to contain the first part of an SQL query.

## Return Value

0 (zero) if successful, otherwise an error number is returned. (For details of the 307 error code, call the [SQLErrMsg\(\)](#) function).

## Related Functions

### Example

```
hSQL = SQLConnect("DSN=QEDBF");
nError = SQLBeginTran(hSQL);
nError = SQLSet(hSQL, "SELECT *");
nError = SQLAppend(hSQL, " FROM EMP");
nError = SQLAppend(hSQL, " ORDER BY last_name");
hRec = SQLGetRecordset(hSQL, "");
```

## See Also

[SQL Functions](#)

## SQLTraceOff

Turns off the debug trace. Use this function to stop tracing function calls that are made to the database. The trace can be turned off globally for all connections or for a specific one.

This function can be called in the foreground or background.

## Syntax

**SQLTraceOff(*hSQL*)**

*hSQL*:

The handle to the DB connection object: INVALID HANDLE - The trace is deactivated for all DB connections (default), otherwise - The trace is deactivated for this specific DB connection object.

## Return Value

0 (zero) if successful, otherwise an error number is returned. (For details of the 307 error code, call the [SQLErrMsg](#) function).

## Related Functions

[SQLErrMsg](#)

## Example

See [SQLFieldInfo](#).

## See Also

[SQL Functions](#)

## SQLTraceOn

Turns on a debug trace. Use this function to begin tracing function calls that are made to the database. The information is written to TraceLog.dat file (the strFileName argument is currently ignored). The trace can be turned on globally for all connections or for a specific connection.

---

**Note:** Currently the SQL tracing functionality uses the standard SCADA logging mechanism. All SQL traces are Information logs and can be written to TraceLog.dat file only if appropriate logging ini parameters are set respectively, for example [\[Debug\]SeverityFilterMode](#).

---

This function can be called in the foreground or background.

## Syntax

**SQLTraceOn(*sFileName*, *hSQL*, *nTraceLevel*)**

*sFileName*:

The output file name for the debug trace. Currently ignored.

*hSQL*:

The handle to the DB connection object: INVALID HANDLE - The trace is activated for all DB connections (default), otherwise - The trace is activated for this specific DB connection object.

*nTraceLevel*:

Defines the level of details written to the trace file. The following values are allowed:

0 - as per 1, but without values (default).

1 - the highest level of details including values of read cells

## Return Value

0 (zero) if successful, otherwise an error number is returned. (For details of the 307 error code, call the

[SQLErrMsg](#) function).

## Related Functions

[SQLErrMsg](#), [SQLTraceOff](#)

## Example

See [SQLFieldInfo](#)

## See Also

[SQL Functions](#)

## SPC Functions

Following are functions relating to Statistical Process Control:

<a href="#">SPCAlarms</a>	Returns the status of the specified SPC alarm.
<a href="#">SPCClientInfo</a>	Returns SPC data for the given SPC tag.
<a href="#">SPCGetHistogramTable</a>	Returns an array containing the frequency of particular ranges for the given SPC tag.
<a href="#">SPCGetSubgroupTable</a>	Returns an array containing the specified subgroup's elements with the mean, range and standard deviation.
<a href="#">SPCPlot</a>	Generates a single page print showing three separate trends of the SPC Mean, Range, and Standard Deviation.
<a href="#">SPCProcessXRSGet</a>	Gets the process mean, range and standard deviation overrides.
<a href="#">SPCProcessXRSSet</a>	Sets the process mean, range and standard deviation overrides.
<a href="#">SPCSelLimit</a>	Sets the upper or lower control limits of X-bar, range, or standard deviation charts.
<a href="#">SPCSpecLimitGet</a>	Gets the upper and lower specification limits for the specified tag.
<a href="#">SPCSpecLimitSet</a>	Sets the upper and lower specification limits for the specified tag.
<a href="#">SPCSubgroupSizeGet</a>	Gets the size of a subgroup for the specified SPC tag.

SPCSubgroupSizeSet	Sets the subgroup size for the specified SPC tag.
--------------------	---

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## SPCAlarms

Returns the status of the specified SPC alarm. This function is used to configure SPC alarms, by defining alarms with this trigger in Advanced Alarms.

This function can only be used if the Alarm Server is on the current machine. When the Alarm Server is not in the calling process, this function will become blocking and cannot be called from a foreground task. In this case, the return value will be undefined and a Cicode hardware alarm will be raised.

## Syntax

**SPCAlarms(*sSPCTag*, *AlarmType*)**

*sSPCTag*:

The SPC Tag name as defined in SPC Tags.

*AlarmType*:

The description of the alarm type. The following types are valid:

XFreak

XOutsideCL

XAboveUCL

XBelowLCL

XOutsideWL

XGradualUp

XGradualDown

XUpTrend

XDownTrend

XErratic

XStratification

XMixture

ROutsideCL

RAboveUCL

RBelowLCL

## Return Value

Alarm status, ON (1) or OFF (0).

## Related Functions

[AlarmAck](#)

## Example

Advanced Alarm

Alarm Tag	Feed_SPC_XBLCL
Alarm Desc	Process mean below LCL
Expression	SPCALarms("Feed_SPC", XBelowLCL)
Comment	Trigger an alarm when XBelowLCL condition becomes true.

Advanced Alarm

Alarm Tag	Temp_SPC_GRADUP
Alarm Desc	Mean is drifting up
Expression	SPCALarms("Temp_SPC", XGradualUp)
Comment	Trigger an alarm if mean drifts up.

## See Also

[SPC Functions](#)

### SPCClientInfo

Returns SPC data for the given SPC tag. The information retrieved through this function is from the cache maintained by the client. This function will give a faster response than the related functions which access the SPC (trend) server.

This function can only be called while the SPC tag is being displayed on an SPC page.

## Syntax

**SPCClientInfo(*sSPCTag*, *iType*)**

*sSPCTag*:

The SPC Tag name as defined in SPC Tags.

*iType*:

The information to be returned:

1 - Subgroup Size

2 - No. of Subgroups

- 3 - Process Mean (x double bar)
- 4 - Process Range
- 5 - Process Standard Deviation
- 6 - Lower Specification Limit (LSL)
- 7 - Upper Specification Limit (USL)
- 8 - Cp - Process Capability Actual
- 9 - Cpk - Process Capability Potential
- 10 - Process Skewness
- 11 - Process kurtosis

## Return Value

The requested data specified by iType. It is of type REAL.

## Related Functions

[SPCSpecLimitGet](#), [SPCProcessXRSGet](#), [SPCSubgroupSizeGet](#)

## Example

```
/* This function will check the capability of a particular SPC tag.*/
REAL
FUNCTION
CheckCapability(STRING sTAG)
    REAL rReturn;
    rReturn = SPCClientInfo(sTag, 8);
    !rReturn holds the inherent capability value
    IF rReturn > 1.0 THEN
        Message(sTag + "Assessment","The process is Capable.",64);
    ELSE
        Message(sTag + "Assessment","The process is not Capable.",64);
    END
    Return rReturn;
END
```

## See Also

[SPC Functions](#)

### SPCGetHistogramTable

Returns an array containing the frequencies of particular ranges for the given SPC tag. The histogram structure is implied in the order of the table as follows - the first array element is the data less than -3 sigma. The second value is the data between -3 sigma and -3 sigma plus the bar width etc. The last value is the data greater than +3 sigma.

This function can only be called while the SPC tag is being displayed on an SPC page.

## Syntax

**SPCGetHistogramTable(*sSPCTag*, *iNoBars*, *TableVariable*)**

*sSPCTag*:

The SPC Tag name as defined in SPC Tags.

*iNoBars*:

The number of bars in the table. The valid range is restricted to values from 7 to 100. This also indicates the size of the array to be returned.

*TableVariable*:

The Cicode array that will store the histogram data. The number of elements in the array needs to be equal to (or greater than) *iNoBars*. Must be a Real type global array variable.

## Return Value

0 (zero) if successful, otherwise an error number is returned. The histogram table is written to *TableVariable*.

## Related Functions

TableMath

## Example

```
/* This function will get the maximum frequency present in the
histogram of a particular SPC tag.*/
REAL iFrequency[7];
! This variable needs to be global to the file so is declared outside
of the function
INT
FUNCTION
GetMaxFreq(STRING sTAG)
    INT iError;
    INT iMax = -1;
    iError = SPCGetHistogramTable(sTag, 7, iFrequency);
    !The elements of iFrequency now hold the histogram table frequencies.
    IF iError = 0 THEN
        ! Get maximum
        iMax = TableMath(iFrequency,7,1,0);
    END
    Return iMax;
END
```

## See Also

[SPC Functions](#)

## SPCGetSubgroupTable

Returns an array containing the specified subgroup's elements with the mean, range and standard deviation. The data will be in the following order:

Element0, Element1, ... , Element(**n**-1), Mean, Range, StdDev

where **n** is the subgroup size.

This function can only be called while the SPC tag is being displayed on an SPC page.

## Syntax

**SPCGetSubgroupTable**(*sSPCTag*, *iSubgroup*, *TableVariable*)

*sSPCTag*:

The SPC Tag name as defined in SPC Tags.

*iSubgroup*:

The number of the subgroup being displayed whose data is to be retrieved. Zero ('0') represents the latest subgroup.

*TableVariable*:

The first element of the Cicode array that will store the sample data. The number of elements in the array needs to be equal to (or greater than) the subgroup size + 3. Must be a Real type global array variable.

## Return Value

0 (zero) if successful, otherwise an error number is returned. The subgroup's data is written to *TableVariable*.

## Related Functions

[TableMath](#)

## Example

```
/* This function will get the minimum value present in the sample
data of a particular SPC tag.*/
REAL rSubgroup[8]; ! 5 samples + mean + range + stddev.
! This variable needs to be global to the file, so is declared outside
of the function
REAL
FUNCTION
GetMinSample(STRING sTAG)
INT iError;
REAL iMin = 0;
iError = SPCGetSubgroupTable(sTag, 7, rSubgroup);
!The elements of rSubgroup now hold the group samples, mean, range and stddev.
IF iError = 0 THEN
! Get minimum. Be aware that the range of data is 5
iMin = TableMath(rSubgroup,5,0,0);
END
Return iMin;
```

**END**

## See Also

[SPC Functions](#)

### SPCPlot

This function is designed to work only on an SPCXRSChart page. It prints a single page showing three separate trends of the SPC Mean, Range, and Standard Deviation. The Mean needs to be at *nAN*, the Range at *AN + 1*, and the Standard Deviation at *AN + 2*. You can specify a title and a comment for the plot, and whether it is printed in color or in black and white.

## Syntax

**SPCPlot(*sPort*, *nAN* [, *sTitle*] [, *sComment*] [, *nMode*] )**

*sPort*:

The name of the printer port to which the plot will be printed. This name needs to be enclosed within quotation marks. For example LPT1:, to print to the local printer, or \\Pserver\canon1 using UNC to print to a network printer.

*nAN*:

The animation point at which the Mean chart is currently situated. The Range and Standard Deviation charts need to be on the next two consecutive animation numbers. For example, if the Mean chart is at animation point 40, the Range chart needs to be at animation point 41, and the Standard Deviation chart needs to be at animation point 42.

*sTitle*:

The title of the trend plot.

*sComment*:

The comment that is to display beneath the title of the trend plot. You do not have to enter a comment.

*nMode*:

The color mode of the printer.

0 - Black and White (default)

1 - Color

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnPlot](#), [TrnComparePlot](#), [TrnPrint](#), [PlotOpen](#)

## Example

```
/* This function will print the Mean trend (currently displayed at
animation point 40), the Range trend (currently at animation point
41), and the Standard Deviation trend (currently at animation
point 42). The result is a one page, black and white combination
of all three trends, printed to LPT1. */
SPCPlot("LPT1:",40, "Plant SCADA SPC Chart","Gradually increasing trend",0);
```

## See Also

[SPC Functions](#)

### SPCProcessXRSGet

Gets the process mean, range, and standard deviation overrides for the specified SPC tag. The values that are returned are the values that are currently being used by the SPC (trend) server, not necessarily the values specified in the SPC Tag definition.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

This function can only be called while the SPC tag is being displayed on an SPC page.

## Syntax

**SPCProcessXRSGet(*sSPCTag*, *XVariable*, *RVariable*, *SVariable* [, *sClusterName*] )**

***sSPCTag*:**

The SPC Tag name as defined in SPC Tags.

***XVariable*:**

The Cicode variable that stores the process mean (X double bar). A constant is not allowed. Must be a Real type global variable.

***RVariable*:**

The Cicode variable that stores the range (R). A constant is not allowed. Must be a Real type global variable.

***SVariable*:**

The Cicode variable that stores the standard deviation (S). A constant is not allowed. Must be a Real type global variable.

***sClusterName*:**

Specifies the name of the cluster of the SPC tag.

## Return Value

0 (zero) if successful, otherwise an error number is returned. The process mean is written to XVariable, the process range to RVariable, and the standard deviation to SVariable.

## Related Functions

[SPCClientInfo](#), [SPCProcessXRSSet](#)

## Example

```
/* This function will set a new override value for Mean, without
overwriting the values already in place for Standard Deviation and
Range */
REAL rOldMean;
REAL rRange;
REAL rStdDev;
! These variables need to be global to the file, so are declared
outside of the function
INT
FUNCTION
Tank1SPCNewMean(REAL rNewMean)
    INT iError;
    iError = SPCProcessXRSGet("TANK_1_TEMP", rOldMean, rRange, rStdDev);
    ! If no error, rOldMean, rRange and rStdDev now hold the current values of XRS.
    IF iError = 0 THEN
        iError = SPCProcessXRSSet("TANK_1_TEMP", rNewMean, rRange, rStdDev);
    END
    Return iError;
END
```

## See Also

[SPC Functions](#)

### SPCProcessXRSSet

Sets the process mean, range and standard deviation overrides for the specified SPC tag. The values entered here will override Plant SCADA's automatic calculations, and the overrides specified in the SPC Tags definition.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

This function can only be called while the SPC tag is being displayed on an SPC page.

## Syntax

**SPCProcessXRSSet(*sSPCTag*, *rMean*, *rRange*, *rStdDev* [, *sClusterName*] )**

*sSPCTag*:

The SPC Tag name as defined in SPC Tags.

*rMean*:

The new value of process mean (x double bar) to set.

*rRange*:

The new value of process range to set.

*rStdDev*:

The new value of process standard deviation to set.

*sClusterName*:

Specifies the name of the cluster of the SPC tag.

## Return Value

0 (zero) if successful, otherwise an error number is returned.

## Related Functions

[SPCProcessXRSGet](#)

## Example

See [SPCProcessXRSGet](#).

## See Also

[SPC Functions](#)

## SPCSetLimit

Sets the upper or lower control limits of X-bar, range, or standard deviation charts. Using this function will only set the controller limits on the Client display which will not affect the SPC Alarms. To set the server control limits, use the SPCProcessXRSSet function.

## Syntax

**SPCSetLimit(*nAN*, *Type*, *Value*, *Setting*)**

*nAN*:

The AN where the SPC chart is located.

*nType*:

The SPC type:

1 - X-bar upper control limit

2 - X-bar lower control limit

3 - Range upper control limit

4 - Range lower control limit

5 - Standard deviation upper control limit

6 - Standard deviation lower control limit

7 - X-bar centre line

8 - Range centre line

9 - Standard deviation centre line

**Value:**

The value for the control limit.

**Setting:**

Automatic calculation or manual setting of control limits:

0 - Automatic

1 - Manual

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Example

```
SPCSetLimit(40,1,250,1);
! Sets X-bar upper control limit to 250 at AN40.
```

## See Also

[SPC Functions](#)

## SPCSpecLimitGet

Gets the process Upper and Lower Specification Limits (USL and LSL) for the specified SPC tag. The values that are returned are the values that are currently being used by the SPC (trend) server, not necessarily the values specified in the SPC Tag definition.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

## Syntax

```
SPCSpecLimitGet(sSPCTag, LSLVariable, USLVariable [, sClusterName] )
```

***sSPCTag:***

The SPC Tag name as defined in SPC Tags.

***LSLVariable:***

The Cicode variable that stores the Lower Specification Limit (LSL). Do not specify a constant in this field. Must be a Real type global variable.

***USLVariable:***

The Cicode variable that stores the Upper Specification Limit (USL). Do not specify a constant in this field. Must be a Real type global variable.

***sClusterName:***

Specifies the name of the cluster of the SPC tag.

## Return Value

0 (zero) if successful, otherwise an error number is returned. The LSL is written to LSLVariable, while the USL is written to USLVariable.

## Related Functions

[SPCClientInfo](#), [SPCSpecLimitSet](#)

## Example

```
/* This function will increase the current USL and LSL of the
specified Tag by 10 percent.*/
REAL rLSL;
REAL rUSL;
! These variables need to be global to the file, so are declared
outside of the function
INT
FUNCTION
ExpSLbyPercent(STRING sTAG)
    REAL rIncPercent = 1.1;
    REAL rDecPercent = 0.9;
    INT iError;
    iError = SPCSpecLimitGet(sTag, rLSL, rUSL);
! If no error, rLSL and rUSL now hold the current values of
    LSL and USL for sTAG
    rLSL = rLSL * rDecPercent;
    rUSL = rUSL * rIncPercent;
    IF iError = 0 THEN
        iError = SPCSpecLimitSet(sTAG, rLSL, rUSL);
    END
    Return iError;
END
```

The function would be called as follows:

Page Button

Button Text	Expand Temperature Limits
Expression	ExpSLby10Percent("TANK_1_TEMP");

## See Also

[SPC Functions](#)

### SPCSpecLimitSet

Sets the process Upper and Lower Specification Limits (USL and LSL) for the specified SPC tag. The values entered here will override those specified in the SPC Tags definition.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

## Syntax

**SPCSpecLimitSet(*sSPCTag*, *rLSL*, *rUSL* [, *sClusterName*] )**

*sSPCTag*:

The SPC Tag name as defined in SPC Tags.

*rLSL*:

The new value of Lower Specification Limit (LSL) to set.

*rUSL*:

The new value of Upper Specification Limit (USL) to set.

*sClusterName*:

Specifies the name of the cluster of the SPC tag.

## Return Value

0 (zero) if successful, otherwise an error number is returned.

## Related Functions

[SPCSpecLimitGet](#)

## Example

See [SPCSpecLimitGet](#).

## See Also

[SPC Functions](#)

## SPCSubgroupSizeGet

Gets the subgroup size for the specified SPC tag. The value that is returned is the value that is currently being used by the SPC (trend) server, not necessarily the value specified in the SPC Tag definition.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

This function can only be called while the SPC tag is being displayed on an SPC page.

## Syntax

**SPCSubgroupSizeGet(*sSPCTag*, *SizeVariable* [, *sClusterName*] )**

*sSPCTag*:

The SPC Tag name as defined in SPC Tags.

*SizeVariable*:

The Cicode variable that stores the subgroup size. This variable needs to be defined as a global of type INT. A

constant is not allowed. Must be a Long type variable.

*sClusterName*:

Specifies the name of the cluster of the SPC tag.

## Return Value

0 (zero) if successful, otherwise an error number is returned. The subgroup size is written to *SizeVariable*.

## Related Functions

[SPCClientInfo](#), [SPCSubgroupSizeSet](#)

## Example

See [SPCSubgroupSizeSet](#).

## See Also

[SPC Functions](#)

## SPCSubgroupSizeSet

Sets a new subgroup size for the specified SPC tag. The new subgroup size becomes the new size as long as the SPC (trend) server is running. The subgroup size is updated first in the SPC server, which then informs the clients to update. This will force re-calculation of SPC values (UCL and LCL) across the span of any displayed charts.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

This function can only be called while the SPC tag is being displayed on an SPC page.

## Syntax

**SPCSubgroupSizeSet(*sSPCTag*, *iSize* [, *sClusterName*] )**

*sSPCTag*:

The SPC Tag name as defined in SPC Tags.

*iSize*:

The new size of the subgroup to set.

*sClusterName*:

Specifies the name of the cluster of the SPC tag.

## Return Value

0 (zero) if successful, otherwise an error number is returned.

## Related Functions

[SPCSubgroupSizeGet](#)

## Example

```
/* This function increments the subgroup size for FEED_RATE_1 by
the specified amount. */
INT iSize;
! This variable needs to be global to the file, so is declared outside
of the function
INT
FUNCTION
IncSubgroupSize(INT iIncrement)
    INT iError;
    iError = SPCSubgroupSizeGet("FEED_RATE_1", iSize);
    ! If no error, iSize now contains the current subgroup size of FEED_RATE_1
    iSize = iSize + iIncrement;
    IF iError = 0 and (isize > 1) THEN
        iError = SPCSubgroupSizeSet("FEED_RATE_1", iSize );
    END
    Return iError;
END
```

## See Also

[SPC Functions](#)

## String Functions

String functions allow you to manipulate strings in various ways. You can extract characters or substrings from a string, convert strings into other data types, format strings, search for strings, and perform other operations.

Following are functions relating to Strings:

<a href="#">CharToStr</a>	Converts an ASCII code into a string.
<a href="#">HexToStr</a>	Converts a number into a hexadecimal string.
<a href="#">IntToStr</a>	Converts an integer variable into a string.
<a href="#">PathToStr</a>	Converts a Plant SCADA path into a string.
<a href="#">RealToStr</a>	Converts a floating-point variable into a string.
<a href="#">StrCalcWidth</a>	Retrieves the pixel width of a string using a particular font.
<a href="#">StrClean</a>	Removes control characters from a string.
<a href="#">StrEndsWith</a>	Verifies whether the given string ends with a specific

	string.
StrFill	Fills a string with characters.
StrFormat	Formats a variable into a string.
StrGetChar	Gets a single character from a string or buffer.
StrLeft	Gets the left-hand characters from a string.
StrLength	Gets the length of a string.
StrListContainsItem	Checks whether the string passed is an item contained in a delimited list of strings.
StrLower	Converts a string to lower-case.
StrMid	Gets characters from the middle of a string.
StrPad	Pads a string to the required length.
StrReplace	Replaces a substring in a string with replacement substring.
StrRight	Gets the right-hand characters from a string.
StrSearch	Searches for a string within a string.
StrSetChar	Sets a single character into string or buffer.
StrSplit	Checks whether the string passed is an item contained in a delimited list of strings.
StrToBool	Converts a string value to a Boolean value.
StrToChar	Converts a string to an ASCII code.
StrToDate	Converts a string into a date variable.
StrToFmt	Converts a string into format fields.
StrToGrp	Converts a string into a group.
StrToHex	Converts a hexadecimal string into an integer.
StrToInt	Converts a string into an integer variable.
StrToLines	Converts a string into lines of limited length.
StrToLocalText	Converts a string from Native language to Local language.
StrToPeriod	Converts a string into a (time) period.

<a href="#">StrToReal</a>	Converts a string into a floating-point variable.
<a href="#">StrToTime</a>	Converts a string into a time variable.
<a href="#">StrToValue</a>	Converts a string into a floating-point variable.
<a href="#">StrTrim</a>	Trims spaces from a string.
<a href="#">StrTruncFont</a>	Returns the truncated string using a particular font (specified by name) or the specified number of characters.
<a href="#">StrTruncFontHnd</a>	Returns the truncated string using a particular font (specified by font number) or the specified number of characters.
<a href="#">StrTruncFontTooltip</a>	Returns a truncated string with ellipsis, and sets an AN with a tooltip containing the complete string if a truncation occurs.
<a href="#">StrUpper</a>	Converts a string to upper-case.
<a href="#">StrWord</a>	Gets a word from a string.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## CharToStr

Converts an ASCII code into a string.

## Syntax

**CharToStr(ASCIICode)**

*ASCIICode:*

The ASCII code to convert.

## Return Value

A string containing the converted ASCII code.

## Related Functions

[StrSetChar](#)

## Example

```
str = CharToStr(65);
! Sets str to "A".
```

## See Also

[String Functions](#)

## HexToStr

Converts a number into a hexadecimal string. The string is the width specified (padded with zeros).

## Syntax

**HexToStr(Number, Width)**

*Number:*

The number to convert.

*Width:*

The width of the string (0 to 8).

---

**Note:** The function does not truncate the result if it is longer than the value specified.

---

## Return Value

A string containing the converted number.

## Related Functions

[StrToHex](#), [IntToStr](#)

## Example

```
Variable=HexToStr(123, 4);
! Sets Variable to "007b".
Variable = HexToStr(0x12ABFE, 8);
! Sets Variable to "0012abfe"
```

## See Also

[String Functions](#)

## IntToStr

Converts a number into a string.

## Syntax

**IntToStr(*Number*)**

*Number*:

The number to convert.

## Return Value

A string containing the converted number.

## Related Functions

[StrFormat](#)

## Example

```
Variable=IntToStr(5);
! Sets Variable to "5".
```

## See Also

[String Functions](#)

[Converting and Formatting Cicode Variables](#)

## PathToStr

Converts a Plant SCADA path into a string. The path string can contain one of the standard Plant SCADA path operators, BIN, DATA, RUN, USER or a user-configured path. If the string does not contain a Plant SCADA path, it is unchanged.

## Syntax

**PathToStr(*sPath*)**

*sPath*:

The Plant SCADA path to convert.

## Return Value

A string containing the converted path.

## Example

```
Variable=PathToStr("[data]:test.txt");
! Sets Variable to "c:\MyApplication\data\test.txt".
! assuming that DATA=C:\MyApplication\DATA
```

## See Also

[String Functions](#)

### RealToStr

Converts a floating-point number into a string.

## Syntax

**RealToStr(*Number*, *Width*, *Places*[, *Separator*])**

*Number*:

The floating-point number to convert.

*Width*:

The width of the string.

*Places*:

The number of decimal places contained in the string.

*Separator*:

Optionally, the decimal separator contained in the string. Defaults to empty string.

- "." - period
- "," - comma
- "" - empty string

If an empty string or an invalid separator is passed as a parameter, the string will contain the decimal separator used in the current locale.

## Return Value

The floating-point number (as a string).

## Related Functions

[StrToReal](#)

## Example

```
Variable=RealToStr(12.345,10,1);
! Sets Variable to " 12.3" (10 characters long).
```

## See Also

[String Functions](#)

[Converting and Formatting Cicode Variables](#)

## StrCalcWidth

Retrieves the pixel width of a string using a particular font.

## Syntax

**StrCalcWidth(*sText, iFont*)**

*sText:*

The text to determine the pixel width of

*iFont:*

The font number used to calculate the pixel width of the text. (To use the default font, set to -1).

## Return Value

The pixel width of a string using the particular font.

## Related Functions

[StrTruncFont](#), [StrTruncFontHnd](#), [DspFont](#), [DspFontHnd](#)

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrClean

Removes control characters from a string. Any character that is not a displayable ASCII character is removed from the string.

## Syntax

**StrClean(*String*)**

*String:*

The source string.

## Return Value

The string with all control characters removed.

## Related Functions

[StrTrim](#)

## Example

```
Variable=StrClean("*****Text*****");
/* Sets Variable to "Text" (the "*" character in this example
represents an unprintable character). */
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrEndsWith

Verifies whether the given string ends with a specific string.

## Syntax

**StrEndsWith(STRING *sSource*, STRING *sSearchTerm*)**

*sSource*

The source string to check.

*sSearchTerm*

The string to search for in the source string.

## Return Value

Returns TRUE if *sSource* ends with *sSearchTerm*, otherwise returns FALSE.

## Example

```
STRING sTest = "MyPage_UHD4K";
INT bResult = StrEndsWith("MyPage_UHD4K", "_UHD4K");
```

## Related Functions

[StrListContainsItem](#), [StrSplit](#), [StrTruncFontTooltip](#)

## See Also

[String Functions](#)

## StrFill

Fills a string with a number of occurrences of another string.

## Syntax

**StrFill(*String*, *Length*)**

*String*:

The string to be repeated.

*Length*:

The length of the string.

## Return Value

The filled string.

## Related Functions

[StrPad](#)

## Example

```
Variable=StrFill("abc",10);
! Sets Variable to "abcabca".
Variable=StrFill("x",10);
! Sets Variable to "xxxxxxxxxx".
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrFormat

Converts a variable into a formatted string. This function is the equivalent of the Cicode " ##### " operator.

## Syntax

**StrFormat(*Variable*, *Width*, *DecPlaces*, *EngUnits*)**

*Variable*:

The variable to format into a string.

*Width*:

The width of the variable after it has been converted to string format.

*DecPlaces*:

The number of decimal places in the converted string.

*EngUnits*:

The engineering units of the variable.

## Return Value

The variable (as a formatted string).

## Related Functions

[StrToReal](#), [StrToInt](#), [RealToStr](#), [IntToStr](#)

## Example

```
Variable=StrFormat(10.345,5,2,"%");
! Sets Variable to "10.35%".
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrGetChar

Gets a single character from a string or buffer. Use this function to read a string, character by character.

## Syntax

**StrGetChar**(*String*, *iOffset*)

*String*:

The source string. Must be a String type variable.

*iOffset*:

The offset in the string, commencing at 0.

## Return Value

The character at the offset in the string.

## Related Functions

[StrSetChar](#)

## Example

```
FOR i = 0 To length DO
char = StrGetChar(str, i);
! Get char from string
END
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrLeft

Gets the left-most characters from a string.

## Syntax

**StrLeft(*String*, *N*)**

*String*:

The source string.

*N*:

The number of characters to get from the source string.

## Return Value

A string containing the left-most *N* characters of *String*.

## Related Functions

[StrRight](#), [StrMid](#), [StrLength](#)

## Example

```
Variable=StrLeft("ABCDEF",2);
! Sets Variable to "AB".
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrLength

Gets the length of a string.

## Syntax

**StrLength(*String*)**

*String*:

The source string.

## Return Value

The length of the string (as an integer).

## Related Functions

[StrRight](#), [StrMid](#), [StrLeft](#)

## Example

```
Variable=StrLength("ABCDEF");
! Sets Variable to 6.
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrListContainsItem

Checks whether the string passed is an item contained in a delimited list of strings. String comparison is case insensitive. Leading and trailing spaces are removed before comparing.

## Syntax

**StrListContainsItem**(STRING *sItem*, STRING *sList*, STRING *sDelim*)

*sItem*

Item that you wish to search for.

*sList*

List to search within.

*sDelim*

The string to use as a delimiter. If this is not specified, a comma (,) is used.

## Return Value

Returns TRUE if the item is found in the list, otherwise returns FALSE

## Related Functions

[StrEndsWith](#), [StrSplit](#), [StrTruncFontTooltip](#)

## See Also

[String Functions](#)

### StrLower

Converts a string to lowercase.

## Syntax

**StrLower(*String*)**

*String*:

The source string.

## Return Value

The string (as lowercase).

## Related Functions

[StrUpper](#)

## Example

```
Variable=StrLower("ABCDEF");
! Sets Variable to "abcdef".
Variable=StrLower("AbCdEf");
! Sets Variable to "abcdef".
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

### StrMid

Gets characters from the middle of a string.

## Syntax

**StrMid(*String, Offset, Characters*)**

*String*:

The source string.

*Offset*:

The offset in the string, commencing at 0.

*Characters:*

The number of characters to get, commencing at the offset.

## Return Value

A string containing the number of characters from the offset.

## Related Functions

[StrLeft](#), [StrLength](#), [StrRight](#)

## Example

```
Variable=StrMid("ABCDEF",1,3);
! Sets Variable to "BCD".
Variable=StrMid("ABCDEF",4,1);
! Sets Variable to "E".
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrPad

Pads a string with a number of occurrences of another string. Padding can be added to the left or to the right of a string. If the string is already longer than the required string length, the string is truncated.

## Syntax

**StrPad**(*String*, *PadString*, *Length*)

*String:*

The source string.

*PadString:*

The padding string.

*Length:*

The length of the string. If a positive length is specified, padding will be added to the right of the string. If a negative length is specified, padding will be added to the left of the string.

## Return Value

A padded string.

## Related Functions

[StrFill](#)

## Example

```
Variable=StrPad("Test", " ",10);
! Sets Variable to "Test ".
Variable=StrPad("Test","abc",-14);
! Sets Variable to "abcabcaTest".
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrReplace

Replaces a substring in a string with replacement substring.

## Syntax

**StrReplace(*sString*, *sFindSubstring* [, *sReplaceSubstring* [, *sTerminators*]])**

*sString*:

The source string.

*sFindSubstring*:

The substring to search for.

*sReplaceSubstring*:

The replacement substring. If this is not specified, it defaults to an empty string, that is, the matched substring will be removed from the original string.

*sTerminators*:

Optional set of characters to terminate the substring. If this is not specified, it defaults to an empty string. When it is specified, the substring is only considered a match if it is enclosed by one of the characters specified by this argument.

## Return Value

The replaced string.

## Example

```
// Replaces argument "abc" with 12. Use the sTerminators for higher precision
StrReplace("func(abcde, abc)", "abc", "12"); // func(12de, 12), incorrect
StrReplace("func(abcde, abc)", "abc", "12", " ,()") // func(abcde, 12), correct
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrRight

Gets the rightmost characters from a string.

## Syntax

**StrRight(*String*, *N*)**

*String*:

The source string.

*N*:

The number of characters to get from the source string.

## Return Value

A string containing the rightmost *N* characters of *String*.

## Related Functions

[StrLeft](#), [StrMid](#), [StrLength](#)

## Example

```
Variable=StrRight("ABCDEF",2);  
! Sets Variable to "EF".
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrSearch

Searches for a string within a string, commencing at a specified offset. The result of the search is the index in the source string, where the first character of the sub-string is found. Index 0 is the first character in the string, index 1 is the second, and so on.

## Syntax

**StrSearch(*Offset*, *String*, *Substring*)**

*Offset:*

The offset in the string, commencing at 0.

*String:*

The source string.

*Substring:*

The substring to search for.

## Return Value

The index in the search string, or -1 if the sub-string does not exist in the string.

## Related Functions

[StrLength](#)

## Example

```
Variable=StrSearch(1,"ABCDEF","CD");
! Sets Variable to 2.
Variable=StrSearch(4,"ABCDEF","CD");
! Sets Variable to -1.
Variable=StrSearch(5,"ABCDEF","F");
! Sets Variable to 5.
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrSetChar

Sets a single character into a string or buffer. Use this function to build up a string, character by character, and terminate the string with the end-of-string character 0 (zero). (If you use a string without a terminator in a function that expects a string, or in a Cicode expression, you could get invalid results.) To use the string to build up a buffer, you do not need the terminating 0 (zero).

## Syntax

**StrSetChar**(*sText*, *iOffset*, *Char*)

*sText:*

The destination string. Must be a String type variable.

*iOffset:*

The offset in the string, commencing at 0.

*Char:*

The ASCII character to set into the string.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[StrGetChar](#)

## Example

```
! Set chars in buffer, Buf is NOT a valid string
! and cannot be used where a normal string would be used.
FOR i = 0 To length DO
    StrSetChar(Buf, i, 30 + i);
END
StrSetChar(sStr, 0, 13); ! put CR into string
StrSetChar(sStr, 1, 0); ! terminate so may be used as a normal string
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrSplit

Splits a string into sub-strings based on the specified delimiter. Sub-strings are stored in a new array and a handle to this array is returned. The returned array contains the exact number of split strings. You can use the ArrayCount function to determine the number of strings into which the source was split.

consecutive delimiters are treated as a single delimiter. By default, leading and trailing white space is trimmed from sub-strings.

## Syntax

`StrSplit(STRING sSource, STRING sDelim, INT bNoTrim)`

*sSource*

The source string to split.

*sDelim*

The string to use as a delimiter. If not specified, a comma (,) is used.

*bNoTrim*

Optional flag to disable the trimming of white space from sub-strings. Trimming will occur by default.

## Return Value

Returns a handle to the array containing all sub-strings.

## Example

```
STRING sNames = "Phil, Michael, Bradley"
STRING sName;
INT aNames
INT iItem = 0;
aNames = StrSplit(sNames, ",");
WHILE (sName <> "") DO
sName = ArrayGetString(aNames, iItem);
iItem = iItem + 1;
END
ArrayDestroy(aNames);
```

## Related Functions

[StrEndsWith](#), [StrListContainsItem](#), [StrTruncFontTooltip](#)

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrToInt

Converts a string value to an integer value.

## Syntax

**StrToInt(sVal)**

*sVal:*

Boolean value as a string. Allowable values are: TRUE, FALSE or any numeric value.

## Return Value

Boolean value of 0 or 1.

## Example

```
StrToInt("TRUE"); // returns 1 which is TRUE
StrToInt("false"); // returns 0 which is FALSE
StrToInt("5"); // returns 1
StrToInt("Orange"); // returns 0
StrToInt("0"); // returns 0
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrToChar

Gets the ASCII code of the first character in a string.

## Syntax

**StrToChar(*String*)**

*String*:

The source string.

## Return Value

The ASCII code of the first character in *String*.

## Related Functions

[StrGetChar](#)

## Example

```
Variable=StrToChar("ABC");
! Sets Variable to 65 (ASCII "A").
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrToDate

Converts a "date" string into a time/date variable. This variable is the same as returned from the TimeCurrent() function. To set the order of the day, month, and year, and the delimiter, use the Windows Control panel.

Time/date functions can only be used with dates from 1980 to 2035. You should check that the date you are using is valid with the following Cicode:

```
IF StrToDate(Arg1)>0 THEN
  ...
ELSE
  ...
END
```

## Syntax

**StrToDate(*String*)**

*String:*

The source string.

## Return Value

A time/date variable, or 274 if the time/date is out of range.

## Related Functions

[StrToPeriod](#), [StrToTime](#)

## Example

```
! Australian format (dd/mm/yy) is set in the Windows Control panel.  
DateVariable=DateAdd(StrToDate("3/11/95"),86400);  
NewDate= TimeToStr(DateVariable, 2);  
! Adds 24 hours to 3/11/95 and sets NewDate to "4/11/95".
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrToFmt

Converts a string into field data for a format template. This function is useful for splitting a string into separate strings. After the string is converted, you can call the [FmtGetField\(\)](#) function to extract the individual data from the template fields.

## Syntax

**StrToFmt(*hFmt*, *String*)**

*hFmt:*

The format template handle, returned from the [FmtOpen\(\)](#) function. The handle identifies the table where all data on the associated format template is stored.

*String:*

The source string.

## Return Value

The string (formatted as template field data).

## Related Functions

[FmtOpen](#), [FmtGetField](#)

## Example

```
StrToFmt(hFmt, "CV101 Raw Coal Conveyor");
Name=FmtGetField(hFmt, "Name");
! Sets Name to "CV101".
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrToGrp

Converts a string into a group and places it into a group number. Any existing values in the group are cleared before the new values are inserted. The group string is a series of numbers separated by " , " to list individual values or " .. " to specify a range of values.

## Syntax

**StrToGrp(*hGrp*, *Str*)**

*hGrp*:

The group handle, returned from the [GrpOpen\(\)](#) function. The group handle identifies the table where all data on the associated group is stored.

*Str*:

The string to convert.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[GrpOpen](#)

## Example

```
hGrp=GrpOpen("MyGrp",1);
! Set group to 1 ... 10 and 20, 30 and 40.
StrToGrp(hGrp,"1..10,20,30,40");
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrToHex

Converts a hexadecimal string into an integer. This function will search the string for the first non-blank character, and then start converting until it finds the end of the string or a non-hexadecimal numeric character. If the first non-blank character is not one of the following hexadecimal characters, the return value is 0 (zero):

- **(0-9, a-f, A-F);**
- **A space;**
- A "+" (plus) or a "-" (minus) sign.

## Syntax

**StrToHex(*String*)**

*String*:

The string to convert.

## Return Value

An integer (converted from *String*).

## Related Functions

[StrToReal](#), [StrToValue](#), [HexToStr](#)

## Example

```
Variable=StrToHex("123");
! Sets Variable to hex 123, decimal 291
Variable=StrToHex("F2");
! Sets Variable to hex F2, decimal 242
Variable=StrToHex("G45");
! Sets Variable to 0.
Variable=StrToHex("-FG");
! Sets Variable to hex, -F decimal -15.
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrToInt

Converts a string into an integer. This function will search the string for the first non-blank character, and then start converting until it finds the end of the string or a non-numeric character. If the first non-blank character is not a numeric character (0-9), a space, a " + " or a " - " sign, the return value is 0 (zero).

## Syntax

**StrToInt(*String*)**

*String*:

The string to convert.

## Return Value

An integer (converted from *String*).

## Related Functions

[StrToReal](#), [StrToValue](#)

## Example

```
Variable=StrToInt("45");
! Sets Variable to 45.
Variable=StrToInt("45.23");
! Sets Variable to 45.
Variable=StrToInt("A45");
! Sets Variable to 0.
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

[Converting and Formatting Cicode Variables](#)

## StrToLines

Converts a string into separate lines that contain no more than the number of characters specified in the *MaxChars* argument.

The function splits the string by inserting newline characters into the text string, thus dividing it into separate lines. The string will be split at a whitespace character if possible, and that whitespace will be replaced by the newline character. If no whitespace characters are available then the insertion will be made at the maximum number of characters from the previous line break.

## Syntax

**StrToLines(*String*, *MaxChars*, *nLines*)**

*String*:

The string to convert.

*MaxChars*:

The maximum number of characters permitted in each new line produced by the StrToLines() function.

*nLines*:

Output Parameter: Returns the number of lines produced by the StrToLines() function from the input string. This should be a Cicode INT type variable.

## Return Value

A string containing the input string with new line characters inserted into it.

## Example

```
BrokenString=StrToLines("Was that a real Stegosaur?", 5, nLines);
!The function returns the value 6 in nLines, and Broken String now contains:
Was
that
a
real
Stego
saur?
BrokenString=StrToLines("It breaks the string by inserting newline characters into
the text.", 16, nLines);
!The function returns the value 6 in nLines, and Broken String now contains:
It breaks the
string by
inserting
newline
characters into
the text.
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrToLocalText

Converts a native string into the local version of that string. (The string needs to be contained within quotation marks, as shown in the example below.) The local version is taken from the current language database(as specified using the [Language]LocalLanguage parameter).

**StrToLocalText(*sText*)**

*sText*:

The string for which you would like the local translation returned. This string needs to be enclosed in quotation marks. For example:

```
"@(Motor Overload)"
```

## Return Value

The local version of the text if it was found, otherwise the native text or "#MESS" is returned, depending on the setting of the [\[Language\]DisplayError](#) parameter.

## Related Functions

[LanguageFileTranslate](#)

## Example

```
StrToLocalText("@(Motor Overload)");
! Returns the Local translation of Motor Overload.
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrToPeriod

Converts a string into a time period. You would normally use this function to convert operator input, for example, to set a trend period.

A valid period string is in the format HH:MM:SS, MM:SS or SS, where HH is the hours, MM is the minutes and SS is the seconds. The colon character (':') represents the time delimiter for these fields, which will be the current system time delimiter as set in the Windows Control Panel.

If minutes are specified, seconds need to be in the range 0-59. If hours are specified, minutes need to be in the range 0-59.

## Syntax

**StrToPeriod(*String*)**

*String*:

The string to convert.

## Return Value

A period (converted from *String*), or -1 if no conversion can be performed.

## Related Functions

[StrToTime](#), [StrToDate](#)

## Example

```
Variable=StrToPeriod("200");
! Sets Variable to 200 (seconds).
Variable=StrToPeriod("200:40");
! Sets Variable to 12040 (12000 + 40 seconds).
Variable=StrToPeriod("48:00:40");
! Sets Variable to 172840 (172800 + 40 seconds).
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrToInt

Converts a string into a floating-point number. This function will search the string for the first non-blank character, and then start converting until it finds the end of the string or a non-numeric character. If the first non-blank character is not a numeric character (0-9), a space, a decimal point, a "+" or a "-" sign, the return value is 0 (zero).

## Syntax

**StrToInt(*String*)**

*String*:

The string to convert.

## Return Value

A floating-point number (converted from *String*).

## Related Functions

[StrToInt](#), [StrToValue](#)

## Example

```
Variable=StrToInt("45");
! Sets Variable to 45.
Variable=StrToInt("45.23");
! Sets Variable to 45.23
Variable=StrToInt("A45");
```

```
! Sets Variable to 0.
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

[Converting and Formatting Cicode Variables](#)

## StrToTime

Converts a "time" string into a time/date variable. The value returned is the number of seconds from midnight. You can add this value to the date to get the current time value. To set the time delimiter, use the Windows Control Panel.

A valid time string is in the format HH:MM:SS or HH:MM:SS tt, where HH is the hour in the range 0-23, MM is the minute in the range 0-59, SS is the second in the range 0-59 and tt is the time extension; for example,, am or pm. The colon character ':' represents the time delimiter for these fields, which will be the current system time delimiter as set in the Windows control panel.

Times may also be passed in the for HH or HH:MM. In other words, you may omit the right-hand fields if they are 0.

---

**Note:** When you call the StrToTime function, it does not have any context of the day or date. This means it does not take into account daylight savings transitions. To accommodate days where a daylight savings time transition occurs, you need to use alternative Cicode functions that accept both date and time as arguments (such as [StrToTimestamp](#) or [TimestampToTimeInt](#)).

---

## Syntax

**StrToTime(String)**

*String:*

The string to convert.

## Return Value

A time/date variable, or -1 if no conversion can be performed.

## Related Functions

[Time, Date](#)

## Example

```
Variable=StrToTime("11:43:00");
! Sets Variable to (11*3600+43*60+0) seconds.
Variable=StrToTime("9:02");
! Sets Variable to (9*3600+2*60) seconds.
Variable=StrToTime("2");
```

```
! Sets Variable to (2*3600) seconds.
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrToValue

Converts a string into a floating-point number. This function is similar to the [StrToReal\(\)](#) function except that the function halts if it is passed an invalid string. The function will search the string for the first non-blank character, and then start converting until it finds the end of the string or a non-numeric character. If the first non-blank character is not a numeric character (0-9), a space, a decimal point, a "+" or a "-" sign, the function will halt.

Use this function to check keyboard input from the operator by setting control points (for example, it minimizes the likelihood of a setpoint being set to 0 if the operator presses ENTER or enters invalid data by mistake).

## Syntax

**StrToValue(*String*)**

*String*:

The string to convert.

## Return Value

A floating-point number (converted from *String*).

## Related Functions

[StrToReal](#), [StrToInt](#)

## Example

System Keyboard

Key Sequence	F3 ##### Enter
Command	SP123 = StrToValue(Arg1);
Comment	Set setpoint 123 to value unless value is invalid

**Note:** If value is invalid the Cicode is halted. Any other Cicode after the StrToValue() function will not execute.

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrTrim

Removes leading and trailing spaces from a string. Internal spaces are not removed from the string.

## Syntax

**StrTrim(*String*)**

*String*:

The source string.

## Return Value

*String* with leading and trailing spaces removed.

## Related Functions

[StrPad](#), [StrFill](#)

## Example

```
Variable=StrTrim(" Test String ");
! Sets Variable to "Test String".
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrTruncFont

Returns the truncated string using a particular font (specified by name) or the specified number of characters.

## Syntax

**StrTruncFont(*sText*, *sFont* [, *iLength*] [, *iLengthMode*])**

*sText*:

The text to truncate

*sFont*:

The name of the font that is used to display the text. The Font Name needs to be defined in the Fonts database. If the font is not found, the default font is used.

*iLength*:

Length of the Text to display, either in characters or pixels depending on *iLengthMode* (default -1, no truncation)

*iLengthMode*:

The length mode of the text string:

- 0 - Length as pixels truncated (default)
- 1 - Length as pixels truncated with ellipsis
- 2 - Length interpreted as characters.

## Return Value

A truncated string or the original one.

## Related Functions

[StrCalcWidth](#), [StrTruncFontHnd](#)

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrTruncFontHnd

Returns the truncated string using a particular font (specified by font number) or the specified number of characters.

## Syntax

**StrTruncFontHnd**(*sText*, *hFont* [, *iLength*] [, *iLengthMode*])

*sText*:

The text to truncate

*hFont*:

The font handle used to calculate the pixel width of the text. (To use the default font, set to -1).

*iLength*:

Length of the Text to display, either in characters or pixels depending on *iLengthMode* (default -1, no truncation)

*iLengthMode*:

The length mode of the text string:

- 0 - Length as pixels truncated (default)
- 1 - Length as pixels truncated with ellipsis
- 2 - Length interpreted as characters.

## Return Value

A truncated string or the original one.

## Related Functions

[StrCalcWidth](#), [StrTruncFont](#), [DspFont](#), [DspFontHnd](#)

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrTruncFontTooltip

Returns a truncated string with ellipsis, and sets an AN with a tooltip containing the complete string if a truncation occurs.

## Syntax

**StrTruncFontTooltip(INT *nANForTooltip*, STRING *sText*, STRING *sFont*, INT *nPixels*)**

*nANForTooltip*

The AN for which the tooltip needs to be set.

*sText*

The text to truncate.

*sFont*

The name of the font that is used to display the text. The font name needs to be defined in the Fonts database. If the font is not found, the default font is used.

*nPixels*

Number of pixels that the text can display before truncation occurs.

## Return Value

Returns the truncated string.

## Example

```
StrTruncFontTooltip(10, "My really, really, really long string", "SA_ButtonFont", 50)
```

## Related Functions

[StrEndsWith](#), [StrListContainsItem](#), [StrSplit](#)

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrUpper

Converts a string to uppercase.

## Syntax

**StrUpper(*String*)**

*String*:

The source string.

## Return Value

The string (as uppercase).

## Related Functions

[StrLower](#)

## Example

```
Variable=StrUpper("abcdef");
! Sets Variable to "ABCDEF".
Variable=StrUpper("AbCdEf");
! Sets Variable to "ABCDEF".
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## StrWord

Gets the first word from a string. The word is removed from the string to allow the function to be repeated. Word separators can be a space, newline, carriage return, or tab character.

## Syntax

**StrWord(*String*)**

*String*:

The source string.

## Return Value

A string containing the first word from *String*.

## Related Functions

[StrSearch](#)

## Example

```
Str="THIS IS A STRING";
Variable=StrWord(Str);
! Sets Variable to "THIS".
Variable=StrWord(Str);
! Sets Variable to "IS".
Variable=StrWord(Str);
! Sets Variable to "A".
Variable=StrWord(Str);
! Sets Variable to "STRING".
```

## See Also

[String Functions](#)

[Using the Caret Escape Sequence Character](#)

## Super Genie Functions

The Super Genie functions allow you to use Super Genies in your runtime system.

Following are functions relating to Super Genies:

<a href="#">Ass</a>	Associates a variable tag with a Super Genie.
<a href="#">AssChain</a>	Chains the associations from the current Super Genie to a new Super Genie.
<a href="#">AssChainPage</a>	Chains the associations from the current Super Genie to a new Super Genie, and displays the new Super Genie (in the current window).
<a href="#">AssChainPopUp</a>	Chains the associations from the current Super Genie to a new Super Genie, and displays the new Super Genie in a new popup window.
<a href="#">AssChainWin</a>	Chains the associations from the current Super Genie to a new Super Genie, and displays the new Super Genie in a new window. The new window will be of the same type as the current window.
<a href="#">AssChainWinFree</a>	Stores the tag associations on an existing Super Genie, closes it, then assigns the tags to a new window.
<a href="#">AssEquipParameters</a>	Associates a set of equipment parameters defined in Plant SCADA Studio's System Model   Equipment

	Runtime Parameters with a page.
AssEquipReferences	Creates Super Genie associations for each piece of equipment referenced by the specified equipment.
AssGetProperty	Gets association information about the current Super Genie from the datasource
AssGetScale	Gets scale information about the associations of the current Super Genie from the datasource
AssInfo	Gets association information about the current Super Genie (that is information about a variable tag that has been substituted into the Super Genie).
AssInfoEx	Gets association information about the current Super Genie (that is information about a variable tag that has been substituted into the Super Genie). Replacement for AssInfo supporting online changes.
AssMetadata	Performs Super Genie associations using the "Name" and "Value" fields.
AssMetadataPage	Uses the metadata information from the current animation point for the page associations for a new Super Genie page, and displays the new Super Genie in the current page.
AssMetadataPopUp	Uses the metadata information from the current animation point for the associations for a new Super Genie page, and displays the new Super Genie in a new pop up window.
AssMetadataWin	Uses the metadata information from the current animation point for the associations for a new Super Genie page, and displays the new Super Genie in a new window.
AssPage	Associates up to eight variable tags with a Super Genie and displays the Super Genie in the current window.
AssPopUp	Associates up to eight variable tags with a Super Genie and displays the Super Genie in a popup window.
AssScaleStr	Gets scale information about the associations of the current Super Genie (that is scale information about a variable tag that has been substituted into the Super Genie).
AssTag	Associates a variable tag with the current Super Genie. The association will be created for the current Super

	Genie only, and will only come into effect after you re-display the Super Genie.
<a href="#">AssTitle</a>	Sets the runtime window title to the tag name of the first variable substituted into the Super Genie.
<a href="#">AssVarTags</a>	Associates up to eight variable tags with a Super Genie. This association is only made for the next Super Genie you display (either in the current window or in a new window). You can use this function repeatedly to associate more than 8 variable tags to a Super Genie.
<a href="#">AssWin</a>	Associates up to eight variable tags with a Super Genie, and displays the Super Genie in a new window.
<a href="#">AssWinReplace</a>	This function removes the associations on a specified Super Genie window and applies any pending associations.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## Ass

Associates a variable tag or equipment with a Super Genie. This association is only made for the next Super Genie you display (either in the current window or in a new window). You cannot create an association for a Super Genie that is already displayed. You need to call this function once for every Super Genie substitution string in the Super Genie, otherwise the variable (substitution string) will remain uninitialized and it will be displayed as #ASC.

---

**Note:** If you need to replace an association for a Super Genie that is already displayed, you can use the function [AssWinReplace](#).

This function provides the lowest level of support for associating Super Genie variables with physical tags. The higher level functions (listed below) are simpler to use.

## Syntax

**Ass(*hWin*, *nArg*, *sTag*, *nMode* [, *sClusterName*] )**

*hWin*:

The association will be created for the next Super Genie to display in the window specified here; enter the window number or:

- -3 - for the current window when the page is changed. The page can be changed by using Page Cicode functions like PageDisplay, PageGoto, etc.
- -2 - for the next new window or page displayed.

**nArg:**

The argument number or name (substitution string number or name) of the Super Genie string to be replaced by sTag. For example, to replace ?INT 3? with sTag, set nArg to 3 ,or ?Level? set nArg to Level.

**sTag:**

Variable tag, or equipment and item name reference of a variable tag (using equipment.item notation) that will replace the Super Genie association. Partial tag names or equipment.item tag references can also be used.

The name of the tag can be prefixed by the name of the cluster for example, "*ClusterName.Tag*" or "*ClusterName.Equipment.Item*".

---

**Note:** If the tag name exceeds the length limit of 254 characters the hardware alarm "Tag name exceed length limit" will be raised.

---

**nMode:**

The mode of the association. Set to 0.

**sClusterName:**

Specifies the name of the cluster in which the Variable Tag resides. This is optional if you have one cluster or are resolving the tag via the page's current cluster context. The argument is enclosed in quotation marks "".

Resolution of the tag's cluster context occurs when the page is displayed. It is resolved to the page's cluster context, not the context in force when this function is called.

## Return Value

0 (zero) if successful, otherwise an error code is returned. When using partial associations, an error 274 or 289 may be returned even though the association is successful, and can be ignored.

## Related Functions

[AssChain](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPage](#),  
[AssChainPopUp](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssInfoEx](#), [AssPage](#),  
[AssPopUp](#), [AssScaleStr](#), [AssTag](#), [AssTitle](#), [AssVarTags](#), [AssWin](#)

## Example

```
//Using a string identifier for the substitution parameter
    Ass(-2,"Level", "MI1K_LEVEL",0);
// Associate variable tag PV123 with the next Genie to display in the current window
Ass(-3, 5, "PV123", 0);
```

## See Also

[Super Genie Functions](#)

## AssChain

Chains the associations from the current Super Genie to a new Super Genie. Use this function to display a new Super Genie when you already have one displayed. The new Super Genie will inherit the associations of the first Super Genie.

This function provides the lowest level of support for chaining associations from one Super Genie to another. You should call the higher level functions [AssChainPage\(\)](#), [AssChainWin\(\)](#), and [AssChainPopUp\(\)](#) - these functions are simpler to use.

## Syntax

**AssChain(*hDest*, *hSource*, *nMode*)**

*hDest*:

The next Super Genie to display in the window specified here will inherit the associations of the current Super Genie - enter the window number, or:

- -3: for the current window when the page is changed. The page can be changed by using the Page Cicode functions like PageDisplay, PageGoto, etc.
- 2: for the next new window or page displayed.

*hSource*:

The number of the window containing the source Super Genie (that is the Super Genie from which the associations will be inherited).

*nMode*:

The mode of the association. Set to 0.

## Return Value

Returns the number of associations copied.

## Related Functions

[Ass](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPage](#), [AssChainPopUp](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssInfoEx](#), [AssPage](#), [AssPopUp](#), [AssScaleStr](#), [AssTag](#), [AssTitle](#), [AssVarTags](#), [AssWin](#)

## Example

```
// Copy associations from the current Super Genie to !NewGenie
AssChain(WinNumber(), WinNumber(), 0);
PageDisplay("!NewGenie");
```

## See Also

[Super Genie Functions](#)

## AssChainPage

Chains the associations from the current Super Genie to a new Super Genie, and displays the new Super Genie (in the current window). Use this function to display a new Super Genie when you already have one displayed. The new Super Genie will inherit the associations of the first Super Genie.

## Syntax

**AssChainPage(*sPage*)**

*sPage*:

The page name of the Super Genie. If you prefixed your Super Genie page name with an exclamation mark (!), remember to include it here.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[Ass](#), [AssChain](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPopUp](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssInfoEx](#), [AssPage](#), [AssPopUp](#), [AssScaleStr](#), [AssTag](#), [AssTitle](#), [AssVarTags](#), [AssWin](#)

## Example

```
// Display new Super Genie in current window, using current associations
AssChainPage("!NewGenie");
```

## See Also

[Super Genie Functions](#)

## AssChainPopUp

Chains the associations from the current Super Genie to a new Super Genie, and displays the new Super Genie in a new popup window. Use this function to display a new Super Genie in a new popup window when a Super Genie is already displayed. The new Super Genie will inherit the associations of the first.

**Note:** This function helps to prevent the Super Genie from being opened more than once (at the same time). However, the same Super Genie with different associations can be opened.

## Syntax

**AssChainPopUp(*sPage*)**

*sPage*

The page name of the Super Genie. If you prefixed your Super Genie page name with an exclamation mark (!), remember to include it here.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[Ass](#), [AssChain](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPage](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssInfoEx](#), [AssPage](#), [AssPopUp](#), [AssScaleStr](#), [AssTag](#), [AssTitle](#), [AssVarTags](#), [AssWin](#)

## Example

```
// Display new Super Genie in new popup using current associations
AssChainPopUp("!NewGenie");
```

## See Also

[Super Genie Functions](#)

## AssChainWin

Chains the associations from the current Super Genie to a new Super Genie, and displays the new Super Genie in a new window. The new window will be of the same type as the current window. Use this function to display a new Super Genie in a new window when a Super Genie is already displayed. The new Super Genie will inherit the associations of the first.

## Syntax

**AssChainWin(*sPage*, *X*, *Y*, *Mode*)**

*sPage*:

The page name of the Super Genie. If you prefixed your Super Genie page name with an exclamation mark (!), remember to include it here.

*X*:

The x pixel coordinate of the top left corner of the window.

*Y*:

The y pixel coordinate of the top left corner of the window.

*Mode*:

The mode of the window:

0 - Normal page.

1 - Page child window. The window is closed when a new page is displayed, for example, when the PageDisplay() or PageGoto() function is called. The parent is the current active window.

2 - Window child window. The window is closed automatically when the parent window is freed with the WinFree() function. The parent is the current active window.

4 - No re-size. The window is displayed with thin borders and no maximize/minimize icons. The window cannot be re-sized.

8 - No icons. The window is displayed with thin borders and no maximize/minimize or system menu icons. The window cannot be re-sized.

16 - No caption. The window is displayed with thin borders, no caption, and no maximize/minimize or system menu icons. The window cannot be re-sized.

32 - Echo enabled. When enabled, keyboard echo, prompts, and error messages are displayed on the parent window. This mode should only be used with child windows (for example, Mode 1 and 2).

64 - Always on top.

128 - Open a unique window. This mode helps to prevent this window from being opened more than once.

256 - Display the entire window. This mode helps to ensure that no parts of the window will appear off the screen

512 - Open a unique Super Genie. This mode helps to prevent a Super Genie from being opened more than once (at the same time). However, the same Super Genie with different associations can be opened.

1024 - Disables dynamic resizing of the new window, overriding the setting of the [Page]DynamicSizing parameter.

You can select multiple modes by adding modes together (for example, set *Mode* to 9 to open a page child window without maximize, minimize, or system menu icons).

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[Ass](#), [AssChain](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPage](#), [AssChainPopUp](#), [AssChainWinFree](#), [AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssInfoEx](#), [AssPage](#), [AssPopUp](#), [AssScaleStr](#), [AssTag](#), [AssTitle](#), [AssVarTags](#), [AssWin](#)

## Example

```
// Displays a new super genie in a new window using the current associations
AssChainWin("!NewGenie", 100, 200, 1 + 8);
```

## See Also

[Super Genie Functions](#)

## AssChainWinFree

Stores the associations on an existing Super Genie, closes it, then assigns the tags to a new window. This allows a Super Genie popup window to call another popup window, and close the parent popup.

This function is effectively the same as the AssChainWin() function, but frees the current Super Genie.

## Syntax

**AssChainWinFree(*sPage*, *X*, *Y*, *Mode*)**

*sPage*:

The page name of the Super Genie. If you prefixed your Super Genie page name with an exclamation mark (!), remember to include it here.

X:

The x pixel coordinate of the top left corner of the window.

Y:

The y pixel coordinate of the top left corner of the window.

*Mode*:

The mode of the window:

0 - Normal page.

1 - Page child window. The window is closed when a new page is displayed, for example, when the PageDisplay() or PageGoto() function is called. The parent is the current active window.

2 - Window child window. The window is closed automatically when the parent window is freed with the WinFree() function. The parent is the current active window.

4 - No re-size. The window is displayed with thin borders and no maximize/minimize icons. The window cannot be re-sized.

8 - No icons. The window is displayed with thin borders and no maximize/minimize or system menu icons. The window cannot be re-sized.

16 - No caption. The window is displayed with thin borders, no caption, and no maximize/minimize or system menu icons. The window cannot be re-sized.

32 - Echo enabled. When enabled, keyboard echo, prompts, and error messages are displayed on the parent window. This mode should only be used with child windows (for example, Mode 1 and 2).

64 - Always on top.

128 - Open a unique window. This mode helps to prevent this window from being opened more than once.

256 - Display the entire window. This mode helps to ensure that no parts of the window will appear off the screen

512 - Open a unique Super Genie. This mode helps to prevent a Super Genie from being opened more than once (at the same time). However, the same Super Genie with different associations can be opened.

1024 - Disables dynamic resizing of the new window, overriding the setting of the [Page]DynamicSizing parameter.

You can select multiple modes by adding modes together (for example, set *Mode* to 9 to open a page child window without maximize, minimize, or system menu icons).

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[Ass](#), [AssChain](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPage](#), [AssChainPopUp](#), [AssChainWin](#), [AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssInfoEx](#), [AssPage](#), [AssPopUp](#), [AssScaleStr](#), [AssTag](#), [AssTitle](#), [AssVarTags](#), [AssWin](#)

## Example

```
// Close the current genie window and display a new genie using the current associations  
AssChainWinFree("!GeniePopup", 200, 300, 1 + 8);
```

## See Also

[Super Genie Functions](#)

### AssEquipParameters

Associates a set of equipment parameters defined in Plant SCADA Studio **System Model | Equipment | Runtime Parameters** with a page. This association is only made for the next page you display (either in the current window or in a new window). You cannot create an association for a page that is already displayed.

## Syntax

**AssEquipParameters(*hWin*, *sClusterName*, *sEquipmentName* )**

*hWin*:

The association will be created for the next page to display in the window specified here; enter the window number or:

- -3 for the current window when the page is changed. The page can be changed by using the Page Cicode functions like PageDisplay, PageGoto, etc.
- -2 for the next new window or page displayed.

*sClusterName*:

Specifies the name of the cluster in which the equipment resides. The cluster name is optional if you have one cluster or are resolving the tag via the page's current cluster context. The argument is enclosed in quotation marks "".

Resolution of the tag's cluster context occurs when the page is displayed. It is resolved to the page's cluster context, not the context in force when this function is called.

The value field of the Equipment Runtime Parameters table can work like a tag reference when the Is Tag column is set to true or default.

It can contain Variable tag, or equipment and item name reference of a variable tag (using equipment.item notation) that will replace the Super Genie association. Partial tag names or equipment.item tag references can also be used.

The name of the tag can be prefixed by the name of the cluster for example, "*ClusterName.Tag*" or "*ClusterName.Equipment.Item*".

## Return Value

0 (zero) if successful, otherwise an error code is returned. When using partial associations, an error 274 or 289 may be returned even though the association is successful, and can be ignored.

## Related Functions

[AssEquipReferences](#), [Ass](#), [AssChain](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPage](#), [AssChainPopUp](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssInfoEx](#), [AssPage](#), [AssPopUp](#), [AssScaleStr](#), [AssTag](#), [AssTitle](#), [AssVarTags](#), [AssWin](#)

## Example

```
// Associate equipment runtime parameters for equipment on Cluster1 called Pump1 with the
next window
AssEquipParameters(-2, "Cluster1", "Pump1");
```

## See Also

[Super Genie Functions](#)

## AssEquipReferences

Creates Super Genie associations for each piece of equipment referenced by the specified equipment.

## Syntax

**AssEquipReferences** (INT *hwin*, STRING *sClusterAndEquipment*, STRING *sCategory*)

*hWin*:

The association will be created for the next Super Genie to display in the window specified here; enter the window number or:

- -3 - for the current window when the page is changed. The page can be changed by using Page Cicode functions like PageDisplay and PageGoto.
- -2 - for the next new window or page displayed.

*sClusterAndEquipment*

The cluster and prefixed equipment name

*sCategory*

The category of reference to create associations for.

If no association name exists then the equipment reference will be skipped.

## Example

```
AssEquipReferences(-3, "Cluster1.Pump1", "MyCategory");
```

This example creates associations on the next page displayed in current window (-3 = same as other Ass Cicode functions) for references from "Pump1" on "Cluster1" with the category "MyCategory".

## Related Functions

[AssEquipParameters](#), [Ass](#), [AssChain](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#),

[AssChainPage](#), [AssChainPopUp](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetProperty](#), [AssGetScale](#), [AssInfo](#),  
[AssInfoEx](#), [AssPage](#), [AssPopUp](#), [AssScaleStr](#), [AssTag](#), [AssTitle](#), [AssVarTags](#), [AssWin](#)

## See Also

[Super Genie Functions](#)

### AssGetProperty

This function gets association information about the current Super Genie from the data source (that is, information about a variable tag that has been substituted into the Super Genie). You can only call this function on a Super Genie after the associations are completed.

Use this function to display association information as part of the Super Genie. For example, if you have a Super Genie that is a loop controller, you could display the name of the loop at the top of the loop controller box. Each time the Super Genie is used with different associations (specifically a different tag name association) the correct loop name will be displayed.

If a constant value is associated, then only the constant value can be retrieved through the *TagName* property. The remaining properties are not valid.

This function replaces [AssInfo](#).

## Syntax

**AssGetProperty**(*sArg*, *sProperty* [, *iCachedMode*] )

*sArg*:

The argument number or name (integer or string) of the association from which to get information.

When you associate a partial variable tag name (or partial equipment.item to reference variable tag) a prefix or suffix needs be added to the association to form a valid tag reference. The Argument *sArg* needs to be parsed as a string when partial tag names are used for the association, such as "[a]?1?[b]".

For example, where the substitution is a valid equipment reference and ?INT 1?.HourRun is a valid 'equipment.item' reference to a tag, set *sArg* to:

"?INT 1?.HourRun"

*sProperty*:

The property to read. Property names are case sensitive. Supported properties are:

Address - The configured address of the referenced tag (as specified in the Variable Tags properties).

ArraySize - Array size of the referenced tag. Returns 1 for non-array types.

AssFullName - Full name of the referenced tag in the form cluster.tagname even if the tag is not resolved.

ClusterName - Name of the cluster the referenced tag resides on.

DataBitWidth - Number of bits used to store the value.

Description - Description of the referenced tag.

EngUnitsHigh - Maximum scaled value.

EngUnitsLow - Minimum scaled value.

Equipment - Name of the equipment.

Format - Format bit string. The format information is stored in the integer as follows:

- Bits 0-7 - format width
- Bits 8-15 - number of decimal places
- Bits 16 - zero-padded
- Bit 17 - left-justified
- Bit 18 - display engineering units
- Bit 20 - exponential (scientific) notation.

ErrorValUsed - Returns 1 if the defined error value was used for the SuperGenie association. This means that tag name is invalid/unresolved or the substitutions are not complete. This is only relevant for named Super Genies. Returns zero (0) if the association string provided a value, or a default value was not defined.

FormatDecPlaces - Number of decimal places for default format.

FormatWidth - Number of characters used in default format.

FullName - Full name of the referenced tag in the form cluster.tagname If the referenced tag is not resolved, returns an empty string.

Item - Name of the equipment item associated with the Tag. If the tag is not resolved, returns an empty string.

Literal - Returns 1 if the substitution is a literal value, returns 0 if the substitution is a tag name.

RangeHigh - Maximum unscaled value.

RangeLow - Minimum unscaled value.

TagName - Name of the tag for the specified association. Will retrieve the tag name regardless if you used equipment.item to reference the tag.

Type - General type of associated tag. Allowed values are:

- 0 - Digital
- 1 - Byte
- 2 - Integer16
- 3 - UInteger16
- 4 - Long
- 5 - Real
- 6 - String
- 7 - ULong
- 8 - Undefined

Units - Engineering Units for example, %, mm, Volts.

Custom1 ... Custom8 - User-defined strings.

*iCachedMode*:

Optional parameter that specifies from where to retrieve the value for the property.

- -1 - The mode is determined by the INI parameter [\[Client\]TagReadCachedMode](#).
- 0 - The property value is retrieved direct from the server in blocking mode and an error code is returned if it is not on the server (or the server is unavailable).
- 1 - The property value is retrieved from the cache and an error code is returned if the cache is not loaded yet.
- 2 - The property value is retrieved from the local configuration and an error code is returned if it is not available (this is the traditional behavior).

- 3 - The property value is retrieved from the cache, if the cache is loaded, and from the local configuration, if the cache is not loaded yet.

Default value is -1.

**Note:** When retrieving bit width ("DataBitWidth" property) or array size ("ArraySize" property) from the local configuration (iCachedMode 2), the value is related to the data structure in the device. For example, MODNET, is a 16 bit device and does not have native LONG and REAL numbers. So when you have a LONG variable, an array of 2 INTEGERS are needed to store it. In the example, the property "DataBitWidth" against the variable will return 16 and property "ArraySize" will return 2. The combination of these two return values will add up to the correct bits.

## Return Value

String representation of the property of the referenced tag. On detection of an error, an empty string and an error are set.

## Related Functions

[Ass](#), [AssChain](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPage](#), [AssChainPopUp](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetScale](#), [AssInfo](#), [AssInfoEx](#), [AssPage](#), [AssPopUp](#), [AssScaleStr](#), [AssTag](#), [AssTitle](#), [AssVarTags](#), [AssWin](#), [TagGetProperty](#), [TagGetScale](#), [TagScaleStr](#), [TagInfo](#)

## Example

```
//Using a string identifier for the substitution parameter
AssGetProperty("MILK_LEVEL", "TagName", 0);
// Get the engineering full scale value for the 2nd
// argument of the association of the current Super Genie
EngFullScale = AssGetProperty(2, "EngUnitsHigh", 0);
// Get the cached engineering units for the 3rd argument
// of the association of the current Super Genie
MeasureUnits = AssGetProperty(3, "Units", 1);
```

## See Also

[Super Genie Functions](#)

## AssGetScale

Gets scale information about the tag references for the current Super Genie from the datasource (that is scale information about a variable tag that has been substituted into the Super Genie). You can only call this function on a Super Genie after the associations are completed.

Use this function to display association scale information as part of the Super Genie. For example, if you have a bar graph illustrating output, you could indicate zero, 50%, and full scale output on the vertical axis of the graph. Each time the Super Genie is used with different associations the correct scale values will be displayed.

The value is returned as a formatted string using the association format specification and (optionally) the engineering units.

This function replaces [AssScaleStr](#).

## Syntax

**AssGetScale(sArg, iPercent, iEngUnits [, iCached] )**

*sArg:*

The argument number or name of the association from which to get information.

When you associate a partial variable tag name (or partial equipment.item to reference variable tag) a prefix or suffix needs be added to the association to form a valid tag reference. The Argument sArg needs to be parsed as a string when partial tag names are used for the association ie "[a]?1?[b]".

For example, where the substitution is a valid equipment reference and ?INT 1?.HourRun is a valid equipment.item reference to a tag.

```
set sArg to "?INT 1?.HourRun"
```

*iPercent:*

The percentage of full scale of the returned value.

*iEngUnits:*

Flag to determine if the value is returned with engineering units:

0 - Return the value without engineering units

1 - Return the value with engineering units

*iCached:*

Optional flag to attempt to retrieve the cached value for the property rather than the current value. This makes the function non-blocking. If the property has not yet been cached, an error is set.

0 - Do not force cached read. Cicode is blocking

1 - Force cached read. Cicode is non-blocking

Default value is 1 (true).

## Return Value

The scale of the referenced tag (as a string).

## Related Functions

[Ass](#), [AssChain](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPage](#), [AssChainPopUp](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetProperty](#), [AssInfo](#), [AssInfoEx](#), [AssPage](#), [AssPopUp](#), [AssScaleStr](#), [AssTag](#), [AssTitle](#), [AssVarTags](#), [AssWin](#), [TagGetProperty](#), [TagGetScale](#), [TagScaleStr](#), [TagInfo](#)

## Example

```
//Using a string identifier for the substitution parameter
AssGetScale("MILK_LEVEL", 50, 1, 0);

// Display the zero, 50% and full scale of the 2nd argument
// of the association of the current Super Genie
```

```
DspText(31,0,AssGetScale(2, 0, 1));
DspText(32,0,AssGetScale(2, 50, 1));
DspText(33,0,AssGetScale(2, 100, 1));
```

## See Also

[Super Genie Functions](#)

### AssInfo

Gets association information about the current Super Genie (that is information about a variable tag that has been substituted into the Super Genie). You can only call this function on a Super Genie after the associations are completed.

Use this function to display association information as part of the Super Genie. For example, if you have a Super Genie that is a loop controller, you could display the name of the loop at the top of the loop controller box. Each time the Super Genie is used with different associations (specifically a different tag name or equipment.item association) the correct loop name will be displayed.

**Note:** In some cases it may be beneficial to replace AssInfo with either the [AssGetProperty](#) or the [AssInfoEx](#) function. If the Tag properties are updated, AssInfo does not get the updated values whereas AssGetProperty does. In addition, the function [AssInfoEx](#) has been introduced to make it easier to make legacy Cicode compatible with online changes. In a large number of cases AssInfo can be replaced with AssInfoEx using Find and Replace (see [Using Find and Replace](#)). Please be aware that if you are replacing an instance of AssInfo with AssInfoEx in a loop, you may want to make AssInfoEx blocking using the *iCached* argument to verify you are using the correct value for the Tag in your logic.

## Syntax

**AssInfo(sArg, nType [, iCachedMode])**

**sArg:**

When you associate variable tags (can use equipment.item to reference the variable tag) with Super Genies, the Super Genie substitution strings are replaced by variable tags. The sArg argument allows you to get information about one of those variable tags. What you need to know is which substitution string it replaced when the association was performed.

Enter the argument number or name (substitution string number or name) of the relevant substitution string. For example, if you want information about the variable that replaced substitution string

?INT 3?

set sArg to 3.

Or

?Level?

set sArg to Level

When you associate a partial variable tag name (or partial equipment.item to reference variable tag) a prefix or suffix needs be added to the association to form a valid tag reference. The Argument sArg needs to be parsed as a string when partial tag names are used for the association ie "[a]?1?[b]".

eg where the substitution is a valid equipment reference and ?INT 1?.HourRun is a valid equipment.item reference to a tag.

set sArg to "?INT 1?.HourRun"

*nType:*

The type of information to get:

0 - Tag reference as defined in sArg. If the referenced tag is not resolved, returns an empty string.

1 - Engineering units

2 - Raw zero scale

3 - Raw full scale

4 - Engineering zero scale

5 - Engineering full scale

6 - Width of the format

7 - Number of decimal places of format

8 - The Tag format as a long integer. The format information is stored in the integer as follows:

- Bits 0-7 - format width
- Bits 8-15 - number of decimal places
- Bits 16 - zero-padded
- Bit 17- left-justified
- Bit 18 - display engineering units
- Bit 20 - exponential (scientific) notation

9 - Logical Unit Number - I/O device number (for internal use)

10 - Raw Type - Protocol's raw data type number for this tag. Type numbers are:

- 0 - Digital
- 1 - Integer
- 2 - Real
- 3 - BCD
- 4 - Long
- 5 - Long BCD
- 6 - Long Real
- 7 - String
- 8 - Byte
- 9 - Void
- 10 - Unsigned integer

11 - Bit Width - Tag's size in bits. For example, an INT is 16 bits

12 - Unit Type - Protocol's unit type number for this tag

13 - Unit Address - Tag's address after the protocol DBF's template is applied.

14 - Unit Count - Array size. For example, if the tag's address is I1[50], the unit count is 50.

15 - Record Number - Tag's record number in variable.DBF - 1. That is, the first tag has a record number of 0.

16 - Comment - As defined in the variable tags list.

17 - ClusterName of the tag. If the referenced tag is not resolved, returns an empty string.

18 - Full name (*cluster.tagname*) of the referenced tag. If the tag is not resolved, returns an empty string.

19 - Full name (*cluster.tagname*) of the referenced tag even if the tag is not resolved.

20 - Configured Address of the tag. If the tag is not resolved, returns an empty string.

21 - Network Number - I/O device number (as defined by the Number field of the I/O Devices dialog).

22 - Name of the equipment associated with the Tag. If the referenced tag is not resolved, returns an empty string.

23 - General Type.

- 0 - Digital
- 1 - Byte
- 2 - Integer16
- 3 - UInteger 16
- 4 - Long
- 5 - Real
- 6 - String
- 7 - ULONG
- 8 - Undefined

24 - Reserved for internal use.

25 - Name of the equipment item associated with the Tag. If the tag is not resolved, returns an empty string.

If the tag is a local variable, mode 9 error code 348 is retrieved when cache mode is 0,1,2,3. Modes 12, 13, 14 and 15 will return an empty string (error 274 is trapped).

Type 8 - 16 are only supported when *iCachedMode* equals to 2 or 3.

26 - Custom 1 - a user-defined string.

27 - Custom 2 - a user-defined string.

28 - Custom 3 - a user-defined string.

29 - Custom 4 - a user-defined string.

30 - Custom 5 - a user-defined string.

31 - Custom 6 - a user-defined string.

32 - Custom 7 - a user-defined string.

33 - Custom 8 - a user-defined string.

#### *iCachedMode:*

Optional parameter that specifies from where to retrieve the value for the property.

- -1 - The mode is determined by the INI parameter [Client]TagReadCachedMode.
- 0 - The property value is retrieved direct from the server in blocking mode and an error code is returned if it is not on the server (or the server is unavailable).
- 1 - The property value is retrieved from the cache and an error code is returned if the cache is not loaded yet.
- 2 - The property value is retrieved from the local configuration and an error code is returned if it is not available (this is the traditional behavior).
- 3 - The property value is retrieved from the cache, if the cache is loaded, and from the local configuration, if the cache is not loaded yet.

Default value is -1.

---

**Note:** When retrieving bit width (Type 11) or unit count (Type 14) from local configuration (iCachedMode 2), the value is related to the data structure in the device. For example, MODNET, This is a 16 bit device does not have native LONG and REAL numbers. So when you have a LONG variable, then 2 INTEGERS are needed to store it. In the case, Type 11 against the variable will return 16 and Type 14 will return 2. The combination of these two return values will add up to the correct bits.

---

## Return Value

The value of the information as a string.

## Related Functions

[Ass](#), [AssChain](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPage](#), [AssChainPopUp](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetProperty](#), [AssGetScale](#), [AssInfoEx](#), [AssPage](#), [AssPopUp](#), [AssScaleStr](#), [AssTag](#), [AssTitle](#), [AssVarTags](#), [AssWin](#), [TagGetProperty](#), [TagGetScale](#), [TagScaleStr](#), [TagInfo](#)

## Example

```
//Using a string identifier for the substitution parameter
AssInfo("MILK_LEVEL", 1); (to get Engineering Units of the association )

sTag = AssInfo(1, 0); // Get the name of association 1
sEngLow = AssInfo(1, 4); // get the low engineering scale of association 1

sTag = AssInfo("?INT 1?.HourRun", 0) //Get the name where the super genie association is
only part of the tag name
```

## See Also

[Super Genie Functions](#)

## AssInfoEx

Gets association information about the current Super Genie (that is information about a variable tag that has been substituted into the Super Genie). You can only call this function on a Super Genie after the associations are completed.

Use this function to display association information as part of the Super Genie. For example, if you have a Super Genie that is a loop controller, you could display the name of the loop at the top of the loop controller box. Each time the Super Genie is used with different associations (specifically a different tag name association) the correct loop name will be displayed.

---

**Note:** When replacing an instance of AssInfo with AssInfoEx in a loop, you may want to make AssInfoEx blocking using the iCached argument to verify you are using the correct value for the Tag in your logic.

---

## Syntax

**AssInfoEx(sArg, nType [, iCachedMode] )**

*sArg:*

When you associate variable tags with super Genies, the Super Genie substitution strings are replaced by variable tags. The sArg argument allows you to get information about one of those variable tags. What you need to know is which substitution string it replaced when the association was performed.

Enter the argument number (substitution string number) of the relevant substitution string. For example, if you want information about the variable that replaced substitution string

?INT 3?

set sArg to 3.

When you associate a partial variable tag name (or partial equipment.item to reference variable tag) a prefix or suffix needs be added to the association to form a valid tag reference. The Argument sArg needs to be parsed as a string when partial tag names are used for the association ie "[a]?1?[b]".

eg where the substitution is a valid equipment reference and ?INT 1?.HourRun is a valid equipment.item reference to a tag.

set sArg to "?INT 1?.HourRun"

*nType:*

The type of information to get:

0 - The referenced tag from the variables table. This is the same as sName argument. (Returned to be compatible with the AssInfo() function).

1 - Engineering units

2 - Raw zero scale

3 - Raw full scale

4 - Engineering zero scale

5 - Engineering full scale

6 - Width of the format

7 - Number of decimal places of format

8 - The Tag format as a long integer. The format information is stored in the integer as follows:

- Bits 0-7 - format width

- Bits 8-15 - number of decimal places

- Bits 16 - zero-padded

- Bit 17- left-justified

- Bit 18 - display engineering units

- Bit 20 - exponential (scientific) notation

9 - Logical Unit Number - I/O device number (for internal use)

10 - Raw Type - Protocol's raw data type number for this tag. Type numbers are:

- 0 - Digital

- 1 - Integer

- 2 - Real

- 3 - BCD

- 4 - Long

- 5 - Long BCD

- 6 - Long Real

- 7 - String
- 8 - Byte
- 9 - Void
- 10 - Unsigned integer

11 - Bit Width - Tag's size in bits. For example, an INT is 16 bits

12 - Unit Type - Protocol's unit type number for this tag

13 - Unit Address - Tag's address after the protocol DBF's template is applied.

14 - Unit Count - Array size. For example, if the tag's address is I1[50], the unit count is 50.

15 - Record Number - Tag's record number in variable.DBF - 1. That is, the first tag has a record number of 0.

16 - Comment - As defined in the variable tags list.

17 - ClusterName of the tag.

18 - Full name (*cluster.tagname*) of the tag.

19 - Full name (*cluster.tagname*) of the referenced tag even if the tag is not resolved.

20 - Configured Address of the tag. If the tag is not resolved, returns an empty string.

21 - Network Number - I/O device number (as defined by the Number field of the I/O Devices dialog).

22 - Name of the equipment associated with the Tag. If the association tag is not resolved, returns an empty string.

If the tag is a local variable, mode 9 error code 348 is retrieved when cache mode is 0,1,2,3. Modes 12, 13, 14 and 15 will return an empty string (error 274 is trapped).

23 - General Type.

- 0 - Digital
- 1 - Byte
- 2 - Integer16
- 3 - UInteger 16
- 4 - Long
- 5 - Real
- 6 - String
- 7 - ULong
- 8 - Undefined

24 - Reserved for internal use.

25 - Name of the equipment item associated with the Tag. If the tag is not resolved, returns an empty string.

26 - Custom 1 - a user-defined string.

27 - Custom 2 - a user-defined string.

28 - Custom 3 - a user-defined string.

29 - Custom 4 - a user-defined string.

30 - Custom 5 - a user-defined string.

31 - Custom 6 - a user-defined string.

32 - Custom 7 - a user-defined string.

33 - Custom 8 - a user-defined string.

***iCachedMode:***

Optional parameter that specifies from where to retrieve the value for the property.

-1 - The mode is determined by the INI parameter [\[Client\]TagReadCachedMode](#).

0 - The property value is retrieved direct from the server in blocking mode and an error code is returned if it is not on the server (or the server is unavailable).

1 - The property value is retrieved from the cache and an error code is returned if the cache is not loaded yet.

2 - The property value is retrieved from the local configuration and an error code is returned if it is not available (this is the traditional behaviour).

3 - The property value is retrieved from the cache, if the cache is loaded, and from the local configuration, if the cache is not loaded yet.

Default value is -1.

**Note:** When retrieving bit width (Type 11) or unit count (Type 14) from local configuration (*iCachedMode* 2), the value is related to the data structure in the device. For example, MODNET, This is a 16 bit device does not have native LONG and REAL numbers. So when you have a LONG variable, then 2 INTEGERS are needed to store it. In the case, Type 11 against the variable will return 16 and Type 14 will return 2. The combination of these two return values will add up to the correct bits.

## Return Value

The value of the information as a string. If an error is detected an empty string is returned. The error code can be obtained by calling the [IsError](#) Cicode function.

## Related Functions

[Ass](#), [AssChain](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPage](#),  
[AssChainPopUp](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssPage](#), [AssPopUp](#),  
[AssScaleStr](#), [AssTag](#), [AssTitle](#), [AssVarTags](#), [AssWin](#), [TagGetProperty](#), [TagGetScale](#), [TagScaleStr](#), [TagInfo](#)

## Example

```
//Using a string identifier for the substitution parameter
AssInfoEx("MILK_LEVEL", 1, 0); //to get Engineering units, no cache read)
sTag = AssInfoEx(1, 0); // Get the name of association 1 in non-blocking mode
sEngLow = AssInfoEx(1, 4, 0); // get the low engineering scale of association 1,
block until data is available

sTag = AssInfoEx("?INT 1?.HourRun", 0) //Get the name where the super genie association is
only part of the tag name
```

## See Also

[Super Genie Functions](#)

## AssMetadata

This non-blocking function performs Super Genie associations using the "Name" and "Value" fields defined on

the Object Properties - Metadata tab, and matches it to the 'Name' field in the page associations table. While performing associations any additional metadata entries are ignored.

## Syntax

**AssMetadata(*hWin* [, *nAn*])**

*hWin*:

The associations will be created for the next Super Genie to display in the window specified. Enter the window number or:

- -3: for the current window when the page is changed. The page can be changed by using the Page Cicode functions like PageDisplay, PageGoto, etc.
- -2: for the next new window or page displayed.

*nAN*:

An animation number that uniquely identifies an object. This object contains the list of metadata definitions that will be used to perform the association operations. This parameter is optional with -2 being the default value. When -2 is specified, it is equivalent to [DspGetAnCur\(\)](#) which returns the animation number of the current active command cursor, please refer [DspGetAnCur\(\)](#) for usage and limitations.



### UNINTENDED EQUIPMENT OPERATION

If called after other Cicode functions in a command expression field, retrieve the animation number first, then pass it through the *nAN* parameter.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Return Value

0 (zero) if successful, otherwise an error code is returned. When using partial associations, an error 274 may be returned even though the association is successful, and can be ignored.

## Example

```
/* Example of calling AssMetadata after other cicode functions */
An = DspGetAnCur();
SomeVal = TagRead("SomeTag"); // do additional work
AssMetadata(-2, An);
PageOpen("!TestSG");
```

## Related Functions

[Ass](#), [AssChain](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPage](#), [AssChainPopUp](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssInfoEx](#), [AssPage](#), [AssPopUp](#), [AssScaleStr](#), [AssTag](#), [AssTitle](#), [AssVarTags](#), [AssWin](#)

## See Also

[Super Genie Functions](#)

### AssMetadataPage

Uses the metadata information from the current object for the page associations for a new Super Genie page, and displays the new Super Genie (in the current page).

## Syntax

**AssMetadataPage(*sPage* [,*nAN*])**

*sPage*:

The name of the Super Genie page to open.

*nAN*:

An animation number that uniquely identifies an object. This object contains the list of metadata definitions that will be used to perform the association operations. This parameter is optional with -2 being the default value. When -2 is specified, it is equivalent to DspGetAnCur() which returns the animation number of the current active command cursor, please refer DspGetAnCur() for usage and limitations.



#### UNINTENDED EQUIPMENT OPERATION

If called after other cicode functions in a command expression field, retrieve the animation number first, then pass it through the *nAN* parameter.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Example

```
/* Example of calling AssMetadataPage after other cicode functions */
An = KeyGetCursor();
SomeVal = TagRead("SomeTag"); // do additional work
AssMetadataPage("!TestSG", An);
```

## Related Functions

[Ass](#), [AssChain](#), [AssMetadata](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPage](#), [AssChainPopUp](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssInfoEx](#), [AssPage](#), [AssPopUp](#), [AssScaleStr](#), [AssTag](#), [AssTitle](#), [AssVarTags](#), [AssWin](#)

## See Also

[Super Genie Functions](#)

### AssMetadataPopUp

Uses the metadata information from the current animation point for the associations for a new Super Genie page, and displays the new Super Genie in a pop up window.

## Syntax

**AssMetadataPopUp(*sPage* [,*nAN*])**

*sPage*

The name of the Super Genie page to open.

*nAN*:

An animation number that uniquely identifies an object. This object contains the list of metadata definitions that will be used to perform the association operations. This parameter is optional with -2 being the default value. When -2 is specified, it is equivalent to [DspGetAnCur\(\)](#) which returns the animation number of the current active command cursor, please refer [DspGetAnCur\(\)](#) for usage and limitations.



#### UNINTENDED EQUIPMENT OPERATION

If called after other cicode functions in a command expression field, retrieve the animation number first, then pass it through the *nAN* parameter.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Example

```
/* Example of calling AssMetadataPopUp after other cicode functions */
An = DspGetAnCur();
SomeVal = TagRead("SomeTag"); // do additional work
AssMetadataPopUp("!TestSG", An);
```

## Related Functions

[Ass](#), [AssChain](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataWin](#), [AssChainPage](#), [AssChainPopUp](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssInfoEx](#), [AssPage](#), [AssPopUp](#), [AssScaleStr](#), [AssTag](#), [AssTitle](#), [AssVarTags](#), [AssWin](#)

## See Also

[Super Genie Functions](#)

### AssMetadataWin

Uses the metadata information from the current animation-point for the associations for a new Super Genie page, and displays the new Super Genie in a new window.

## Syntax

**AssMetadataWin(*sPage*, INT *x*, INT *y*, INT *mode* [,*nAN*])**

*sPage*:

The name of the Super Genie page to open.

*X*:

The x pixel coordinate of the top left corner of the window. Default value is 0.

*Y*:

The y pixel coordinate of the top left corner of the window. Default value is 0.

*Mode*:

The mode of the window:

0 - Normal page (default value).

1 - Page child window. The window is closed when a new page is displayed, for example, when the PageDisplay() or PageGoto() function is called. The parent is the current active window.

2 - Window child window. The window is closed automatically when the parent window is freed with the WinFree() function. The parent is the current active window.

4 - No re-size. The window is displayed with thin borders and no maximize/minimize icons. The window cannot be re-sized.

8 - No icons. The window is displayed with thin borders and no maximize/minimize or system menu icons. The window cannot be re-sized.

16 - No caption. The window is displayed with thin borders, no caption, and no maximize/minimize or system menu icons. The window cannot be re-sized.

32 - Echo enabled. When enabled, keyboard echo, prompts, and error messages are displayed on the parent window. This mode should only be used with child windows (for example, Mode 1 and 2).

64 - Always on top.

128 - Open a unique window. This mode stops this window from being opened more than once.

256 - Display the entire window. This mode commands that no parts of the window will appear off the screen

512 - Open a unique Super Genie. This mode stops a Super Genie from being opened more than once (at the same time). However, the same Super Genie with different associations can be opened.

1024 - Disables dynamic resizing of the new window, overriding the setting of the [\[Page\]DynamicSizing](#) parameter.

4096 - Allows the window to be resized without maintaining the current aspect ratio. The aspect ratio defines the relationship between the width and the height of the window, which means this setting allows you to stretch

or compress the window to any proportions. This option overrides the setting of the [Page]MaintainAspectRatio parameter.

8192 - Text on a page will be resized in proportion with the maximum scale change for a resized window. For example, consider a page that is resized to three times the original width, and half the original height. If this mode is set, the font size of the text on the page will be tripled (in proportion with the maximum scale). This option overrides the setting of the [\[Page\]ScaleTextToMax](#) parameter.

16384 - Hide the horizontal scroll bar.

32768 - Hide the vertical scroll bar.

65536 - Disable horizontal scrolling.

131072 - Disable vertical scrolling.

You can select multiple modes by adding modes together (for example, set Mode to 9 to open a page child window without maximize, minimize, or system menu icons).

nAN:

An animation number that uniquely identifies an object. This object contains the list of metadata definitions that will be used to perform the association operations. This parameter is optional with -2 being the default value. When -2 is specified, it is equivalent to [DspGetAnCur\(\)](#) which returns the animation number of the current active command cursor, please refer [DspGetAnCur\(\)](#) for usage and limitations.

## WARNING

### UNINTENDED EQUIPMENT OPERATION

If called after other Cicode functions in a command expression field, retrieve the animation number first, then pass it through the nAN parameter.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Example

```
/* Example of calling AssMetadataWin after other cicode functions */
An = DspGetAnCur();
SomeVal = TagRead("SomeTag"); // do additional work
AssMetadataWin("!TestSG", 50, 50, 1, An);
```

## Related Functions

[Ass](#), [AssChain](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPage](#), [AssChainPopUp](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssInfoEx](#), [AssPage](#), [AssPopUp](#), [AssScaleStr](#), [AssTag](#), [AssTitle](#), [AssVarTags](#), [AssWin](#)

## See Also

[Super Genie Functions](#)

## AssPage

Associates up to eight variable tags or equipment.item tag references with a Super Genie and displays the Super Genie in the current window. Partial tag names and equipment.item tag references can also be used. The first variable tag (*sTag1*) replaces Super Genie substitution string 1. The second variable tag (*sTag2*) replaces substitution string 2, and so on.

This function has the same effect as calling Ass() or AssTag() eight times, and then calling the PageDisplay() function. The AssPage() function provides a quick way of associating eight Super Genie variables and displaying the Super Genie - at the same time.

If you want to associate more than eight tags with the Super Genie, it is strongly recommended you call the AssVarTags(), AssTag(), or Ass() function to create the associations before you call this function.

## Syntax

**AssPage(*sPage*, *sTag1*, [*sTag2..8*] )**

*sPage*:

The page name of the Super Genie. If you prefixed your Super Genie page name with an exclamation mark (!), remember to include it here.

*sTag1..sTag8*:

The first eight physical tags to be associated with the Super Genie. For any given Super Genie, the variable tags will replace the Super Genie substitution strings as follows:

Variable tag	replaces substitution string
<i>sTag1</i>	1
<i>sTag2</i>	2
<i>sTag3</i>	3
<i>sTag4</i>	4
<i>sTag5</i>	5
<i>sTag6</i>	6
<i>sTag7</i>	7
<i>sTag8</i>	8

Because there is a strict correlation between the variable tag numbers and the substitution string numbers, you need to know how your Super Genie substitutions are numbered. For example, if your Super Genie has three unique substitution strings, numbered 1, 3, & 4, you need to enter a blank ("") for *sTag2*.

The name of the tag can be prefixed by the name of the cluster for example, "*ClusterName.Tag*".

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[Ass](#), [AssChain](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPage](#),  
[AssChainPopUp](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssInfoEx](#), [AssPage](#),  
[AssPopUp](#), [AssScaleStr](#), [AssTag](#), [AssTitle](#), [AssVarTags](#), [AssWin](#)

## Example

```
// Associate 3 tags with the Super Genie then display the Super Genie
AssPage("!MyGenie", "PV123", "OP123", "SP123");
```

## See Also

[Super Genie Functions](#)

## AssPopUp

Associates up to eight variable tags or equipment.item tag references with a Super Genie and displays the Super Genie in a popup window. Partial tag names or equipment.item tag references can also be used. The first variable tag (*sTag1*) replaces Super Genie substitution string 1. The second variable tag (*sTag2*) replaces substitution string 2, and so on.

This function has the same effect as calling the [Ass\(\)](#) function or the [AssTag\(\)](#) function eight times, and then calling the [WinNewAt\(\)](#) function to create a window at the position of the mouse. The [AssPopUp\(\)](#) function is a quick way of associating eight Super Genie variables and displaying the Super Genie in a new window at the same time.

If you want to associate more than eight tags with the Super Genie, you need to call the [AssVarTags\(\)](#), [AssTag\(\)](#), or [Ass\(\)](#) function to create the associations before you call this function.

---

**Note:** This function helps to prevent the Super Genie from being opened more than once (at the same time). However, the same Super Genie with different associations can be opened.

---

## Syntax

**AssPopUp(*sPage*, *sTag1*, [*sTag2..8*] )**

*sPage*:

The page name of the Super Genie. If you prefixed your Super Genie page name with an exclamation mark (!), remember to include it here.

*sTag1..sTag8*:

The first 8 physical tags to be associated with the Super Genie. For any given Super Genie, the variable tags will replace the Super Genie substitution strings as follows:

Variable tag	replaces substitution string
sTag1	1
sTag2	2
sTag3	3
sTag4	4
sTag5	5
sTag6	6
sTag7	7
sTag8	8

Because there is a strict correlation between the variable tag numbers and the substitution string numbers, you need to know how your Super Genie substitutions are numbered. For example, if your Super Genie has three unique substitution strings, numbered 1, 3, & 4, you have to enter a blank ("") for sTag2.

The name of the tag can be prefixed by the name of the cluster for example, "*ClusterName.Tag*".

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[Ass](#), [AssChain](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPage](#), [AssChainPopUp](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssInfoEx](#), [AssPage](#), [AssScaleStr](#), [AssTag](#), [AssTitle](#), [AssVarTags](#), [AssWin](#)

## Example

```
// Associate 3 tags with the Super Genie then display it
AssPopUp("!MyGenie", "PV123", "OP123", "SP123");
```

## See Also

[Super Genie Functions](#)

## AssScaleStr

Gets scale information about the associations of the current Super Genie (that is scale information about a variable tag that has been substituted into the Super Genie). You can only call this function on a Super Genie after the associations are completed.

Use this function to display association scale information as part of the Super Genie. For example, if you have a

bar graph illustrating output, you could indicate zero, 50%, and full scale output on the vertical axis of the graph. Each time the Super Genie is used with different associations the correct scale values will be displayed.

The value is returned as a formatted string using the association format specification and (optionally) the engineering units.

---

**Note:** This function is being deprecated and is replaced by the [AssGetScale](#) function. If the Tag properties are updated AssScaleStr does not get the updated values whereas AssGetScale does.

---

## Syntax

STRING **AssScaleStr**(STRING *Arg*, INT *Percent*, INT *EngUnits*[,INT *CachedMode*])

*sArg*:

When you associate variable tags with Super Genies, the Super Genie substitution strings are replaced by variable tags. The *nArg* argument allows you to get scale information about a particular variable tag. You need to know which substitution string the tag replaced when the association was performed.

Enter the argument number or name (substitution string number or name) of the relevant substitution string. For example, if you want scale information about the variable that replaced substitution string:

?INT 3?

set *nArg* to 3.

or

?Level?

set *nArg* to Level

When you associate a partial variable tag name (or partial equipment.item to reference variable tag) a prefix or suffix needs be added to the association to form a valid tag reference. The Argument *sArg* needs to be parsed as a string when partial tag names are used for the association ie "[a]?1?[b]".

For example, where the substitution is a valid equipment reference and ?INT 1?.HourRun is a valid equipment.item reference to a tag.

set *sArg* to "?INT 1?.HourRun"

*Percent*:

The percentage of full scale of the returned value.

*EngUnits*:

Determines if the value is returned with engineering units:

0 - Do not return the value with engineering units

1 - Return the value with engineering units

*CachedMode*:

Optional parameter that specifies from where to retrieve the value for the property.

-1 - The mode is determined by the INI parameter [\[Client\]TagReadCachedMode](#).

0 - The property value is retrieved direct from the server in blocking mode and an error code is returned if it is not on the server (or the server is unavailable).

1 - The property value is retrieved from the cache and an error code is returned if the cache is not loaded yet.

2 - The property value is retrieved from the local configuration and an error code is returned if it is not available (this is the traditional behaviour).

3 - The property value is retrieved from the cache, if the cache is loaded, and from the local configuration, if the cache is not loaded yet.

Default value is -1.

## Return Value

The scale of the referenced tag(as a string).

## Related Functions

[Ass](#), [AssChain](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPage](#),  
[AssChainPopUp](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssInfoEx](#), [AssPage](#),  
[AssPopUp](#), [AssTag](#), [AssTitle](#), [AssVarTags](#), [AssWin](#)

## Example

```
//Using a string identifier for the substitution parameter
AssScaleStr("MILK_LEVEL", 50, 1);

// Display the zero, 50% and full scale of the variable that was substituted for
Super Genie arg no. 3DspText(31,0,AssScaleStr(3, 0, 1));DspText(32,0,AssScaleStr(3, 50,
1));DspText(33,0,AssScaleStr(3, 100, 1));
```

## See Also

[Super Genie Functions](#)

## AssTag

Associates a variable tag or equipment.item tag reference with a Super Genie. The association will only be created for the next Super Genie you display in the current window, and will only come into effect after you re-display the Super Genie. You need to call this function once for every substitution string in the current Super Genie, or the super-genie variable (substitution string) will remain uninitialized and it will display as #ASC. You cannot use this function to create associations for variables that will display in new windows.

## Syntax

**AssTag**(*nArg*, *sTag* [, *sClusterName*] )

*nArg*:

The argument name (substitution string name) of the Super Genie string to be replaced by *sTag*. For example, to replace ?INT equip? with *sTag*, set *nArg* to equip.

*sTag*:

The variable tag or equipment.item tag reference that will replace the Super Genie substitution string. Partial tag names or equipment.item tag references can also be used.

The name of the tag can be prefixed by the name of the cluster for example, "ClusterName.Tag" or "ClusterName.equipment.item".

***sClusterName:***

Specifies the name of the cluster in which the Variable Tag resides. This is optional if you have one cluster or are resolving the tag via the current cluster context. The argument is enclosed in quotation marks "".

Resolution of the tag's cluster context occurs when the page is displayed. It is resolved to the page's cluster context, not the context in force when this function is called.

**Return Value**

0 (zero) if successful, otherwise an error code is returned. When using partial associations, an error 274, or 289 may be returned even though the association is successful, and can be ignored.

**Related Functions**

[Ass](#), [AssChain](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPage](#), [AssChainPopUp](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssInfoEx](#), [AssPage](#), [AssPopUp](#), [AssScaleStr](#), [AssTitle](#), [AssVarTags](#), [AssWin](#)

**Example**

```
// Associate variable tag PV123 and PV124 with !MyGenie
AssTag(1, "PV123");
AssTag(2, "PV124");
// Re-display the current Super Genie
PageDisplay("!MyGenie");
```

**See Also**

[Super Genie Functions](#)

**AssTitle**

Sets the runtime window title to the tag name of the first variable substituted into the Super Genie.

---

**Note:** This function only supports associations using tag names or equipment.item references.

See [Page Properties - General](#) for information regarding using named associations in the Window Title field.

**Syntax**

**AssTitle( [Mask] [, Prefix] [, Suffix])**

***Mask:***

The number of characters to mask (hide) from the right of the title string (optional).

***Prefix:***

A string to add to the beginning of the title string (optional).

***Suffix:***

A string to add to the end of the title string (optional).

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[Ass](#), [AssChain](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPage](#),  
[AssChainPopUp](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssInfoEx](#), [AssPage](#),  
[AssPopUp](#), [AssScaleStr](#), [AssTag](#), [AssVarTags](#), [AssWin](#)

## See Also

[Super Genie Functions](#)

## AssVarTags

Associates up to eight variable tags or equipment.item tag references with a Super Genie. Partial tag names or equipment.item tag references can also be used. This association is only made for the next Super Genie you display (either in the current window or in a new window). This function has an offset that allows you to specify which substitution string the first variable tag will replace. This means that if you have a Super Genie with more than 8 substitution strings, you can use this function repeatedly (while increasing the offset), until you have associated the necessary variable tags.

This function has the same effect as calling the Ass() function or the AssTag() function eight times. The AssVarTags() function is a quick way of associating up to eight Super Genie variables at the same time.

**Note:** This function does not support named associations. To use named associations refer to [AssMetadata](#).

## Syntax

**AssVarTags(*hWin*, *nOffset*, *sTag1*, [*sTag2..8*] )**

*hWin*:

The association will be created for the next Super Genie to display in the window specified here - enter the window number or:

- -3 - for the current window when the page is changed. The page can be changed by using the Page Cicode functions like PageDisplay, PageGoto, etc.
- -2 - for the next new window or page displayed.

*nOffset*:

By default, the first variable tag (*sTag1*) will replace substitution string 1, and *sTag2* will replace substitution string 2, and so on. Enter an offset to change this so that *sTag1* replaces a substitution string other than the first. For example, an offset of 8 means that *sTag1* replaces string 9 instead of the default string 1 (8+1=9), and *sTag2* replaces string 10 instead of string 2 (8+2=10) etc. This means that you can use this function repeatedly to associate more than eight variables.

*sTag1..8*:

The physical variable tags (up to eight) to be associated with the Super Genie. For any given Super Genie, the variable tags will replace the Super Genie substitution strings as follows:

If <i>nOffset</i> is 0...	<i>sTag1</i> will replace the substitution string 1,
	<i>sTag2</i> will replace the substitution string 2, etc.
If <i>nOffset</i> is 8...	<i>sTag1</i> will replace the substitution string 9,
	<i>sTag2</i> will replace the substitution string 10, etc.

Because there is a strict correlation between the variable tag numbers and the substitution string numbers, you need to know how your Super Genie substitutions are numbered. For example, if your Super Genie has three unique substitution strings, numbered 1, 3, & 4, you need to enter a blank ("") for *sTag2*.

The name of the tag can be prefixed by the name of the cluster for example, "*ClusterName.Tag*".

## Return Value

No value is returned.

## Related Functions

[Ass](#), [AssChain](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPage](#), [AssChainPopUp](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssInfoEx](#), [AssPage](#), [AssPopUp](#), [AssScaleStr](#), [AssTag](#), [AssTitle](#), [AssWin](#)

## Example

```
// Associate 12 variables to the Super Genie
AssVarTags(WinNumber(), 0, "PV123", "SP123", "OP123", "PV124", "SP124", "OP124",
            "PV125", "SP125");
AssVarTags(WinNumber(), 8, "OP125", "PV126", "SP126", "OP126");
PageDisplay("!MyGenie"); // Display the Super Genie
```

## See Also

[Super Genie Functions](#)

## AssWin

Associates up to eight variable tags or equipment.item tag references with a Super Genie, and displays the Super Genie in a new window. This function has the same effect as calling the [Ass\(\)](#) or [AssTag\(\)](#) function eight times, and then calling the [WinNewAt\(\)](#) function. The [AssWin\(\)](#) function is a quick way of associating eight Super Genie variables and creating a new window - at the same time.

If you want to associate more than eight tags with the Super Genie you need to call the [AssVarTags\(\)](#), [AssTag\(\)](#), or [Ass\(\)](#) function to create the associations before you call this function.

## Syntax

**AssWin(*sPage*, *X*, *Y*, *Mode*, *sTag1*, [*sTag2..8*])**

*sPage:*

The page name of the Super Genie. If you prefixed your Super Genie page name with an exclamation mark (!), remember to include it here.

X

The x pixel coordinate of the top left corner of the window.

Y

The y pixel coordinate of the top left corner of the window.

*Mode:*

The mode of the window:

0 - Normal page.

1 - Page child window. The window is closed when a new page is displayed, for example, when the PageDisplay() or PageGoto() function is called. The parent is the current active window.

2 - Window child window. The window is closed automatically when the parent window is freed with the WinFree() function. The parent is the current active window.

4 - No re-size. The window is displayed with thin borders and no maximize/minimize icons. The window cannot be re-sized.

8 - No icons. The window is displayed with thin borders and no maximize/minimize or system menu icons. The window cannot be re-sized.

16 - No caption. The window is displayed with thin borders, no caption, and no maximize/minimize or system menu icons. The window cannot be re-sized.

32 - Echo enabled. When enabled, keyboard echo, prompts, and error messages are displayed on the parent window. This mode should only be used with child windows (for example, Mode 1 and 2).

64 - Always on top.

128 - Open a unique window. This mode stops this window from being opened more than once.

256 - Display the entire window. This mode commands that no parts of the window will appear off the screen

512 - Open a unique Super Genie. This mode stops a Super Genie from being opened more than once (at the same time). However, the same Super Genie with different associations can be opened.

1024 - Disables dynamic resizing of the new window, overriding the setting of the [\[Page\]DynamicSizing](#) parameter.

You can select multiple modes by adding modes together (for example, set Mode to 9 to open a page child window without maximize, minimize, or system menu icons).

*sTag1..8:*

The first eight physical tags to be associated with the Super Genie. For any given Super Genie, the variable tags will replace the Super Genie substitution strings as follows:

Variable tag...	replaces substitution string...
sTag1	1
sTag2	2
sTag3	3
sTag4	4

sTag5	5
sTag6	6
sTag7	7
sTag8	8

Because there is a strict correlation between the variable tag numbers and the substitution string numbers, you need to know how your Super Genie substitutions are numbered. For example, if your Super Genie has three unique substitution strings, numbered 1, 3, & 4, you need to enter a blank ("") for sTag2.

The variable tags that you specify here need to be the same data type as that specified by the relevant Super Genie substitution strings. For example, only a digital tag could replace the substitution string ?DIGITAL 4?. If the substitution string does not specify a type (for example, ?5?), you can use any type except STRING.

The name of the tag can be prefixed by the name of the cluster for example, "*ClusterName.Tag*".

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[Ass](#), [AssChain](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPage](#), [AssChainPopUp](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssInfoEx](#), [AssPage](#), [AssPopUp](#), [AssScaleStr](#), [AssTag](#), [AssTitle](#), [AssVarTags](#)

## Example

```
// Associate 3 tags with the Super Genie
// then display the new window at (100,200) in mode 9
AssWin("!MyGenie", 100, 200, 1 + 8, "PV123", "OP123", "SP123");
```

## See Also

[Super Genie Functions](#)

## AssWinReplace

This function removes the associations on a specified Super Genie window and applies any pending associations. This may be a useful alternative to calling [PageDisplay](#) or [WinNewAt](#) to update associations, as a full page reload is not required.

You need to call this function once for every Super Genie substitution string in the specified window, otherwise the variable (substitution string) will remain uninitialized and will be displayed as #ASC.

## Syntax

**AssWinReplace(*nTargetWindow*)**

*nTargetWindow*:

The number of the window where associations will be replaced.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Example

```
Ass(-3, "Level", "MILK_LEVEL", 0);
AssWinReplace(2);
```

## Related Functions

[Ass](#), [AssChain](#), [AssMetadata](#), [AssMetadataPage](#), [AssMetadataPopUp](#), [AssMetadataWin](#), [AssChainPage](#), [AssChainPopUp](#), [AssChainWin](#), [AssChainWinFree](#), [AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssInfoEx](#), [AssPage](#), [AssPopUp](#), [AssScaleStr](#), [AssTag](#), [AssTitle](#), [AssVarTags](#), [AssWin](#)

## See Also

[Super Genie Functions](#)

## Table (Array) Functions

Table functions perform mathematical functions on entire tables (or arrays), such as the calculation of minimum, maximum, average, and standard deviation values.

---

**Note:** This function only supports arrays declared in Cicode and not variable tag arrays.

---

Following are functions relating to Tables:

<a href="#">TableLookup</a>	Gets a value from a table.
<a href="#">TableMath</a>	Performs mathematical operations on a table, for example, average, maximum, minimum, etc.
<a href="#">TableShift</a>	Shifts a table, left or right.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## TableLookup

Searches for a value in a table, and returns the position (offset) of the value in the table. Be aware that the first item in a table is offset 0 (zero), the next item is offset 1, etc.

**Note:** This function only supports arrays declared in Cicode and not variable tag arrays.

## Syntax

**TableLookup(*Table*, *Size*, *Value*)**

*Table*:

The table to search. The table needs to be an array of real numbers. Must be a Real type variable.

*Size*:

The maximum number of items in the table.

*Value*:

The value to locate.

## Return Value

The offset to the table value, or -1 if the value does not exist.

## Related Functions

[TableMath](#)

## Example

```
REAL Levels[5]=10,15,50,100,200;  
Variable=TableLookup(Levels,5,50);  
! Sets Variable to 2.  
Variable=TableLookup(Levels,5,45);  
! Sets Variable to -1.
```

## See Also

[Table \(Array\) Functions](#)

## TableMath

Performs mathematical operations on a table of real (floating-point) numbers. This function supports minimum, maximum, average, standard deviation, and total operations on a table of values. Use this function for operating on tables returned from the trend system with the [TrnGetTable\(\)](#) function. You can set the *Mode* to either accept or ignore invalid or gated data returned from [TrnGetTable\(\)](#).

**Note:**

- This function cannot check the length of any arrays passed to it. If the array is shorter than the size argument,

unpredictable results can occur, such as data in memory being overwritten, or a general protection fault.

- This function only supports arrays declared in Cicode and not variable tag arrays.

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

Always confirm that arrays are of appropriate length before passing them to the TableMath function.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## **Syntax**

**TableMath(Table, Size, Command [, Mode] )**

*Table:*

Any table of floating-point numbers. Must be a Real type variable.

*Size:*

The maximum number of items in the table.

*Command:*

The mathematical operation to perform on the table:

0 - Minimum

1 - Maximum

2 - Average

3 - Standard deviation

4 - Total

*Mode:*

The mode of the operation:

0 - Operate on all data - default

1 - Ignore invalid or gated data returned from the TrnGetTable() function

## **Return Value**

Returns the value related to the requested mathematical operation performed on the table (Minimum, Maximum, Average, Standard deviation or Total).\\

## **Related Functions**

[TableLookup](#) [TrnGetTable](#)

## **Example**

```
REAL Array[5]=10,15,50,100,200;  
REAL Min,Avg;  
! Get the minimum value.
```

```
Min=TableMath(Array, 5, 0, 0); ! Sets Min to 10.  
! Get the average value.  
Avg=TableMath(Array, 5, 2, 0); ! Sets Avg to 75.
```

## See Also

[Table \(Array\) Functions](#)

## TableShift

Shifts table items in a table by a number of positions. You can shift the table left or right. Items shifted off the end of the table are lost. Items within a table that are not replaced by other items (that have moved) are set to 0.

---

**Note:** This function only supports arrays declared in Cicode and not variable tag arrays.

---

## Syntax

**TableShift**(*Table*, *Size*, *Count*)

*Table*:

The table to shift, an array of real numbers. Must be a Real type variable.

*Size*:

The maximum number of items in the table.

*Count*:

The number of positions to shift the table items. A negative Count moves items to the right and a positive Count moves items to the left.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TableMath](#), [TableLookup](#)

## Example

```
REAL Levels[5]=10,15,50,100,200;  
TableShift(Levels,5,2);  
/* Shifts the table items by 2 positions to the left, that is  
Levels[0]=50  
Levels[1]=100  
Levels[2]=200  
Levels[3]=0  
Levels[4]=0 */
```

## See Also

[Table \(Array\) Functions](#)

## Tag Functions

The Tag Functions allow you to read the values of variables in I/O devices such as PLCs, and to write data into these I/O device variables. These functions also allow you to control I/O devices and to display information about I/O devices.

Following are functions relating to tags:

<a href="#">SubscriptionAddCallback</a>	Adds a callback function to a tag subscription.
<a href="#">SubscriptionGetAttribute</a>	Reads an attribute value of a tag subscription.
<a href="#">SubscriptionGetInfo</a>	Reads the specified text information about a subscribed tag.
<a href="#">SubscriptionGetQuality</a>	Reads quality of a subscribed tag.
<a href="#">SubscriptionGetTag</a>	Reads a value, quality and timestamps of a subscribed tag.
<a href="#">SubscriptionGetTimestamp</a>	Reads the specified timestamp of a subscribed tag.
<a href="#">SubscriptionGetValue</a>	Reads a value of a subscribed tag.
<a href="#">SubscriptionRemoveCallback</a>	Removes a callback function from a tag subscription.
<a href="#">TagBrowseClose</a>	Terminates an active data browse session and cleans up all resources associated with the session.
<a href="#">TagBrowseFirst</a>	Places the data browse cursor at the first record.
<a href="#">TagBrowseGetField</a>	Retrieves the value of the specified field from the record the data browse cursor is currently referencing.
<a href="#">TagBrowseNext</a>	Moves the data browse cursor forward one record.
<a href="#">TagBrowseNumRecords</a>	Gets the number of records for a given browsing session.
<a href="#">TagBrowseOpen</a>	Initiates a new browse session and returns a handle to the new session that can be used in subsequent data browse function calls.
<a href="#">TagBrowsePrev</a>	Moves the data browse cursor back one record.
<a href="#">TagDebug</a>	Displays a dialog which allows you to select any configured tag to read or change (write) its value.

<a href="#">TagDebugForm</a>	Displays a dialog that allows you to select a variable tag and perform some basic read/write operations.
<a href="#">TagEventFormat</a>	Returns a handle to the format of the data used by the TagEventQueue().
<a href="#">TagEventQueue</a>	Opens the tag update event queue.
<a href="#">TagGetProperty</a>	Gets a property for a variable tag from the datasource.
<a href="#">TagGetScale</a>	Gets the value of a tag at a specified scale from the datasource.
<a href="#">TagGetValue</a>	Gets the value, quality and timestamp of a tag based on the tag subscription.
<a href="#">TagInfo</a>	Gets information about a variable tag.
<a href="#">TagInfoEx</a>	Gets information about a variable tag.
<a href="#">TagRamp</a>	Increments a tag by a percentage amount.
<a href="#">TagRDBReload</a>	Reloads the variable tag database so when TagInfo is called it picks up online changes to the tag database.
<a href="#">TagRead</a>	Reads a variable from an I/O Device.
<a href="#">TagReadEx</a>	Reads the value, quality and timestamp of a particular tag element.
<a href="#">TagResolve</a>	Increments a reference count on a tag to keep it resolved.
<a href="#">TagScaleStr</a>	Gets the value of a tag at a specified scale.
<a href="#">TagSetOverrideBad</a>	Sets a quality Override element for a specified tag to Bad Non Specific.
<a href="#">TagSetOverrideGood</a>	Sets a quality Override element for a specified tag to Good Non Specific.
<a href="#">TagSetOverrideUncertain</a>	Sets a quality Override element for a specified tag to Uncertain Non Specific.
<a href="#">TagSetOverrideQuality</a>	Sets a quality Override element for a specified tag.
<a href="#">TagSubscribe</a>	Subscribes a tag for periodic monitoring and event handling.
<a href="#">TagUnsubscribe</a>	Unsubscribes a tag for periodic monitoring and event handling.

<a href="#">TagUnresolve</a>	Unresolves a reference count implemented on a tag by TagResolve().
<a href="#">TagWrite</a>	Writes to an I/O Device variable.
<a href="#">TagWriteEventQue</a>	Opens the tag write event queue.
<a href="#">TagWriteIntArray</a>	Writes an array of integers to a tag.
<a href="#">TagWriteRealArray</a>	Writes an array of reals to a tag.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## SubscriptionAddCallback

Adds a function callback to a tag subscription. When the value change for a subscribed tag is detected, a callback function can be called. This implements change based Cicode and avoids continuously polling a tag value to monitor changes.

Multiple callbacks are possible to the same subscription.

To remove a callback from a subscription use the [SubscriptionRemoveCallback](#) function.

## Syntax

**SubscriptionAddCallback(*iHandle*, *sCallback*)**

*iHandle*

Integer handle of the subscription to add a callback to.

*sCallback*

String stating the name of a function to call when the value is updated. The function should have the structure:

```
FUNCTION evtHandler(INT subsHandle)
```

..

End

Where *subsHandle* is the subscription that raised the event.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TagSubscribe](#), [TagUnsubscribe](#), [SubscriptionGetAttribute](#), [SubscriptionRemoveCallback](#)

## See Also

[Tag Functions](#)

### SubscriptionGetAttribute

Reads the specified attribute value of a subscribed tag. Similar to TagRead.

**Note:** This function has been superseded and may be deprecated in a future release. Please use the following functions: [SubscriptionGetInfo](#), [SubscriptionGetTimestamp](#), [SubscriptionGetQuality](#), [SubscriptionGetValue](#) or [SubscriptionGetTag](#) as substitutions in new projects.

## Syntax

**SubscriptionGetAttribute**(*iHandle*, *sAttribute* [, *iOffset*] )

*iHandle*

Integer handle of the subscription to read from.

*sAttribute*

Attribute of the tag to read. Supported Attributes are:

**ClusterName** - Returns the resultant cluster context of the subscription. For example, for the tag subscribed as "cluster1.tagname", the return value is "cluster1".

There are several possible outcomes where no cluster is specified in the subscription:

- If the tag is a local tag, an empty string will be returned.
- If the tag is a variable tag and the system only contains one cluster, this cluster will be returned.
- If the tag is a variable tag and there is a default cluster being run in the Cicode, the default cluster will be returned.
- If none of these options are true, an empty string will be returned.

**FullName** - Return the full subscription name. For example, for the tag subscribed as "cluster1.tagname", return value is "cluster1.tagname". If the tag was subscribed without a cluster the return value will be the tagname.

**TagName** - Return the tagname for the subscription. For example, for the tag subscribed as "cluster1.tagname", return value is "tagname".

**Value** - The current value of the tag.

**ValueQuality** - An indication of the current quality of the tag as an integer number.

**Note:** The return value is not compatible with the QUALITY data type or the quality Cicode functions. Use [SubscriptionGetQuality](#).

**ValueTimestamp** - The time when the tag last changed. It is returned as an integer value compatible with a time/data variable.

**Note:** The return value is not compatible with the TIMESTAMP data type. Use [SubscriptionGetTimestamp](#).

**ValueTimestampMS** - The millisecond part of the time when the tag last changed.

*iOffset*

Optional integer expressing the zero based index of an array attribute. This is only valid for the Value attribute. Default value is 0.

## Return Value

String representation of the cached value for a subscribed tag. On error, an empty string and an error is set.

## Related Functions

[SubscriptionGetInfo](#), [SubscriptionGetTimestamp](#), [SubscriptionGetQuality](#), [SubscriptionGetValue](#),  
[SubscriptionGetTag](#)

## See Also

[Tag Functions](#)

### SubscriptionGetInfo

Reads the specified text information about a subscribed tag.

## Syntax

**SubscriptionGetInfo**(*iHandle*, *sAttribute* )

*iHandle*

Integer handle of the subscription to read from.

*sAttribute*

Attribute of the tag to read. Supported Attributes are:

**ClusterName** - Return the cluster context of the subscription. For example, for the tag subscribed as "cluster1.tagname", return value is "cluster1". If the tag was subscribed without a cluster the return value will be an empty string.

**FullName** - Return the full subscription name. For example, for the tag subscribed as "cluster1.tagname.field", return value is "cluster1.tagname.field". If the tag was subscribed without a cluster, the return value will be the tag name and/or element name. If the tag was subscribed without an element, the return value will be the tag name and/or cluster name.

**TagName** - Return the tagname for the subscription. For example, for the tag subscribed as "cluster1.tagname", return value is "tagname".

**ElementName** - Retrieve the element name of the subscription.

## Return Value

String representation of the requested information for a subscribed tag. On error, returns an empty string and an error is set.

## Related Functions

[SubscriptionGetTimestamp](#), [SubscriptionGetQuality](#), [SubscriptionGetValue](#), [SubscriptionGetTag](#)

## Example

```
STRING TagName = SubscriptionGetInfo(hSub, "TagName");
```

## See Also

[Tag Functions](#)

### SubscriptionGetQuality

Reads quality of a subscribed tag.

## Syntax

**SubscriptionGetQuality(*iHandle*)**

*iHandle*

Integer handle of the subscription to read from.

## Return Value

The quality for a subscribed tag. On error, QUAL\_BAD.

## Related Functions

[SubscriptionGetInfo](#), [SubscriptionGetTimestamp](#), [SubscriptionGetValue](#), [SubscriptionGetTag](#)

## Example

```
QUALITY theQuality = SubscriptionGetQuality(hSub);
```

## See Also

[Tag Functions](#)

### SubscriptionGetTag

Reads a value, quality and timestamps of a subscribed tag.

## Syntax

**SubscriptionGetTag(*iHandle*, *sOffset*)**

*iHandle*

Integer handle of the subscription to read from.

*sOffset*

Optional integer expressing the zero based index of an array attribute. This is only valid for the Value attribute. Default value is 0.

## Return Value

Returns a value, quality and timestamps of a subscribed tag. The type of the returned value depends on a type of the subscribed tag. The quality and timestamps of the subscribed tag are read and passed with the returned value.

Using [SubscriptionGetValue](#) gives similar results as using direct reference to a tag without item ex. tag1, tag1.Field.

On error, returns either 0 for numerical data types or an empty string for strings.

## Related Functions

[SubscriptionGetTimestamp](#), [SubscriptionGetQuality](#), [SubscriptionGetInfo](#), [SubscriptionGetValue](#)

## Example

```
INT Value = SubscriptionGetTag(hSub);
```

## See Also

[Tag Functions](#)

## SubscriptionGetTimestamp

Reads the specified timestamp of a subscribed tag.

## Syntax

**SubscriptionGetTimestamp(*iHandle*, *sAttribute* )**

*iHandle*

Integer handle of the subscription to read from.

*sAttribute*

Attribute of the tag to read. Supported Attributes are:

**Timestamp** - The timestamp when the tag last changed. It is default value used when this argument is not specified.

**QualityTimestamp** - The timestamp when quality of the tag last changed.

**ValueTimestamp** - The timestamp when value of the tag last changed.

## Return Value

The requested timestamp for a subscribed tag. On error, 0 (INVALID TIMESTAMP).

## Related Functions

[SubscriptionGetInfo](#), [SubscriptionGetQuality](#), [SubscriptionGetValue](#), [SubscriptionGetTag](#)

## Example

```
TIMESTAMP theTime = SubscriptionGetTimestamp(hSub, "ValueTimestamp");  
TIMESTAMP theTime = SubscriptionGetTimestamp(hSub);
```

## See Also

[Tag Functions](#)

## SubscriptionGetValue

Reads a value of a subscribed tag.

## Syntax

**SubscriptionGetValue(*iHandle*, *sOffset*)**

*iHandle*

Integer handle of the subscription to read from.

*sOffset*

Optional integer expressing the zero based index of an array attribute. Default value is 0.

## Return Value

Returns a value of a subscribed tag. The type of the returned variable depends on a type of the subscribed tag. The quality and timestamps of the subscribed tag are not read i.e. quality of the returned value can be consider as GOOD and its timestamps as 0 (INVALID TIMESTAMP).

Using [SubscriptionGetValue](#) gives similar results as using direct reference to tag's v item ex. tag1.v, tag1.Field.v.

On error, returns either 0 for numerical data types or an empty string for strings.

## Related Functions

[SubscriptionGetTimestamp](#), [SubscriptionGetQuality](#), [SubscriptionGetInfo](#), [SubscriptionGetTag](#)

## Example

```
INT Value = SubscriptionGetValue(hSub);
```

## See Also

[Tag Functions](#)

## SubscriptionRemoveCallback

Removes a function callback from a tag subscription. The subscription handle and callback function needs to match those used when adding the callback.

## Syntax

**SubscriptionRemoveCallback(*iHandle*, *sCallback*)**

*iHandle*

Integer handle of the subscription of the callback.

*sCallback*

String stating the name of the callback function.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TagSubscribe](#), [TagUnsubscribe](#), [SubscriptionAddCallback](#), [SubscriptionGetAttribute](#)

## See Also

[Tag Functions](#)

## TagBrowseClose

The TagBrowseClose function terminates an active data browse session and cleans up resources associated with the session.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT TagBrowseClose(LONG *Session*)**

*Session*:

The handle to a browse session previously returned by a [TagBrowseOpen](#) call.

## Return Value

0 if successful -1 if unsuccessful.

## Related Functions

[TagInfoEx](#), [TagInfo](#), [TagBrowseFirst](#), [TagBrowsePrev](#), [TagBrowseNumRecords](#), [TagBrowseOpen](#),  
[TagBrowseNext](#),[TagBrowseGetField](#)

## Example

```
// close
TBResult = TagBrowseClose(TBHandle);
End
```

## See Also

[Tag Functions](#)

## TagBrowseFirst

The TagBrowseFirst function places the data browse cursor at the first record.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **TagBrowseFirst**(LONG *Session*)

*Session*:

The handle to a browse session previously returned by a [TagBrowseOpen](#) call.

## Return Value

0 if successful -1 if unsuccessful.

## Related Functions

[TagInfoEx](#), [TagInfo](#), [TagBrowseClose](#), [TagBrowsePrev](#), [TagBrowseNumRecords](#), [TagBrowseOpen](#),  
[TagBrowseNext](#),[TagBrowseGetField](#)

## Example

```
// first
count = 1;
TBResult = TagBrowseFirst(TBHandle);
ErrLog("First: " + IntToStr(TBResult) + ", Error = " + IntToStr(IsError()));
WHILE (TBResult <> -1) DO
ErrLog("Entry " + IntToStr(count) + ": " +
"Tag: " + TagBrowseGetField(TBHandle , "TAG") + ", " +
"Type: " + TagBrowseGetField(TBHandle , "TYPE") + ", " +
"Addr: " + TagBrowseGetField(TBHandle , "ADDR") + ", " +
"Error = " + IntToStr(IsError()));
```

```
TBResult = TagBrowseNext(TBHandle);
ErrLog("Next: " + IntToStr(TBResult) +
", Error = " + IntToStr(IsError()));
count = count + 1;
END
```

## See Also

[Tag Functions](#)

### TagBrowseGetField

The TagBrowseGetField function retrieves the value of the specified field from the record the data browse cursor is currently referencing.

## Syntax

```
STRING TagBrowseGetField(LONG Session, STRING FieldName)
```

*Session:*

The handle to a browse session previously returned by a TagBrowseOpen call.

*Fieldname:*

The name of the field that references the value to be returned. Supported fields are:

ADDR, ARR\_SIZE, CLUSTER, COMMENT, DEADBAND, ENG\_FULL, ENG\_UNITS, ENG\_ZERO, EQUIPMENT, FORMAT, IODEV, PSI\_TYPE, RAW\_FULL, RAW\_ZERO, SCALED\_TYPE, TAG, ITEM, TYPE

For details on these fields refer to the [Browse Function Field Reference](#).

## Return Value

The value of the specified field as a string if successful. An empty string may or may not be an indication that an error has been detected. The last error should be checked in this instance to determine if an error has actually occurred. -1 if unsuccessful.

## Related Functions

[TagInfoEx](#), [TagInfo](#), [TagBrowseClose](#), [TagBrowseFirst](#), [TagBrowseNumRecords](#), [TagBrowseOpen](#), [TagBrowseNext](#)

## Example

```
// GetField
ErrLog("CLUSTER: " + TagBrowseGetField(TBHandle , "CLUSTER") +
", Error = " + IntToStr(IsError()));
ErrLog("IODEV: " + TagBrowseGetField(TBHandle , "IODEV") +
", Error = " + IntToStr(IsError()));
ErrLog("TAG: " + TagBrowseGetField(TBHandle , "TAG") +
", Error = " + IntToStr(IsError()));
END
```

## See Also

[Tag Functions](#)

### TagBrowseNext

The TagBrowseNext function moves the data browse cursor forward one record.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **TagBrowseNext**(LONG *Session*)

*Session*:

The handle to a browse session previously returned by a [TagBrowseOpen](#) call.

## Return Value

0 if successful -1 if unsuccessful.

## Related Functions

[TagInfoEx](#), [TagInfo](#), [TagBrowseClose](#), [TagBrowseFirst](#), [TagBrowsePrev](#), [TagBrowseNumRecords](#),  
[TagBrowseOpen](#), [TagBrowseGetField](#)

## Example

```
// next without first
TBResult = TagBrowseNext(TBHandle);
ErrLog("Next: " + IntToStr(TBResult) + ", Error = " + IntToStr(IsError()));
END
```

## See Also

[Tag Functions](#)

### TagBrowseNumRecords

The TagBrowseNumRecords function gets the number of records for a given browsing session. This function may not be accurate if the following occurs:

- Project misconfiguration on two separate machines
- Online changes happening on two separate machines.

The count will be a consolidated sum of all browse records per I/O Device as returned by each I/O Server in response to opening a browse session. The count reconciliation is performed on the basis of the "best device

"status", that is records with the "best status" are picked and the others are discarded. "Best device status" is calculated as follows:

- Data Source online state (ONLINE is picked over OFFLINE), then
- Data Source active state (ACTIVE is picked over INACTIVE), then
- Data Source priority (higher priority is picked over a lower one, with highest being 1 – for primary I/O device).

## Syntax

**INT TagBrowseNumRecords(LONG Session)**

*Session:*

The handle to a browse session previously returned by a [TagBrowseOpen](#) call.

## Return Value

Number of records (INT) if successful -1 if unsuccessful.

## Related Functions

[TagInfoEx](#), [TagInfo](#), [TagBrowseClose](#), [TagBrowseFirst](#), [TagBrowsePrev](#), [TagBrowseOpen](#), [TagBrowseNext](#), [TagBrowseGetField](#)

## Example

```
// number of records
TBResult = TagBrowseNumRecords(TBHandle);
ErrLog("Num Records: " + IntToStr(TBResult));
END
```

## See Also

[Tag Functions](#)

## TagBrowseOpen

The TagBrowseOpen function initiates a new browse session and returns a handle to the new session that can be used in subsequent data browse function calls. Tag records are sorted on the server and, arrive at the client in the order specified by the sorting fields parameter, or default sorting order as described below under the Sort parameter.

Tag browsing provides a snapshot of tag configuration data. It is not intended to support live data updates.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

**Note:** In order to conserve memory the following recommendations should be observed, as multiple open tag browsing sessions will use a large amount of memory:

1. Close an opened session at the completion of the browsing session using [TagBrowseClose](#).

---

2. List only the fields that you wish to browse, not all fields.

---

## Syntax

**LONG TagBrowseOpen(STRING Filter, STRING Fields, STRING Sort[, STRING Clusters])**

*Filter:*

Semicolon delimited list of predefined field name filters specifying the records to return during the browsing session. If you do not specify a filter string then all tag records will be displayed.

All string fields can be filtered based on regular expressions. Using an operator other than = will cause strings to not match the filter criteria. The following regular expressions are supported \*expr, expr\*, and \*expr\*.

If any of the filter names are invalid, opening a browsing session will not succeed and will return an invalid handle.

*Field:*

Comma separated list of record fields to be returned during the browsing session. An empty field string will return all possible fields.

Supported fields are:

### Configuration Fields

ADDR, ARR\_SIZE, CLUSTER, COMMENT, CUSTOM1, CUSTOM2, CUSTOM3, CUSTOM4, CUSTOM5, CUSTOM6, CUSTOM7, CUSTOM8, DEADBAND, ENG\_FULL, ENG\_UNITS, ENG\_ZERO, EQUIPMENT, FULLNAME, HISTORIAN, ITEM, RAW\_FULL, RAW\_ZERO, SCALED\_TYPE, TAG, TYPE, WRITE\_ROLES

For full details on these fields refer to the [Browse Function Field Reference](#).

The fields LOG\_UNIT and NET\_UNIT are available when using the Cicode functions and [TagInfoEx](#) are also supported.

### Runtime Fields

Field Value	Description (predefined values)	Possible values
OVR_MODE	Override Mode	Integer (as per override mode element specification).
CTRL_MODE	Control Mode (0-1)	Integer (as per control mode element specification).
NUM_SUBS	Number of current subscriptions to the tag	Integer.
VALUE	Default value	As per value type.
FIELD	Field value	As per value type.
OVERRIDE	Override value	As per value type.
VALID	Last Valid value	As per value type.
STATUS	Last Status value	Integer (as per status element specification).

**Sort:**

Comma delimited list of record fields to be returned in order of sorting preferences (each with an indication of whether the sort is A - ascending or D - descending). For example "TYPE:D". By default tag browsing records are sorted by the tag primary key Cluster.TagName. which is implicitly appended to the list of sorting fields, since all the other fields may end up not being unique.

**Clusters:**

An optional parameter that specifies via a comma delimited string the subset of the clusters to browse. An empty string indicates that all clusters will be browsed.

## Return Value

Session handle (LONG) if successful, -1 if unsuccessful.

## Related Functions

[TagInfoEx](#), [TagInfo](#), [TagBrowseClose](#), [TagBrowseFirst](#), [TagBrowsePrev](#), [TagBrowseNumRecords](#),  
[TagBrowseNext](#), [TagBrowseGetField](#)

## Example 1

```
// open
TBResult = TagBrowseOpen("ADDR=*2; TYPE<=2", "TAG,TYPE,ADDR", "ADDR:D", "");
ErrLog("Open Session ID: " + IntToStr(TBResult) +
", Error = " + IntToStr(IsError()));
TBHandle = TBResult;
END
```

## Example 2 - Filters

The following variable tags have been defined - Tag01, Tag02 and Tag03.

```
// Example Filter
FUNCTION TagBrowseTest1()
    STRING sFilter = "TAG=Tag01|Tag03";
    STRING sField = "TAG";
    STRING sTag = "";
    INT iStatus = -1;
    INT hTagBrowse = TagBrowseOpen(sFilter,sField,sField+":A");
    IF (hTagBrowse <> -1) THEN
        iStatus = TagBrowseFirst(hTagBrowse);
        WHILE iStatus = 0 DO
            sTag = TagBrowseGetField(hTagBrowse,sField);
            Message(sTag,sTag,64); // Tag01, Tag03
            iStatus = TagBrowseNext(hTagBrowse);
        END
        TagBrowseClose(hTagBrowse);
    END
END
```

## Example 3 - Filters

```
// Example Filter
FUNCTION TagBrowseTest2()
    STRING sFilter = "TAG=Tag01 OR TAG=Tag03";
    STRING sField = "TAG";
    STRING sTag = "";
    INT iStatus = -1;
    INT hTagBrowse = TagBrowseOpen(sFilter,sField,sField+":A");
    IF (hTagBrowse <> -1) THEN
        iStatus = TagBrowseFirst(hTagBrowse);
        WHILE iStatus = 0 DO
            sTag = TagBrowseGetField(hTagBrowse,sField);
            Message(sTag,sTag,64); // Tag01, Tag03
            iStatus = TagBrowseNext(hTagBrowse);
        END
        TagBrowseClose(hTagBrowse);
    END
END
```

### Results

Good: STRING sFilter = "TAG=Tag01|Tag03";  
Good: STRING sFilter = "TAG=Tag01 OR TAG=Tag03";  
Fail: STRING sFilter = "TAG=Tag01 OR Tag03";

## See Also

[Tag Functions](#)

### TagBrowsePrev

The TagBrowsePrev function moves the data browse cursor back one record. If you call this function after you have reached the beginning of a browse, error code 412 is returned (Databrowse session EOF).

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

INT **TagBrowsePrev**(LONG *Session*)

*Session*:

The handle to a browse session previously returned by a TagBrowseOpen call.

## Return Value

0 if successful -1 if unsuccessful.

## Related Functions

[TagInfoEx](#), [TagInfo](#), [TagBrowseClose](#), [TagBrowseFirst](#), [TagBrowseNumRecords](#),

[TagBrowseOpen](#), [TagBrowseNext](#), [TagBrowseGetField](#)

## Example

```
// previous without first
TBRResult = TagBrowsePrev(TBHandle);
ErrLog("Prev: " + IntToStr(TBRResult) + ", Error = " + IntToStr(IsError()));
END
```

## See Also

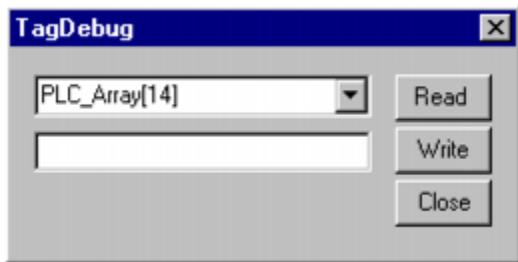
[Tag Functions](#)

## TagDebug

Displays a dialog which allows you to select from a list of the configured variable tags in your system. Once you have selected a tag, you can either press the **Read** button to get the tag's current value; or change the value by entering a new one, and pressing the **Write** button. This function should only be used for debugging or commissioning.

**Note:** An expanded version of the TagDebug dialog is available via the Cicode function [TagDebugForm](#).

To read or change (write) the value of an element in an array variable, type it into the dialog's variable tag field as follows:



## Syntax

`TagDebug()`

## Return Value

The name of the tag entered in the dialog.

## Related Functions

[TagDebugForm](#), [TagInfo](#), [TagReadEx](#), [TagWrite](#)

## Example

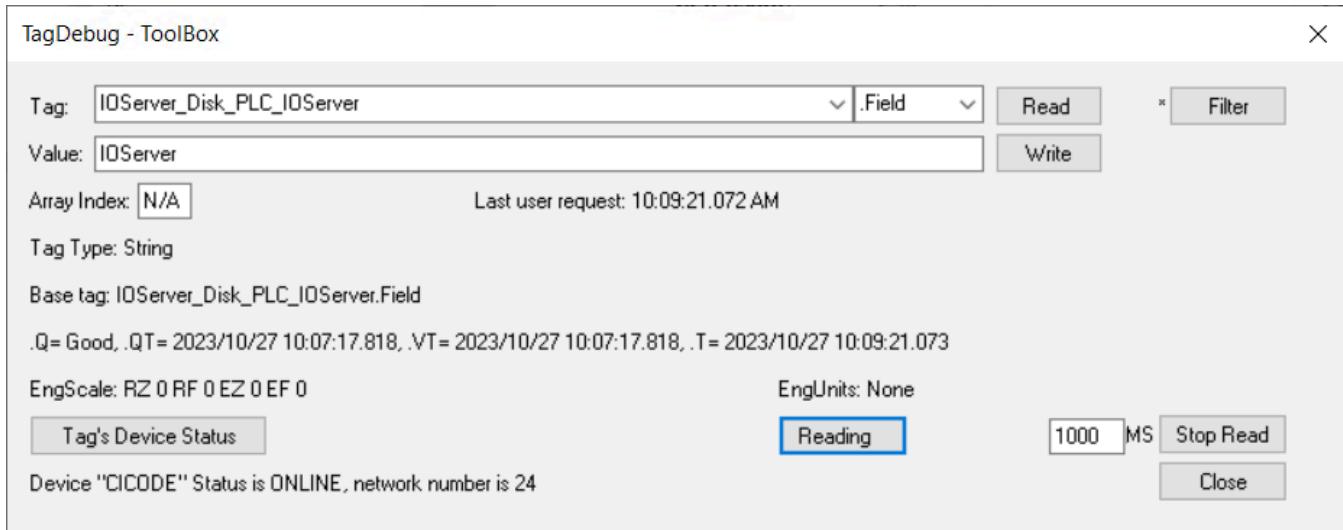
```
TagDebug(); /* Display debug form to allow user to debug */
```

## See Also

[Tag Functions](#)

### TagDebugForm

The TagDebugForm Cicode function displays a dialog that allows you to select a variable tag and perform some basic read/write operations on it. This function is useful for debugging or commissioning a project.



When the TagDebug form appears, you can perform the following tasks:

- **Select a tag**

You can manually enter the name of the tag you would like to debug, or you can select it from the list included in the drop-down menu to the right of the **Tag** field. Be aware that the drop-down list is limited to 500 tags.

**Note:** Local variables will not be included in the **Tag** field drop-down list.

You can use the **Filter** button to help locate specific items from the tags database. A filter allows you to reduce the list of tags included in the tag drop-down menu to only those that match the characters specified in the Tag Filter dialog. To apply a filter, click on the Filter button and enter part of the name of the tag you would like to locate. The text entered can be a common suffix, it does not need to be the start of a tag name.

Click OK to apply the filter, or Clear to remove it.

If a filter is currently implemented, an "F" will appear next to the Filter button. An asterisk (\*) indicates no filter is currently being used.

Once you have a tag selected, you can specify additional properties in the field to the right of the Tag field. The drop-down menu includes a list of available extensions (for example, ".Field", ".Valid", ".Override", etc.). To automatically display these additional properties when a read occurs, you should use the drop-down menu in this field instead of manually adding an extension to a tag name.

- **Read a tag**

The **Read** button will retrieve the value for the tag you have specified, or an error code will be displayed (for example, <Error 424: Tag not found>).

If successful, a read will display the following information about the selected tag:

- Last user request: the time the most recent read occurred.
- Tag Type: the data type.
- Base tag: confirmation of the tag name, including the selected extension.
- .Q (quality), .QT (quality timestamp), .VT (value timestamp), and .T (the object timestamp, which is the most recent of the two timestamps).
- Eng Scale: RZ = raw zero; RF = Raw Full Scale; EZ = engineering zero; EF = engineering full scale.
- Eng Units.

- **Read from a tag array**

You view information for a particular item within a tag array using the **Array Index** field. This field allows you to specify the index number of the required item within the selected tag array.

- **Write to a tag**

To write a value to the selected tag, key in the required value and click on the **Write** button.

When a write has occurred, a log entry similar to the following will be added to the syslog.dat file:

```
<timestamp> TagDebug() Write: User 'Engineer' set tag 'Entry_ScrapWeight' to value ' 23.00' - NO ERROR
```

- **Check the device status**

The **Tag's Device Status** button allows you to confirm if the host device for the selected tag is currently online. In most cases, this will be useful in determining the cause of an error code, for example, <Error 424: Tag not found>. Clicking the button will update the current status of the device.

- **Use the auto-read feature**

The form allows for reads to be automatically repeated at a specified interval, allowing you observe how a tag value is changing over time.

Clicking the **Auto Read** button will start the read cycle; the **Stop Read** button will stop it. To adjust the read period, enter a value (milliseconds) in the field next to the Auto Read button.

## Syntax

`TagDebugForm()`

## Return Value

The name of the tag entered in the dialog.

## Related Functions

[TagDebug](#), [TagInfo](#), [TagReadEx](#), [TagWrite](#)

## Example

```
TagDebug(); /* Display debug form to allow user to debug */
```

## See Also

[Tag Functions](#)

## TagEventFormat

Returns a handle to the format of the data used by the [TagEventQueue\(\)](#).

## Syntax

**TagEventFormat()**

Parameters - None

## Return Value

The format handle.

## Related Functions

[QuePeek](#), [QueRead](#), [TagEventQueue](#)

## See Also

[Tag Functions](#)

## TagEventQueue

Opens the tag update event queue. The I/O server writes events into this queue as they are processed. These events include tag updates from drivers that support time-stamped data.

To read events from this queue, use the [QueRead\(\)](#) or [QuePeek\(\)](#) functions. The data put into the queue contains the following fields:

- Driver (the driver used to communicate with the I/O device)
- Port (the name of the port to which the I/O device is connected)
- Unit (the name of the I/O device)
- Tag (the variable tag name)
- Seconds (a UTC timestamp representing the number of seconds since 1970)
- Milliseconds (number of milliseconds since the last second)
- Value (the tag value)
- Quality (OPC component only equivalent to QualityGetPart, mode 7)

To use this function, you need to enable the tag update event queue with the [\[IOServer\]EnableEventQueue](#) parameter. This parameter will tell the I/O Server to start placing events into the queue. The function [TagEventFormat\(\)](#) returns a handle to the format of the data placed into the string field.

Enabling this formatting feature can increase CPU loading and reduce performance of the I/O Server as every tag update event is formatted and placed in the queue. You should reconsider using this feature if a decrease in performance is noticeable.

The maximum length of each queue is controlled by the [\[Code\]Queue](#) parameter. You may need to adjust this

parameter so as not to miss alarm events. When the queue is full, the I/O Server will discard events.

## Syntax

**TagEventQueue()**

Parameters - None

## Return Value

The queue handle of the Tag Update Event queue.

## Related Functions

[QuePeek](#), [QueRead](#), [TagEventFormat](#)

## Example

```
FUNCTION
ReadEvents()
INT status;
INT queue;
INT format;
INT eventId;
STRING event;
INT sec;
INT ms;
TIMESTAMP time;

queue = TagEventQueue();
format = TagEventFormat();

IF (queue <> -1 AND format <> -1) THEN
WHILE (true) DO
status = QueRead(queue, eventId, event, 1);
IF status = 0 THEN
ErrLog("eventId: " + IntToStr(eventId));
StrToFmt(format, event);
ErrLog(" driver: " + FmtGetField(format, "Driver"));
ErrLog(" port: " + FmtGetField(format, "Port"));
ErrLog(" unit: " + FmtGetField(format, "Unit"));
ErrLog(" tag: " + FmtGetField(format, "Tag"));
sec = FmtGetField(format, "Seconds");
ms = FmtGetField(format, "Milliseconds");
time = TimeIntToTimestamp(sec, ms, 1);
ErrLog(" time: " + TimestampToStr(time,14));
ErrLog(" value: " + FmtGetField(format, "Value"));
ErrLog(" quality: " + FmtGetField(format, "Quality"));
END
END
END
END
```

## See Also

[Tag Functions](#)

### TagGetProperty

This function reads a property of a variable tag from the data source. This function replaces [TagInfo](#).

## Syntax

```
STRING TagGetProperty(STRING Name, STRING Property [, INT CachedMode] [, STRING ClusterName] )
```

### *Name:*

The name of the tag or the equipment and item name of a variable tag (using equipment.item notation) from which to get information. The tag can be prefixed by the name of the cluster that is "ClusterName.Tag" or "ClusterName.Equipment.Item".

---

**Note:** If the tag name exceeds the length limit of 254 characters the hardware alarm "Tag name exceed length limit" will be raised.

---

### *Property:*

The property to read. Property names are case sensitive. Supported properties are:

Address - Returns the configured address of the tag (as specified in the variable tags form).

ArraySize - Array size of the associated tag. Returns 1 for non-array types.

ClusterName - Name of the cluster the tag resides on.

DataBitWidth - Number of bits used to store the value

Description - Tag description

EngUnitsHigh - Maximum scaled value

EngUnitsLow - Minimum scaled value

Equipment - Name of the equipment associated with the Tag.

Format - Format bit string. The format information is stored in the integer as follows:

- Bits 0-7 - format width
- Bits 8-15 - number of decimal places
- Bits 16 - zero-padded
- Bit 17- left-justified
- Bit 18 - display engineering units
- Bit 20 - exponential (scientific) notation

FormatDecPlaces - Number of decimal places for default format

FormatWidth - Number of characters used in default format

FullName - Full name of the tag in the form cluster.tagname.

Item - Name of the equipment item associated with the Tag. If the tag is not resolved, returns an empty string.

RangeHigh - Maximum unscaled value

RangeLow - Minimum unscaled value

TagName - Name of the tag specified.

Type - General type of tag. Allowed values are:

- 0 - Digital
- 1 - Byte
- 2 - Integer16
- 3 - UInteger16
- 4 - Long
- 5 - Real
- 6 - String
- 7 - ULONG
- 8 - Undefined

Units - Engineering Units for example, %, mm, Volts.

Custom1 ... Custom8 - User-defined strings.

*CachedMode:*

Optional parameter that specifies from where to retrieve the value for the property.

-1 - The mode is determined by the INI parameter [\[Client\]TagReadCachedMode](#).

0 - The property value is retrieved direct from the server in blocking mode and an error code is returned if it is not on the server (or the server is unavailable). For the first call of TagGetProperty to return a proper value for custom fields, mode 0 needs to be specified.

1 - The property value is retrieved from the cache and an error code is returned if the cache is not loaded yet.

2 - The property value is retrieved from the local configuration and an error code is returned if it is not available (this is the traditional behaviour).

3 - The property value is retrieved from the cache, if the cache is loaded, and from the local configuration, if the cache is not loaded yet.

Default value is -1.

*ClusterName:*

Specifies the name of the cluster in which the Tag resides. The argument is enclosed in quotation marks.

---

**Note:** When retrieving bit width ("DataBitWidth" property) or array size ("ArraySize" property) from the local configuration (iCachedMode 2), the value is related to the data structure in the device. For example, MODNET, is a 16 bit device and does not have native LONG and REAL numbers. So when you have a LONG variable, an array of 2 INTEGERS are needed to store it. In the example, the property "DataBitWidth" against the variable will return 16 and property "ArraySize" will return 2. The combination of these two return values will add up to the correct bits.

## Return Value

String representation of the property of the tag. If unsuccessful, an empty string and an error code is set.

## Related Functions

[AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssScaleStr](#), [TagGetProperty](#), [TagScaleStr](#), [TagInfo](#)

## Example

```
// Get the engineering full scale value for the variable "PV131"  
EngFullScale = TagGetProperty("PV131", "EngUnitsHigh", 0);  
// Get the cached array size for the array variable "PLC_Array"  
ArrayLength = TagGetProperty("PLC_Array", "ArraySize", 1);
```

## See Also

[Tag Functions](#)

### TagGetScale

Gets the value of a tag at a specified scale from the datasource. The value is returned as a formatted string using the tags format specification and (optionally) the engineering units. Use this function to write generic Cicode that will work with any type of tag. This function replaces [TagScaleStr](#).

## Syntax

**TagGetScale(STRING Name, INT Percent, INT EngUnits [, INT CachedMode] [, STRING ClusterName] )**

*Name:*

The name of the tag, or the equipment and item name of a variable tag (using equipment.item notation). The tag can be prefixed by the name of the cluster that is "ClusterName.Tag".

---

**Note:** If the tag name exceeds the length limit of 254 characters the hardware alarm "Tag name exceed length limit" will be raised.

*Percent:*

The percentage of full scale of the returned value.

*EngUnits:*

Flag to determine if the value is returned with engineering units:

0 - Do not return the value with engineering units

1 - Return the value with engineering units

*CachedMode:*

Optional parameter that specifies from where to retrieve the value for the property.

-1 - The mode is determined by the INI parameter [\[Client\]TagReadCachedMode](#).

0 - The property value is retrieved direct from the server in blocking mode and an error code is returned if it is not on the server (or the server is unavailable).

1 - The property value is retrieved from the cache and an error code is returned if the cache is not loaded yet.

2 - The property value is retrieved from the local configuration and an error code is returned if it is not available (this is the traditional behaviour).

3 - The property value is retrieved from the cache, if the cache is loaded, and from the local configuration, if the cache is not loaded yet.

Default value is -1.

*ClusterName:*

Specifies the name of the cluster in which the Tag resides. The argument is enclosed in quotation marks.

## Return Value

The scale of the tag (as a string).

## Related Functions

[AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssScaleStr](#), [TagGetProperty](#), [TagScaleStr](#), [TagInfo](#)

## Example

```
// Display the zero, 50% and full scale of the tag CV_123_PV
DspText(31,0,TagGetScale("CV_123_PV", 0, 1));
DspText(32,0,TagGetScale("CV_123_PV", 50, 1));
DspText(33,0,TagGetScale("CV_123_PV", 100, 1));
```

## See Also

[Tag Functions](#)

## TagGetValue

Reads the value, quality and timestamp of a tag based on the tag subscription.

This function sets up an asynchronous subscription that checks at the specified polling period whether the value of a tag has changed. If the function fails to get a value from the tag initially, subsequent calls at periodic intervals retrieve the tag value. Unused subscriptions initiated by the function automatically expire at the end of the timeout period.

This is a non-blocking function.

## Syntax

**TagGetValue**(*STRING TagName [, INT Item] [, INT Index] [, LONG PollTime] [, INT ScaleMode] [, REAL Deadband] [, INT Lightweight] [, LONG KeepAliveSeconds]*)

### *TagName*

String representing the tag/tag element/ the equipment and item name (using equipment.item notation) associated with that tag, to subscribe to in the form of "cluster\_name.tag\_name.element name" or "cluster\_name.equipment. item.element\_name". If the element name (for example, 'Field') is not specified, it will be resolved at runtime as an unqualified tag reference. . If you need to access a specific tag element item to get timestamp and quality values, use the *nItem* parameter as well (for further information, refer to the [Tag Extensions](#) documentation in the main help).

Cluster name is optional for single-cluster projects.

If the *TagName* represents an array, the entire array is subscribed and will be reused. Note that square brackets for array indexing are unsupported. To retrieve values from individual elements of the array, use the *nIndex* parameter.

---

**Note:** If the tag name exceeds the length limit of 254 characters the hardware alarm "Tag name exceeds length limit" will be raised.

---

*Item*

Optional element number to access tag element values. For more information, refer to the Tag Extensions documentation in the main help.

0: Unqualified tag reference (default) – equivalent to empty element name

1: Value. Equivalent to "v" on Tag Extensions.

2: Value timestamp. Equivalent to "vt" on Tag Extensions.

3: Quality. Equivalent to "q" on Tag Extensions.

4: Quality timestamp. Equivalent to "qt" on Tag Extensions.

5: Timestamp. Equivalent to "t" on Tag Extensions.

*Index:*

Optional index for an array variable. If not specified, it has a default value of 0. Since the *TagName* parameter does not support array indexing, use the *Index* parameter to access an element in an array. Note that the address of the first element in an array is 0 (zero).

*PollTime*

Optional integer representing the Datasource Poll time in milliseconds. Default is 250 milliseconds.

*ScaleMode*

Optional number to specify scaling mode of subscription.

0: Equivalent to "Eng" (default)

1: Equivalent to "Raw"

2: Equivalent to "Percent"

(Deadband, Lightweight, KeepAliveSeconds: Unchanged.)

*Deadband*

Optional real value specifying the percentage of the variable tag's engineering range by which a tag needs to change for an update to be sent through the system. Default is -1.0, indicating the deadband specified by the tag definition is to be used.

*Lightweight*

This optional boolean argument indicates whether or not subscription updates use a "lightweight" version of the tag value that does not include a quality timestamp or a value timestamp.

If not used, this option is set to 1 which means lightweight tag values will be used by default.

To retrieve quality and value timestamps for a tag, you need to set the Lightweight option to 0.

*KeepAliveSeconds*

Optional length of time (in seconds) for which the subscription will remain in memory. The default is 30 seconds. If there are multiple requests to the same tag with different timeout values, only the first timeout value will be taken into account.

---

**Note:** ScaleMode and Deadband are ignored for Digital tags.

---

## Return Value

Returns the value, quality and timestamp of a subscribed tag. The data type of the value returned depends on

the type of the subscribed tag and also the tag extension. The quality and timestamp of the subscribed tag are read and passed with the value returned by the function.

Returns 0 for numerical data types or an empty string for strings if an error occurs. A hardware alarm will be also raised for an error. The tag value may be the last known value (LKV) if the quality of the tag is not good.

When a new subscription is being made and while waiting for its initial value, this function may return 0 or an empty string, which is not regarded as an error.

**Note:** If tag quality is important such that only good quality data values from this function need to be used in your project, you need to monitor the tag quality by calling this function separately with tag extensions. The tag value returned by this function does not reflect the original data quality and it is necessary to specify tag quality item number such as 3 (Quality) on Item argument in order to check the data quality.

## Examples

The following example returns the assigned RealPower item tag value on equipment item "Building.External.Light1".

```
REAL RealPower = TagGetValue("Building.External.Light1.RealPower", 0, 0, 250, 0, -1.0, 1, 30);
```

The following example returns the value of the "BIT\_1" tag.

```
TagGetValue("BIT_1");
```

The following example returns the field timestamp of "BIT\_1" tag.

```
TagGetValue("BIT_1.Field", 5, 0, 250, 0, -1.0, 0, 30);
```

The following example returns the the valid quality timestamp of the StatusEnum item tag value on equipment "Plant.Bottler.Bottle1". Note that lightweight mode is turned off to get quality timestamp values.

```
TagGetValue("Plant.Bottler.Bottle1.StatusEnum.Valid", 4, 0, 250, 0, -1.0, 0, 30);
```

The following example returns the value of the 3rd array element of Plot\_1. Note that array index starts from zero.

```
TagGetValue("Plot_1", 0, 2, 250, 0, -1.0, 0, 30);
```

The following example returns the Quality timestamp of the 1st array element of Plot\_1.

```
TagGetValue("Plot_1", 4, 0, 250, 0, -1.0, 0, 30);
```

## See Also

[Tag Extensions](#)

[TagRead](#)

[Tag Functions](#)

## TagInfo

Gets information about a variable tag. This function allows you to develop generic Cicode and Super Genies.

## Syntax

```
STRING TagInfo(STRING Name, INT Type [, STRING ClusterName] [, INT CachedMode] )
```

**Name:**

The name of the tag or the equipment and item reference of a variable tag (using equipment.item notation) from which to get information. The tag can be prefixed by the name of the cluster that is "ClusterName.Tag" or "ClusterName.Equipment.Item".

---

**Note:** If the tag name exceeds the length limit of 254 characters the hardware alarm "Tag name exceed length limit" will be raised.

To get information on a particular element in an array, enter the array name here, followed by the number of the element as follows:

"PLC\_Array[9]"

The above example tells the function to get information on the tenth element in PLC\_Array (remember, the address of the first element in an array is 0 (zero)).

**Type:**

The type of information to get:

0 - The Tag name from the variables table. This is the same as sName argument. (Returned to be compatible with the AssInfo() function). An empty string is returned if Type is 0.

1 - Engineering units

2 - Raw zero scale

3 - Raw full scale

4 - Engineering zero scale

5 - Engineering full scale

6 - Width of the format

7 - Number of decimal places of format

8 - The Tag format as a long integer. The format information is stored in the integer as follows:

- Bits 0-7 - format width

- Bits 8-15 - number of decimal places

- Bits 16 - zero-padded

- Bit 17- left-justified

- Bit 18 - display engineering units

- Bit 20 - exponential (scientific) notation

9 - Logical Unit Number - I/O device number (for internal use)

10 - Raw Type - Protocol's raw data type number for this tag. Type numbers are:

- 0 - Digital

- 1 - Integer

- 2 - Real

- 3 - BCD

- 4 - Long

- 5 - Long BCD

- 6 - Long Real

- 7 - String

- 8 - Byte

- 10 - Unsigned integer
  - 11 - Bit Width - Tag's size in bits. For example, an INT is 16 bits
  - 12 - Unit Type - Protocol's unit type number for this tag
  - 13 - Unit Address - Tag's address after the protocol DBF's template is applied.
  - 14 - Unit Count - Array size. For example, if the tag's address is I1[50], the unit count is 50.
  - 15 - Record Number - Tag's record number in variable.DBF - 1. That is, the first tag has a record number of 0.
  - 16 - Comment - As defined in the variable tags list.
  - 17 - ClusterName of the tag. If the tag is not resolved, returns an empty string.
  - 18 - Full name (*cluster.tagname*) of the tag. If the tag is not resolved, returns an empty string.
  - 19 - Reserved for internal operation.
  - 20 - Configured Address of the tag. If the tag is not resolved, returns an empty string.
  - 21 - Network Number - I/O device number (as defined by the Number field of the I/O Devices dialog).
  - 22 - Name of the equipment associated with the Tag. If the tag is not resolved, returns an empty string.
- If the tag is a local variable, mode 9 error code 348 is retrieved when cache mode is 0,1,2,3. Modes 12, 13, 14 and 15 will return an empty string (error 274 is trapped).
- 23 - General Type.
    - 0 - Digital
    - 1 - Byte
    - 2 - Integer16
    - 3 - UInteger 16
    - 4 - Long
    - 5 - Real
    - 6 - String
    - 7 - ULong
    - 8 - Undefined
  - 24 - Reserved for internal use.
  - 25 - Name of the equipment item associated with the Tag. If the tag is not resolved, returns an empty string.
  - 26 - Custom 1 - a user-defined string.
  - 27 - Custom 2 - a user-defined string.
  - 28 - Custom 3 - a user-defined string.
  - 29 - Custom 4 - a user-defined string.
  - 30 - Custom 5 - a user-defined string.
  - 31 - Custom 6 - a user-defined string.
  - 32 - Custom 7 - a user-defined string.
  - 33 - Custom 8 - a user-defined string.

#### *ClusterName*

Specifies the name of the cluster in which the Tag resides. The argument is enclosed in quotation marks.

#### *CachedMode:*

Optional parameter that specifies from where to retrieve the value for the property.

- 1 - The mode is determined by the INI parameter [Client]TagReadCachedMode.
- 0 - The property value is retrieved direct from the server in blocking mode and an error code is returned if it is not on the server (or the server is unavailable).
- 1 - The property value is retrieved from the cache and an error code is returned if the cache is not loaded yet.
- 2 - The property value is retrieved from the local configuration and an error code is returned if it is not available (this is the traditional behaviour).
- 3 - The property value is retrieved from the cache, if the cache is loaded, and from the local configuration, if the cache is not loaded yet.

Default value is -1.

---

**Note:** When retrieving bit width (Type 11) or unit count (Type 14) from local configuration (iCachedMode 2), the value is related to the data structure in the device. For example, MODNET, This is a 16 bit device does not have native LONG and REAL numbers. So when you have a LONG variable, then 2 INTEGERS are needed to store it. In the case, Type 11 against the variable will return 16 and Type 14 will return 2. The combination of these two return values will add up to the correct bits.

---

## Return Value

The value of the information as a string.

## Related Functions

[AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssScaleStr](#), [TagGetProperty](#), [TagGetScale](#), [TagInfoEx](#), [TagScaleStr](#)

## Example

```
/* Get the engineering full scale value for the variable "PV131" */  
EngFullScale = TagInfo("PV131", 5);  
/* Get the engineering zero scale value for the array variable "PLC_Array" */  
EngZeroScale = TagInfo("PLC_Array", 4);
```

## See Also

[Tag Functions](#)

## TagInfoEx

This function replaces TagInfo and is identical in operation. It supports online changes. It is recommended therefore that instances of TagInfo in legacy code are migrated to either TagInfoEx or TagGetProperty. New Cicode should use TagGetProperty.

Gets information about a variable tag. This function allows you to develop generic Cicode and Super Genies. Execution can be blocking or non-blocking depending on the iCached argument.

**Note:** When replacing an instance of TagInfo with TagInfoEx in a loop, you may want to make TagInfoEx blocking using the iCached argument so that you are using the correct value for the Tag in your logic. You should also be aware that TagInfo has different return values if you are using mode 10 for nType

## Syntax

STRING **TagInfoEx**(*STRING Name*, *INT Type* [, *STRING ClusterName*] [, *INT CachedMode*] )

*Name*:

The name of the tag or the equipment and item name of a variable tag (using equipment.item notation) from which to get information. The name of the tag can be prefixed by the name of the cluster that is "ClusterName.Tag" or "ClusterName.Equipment.Item".

If the tag name exceeds the length limit of 254 characters the hardware alarm "Tag name exceed length limit" will be raised.

To get information on a particular element in an array, enter the array name here, followed by the number of the element as follows:

"PLC\_Array[9]"

The above example tells the function to get information on the tenth element in PLC\_Array (remember, the address of the first element in an array is 0 (zero)).

*Type*:

The type of information to get:

*0* - The Tag name from the variables table. This is the same as the *Name* argument. (Returned to be compatible with the [AssInfo](#) function.)

*1* - Engineering units

*2* - Raw zero scale

*3* - Raw full scale

*4* - Engineering zero scale

*5* - Engineering full scale

*6* - Width of the format

*7* - Number of decimal places of format

*8* - The Tag format as a long integer. The format information is stored in the integer as follows:

- Bits 0-7 - format width
- Bits 8-15 - number of decimal places
- Bits 16 - zero-padded
- Bit 17- left-justified
- Bit 18 - display engineering units
- Bit 20 - exponential (scientific) notation

*10* - General Type - Protocol's general data type number for this tag. Type numbers are:

- *0* - Digital
- *1* - Byte
- *2* - Integer16
- *3* - UInteger16
- *4* - Long
- *5* - Real
- *6* - String

- 7 - ULONG

- 8 - Undefined

11 - Bit Width - Tag's size in bits. For example, an INT is 16 bits

Types 12 to 15 are only supported when *iCachedMode* equals to 2 or 3.

12 - Unit Type - Protocol's unit type number for this tag

13 - Unit Address - Tag's address after the protocol DBF's template is applied.

14 - Unit Count - Array size. For example, if the tag's address is I1[50], the unit count is 50.

15 - Record Number - Tag's record number in variable.DBF - 1. That is, the first tag has a record number of 0.

16 - Comment - As defined in the variable tags list.

17 - ClusterName of the tag.

18 - Full name (*cluster.tagname*) of the tag.

19 - Reserved for internal operation.

20 - Configured Address of the tag. If the tag is not resolved, returns an empty string.

21 - Network Number - I/O device number (as defined by the Number field of the I/O Devices dialog).

22 - Name of the equipment associated with the Tag. If the tag is not resolved, returns an empty string.

If the tag is a local variable, mode 9 error code 348 is retrieved when cache mode is 0,1,2,3. Modes 12, 13, 14 and 15 will return an empty string (error 274 is trapped).

23 - Reserved for internal use.

24 - Reserved for internal use.

25 - Name of the equipment item associated with the Tag. If the tag is not resolved, returns an empty string.

26 - Custom 1 - a user-defined string.

27 - Custom 2 - a user-defined string.

28 - Custom 3 - a user-defined string.

29 - Custom 4 - a user-defined string.

30 - Custom 5 - a user-defined string.

31 - Custom 6 - a user-defined string.

32 - Custom 7 - a user-defined string.

33 - Custom 8 - a user-defined string.

#### *ClusterName*

Specifies the name of the cluster in which the Tag resides. The argument is enclosed in quotation marks.

#### *CachedMode*:

Optional parameter that specifies from where to retrieve the value for the property.

-1 - The mode is determined by the INI parameter [Client]TagReadCachedMode.

0 - The property value is retrieved direct from the server in blocking mode and an error code is returned if it is not on the server (or the server is unavailable).

1 - The property value is retrieved from the cache and an error code is returned if the cache is not loaded yet.

2 - The property value is retrieved from the local configuration and an error code is returned if it is not available (this is the traditional behavior).

3 - The property value is retrieved from the cache, if the cache is loaded, and from the local configuration, if the

cache is not loaded yet.

Default value is -1.

**Note:** When retrieving bit width (Type 11) or unit count (Type 14) from local configuration (iCachedMode 2), the value is related to the data structure in the device. For example, MODNET, This is a 16 bit device does not have native LONG and REAL numbers. So when you have a LONG variable, then 2 INTEGERS are needed to store it. In the case, Type 11 against the variable will return 16 and Type 14 will return 2. The combination of these two return values will add up to the correct bits.

## Return Value

The value of the information as a string. If unsuccessful, an empty string is returned. The error code can be obtained by calling the IsError Cicode function.

## Related Functions

[AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssScaleStr](#), [TagGetProperty](#), [TagGetScale](#), [TagInfo](#), [TagScaleStr](#)

## Example

```
/* Get the engineering full scale value for the variable "PV131".  
Obtain the value from Cluster1 in blocking mode */  
EngFullScale = TagInfoEx("PV131", 5, "Cluster1", 0);  
/* Get the engineering zero scale value for the array variable  
"PLC_Array" in non-blocking mode*/  
EngZeroScale = TagInfoEx("PLC_Array", 4);
```

## See Also

[Tag Functions](#)

## TagRamp

This function will increment a Tag by the amount defined by *iPercentInc*. It is often used for incrementing a tag while a button is held down.

## Syntax

**TagRamp**(STRING *Tag*, INT *PercentInc*)

*Tag*:

The variable tag name (or alarm property name), as a string. The name of the tag can be prefixed by the name of the cluster that is "ClusterName.Tag".

To read a particular element in an array, you can enter the array name here, followed by an index to the element as follows:

"PLC\_Array[9] "

The above example tells the function to read the 10th element in the array variable PLC\_Array (remember, the

address of the first element in an array is 0 (zero)).

If you enter an array offset using the *nOffset* argument, it will be added to the index value specified here. For example, TagRead("PLC\_Array[9]", 4) will read the 14th element in PLC\_Array (because [9] means the 10th element, and an offset of 4 means 4 elements after the 10th = element 14).

*PercentInc:*

The percentage by which you want to increase the value of the variable. A negative number will decrease the variable. The increment is calculated as a percentage of the full range.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TagInfo](#), [TagReadEx](#), [TagWrite](#)

## Example

Buttons

Text	Ramp Up
Repeat Command	TagRamp("PLC_VAR_1",2);
Comment	Continual increment by 2%

## See Also

[Tag Functions](#)

## TagRDBReload

Works in conjunction with the TagInfo function. Reloads the variable tag database so when TagInfo is called it picks up all online changes to the tag database.

## Syntax

**TagRDBReload()**

## Return Value

Returns 1 if the tag database was successfully reloaded, and 0 if the tag database fails to load.

**Note:** This function will fail and return 0 if the parameter [\[General\]TagDB](#) has been set to 0.

## Related Functions

[TagInfo](#)

## See Also

[Tag Functions](#)

### TagRead

Reads a variable from an I/O device or a local variable. The variable tag needs to be defined in the Variable Tags database. Because the variable tag is specified as a string (not as a tag), you can ignore the data type of the variable.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

TagRead is useful when the variable tag name is a calculation, such as `sAlarmExt+".Paging"`. For simple assignment of variables use the assignment operator. For example:

```
MyString = MyCluster.MyAlarm.MyProperty
```

---

**Note:** If the Tag property has not been initialized the timestamp value displayed for that tag will be equal to the Unix Epoch, January 1st 1970.

---

If you try to read many variables at the same time, the TagRead() function may be slow. The offset index for array variables is checked against the size of the array.

---

**Note:** TagRead can only return the values of elements for those tags having "Good" quality. If the quality of a tag is not "Good", TagRead returns an empty string. In order to obtain values of tags having not "Good" quality one can use their 'v' item. For example, `TagRead("MyBadQualityTag.v")`. However, the value returned by a TagRead call on a tag with not "Good" quality may be obsolete (due to TagRead asynchronous nature). Use the [IsError](#) Cicode function to control the returned error code when using this function.

## Syntax

**TagRead(STRING Tag [, INT nOffset [, STRING ClusterName]])**

*Tag:*

A string that can be one of the following:

- A tag name — for example, "Fire1"
- An equipment item reference — for example, "Motor1.Fire"
- A tag property — for example, "Fire1.v"
- An equipment item property — for example, "Motor1.Fire.v"
- An alarm property — for example, "AlarmFire1.On"
- An alarm equipment item property — for example, "Motor1.AlarmFire.On"

Currently only the "v" tag extension item is supported by TagRead. References to other extension items will generate an error message. If the element name is not specified, it will be resolved at runtime as an unqualified tag reference.

The name of the tag can be prefixed by the name of the cluster, for example, "ClusterName.Tag" or "ClusterName.Equipment.Item".

If the tag name exceeds the length limit of 254 characters the hardware alarm "Tag name exceed length limit" will be raised.

In the example below, assuming the following variable tag, equipment and item are defined in your project :

Tag: PLC\_Array

Equipment: PLC

Item: Array

Address: 30000[10]

You can refer to the variable tag using the following syntax:

"PLC\_Array"

"PLC.Array"

To read a particular element in an array, you can enter the array name here, followed by an index to the element as follows:

"PLC\_Array[9]"

"PLC.Array[9]"

The above example tells the function to read the 10th element in the array variable PLC\_Array (remember, the address of the first element in an array is 0 (zero)).

If you enter an array offset using the *nOffset* argument, it will be added to the index value specified here. For example, TagRead("PLC\_Array[9]", 4) will read the 14th element in PLC\_Array (because [9] means the 10th element, and an offset of 4 means 4 elements after the 10th = element 14). If you want to read the value of <Valid> element in the above example,

TagRead("PLC\_Array.Valid.V[9]", 4)

*Offset*:

The offset for an array variable. This argument is optional - if not specified, it has a default value of 0.

If you enter an array index as part of the *sTag* argument, it will be added to this offset value. For example, TagRead("PLC\_Array[9]", 4) will read the 14th element in PLC\_Array (because [9] means the 10th element, and an offset of 4 means 4 elements after the 10th = element 14).

*ClusterName*:

Specifies the name of the cluster in which the Tag resides. The argument is enclosed in quotation marks.

## Return Value

The function can return:

- The tag element value (depending on the *sTag* parameter, of the I/O device variable) when the tag has "Good" quality.
- The tag element value (depending on the *sTag* parameter, of the I/O device variable) and the error 257 (can be checked by IsError() built-in function) when the value of the tag is out of the predefined range.
- An empty string and an error code when the tag has not "Good" quality".

## Related Functions

[TagReadEx](#), [TagWrite](#), [IODeviceControl](#), [IODeviceInfo](#)

## Example

```
STRING sStringTagValue;
STRING sStringTagValueField;
INT nIntTagValue;
REAL fRealTagValue;
// Tag1 has a STRING data type.
sStringTagValue = TagRead("Tag1");
sStringTagValueField = TagRead("Tag1.Field");
// Tag2 has an INTEGER data type.
nIntTagValue = TagRead("Tag2");
// Tag3 has a REAL data type.
fRealTagValue = TagRead("Tag3");
```

## See Also

[Tag Reference and TagReadEx Behavior in Cicode Expressions](#)

[Tag Functions](#)

## TagReadEx

Reads the value, quality or timestamp of a particular tag from the I/O device. The variable tag needs to be defined in the Variable Tags database. Because the variable tag is specified as a string (not as a tag), you can ignore the data type of the variable.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

TagReadEx should only be used when the variable tag name is a calculation such as sAlarmExt+.Paging". For simple assignment of variables use the assignment operator. For example, MyString = MyCluster.MyAlarm.MyProperty.

If you try to read many variables at the same time, the TagReadEx() function may be slow. The offset index for array variables is checked against the size of the array.

---

**Note:** TagReadEx can only return the values of elements for those tags having "Good" quality. If the quality of a tag is not "Good", TagReadEx may return an obsolete value (due to TagRead asynchronous nature).

---

## Syntax

**TagReadEx(STRING Tag [, INT Offset [, STRING ClusterName]])**

*Tag:*

A string that can be one of the following:

- A tag name — for example, "Fire1"
- An equipment item reference — for example, "Motor1.Fire"
- A tag property — for example, "Fire1.v"
- An equipment item property — for example, "Motor1.Fire.v"
- An alarm property — for example, "AlarmFire1.On"
- An alarm equipment item property — for example, "Motor1.AlarmFire.On"

Currently only the "v" tag extension item is supported by TagRead. References to other extension items will

generate an error message. If the element name is not specified, it will be resolved at runtime as an unqualified tag reference.

The name of the tag can be prefixed by the name of the cluster, for example, "ClusterName.Tag" or "ClusterName.Equipment.Item".

If the tag name exceeds the length limit of 254 characters the hardware alarm "Tag name exceed length limit" will be raised.

In the example below, assuming the following variable tag, equipment and item are defined in your project:

Tag: PLC\_Array

Equipment: PLC

Item: Array

Address: 30000[10]

You can refer to the variable tag using the following syntax:

"PLC\_Array"

"PLC.Array"

To read a particular element in an array, you can enter the array name here, followed by an index to the element as follows:

"PLC\_Array[9]"

"PLC.Array[9]"

The above example tells the function to read the 10th element in the array variable PLC\_Array (remember, the address of the first element in an array is 0 (zero)).

If you enter an array offset using the *nOffset* argument, it will be added to the index value specified here. For example, TagRead("PLC\_Array[9]", 4) will read the 14th element in PLC\_Array (because [9] means the 10th element, and an offset of 4 means 4 elements after the 10th = element 14). If you want to read the value of <Valid> element in the above example,

TagReadEx("PLC\_Array.Valid.V[9]", 4)

*Offset*:

The offset for an array variable. This argument is optional - if not specified, it has a default value of 0.

If you enter an array offset using the *nOffset* argument, it will be added to the index value specified here. For example, TagReadEx("PLC\_Array[9]", 4) will read the 14th element in PLC\_Array (because [9] means the 10th element, and an offset of 4 means 4 elements after the 10th = element 14).

*ClusterName*:

Specifies the name of the cluster in which the Tag resides. The argument is enclosed in quotation marks.

## Return Value

The function can return:

- The tag element value (depending on the *sTag* parameter, of the I/O device variable) when the tag has "Good" quality.
- The tag element value (depending on the *sTag* parameter, of the I/O device variable) and the error 257 (can be checked by *IsError()* built-in function) when the value of the tag is out of the predefined range.
- A value and an error code when the tag does not have "Good" quality.

## Related Functions

[TagRead](#), [TagWrite](#), [IODeviceControl](#), [IODeviceInfo](#)

## Example

```
STRING sStringTagValue;
STRING sStringTagValueField;
INT nIntTagValue;
REAL fRealTagValue;
// Tag1 has a STRING data type.
sStringTagValue = TagReadEx("Tag1");
sStringTagValueField = TagReadEx("Tag1.Field");
// Tag2 has an INTEGER data type.
nIntTagValue = TagReadEx("Tag2");
// Tag3 has a REAL data type.
fRealTagValue = TagReadEx("Tag3");
TIMESTAMP t1 = TagReadEx("Tag1.T");
TIMESTAMP t2 = TagReadEx("Tag1.VT");
TIMESTAMP t3 = TagReadEx("Tag1.Qt");
QUALITY q1 = TagReadEx("Tag1.Q");
```

## See Also

[Tag Reference /TagReadEx\(\) Behavior in Cicode Expressions](#)

[Tag Functions](#)

## TagResolve

This function can be used to increment a reference count on a tag to keep it resolved, making it readily available to a client.

If a tag is held in a resolved state, it will not be evicted from the client even if it is not being used. This means a client does not need to locate the tag's associated I/O server when a read or write is required.

While this allows faster access to a tag, it should be only utilised for priority tags, as a large number of resolved tags will increase the amount of memory used.

You can use the function [TagUnresolve](#) to decrement the reference count.

## Syntax

**TagResolve(STRING TagName)**

*TagName:*

The name of the tag or the equipment and item name (using equipment.item notation) associated with that tag to resolve (in the format "clusterName.tagName" or "clusterName.equipment.item" if you need to specify a cluster).

**Note:** If the tag name exceeds the length limit of 254 characters the hardware alarm "Tag name exceed length limit" will be raised.

## Return Value

The returned integer is a handle to the tag that has been resolved, or -1 if an error occurred.

The specific error code is accessible by calling [IsError\(\)](#).

## Related Functions

[TagUnresolve](#)

## See Also

[Tag Functions](#)

### TagScaleStr

Gets the value of a tag at a specified scale. The value is returned as a formatted string using the tags format specification and (optionally) the engineering units. Use this function to write generic Cicode that will work with any type of tag.

---

**Note:** This function is being deprecated and is replaced by the [TagGetScale](#) function. If the Tag properties are updated TagScaleStr does not get the updated values whereas TagGetScale does.

---

## Syntax

STRING **TagScaleStr**(STRING *Tag*, INT *Percent* , INT *EngUnits* [,STRING *ClusterName*] [,INT *CachedMode*] )

*Tag*:

The name of the tag, or the equipment and item name of a variable tag (using equipment.item notation).

---

**Note:** If the tag name exceeds the length limit of 254 characters the hardware alarm "Tag name exceed length limit" will be raised.

---

*Percent*:

The percentage of full scale of the returned value.

*EngUnits*:

Determines if the value is returned with engineering units:

0 - Return the value without engineering units

1 - Return the value with engineering units

*ClusterName*:

Name of the cluster

*CachedMode*:

Optional parameter that specifies from where to retrieve the value for the property.

-1 - The mode is determined by the INI parameter [\[Client\]TagReadCachedMode](#).

0 - The property value is retrieved direct from the server in blocking mode and an error code is returned if it is not on the server (or the server is unavailable).

1 - The property value is retrieved from the cache and an error code is returned if the cache is not loaded yet.

2 - The property value is retrieved from the local configuration and an error code is returned if it is not available (this is the traditional behaviour).

3 - The property value is retrieved from the cache, if the cache is loaded, and from the local configuration, if the cache is not loaded yet.

Default value is -1.

## Return Value

The scale of the tag (as a string).

## Related Functions

[AssGetProperty](#), [AssGetScale](#), [AssInfo](#), [AssScaleStr](#), [TagGetProperty](#), [TagGetScale](#), [TagInfo](#)

## Example

```
// Display the zero, 50% and full scale of the tag CV_123_PV
DspText(31,0,TagScaleStr("CV_123_PV", 0, 1));
DspText(32,0,TagScaleStr("CV_123_PV", 50, 1));
DspText(33,0,TagScaleStr("CV_123_PV", 100, 1));
```

## See Also

[Tag Functions](#)

## TagSetOverrideBad

Sets a quality Override element for a specified tag to Bad Non Specific.

## Syntax

**TagSetOverrideBad**(*STRING Tag* [,*INT Synch* [, *STRING ClusterName*]])

*Tag*:

The variable tag name or the equipment and item name (using equipment.item notation) associated with that tag as a string. The name of the tag can be prefixed by the name of the cluster that is "ClusterName.Tag" or "ClusterName.Equipment.Item".

If the tag name exceeds the length limit of 254 characters the hardware alarm "Tag name exceed length limit" will be raised.

*Synch*:

An optional boolean argument that specifies whether the command is synchronous (blocking) or asynchronous (non-blocking). By default, this parameter is set to "True", synchronous (blocking), and the function will wait until the write has completed and returned from the server before further code execution. If you specify this parameter the other parameters need to be explicitly specified.

*ClusterName*:

Specifies the name of the cluster in which the Tag resides. The argument is enclosed in quotation marks.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TagSetOverrideQuality](#), [TagSetOverrideGood](#), [TagSetOverrideUncertain](#), [QualityIsOverride](#)

## Example

```
TagSetOverrideBad("Tag1");
```

Setting the value of the OverrideMode element to anything other than 3 will overwrite the quality that has been applied to the Override element using this function. See the example below. Also see the topic [Override Mode](#) in Plant SCADA's Scheduler documentation.

```
//The OverrideMode is switched off
Tag1.OverrideMode = 0;
//The quality of Override element is set to Bad
TagSetOverrideUncertain("Tag1");
//The OverrideMode is set to 1 and
//the Field element is copied to the Override element.
//The quality of the Override element is overwritten.
Tag1.OverrideMode = 1;
//The quality is set again to Bad
TagSetOverrideBad("Tag1");
```

## See Also

[Tag Functions](#)

## TagSetOverrideGood

Sets a quality Override element for a specified tag to Good Non Specific.

## Syntax

**TagSetOverrideGood** (STRING *Tag* [,INT *Synch* [, STRING *ClusterName*]])

*Tag*:

The variable tag name or the equipment and item name (using <equipment>.<item> notation) associated with that tag as a string. The name of the tag can be prefixed by the name of the cluster that is "ClusterName.Tag" or "ClusterName.Equipment.Item".

If the tag name exceeds the length limit of 254 characters the hardware alarm "Tag name exceed length limit" will be raised.

*Synch*:

An optional boolean argument that specifies whether the command is synchronous (blocking) or asynchronous (non-blocking). By default, this parameter is set to "True", synchronous (blocking), and the function will wait until the write has completed and returned from the server before further code execution. If you specify this

parameter the other parameters need to be explicitly specified.

*ClusterName:*

Specifies the name of the cluster in which the Tag resides. The argument is enclosed in quotation marks.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TagSetOverrideQuality](#), [TagSetOverrideBad](#), [TagSetOverrideUncertain](#), [TagSetOverrideBad](#), [QualityIsOverride](#)

## Example

```
TagSetOverrideGood("Tag1");
```

Setting the value of the OverrideMode element to anything other than 3 will overwrite the quality that has been applied to the Override element using this function. See the example below. Also see the topic [Override Mode](#) in Plant SCADA's Scheduler documentation.

```
//The OverrideMode is switched off
Tag1.OverrideMode = 0;
//The quality of Override element is set to Good
TagSetOverrideGood("Tag1");
//The OverrideMode is set to 1 and
//the Field element is copied to the Override element.
//The quality of the Override element is overwritten.
Tag1.OverrideMode = 1;
//The quality is set again to Good
TagSetOverrideGood("Tag1");
```

## See Also

[Tag Functions](#)

## TagSetOverrideQuality

Sets a quality of Override element for a specified tag.

## Syntax

**TagSetOverrideQuality**(STRING *Tag*, QUALITY *qualityNew* [,INT *Synch* [, STRING *ClusterName*]])

*Tag*

The variable tag name or the equipment and item name (using equipment.item notation) associated with that tag as a string. The name of the tag can be prefixed by the name of the cluster that is "ClusterName.Tag" or "ClusterName.Equipment.Item".

If the tag name exceeds the length limit of 254 characters the hardware alarm "Tag name exceed length limit" will be raised.

**qualityNew**

The new quality for the tag's Override element.

**Synch**

An optional boolean argument that specifies whether the command is synchronous (blocking) or asynchronous (non-blocking). By default, this parameter is set to "True", synchronous (blocking), and the function will wait until the write has completed and returned from the server before further code execution. If you specify this parameter the other parameters need to be explicitly specified.

**ClusterName**

Specifies the name of the cluster in which the Tag resides. The argument is enclosed in quotation marks.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TagSetOverrideGood](#), [TagSetOverrideBad](#), [QualityIsOverride](#)

## Example

```
QUALITY q1 = QualityCreate(QUAL_UNCR);
TagSetOverrideQuality("Tag1", q1);
```

Setting the value of the OverrideMode element to anything other than 3 will overwrite the quality that has been applied to the Override element using this function. See the example below. Also see the topic [Override Mode](#) in Plant SCADA's Scheduler documentation.

```
//The OverrideMode is switched off
Tag1.OverrideMode = 0;
//The quality of Override element is set to Uncertain
QUALITY q1 = QualityCreate(QUAL_UNCR);
TagSetOverrideQuality("Tag1", q1);
//The OverrideMode is set to 1 and
//the Field element is copied to the Override element.
//The quality of the Override element is overwritten.
Tag1.OverrideMode = 1;
//The quality is set again to Uncertain
QUALITY q1 = QualityCreate(QUAL_UNCR);
TagSetOverrideQuality("Tag1", q1);
```

## See Also

[Tag Functions](#)

### TagSetOverrideUncertain

Sets a quality Override element for a specified tag to Uncertain Non Specific.

## Syntax

**TagSetOverrideUncertain(STRING Tag [,INT Synch [, STRING ClusterName]])**

*Tag:*

The variable tag name or the equipment and item name (using equipment.item notation) associated with that tag as a string. The name of the tag can be prefixed by the name of the cluster that is "ClusterName.Tag" or "ClusterName.Equipment.Item".

If the tag name exceeds the length limit of 254 characters the hardware alarm "Tag name exceed length limit" will be raised.

*Synch:*

An optional boolean argument that specifies whether the command is synchronous (blocking) or asynchronous (non-blocking). By default, this parameter is set to "True", synchronous (blocking), and the function will wait until the write has completed and returned from the server before further code execution. If you specify this parameter the other parameters need to be explicitly specified.

*ClusterName:*

Specifies the name of the cluster in which the Tag resides. The argument is enclosed in quotation marks.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TagSetOverrideQuality](#), [TagSetOverrideGood](#), [TagSetOverrideBad](#), [QualityIsOverrride](#)

## Example

```
TagSetOverrideUncertain("Tag1");
```

Setting the value of the OverrideMode element to anything other than 3 will overwrite the quality that has been applied to the Override element using this function. See the example below. Also see the topic [Override Mode](#) in Plant SCADA's Scheduler documentation.

```
//The OverrideMode is switched off
Tag1.OverrideMode = 0;
//The quality of Override element is set to Uncertain
TagSetOverrideUncertain("Tag1");
//The OverrideMode is set to 1 and
//the Field element is copied to the Override element.
//The quality of the Override element is overwritten.
Tag1.OverrideMode = 1;
//The quality is set again to Uncertain
TagSetOverrideUncertain("Tag1");
```

## See Also

[Tag Functions](#)

## TagSubscribe

Subscribes a tag so that Cicode functions can be called when a tag's value changes. The subscription checks each poll period whether the tag has changed value and if it has, the specified callback function is called. This avoids continuously polling a tag value to monitor changes. To add a function callback to the subscription, use the optional parameter in this command or the [SubscriptionAddCallback](#) function.

Multiple subscriptions are possible to the same tag. Each new subscription returns a new subscription handle. Multiple callbacks are possible to the same subscription.

You can use this function to subscribe to a variable that is an array by specifying an array index in the *TagName* argument. For example:

```
TagSubscribe("tag[5]", ...)
```

When the array value changes, a runtime callback will pass the indexed tag value. If an invalid index value is passed, a hardware error will be generated.

- If a negative array index value is used, an "Invalid argument" hardware alarm will be raised.
- If an index value exceeds the maximum value of the array offset, an "Array overrun" hardware alarm will be raised.

To unsubscribe a tag, use the [TagUnsubscribe](#) function.

## Syntax

```
TagSubscribe(STRING TagName [, INT PollTime] [, STRING ScaleMode] [, REAL Deadband] [, STRING Callback] [,  
INT Lightweight] [, INT NoUpdateForDuplicateValues])
```

### TagName

String representing the tag or tag element or the equipment and item name (using equipment.item notation) associated with that tag, to subscribe to in the form of "cluster\_name.tag\_name.element name" or "cluster\_name.equipment.item.element name". If the element name is not specified, it will be resolved at runtime as an unqualified tag reference.

If the tag name exceeds the length limit of 254 characters the hardware alarm "Tag name exceed length limit" will be raised.

### PollTime

Optional integer representing the Datasource Poll time in milliseconds (default 250).

### ScaleMode

Optional string stating the mode to subscribe. Supported modes are: Raw, Eng, Percent. Default is "Eng".

### Deadband

Optional real value specifying the percentage of the variable tag's engineering range that a tag needs to change by for an update to be sent through the system. Default value is -1.0, indicating the deadband specified by the tag definition is to be used.

### Callback

Optional string stating the name of a function to call when the value is updated. If an empty string is specified, no handler is registered. Default value is "" (empty string).

The function should have the structure:

```
FUNCTION evtHandler([INT handle] [, STRING value] [, QUALITY quality] [, TIMESTAMP timestamp])
```

...

END

Where:

*handle* is the subscription that raised the event.

*value* is the tag value.

*quality* is the value quality.

*timestamp* is the value timestamp.

#### *Lightweight*

This optional boolean argument indicates whether or not subscription updates use a "lightweight" version of the tag value that does not include a quality timestamp or a value timestamp.

If not used, this option is set to 1 which means lightweight tag values will be used by default. For a client to retrieve quality and value timestamps for a tag, you should explicitly specify that a full tag value is required by setting this option to 0.

#### *NoUpdateForDuplicateValues*

This optional boolean argument allows you to specify if the subscription accepts duplicated values. Set this value to 1 to indicate that a duplicated update is not required for values that have already successfully written to the device.

---

**Note:** *ScaleMode* and *Deadband* are ignored for digital tags.

---

## Return Value

Integer representing the subscription handle that can be used to read values, hook to events or unsubscribe. If unsuccessful, -1 is returned and an error is set. Even though a subscription handle is returned immediately, it can't be used to get attributes until the subscription has been confirmed as this is an asynchronous Cicode function call. The typical Cicode error is 423 when a subscription handled is used too soon. We recommend the use of a callback function or the direct use of the tag extension, e.g. <tag>.VT

## Related Functions

[TagUnsubscribe](#), [SubscriptionAddCallback](#), [SubscriptionGetAttribute](#), [SubscriptionRemoveCallback](#)

## Example

The following example subscribes the tag "Conveyor1" in order to manually poll for attributes of the tag.

```
...
// Get the last changed value, quality and timestamp for the tag every 1s
...
// LOOP START

SleepMs(1000);

ErrSet(1);
convValue = SubscriptionGetAttribute(subsHandle, "Value");

IF IsError() = 0
convQual = SubscriptionGetAttribute(subsHandle, "ValueQuality");
convTime = SubscriptionGetAttribute(subsHandle, "ValueTimestamp");
```

```
// Format and use data here  
  
END  
  
// LOOP END  
...  
// Unsubscribe the tag TagUnsubscribe(subsHandle);
```

The following example subscribes the "conveyor1" tag as a percentage and polls it every 100ms to check for changes. When the value changes the functions OnValueChanged and ValChanged2 are called. This is the recommended way to do polling of special variables:

```
...  
INT subsHandle = TagSubscribe("Conveyor1", 100, "Percent",  
"OnValueChanged");  
...  
// Later on if no callback was registered initially, a new one can be added.  
SubscriptionAddCallBack(subsHandle, "ValChanged2");  
...  
Function OnValueChanged(INT handle, STRING tagValue, QUALITY valueQuality, TIMESTAMP  
valueTimestamp)  
STRING sTag;  
INT qualityGeneral;  
INT year;  
INT second;  
  
sTag = SubscriptionGetAttribute(handle, "FullName"); // If the name is needed  
qualityGeneral = QualityGetPart(valueQuality, 0); // If General Quality value is needed  
year = TimestampGetPart(valueTimestamp, 0); // if year part is needed  
second = TimestampGetPart(valueTimestamp, 5); // if second part is needed  
...  
END  
...  
Function ValChanged2(INT handle, STRING tagValue, QUALITY valueQuality, TIMESTAMP  
valueTimestamp)  
STRING sTag;  
INT qualityGeneral;  
INT year;  
INT second;  
sTag = SubscriptionGetAttribute(handle, "FullName"); // If the name is needed  
qualityGeneral = QualityGetPart(valueQuality, 0); // if General Quality value is needed  
year = TimestampGetPart(valueTimestamp, 0); // if year part is needed  
second = TimestampGetPart(valueTimestamp, 5); // if second part is needed  
...  
END  
...  
// Remove all callbacks and unsubscribe  
TagUnsubscribe(subsHandle);
```

## See Also

[Tag Functions](#)

## TagUnresolve

This function is used to decrement a reference count implemented on a tag by [TagResolve](#). This will allow the tag to be evicted if the decrement causes its reference count to reach zero (0).

## Syntax

**TagUnresolve(INT *iHandle*)**

*Handle*:

The tag handle returned from a [TagResolve\(\)](#) function call.

## Return Value

The returned integer represents any error that occurs when the unresolve is attempted (0 for no error).

## Related Functions

[TagResolve](#)

## See Also

[Tag Functions](#)

## TagUnsubscribe

Unsubscribes the tag subscription specified by the integer subscription handle that was returned from the [TagSubscribe](#) function. This function also removes any callbacks that are associated with the subscription.

## Syntax

**TagUnsubscribe(INT *Handle*)**

*Handle*

Integer handle of the subscription to unsubscribe.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TagSubscribe](#), [SubscriptionAddCallback](#), [SubscriptionGetAttribute](#), [SubscriptionRemoveCallback](#)

## See Also

[Tag Functions](#)

### TagWrite

Writes to an I/O device variable by specifying the variable tag name or the variable tag name and the name of the requested element having read/write access. The variable tag needs to be defined in the Variable Tags database.

---

**Note:** For this function to be successful a user needs to be logged in.

This function completes asynchronously to the caller. It will be unsuccessful if the tag does not exist or if a write request could not be sent. This function does not test whether the write succeeded. In cases where the write does not succeed, TagWrite does not return a driver error code. You can use the [TagReadEx](#) function to confirm the write operation took place.

TagWrite should only be used when the variable tag name is a calculation such as `sAlarmExt+".Paging"`. For assignment of variables use the assignment operator. For example, `MyCluster.MyAlarm.MyProperty = MyString`.

---

**Note:** When using this function and parameter [Code]ScaleCheck is set to 1, the attempt to write an out-of-range value to a device will not occur. No hardware alarm will be generated. This function checks a value before writing it to a PLC.

## Syntax

**TagWrite**(*STRING Tag, STRING sValue [, INT Offset] [, INT Synch] [, STRING ClusterName]*)

*Tag:*

A string that can be one of the following:

- A tag name — for example, "Fire1"
- An equipment item reference — for example, "Motor1.Fire"
- A tag property — for example, "Fire1.v"
- An equipment item property — for example, "Motor1.Fire.v"
- An alarm property — for example, "AlarmFire1.On"
- An alarm equipment item property — for example, "Motor1.AlarmFire.On"

If the element name is not specified, the writing will be performed to the Field VQT element.

The name of the tag can be prefixed by the name of the cluster, for example, "ClusterName.Tag" or "ClusterName.Equipment.Item".

If the tag name exceeds the length limit of 254 characters the hardware alarm "Tag name exceed length limit" will be raised.

*Value:*

The value to be written to the I/O device variable. The value is specified as a string, however if an integer or real is used the compiler will convert it to a string. The function converts the string into the correct format, and then writes it to the variable.

To write to a particular element in an array, you can enter the array name here, followed by an index to the element as follows:

```
"PLC_Array[9] "
```

The above example tells the function to write to the 10th element in the array variable PLC\_Array (remember, the address of the first element in an array is 0 (zero)).

If you enter an array offset using the *nOffset* argument, it will be added to the index value. See example below.

*Offset:*

Optional offset for an array variable. Default is 0.

If you enter an array index as part of the *sValue* argument, it will be added to this offset value. For example, TagWrite("PLC\_Array[9]", 24, 4) will set the 14th element in PLC\_Array to 24 (because [9] means the 10th element, and an offset of 4 means 4 elements after the 10th = element 14).

*Synch:*

An optional boolean argument that specifies whether the command is synchronous (blocking) or asynchronous (non-blocking). If it is specified as synchronous (blocking) the function will wait until the write has completed and returned from the server before further code execution. This parameter is "False", or asynchronous, by default. If you specify this parameter the rest of the parameters need to be explicitly specified, including *nOffset* which should be set as 0 if the tag is not an array tag.

*ClusterName:*

Specifies the name of the cluster in which the Tag resides. The argument is enclosed in quotation marks.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TagRead](#), [TagReadEx](#), [IODeviceControl](#), [IODeviceInfo](#)

## Example

```
TagWrite("PLC_VAR1", 123);
TagWrite("PLC_VAR1", 123, 0, TRUE); ! Write to PLC variable
! and block until write is successful.
TagWrite("PLC_VAR_STR", "string data to write");
TagWrite("PLC_ARRAY", 42, 3); ! Write to element 4 in array
TagWrite("PLC_Array[9]", 2); ! Write to element 12 in array
TagWrite ("Tag1", "123");
TagWrite("Tag1.Field", "123");
```

## See Also

[Tag Functions](#)

## TagWriteEventQue

Opens the tag write event queue. The TagWriteEventQue is a queue of data containing details of tag value changes initiated by the process. To read events from the queue, use the [QueRead\(\)](#) or [QuePeek\(\)](#) functions. The

queue contains timestamp, tagname and value data for each change event.

This queue is enabled by the corresponding INI parameter [General]TagWriteEventQue. Writes are logged to the queue for all tags whose I/O devices have their Log Write parameter enabled.

## Syntax

*TagWriteEventQue()*

## Return Value

The handle of the tag write event queue, or -1 if the queue cannot be opened.

## Related Functions

[AlarmEventQue](#), [QueRead](#), [QuePeek](#)

## Example

To enable tag logging:

- On the client, set the INI param [General]TagWriteEventQue = 1.
- Enable the IODevices that you want to be logged by setting their Log Write parameters to TRUE. See I/O Devices Properties.

To read the queue you need to create a Cicode function. For example:

```
FUNCTION
checkWrite()
    STRING sTagAndValue = "";
    INT nDateTime = 0;
    INT hQue = TagWriteEventQue();
    IF hQue = -1 THEN
        RETURN;
    END
    WHILE 1 DO
        QueRead(hQue, nDateTime, sTagAndValue, 1);
        Message("Value written", sTagAndValue, 64);
    END
END
```

Where:

- *nDateTime* is the timestamp of the tag write.
- *sTagAndValue* is the tagname and value written to the queue.

When the function is run, successful writes to tags on the IODevice will show as a message "Value written <tagname> <value>".

---

**Note:** The TagWriteEventQue is enabled on each process and will only log the data changes initiated by this process. By combining this data with the current user and machine name Plant SCADA can generate a user activity log with respect to setting data within the control system. This functionality can also be combined with

---

Plant SCADA Reports to augment the detail of the historian data.

---

## See Also

[Tag Functions](#)

### TagWriteIntArray

This function writes an array of integers to a tag. You can write to all elements of an array tag, or write to a subset. Use the Offset parameter to write to the subset (e.g. if you specify a Length of 5 and an Offset of 2, you will write to the 3rd, 4th, 5th, 6th and 7th elements in the tag array).

---

**Note:** For this function to be successful a user needs to be logged in.

---

This function completes asynchronously to the caller. It will be unsuccessful if the tag does not exist or if a write request could not be sent. This function does not test whether the write succeeded. In cases where the write does not succeed, TagWriteIntArray does not return a driver error code. You can use the [TagReadEx](#) function to confirm the write operation took place.

TagWriteIntArray should only be used when the variable tag name is a calculation such as sAlarmExt+".Paging". For assignment of variables use the assignment operator. For example, MyCluster.MyAlarm.MyProperty = MyString.

---

**Note:** When using this function and parameter [\[Code\]ScaleCheck](#) is set to 1, the attempt to write an out-of-range value to a device will not occur. No hardware alarm will be generated. This function checks a value before writing it to a PLC.

---

## Syntax

**TagWriteIntArray**(*STRING Tag, INT Length, VAR LONG Values, INT Offset, INT Sync, STRING ClusterName*)

*Tag:*

The string can refer to either: the variable tag name, the equipment and item name (using equipment.item notation) associated with that tag, the alarm name and the alarm property name, the tag name and the tag element name. If the element name is not specified, the writing will be performed to the Field VQT element. The name of the tag can be prefixed by the name of the cluster that is "ClusterName.Tag" or "ClusterName.Equipment.Item".

If the tag name exceeds the length limit of 254 characters the hardware alarm "Tag name exceed length limit" will be raised.

*Length:*

The number of elements written to the tag.

*Value:*

The element you select in the array to be the first. The array contains the values you are writing to the variable tag. See example below. Must be a Long type variable.

*Offset:*

Optional offset inside the variable tag array where you start writing to. Default is 0.

*Sync:*

An optional boolean argument that specifies whether the command is synchronous (blocking) or asynchronous

(non-blocking). If it is specified as synchronous (blocking) the function will wait until the write has completed and returned from the server before further code execution. This parameter is "False", or asynchronous, by default. If you specify this parameter the rest of the parameters need to be explicitly specified, including *nOffset* which should be set as 0 if the tag is not an array tag.

*ClusterName*:

Specifies the name of the cluster in which the Tag resides. The argument is enclosed in quotation marks.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TagRead](#), [TagReadEx](#), [IODeviceControl](#), [IODeviceInfo](#)

## Example

```
GLOBAL INT ArrayOfInts[64];
FUNCTION ArrayWriteAscendingInts()
    INT i = 0;
    FOR i = 0 TO 63 DO
        ArrayOfInts[i] = i;
    END
    TagWriteIntArray("ModnetIntArray", 64, ArrayOfInts[0]);
END
```

## See Also

[Tag Functions](#)

## TagWriteRealArray

This function writes an array of REAL values to a variable tag. You can write to all elements of an array tag, or write to a subset. Use the Offset parameter to write to the subset (e.g. if you specify a Length of 5 and an Offset of 2, you will write to the 3rd, 4th, 5th, 6th and 7th elements in the tag array).

---

**Note:** For this function to be successful a user needs to be logged in.

This function completes asynchronously to the caller. It will be unsuccessful if the tag does not exist or if a write request could not be sent. This function does not test whether the write succeeded. In cases where the write does not succeed, TagWriteRealArray does not return a driver error code. You can use the [TagReadEx](#) function to confirm the write operation took place.

---

**Note:** When using this function and parameter [Code]ScaleCheck is set to 1, the attempt to write an out-of-range value to a device will not occur. No hardware alarm will be generated. This function checks a value before writing it to a PLC.

## Syntax

**TagWriteRealArray(STRING Tag, INT Length, VAR REAL Value [, INT Offset] [, INT Sync] [, STRING ClusterName])**

*Tag:*

The string can refer to either: the variable tag name, the equipment and item name (using equipment.item notation) associated with that tag, the alarm name and the alarm property name, the tag name and the tag element name. If the element name is not specified, the writing will be performed to the Field VQT element. The name of the tag can be prefixed by the name of the cluster that is "ClusterName.Tag" or "ClusterName.Equipment.Item".

If the tag name exceeds the length limit of 254 characters the hardware alarm "Tag name exceed length limit" will be raised.

*Length:*

The number of elements written to the tag.

*Value:*

The element you select in the array to be the first. The array contains the values you are writing to the variable tag. See example below.

*Offset:*

Optional offset inside the variable tag array where you start writing to. Default is 0.

If you enter an array index as part of the *sValue* argument, it will be added to this offset value. For example, TagWrite("PLC\_Array[9]", 24, 4) will set the 14th element in PLC\_Array to 24 (because [9] means the 10th element, and an offset of 4 means 4 elements after the 10th = element 14).

*Sync:*

An optional boolean argument that specifies whether the command is synchronous (blocking) or asynchronous (non-blocking). If it is specified as synchronous (blocking) the function will wait until the write has completed and returned from the server before further code execution. This parameter is "False", or asynchronous, by default. If you specify this parameter the rest of the parameters need to be explicitly specified, including *nOffset* which should be set as 0 if the tag is not an array tag.

*ClusterName:*

Specifies the name of the cluster in which the Tag resides. The argument is enclosed in quotation marks.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Example

```
GLOBAL REAL ArrayOfInts[64];
FUNCTION ArrayWriteAscendingInts()
    INT i = 0;
    FOR i = 0 TO 63 DO
        ArrayOfInts[i] = i;
    END
    TagWriteIntArray("ModnetIntArray", 64, ArrayOfInts[0]);
END
```

## Related Functions

[TagRead](#), [TagReadEx](#), [IODeviceControl](#), [IODeviceInfo](#)

## See Also

[Tag Functions](#)

## Task Functions

Task functions support advanced multi-tasking operations in Cicode, handling queues, semaphores, messages, and other process functions. The task functions control the transfer of data between different Cicode tasks and across the network to different computers (by remote procedure calls).

Following are functions relating to tasks:

<a href="#">CodeSetMode</a>	Sets the execution mode of a Cicode task.
<a href="#">EnterCriticalSection</a>	Requests permission for the current thread to have access to a critical shared resource (critical section). If the critical section is already being accessed, the thread will be granted access when it is free.
<a href="#">Halt</a>	Halts the current Cicode task.
<a href="#">LeaveCriticalSection</a>	Relinquishes the current thread's ownership of a critical shared resource (critical section).
<a href="#">MsgBrdcst</a>	Broadcasts a message.
<a href="#">MsgClose</a>	Closes a message.
<a href="#">MsgGetCurr</a>	Gets the handle of the message that called the current report or remote procedure.
<a href="#">MsgOpen</a>	Opens a message session with a Plant SCADA server or client.
<a href="#">MsgRead</a>	Reads a message from a session.
<a href="#">MsgRPC</a>	Calls a remote procedure on another Plant SCADA computer.
<a href="#">MsgState</a>	Verifies the status of a message session.
<a href="#">MsgWrite</a>	Writes a message to a session.
<a href="#">QueClose</a>	Closes a queue.
<a href="#">QueLength</a>	Gets the current length of a queue.

<a href="#">QueOpen</a>	Creates or opens a queue.
<a href="#">QuePeek</a>	Searches a queue for a queue element.
<a href="#">QueRead</a>	Reads elements from a queue.
<a href="#">QueWrite</a>	Writes elements to a queue.
<a href="#">ReRead</a>	ReRead is deprecated in this version.
<a href="#">SemClose</a>	Closes a semaphore.
<a href="#">SemOpen</a>	Creates or opens a semaphore.
<a href="#">SemSignal</a>	Signals a semaphore.
<a href="#">SemWait</a>	Waits on a semaphore.
<a href="#">ServerRPC</a>	Calls a remote procedure on a Plant SCADA server.
<a href="#">Sleep</a>	Suspends the current Cicode task for a specified time.
<a href="#">SleepMS</a>	Suspends the current Cicode task for a specified time (in milliseconds).
<a href="#">TaskCall</a>	Calls a Cicode function by specifying the function name and providing an arguments string.
<a href="#">TaskCluster</a>	Gets the name of the cluster context in which the current task is executing.
<a href="#">TaskGetSignal</a>	Retrieves a value that indicates the stop signal for a specific task.
<a href="#">TaskHnd</a>	Gets the handle of a particular task.
<a href="#">TaskKill</a>	Kills a running task.
<a href="#">TaskNew</a>	Creates a new task.
<a href="#">TaskNew</a>	Creates a new task with a subscription rate.
<a href="#">TaskResume</a>	Resumes a task.
<a href="#">TaskSetSignal</a>	Ends a task by manually triggering its stop signal.
<a href="#">TaskSuspend</a>	Suspends a task.

## See Also

[Cicode Function Categories](#)

## Using Cicode Functions

### CodeSetMode

Sets various execution modes for Cicode tasks in the current thread. Using this function, you can specify whether to:

- Write to a local image of an I/O device.
- Check if a variable is within range before writing it to the I/O device.
- Echo error messages to the operator.
- Check the case of string data in Cicode.

### Syntax

**CodeSetMode(*nType*, *Value*)**

*nType*:

Type of mode:

0 - Write to a local image of an I/O device. If you set Value to 1, this mode is enabled, and Cicode writes its local memory image of the I/O device whenever you write to the I/O device. (Cicode assumes that most writes to the I/O device will be done immediately).

This local image might produce problems, or you might want to verify that the data was actually written to the I/O device. If you set Value to 0 (zero), this check is disabled, and Cicode does not write to the local memory image.

1 - Check if a variable is within range before writing it to the I/O device. If you set Value to 1, this mode is enabled. When a variable tag is modified, Cicode checks the new value of the variable against the Scales specified in the Variable Tags database. If the value of the variable is out of scale, Cicode generates a hardware error, and does not write to the I/O device.

If you set Value to 0 (zero), this check is disabled. Cicode writes the variable to the I/O device without checking if its value is within range.

2 - Echo error messages to the operator. If you set Value to 1, this mode is enabled. When a simple user error occurs (for example, if the PageDisplay() function is passed a bad page name), Cicode displays an error message at the Error AN, and returns an error code from the function.

If you set Value to 0 (zero), the echo is disabled. Cicode does not display the error message, and the output of the DspError() function is stopped.

3 - Ignore the case of string data in the current thread of Cicode. If you set Value to 1, this mode is enabled, and Plant SCADA will ignore case in string data. For example, Plant SCADA will equate "Hello" to "HELLO".

If you set Value to 0 (zero), Plant SCADA will be case sensitive to string data. Case sensitivity is used when a string comparison operation is performed. For example:

**IF sStr1 = sStr2 THEN**

(You can also make Plant SCADA case sensitive to strings in all of your Cicode, using the [Code]IgnoreCase parameter.)

4 - Calls the Cicode Profiler. If you set Value to 1, the Profiler is enabled. This will only enable\disable the Profiler for the current Cicode task.

*Value*:

The value of the mode:

- 0 - Disable
- 1 - Enable

## Return Value

-1 if there is an error, otherwise the last value of the mode.

## Example

```
! disable local image write
CodeSetMode(0, 0);
```

## See Also

[Task Functions](#)

## EnterCriticalSection

Requests permission for the current thread to have access to a critical section (shared critical resource). If the critical section is already being accessed by another thread (using the [EnterCriticalSection\(\)](#) function), the current thread will be granted access when the other thread relinquishes ownership using the [LeaveCriticalSection\(\)](#) function.

Once a thread has ownership of a critical section, it can access the same section repeatedly (using the [EnterCriticalSection\(\)](#) function each time). Remember, however, that [LeaveCriticalSection\(\)](#) needs to be called once for each [EnterCriticalSection\(\)](#) used.

---

**Note:** This function is process-based, not computer-based, and so is only effective for threads within the same process. Any threads attempting to gain access to this critical section will be blocked until the thread relinquishes ownership. This function will have no effect on threads running within other processes.

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Do not attempt to access a critical section that is already in use by another thread. This will cause the thread to be blocked until the critical section is relinquished.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Syntax

**EnterCriticalSection(*sName*)**

*sName*:

The name of the critical section. The name needs to be entered in quotation marks and follow the rules outlined in the topic **Tag Name Syntax** in the main help..

## Return Value

This function does not return a value.

## Related Functions

[LeaveCriticalSection](#)

## Example

```
/* Request access to critical section, execute code and relinquish
ownership of critical section. */
FUNCTION
MyCriticalSection()
    EnterCriticalSection("MyCriticalSection");
    // critical code is placed here
    LeaveCriticalSection("MyCriticalSection");
END
```

## See Also

[Task Functions](#)

## Halt

Stops the execution of the current Cicode task and returns to Plant SCADA. This function does not affect any other Cicode tasks that are running.

Use this function to stop execution in nested function calls. When Halt() is called, Cicode returns to Plant SCADA and does not execute any return function calls.

## Syntax

**Halt()**

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[Assert](#), [TaskKill](#)

## Example

```
INT
FUNCTION
```

```
MyFunc(INT Arg)
  IF Arg<0 THEN
    Prompt("Invalid Arg");
    Halt();
  END
  ...
END
```

## See Also

[Task Functions](#)

### LeaveCriticalSection

Relinquishes the current thread's ownership of a critical section (shared critical resource). Once ownership is relinquished, access to the critical section is available to the next thread that requests it (using the [EnterCriticalSection\(\)](#) function). If a thread has been waiting for access, it will be granted at this point.

`LeaveCriticalSection()` needs to be called once for each `EnterCriticalSection()` used.

---

**Note:** This function is process-based, not computer-based, and so will only prevent access to a critical section within a single process. This function only works between Cicode tasks within the same process.

---

## Syntax

`LeaveCriticalSection(sName)`

*sName:*

The name of the critical section. The name needs to be entered in quotation marks.

## Return Value

This function does not return a value.

## Related Functions

[EnterCriticalSection](#)

## Example

```
/* Request access to critical section, execute code and relinquish
ownership of critical section. */
FUNCTION
MyCriticalSection()
  EnterCriticalSection("MyCriticalSection");
  // critical code is placed here
  LeaveCriticalSection("MyCriticalSection");
END
```

## See Also

[Task Functions](#)

### MsgBrdcst

Broadcasts a message to all the clients of a server. You should call this function only on a Plant SCADA server. The message is only received by clients that have a current message session (opened with the [MsgOpen\(\)](#) function).

## Syntax

**MsgBrdcst**(*Name*, *Type*, *Str* [, *sClusterName*])

*sName*:

The name of the Plant SCADA server.

*nType*:

The message number.

*Str*:

The message text.

*sClusterName*:

The name of the cluster to which the server being communicated with belongs. This is optional if you have one cluster.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[MsgOpen](#), [MsgClose](#), [MsgRead](#), [MsgWrite](#), [MsgRPC](#)

## Example

```
! Send a message to all alarm clients.  
MsgBrdcst("Alarm",0,"Alarm Occurred");
```

## See Also

[Task Functions](#)

### MsgClose

Closes a message. After the message is closed, the message post function (the callback function specified in the [MsgOpen\(\)](#) function) is not called if a message is received. When the server side is closed, all clients are closed. When the client side is closed, only the specified client is closed.

## Syntax

**MsgClose(*Name*, *hMsg*)**

*sName*:

The name of the Plant SCADA server.

*hMsg*:

The message handle, returned from the **MsgOpen()** function. The message handle identifies the table where all data on the associated message is stored.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[MsgOpen](#), [MsgRead](#), [MsgWrite](#), [MsgRPC](#)

## Example

```
MsgClose("Alarm", hMsg);
```

## See Also

[Task Functions](#)

## MsgGetCurr

Gets the handle of the client message that called the report or remote procedure that is currently running. You can call this function only in a report or a remote procedure call.

If the report was called by a client, this function returns that client message handle. The report can then send a message back to the client. If a function was called remotely by [MsgRPC\(\)](#), this function returns the message handle for the remote client.

## Syntax

**MsgGetCurr()**

## Return Value

The handle for the client message. The message handle identifies the table where all data on the associated message is stored. The function returns -1 if no client called the report or function.

## Related Functions

[MsgOpen](#), [MsgRPC](#)

## Example

```
! Send message back to the client.  
hMsg=MsgGetCurr();  
IF hMsg<>-1 THEN  
    MsgRPC(hMsg,"Prompt","^"Hello Client from Report Server^"",1);  
END
```

## See Also

[Task Functions](#)

## MsgOpen

Opens a message session with a Plant SCADA server. You can specify a message post function - a callback function that is automatically called when a message arrives. In this function you can call [MsgRead\(\)](#) to get the message, and perform other tasks common to your message sessions. You can then call [MsgWrite\(\)](#) to send a message back to the caller, [MsgRPC\(\)](#) to call a procedure on the caller, and so on.

A logged on user is required for this function to be successful.

---

**Note:** This function will only work successfully on a full-control client. It will not work on a view-only client.

---

## Syntax

**MsgOpen(*Name*, *Mode*, *Fn* [, *sClusterName*] )**

You should use [MsgState\(\)](#) to check the return value of [MsgOpen\(\)](#), unless you are using mode 1. The message post function is set effectively only if [MsgState\(\)](#) returns 1, which means the message session is online.

You can open a client-server message session or a session between redundant servers. This function does not create extra network sessions; it uses Plant SCADA's existing sessions, so you create sessions to the alarm server, report server, trend server, or a named I/O server.

*sName*:

The name of the server to open, either:

For Mode 0, 1, or 3: "Alarm", "Report", "Trend", or the name of an I/O server.

For Mode 2: The default computer name, as set in the [\[LAN\]Node](#) parameter.

*Mode*:

The mode of the message session to open:

0 - Open the client side.

1 - Open the server side.

2 - Open a session from a server to the default computer name. Set *Name* to the computer name of the computer, as defined by the [\[LAN\]Node](#) parameter.

3 - Open a message session between redundant servers. (Clients cannot tell which server they are connected to

or if something has changed on the server. Changes should be performed on the redundant server as well.)

4 - Open a message session from the calling process to the client process. The Name and Fn are ignored in this mode. The message session opened in this mode does not need to call MsgClose.

*Fn:*

The message post function, that is a callback function for the message event. Set Fn to 0 if no event callback function is required.

*sClusterName:*

The name of the cluster the server being communicated with belongs to, this is used when mode is 0, 1 or 3. This is not required if the client is connected to only one cluster containing a server of the type set in the name parameter.

## Return Value

The message handle, or -1 if the session cannot be opened. The message handle identifies the table where data on the associated message is stored. The exception to this is mode 1 where the handle 4096 will be returned if the session can be opened.

## Related Functions

[MsgClose](#), [MsgRead](#), [MsgWrite](#), [MsgRPC](#)

## Example

```
INT hClient = -1;
INT hServer = -1;
// Open message session on the client, connecting using the
//existing message session to the current Alarm server
FUNCTION
MsgClientOpen()
    INT nState;
    hClient = MsgOpen("Alarm", 0, MsgPostClient);
    IF hClient <> -1 THEN
        nState = MsgState(hClient);
        SELECT CASE nState
        CASE 1
            Prompt("Client Message session is online!");
        CASE -1
            Prompt("Client Message session handle is invalid!");
        CASE 0
            Prompt("Client Message session is offline!");
        CASE 2
            Prompt("Client Message session is connecting!");
        CASE 3
            Prompt("Client Message session is disconnecting!");
        CASE ELSE
            Prompt("Client Message session has unknown status!");
        END SELECT
    ELSE
        Prompt("Client Message session could not be opened!");
    END
```

```
END
// Open message session on the server
FUNCTION
MsgServerOpen()
    INT nState;
    hServer = MsgOpen("Alarm", 1, MsgPostServer);
    // Opening a server connection will result in returning 4096 (if successfully opened)
    // and should not be checked via MsgState
    IF hServer <> 4096 THEN
        ErrLog("Server Message session could NOT be opened!");
    ELSE
        ErrLog("Server Message session is online!");
    END
END
// This function is called when the server receives a message from
//the client
INT
FUNCTION
MsgPostServer()
    INT Type;
    INT hCurr;
    STRING Str;
    // Record the returned handle from the read to use it for the write
    hCurr = MsgRead(Type,Str);
    ErrLog("Server received Message: Type = "+Type:###+" Str = "+Str);
    MsgServerWrite(hCurr);
    RETURN 0;
END
// This function is called when the client receives a message from
//the server
INT
FUNCTION
MsgPostClient()
    INT Type;
    INT replyHandle;
    STRING Str;
    MsgRead(Type,Str);
    Prompt("Client received message: Type = "+Type:###+" Str = "+Str);
    RETURN 0;
END
// Write a message to the server
FUNCTION
MsgClientWrite()
    Prompt("Client Write");
    MsgWrite(hClient, 1, "MyClientMessage");
END
// Write a message to the client
FUNCTION
MsgServerWrite(INT hCurr)
    MsgWrite(hCurr, 1, "MyServerMessage");
END
```

## See Also

[Task Functions](#)

## MsgRead

Reads a message from a message session. You can call this function only in a message post function (the callback function specified in the [MsgOpen\(\)](#) function), to read the current message.

The *nType* and *Str* variables of this function return the message number and the text of the message. The return value of this function is the message handle (allowing a response to be sent back if required).  
you need to open the message session using the [MsgOpen\(\)](#) function, to enable the callback function.

## Syntax

**MsgRead(*Type*, *Str*)**

*nType*:

The message number. Must be a Long type variable.

*Str*:

The message text. Must be a String type variable.

## Return Value

The message handle of the message being read.

## Related Functions

[MsgOpen](#), [MsgClose](#), [MsgWrite](#), [MsgRPC](#)

## Example

```
/* This function will read a message from the session and if
Type=1, will display the string as a prompt. If Type=2 then the
speaker beeps and an acknowledgment is sent back to the caller. */
INT
FUNCTION
MsgPostClient()
    INT Type;
    STRING Str;
    INT hMsg;
    hMsg=MsgRead(Type,Str);
    IF Type=1 THEN
        Prompt("Message"+Str);
    ELSE
        IF Type=2 THEN
            Beep();
            MsgWrite(hMsg,2,"DONE");
        END
    END
END
```

## See Also

[Task Functions](#)

### MsgRPC

Calls a remote procedure on another Plant SCADA computer. You can call any of the built-in Cicode functions remotely, or your own functions. You pass the *sName* of the function as a string, not as the function tag, and pass all the arguments for that function in *Arg*.

You can call the function in synchronous or asynchronous Mode. In synchronous mode, `MsgRPC()` does not return until the remote function is called and the result is returned. In asynchronous mode, `MsgRPC()` returns before the function is called, and the result cannot be returned.

---

#### Note:

- The **Allow RPC** property for a Role determines if a user or group of users can perform remote `MsgRPC` and `ServerRPC` calls. This functionality now requires a user-specific permission. To allow existing users to use `MsgRPC` and `ServerRPC`, you need to manually change the value of **Allow RPC** to "TRUE" in Plant SCADA Studio's **Security | Roles** view. You also need to set the **Allow RPC** property to "TRUE" for each server process that these functions will access.
  - If you want to use `MsgRPC` to call a procedure on a remote client computer, you will need to set the parameter [\[Client\]AllowRPC=1](#) on the client computer.
- 

## Syntax

**MsgRPC(*hMsg*, *sName*, *Arg*, *Mode*)**

*hMsg*:

The message handle, returned from the `MsgOpen()` function. The message handle identifies the table where all data on the associated message is stored.

*sName*:

The name of the function to call remotely, as a string.

If this function returns an error, you should confirm that the name you have used is not a label instead of the actual function name. Some functions are aliased using a label, for example, the function `_AlarmGetFieldRec` is defined in the labels database as "AlarmGetFieldRec". In this case, only "`_AlarmGetFieldRec`" should be passed to `MsgRPC`.

*Arg*:

The arguments to pass to the function, separated by commas (,). Enclose string arguments in quotes "" and use the string escape character (^) around strings enclosed within a string. If you do not enclose the string in quotes, then the string is only the first tag found.

*Mode*:

The mode of the call:

0 - Blocking mode - synchronous.

1 - Non-blocking mode - asynchronous.

## Return Value

The result of the remote function call (as a string). If the function is called in asynchronous mode the result of the remote function cannot be returned, so an empty string is returned.

**Note:** The error code 513 will be returned if the **Allow RPC** property is not set to TRUE in an associated Role or server process.

## Related Functions

[MsgOpen](#), [MsgClose](#), [MsgRead](#), [MsgWrite](#)

## Example

```
! Call remote procedure, call MyRPC() on server. Wait for result
Str=MsgRPC(hMsg,"MyRPC","Data",0);
! Call remote procedure, pass two strings. Don't wait for call to complete.
! be careful of your string delimiters as shown.
MsgRPC(hMsg,"MyStrFn","^"First string^","Second string^",1);
! Call remote procedure, pass Cicode string. Don't wait for call to complete.
STRING sMessage = "this is a message";
MsgRPC(hMsg,"MyStrFn","^" + sMessage + "^",1);
! These functions could be used to acknowledge an alarm by record
from any Plant SCADA Client on the network.
! The AlmAck() function is initialized by the Control Client
(Don't forget that servers are also Control Clients.)
! The Alarm tag is passed into the function as a string and a
message is sent to the Alarms Server to initialize
! the AlmAckMsg() function.
FUNCTION
AlmAck(String AlmTag)
    INT hAlarm1;
    hAlarm1 = MsgOpen("Alarm", 0, 0);
    MsgRPC(hAlarm1,"AlmAckMsg",AlmTag,1);
    MsgClose("Alarm", hAlarm1);
END
! The AlmAckMsg() function is executed on the Alarms Server that
the client is connected to. This could be
! either the primary or standby Alarms Server. The function
performs the alarm acknowledge.
FUNCTION
AlmAckMsg(String AlmTag)
    AlarmAckRec(AlarmFirstTagRec(AlmTag,"",""));
END
```

## See Also

[Task Functions](#)

## MsgState

Verifies the status of a message session. Use **MsgState()** to check the return value of [MsgOpen\(\)](#). A message post

function is set effectively only if `MsgState()` returns 1, which means the message session is online.

## Syntax

**MsgState(*hMsg*)**

*hMsg*:

The message handle, returned from the `MsgOpen()` function. The message handle identifies the table where all data on the associated message is stored.

## Return Value

This function has the following possible return values:

- -1 if the message session handle is invalid
- 0 if the message session is offline
- 1 if the message session is online
- 2 if the message session is connecting
- 3 if the message session is disconnecting.

## Related Functions

[MsgOpen](#)

## Example

```
INT hClient = -1;
// Open message session on the client, connecting using the
existing
// message session to the current Alarm server
FUNCTION
MsgClientOpen()
    INT nState;
    hClient = MsgOpen("Alarm", 0, MsgPostClient);
    IF hClient <> -1 THEN
        nState = MsgState(hClient);
        SELECT CASE nState
        CASE 1
            Prompt("Message session is online!");
            //Send a message to the server
            MsgWrite(hClient, 1, "MyMessage");
        CASE -1
            Prompt("Message session handle is invalid!");
        CASE 0
            Prompt("Message session is offline!");
        CASE 2
            Prompt("Message session is connecting!");
        CASE 3
            Prompt("Message session is disconnecting!");
        CASE ELSE
```

```
Prompt("Message session has unknown status!");
END SELECT
ELSE
    Prompt("Message session could not be opened!");
END
END
```

## See Also

[Task Functions](#)

## MsgWrite

Writes a message to a message session. The message is sent to the remote computer's callback function and can be read by calling [MsgRead\(\)](#). If the remote computer has not opened the session, this message is disregarded.

This function returns immediately after passing the message to Plant SCADA. Plant SCADA sends the message over the LAN in the background.

You need to first open the message session using the [MsgOpen\(\)](#) function, to obtain the message handle.

## Syntax

**MsgWrite(*hMsg*, *Type*, *Str*)**

*hMsg*:

The message handle, returned from the [MsgOpen\(\)](#) function. The message handle identifies the table where all data on the associated message is stored.

*nType*:

The integer message data, that is the message number.

*Str*:

The message text.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## Related Functions

[MsgRead](#), [MsgOpen](#)

## Example

```
MsgWrite(hMsg,10,"MyMsg");
```

## See Also

[Task Functions](#)

## QueClose

Closes a queue opened with the [QueOpen\(\)](#) function. All data is flushed from the queue.

If a Cicode task is waiting on the [QueRead\(\)](#) function, it returns with a "queue empty" status. You should close all queues when they are no longer required, because they consume memory. At shutdown, Plant SCADA closes all open queues.

## Syntax

**QueClose(*hQue*)**

*hQue*:

The queue handle, returned from the [QueOpen\(\)](#) function. The queue handle identifies the table where all data on the associated queue is stored.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[QueLength](#), [QueOpen](#), [QueRead](#), [QueWrite](#), [QuePeek](#)

## Example

```
hQue=QueOpen("MyQue",1);
...
QueClose(hQue);
```

## See Also

[Task Functions](#)

## QueLength

Gets the current length of the queue.

## Syntax

**QueLength(*hQue*)**

*hQue*:

The queue handle, returned from the [QueOpen\(\)](#) function. The queue handle identifies the table where all data on the associated queue is stored.

## Return Value

The current length of the queue. If the queue is closed then 0 is returned.

## Related Functions

[QueClose](#), [QueOpen](#), [QueRead](#), [QueWrite](#), [QuePeek](#)

## Example

```
Length=QueLength(hQue);
```

## See Also

[Task Functions](#)

## QueOpen

Open a queue for reading and writing data elements. Use this function to create a new queue or open an existing queue. Use queues for sending data from one task to another or for other buffering operations.

## Syntax

**QueOpen(*Name*, *Mode*)**

*sName*:

The name of the queue. You need to use the following syntax:

Names need to begin with either an alpha character (A-Z or a-z) or the underscore character (\_).

Any following characters need to be either alpha characters (A-Z or a-z), digit characters (0 - 9), period characters (.), backslash characters (\), or underscore characters (\_).

The use of any other characters will result in the name being modified.

*Mode*:

The mode of the queue open:

0 - Open existing queue.

1 - Create new queue.

2 - Attempts to open an existing queue. If the queue does not exist, it will create it.

4 - Create a queue that can have multiple blocked readers.

## Return Value

The queue handle, or -1 if the queue cannot be opened. The queue handle identifies the table where all data on the associated queue is stored.

## Related Functions

[QueClose](#), [QueLength](#), [QueRead](#), [QueWrite](#), [QuePeek](#)

## Example

```
! Create a queue.  
hQue=QueOpen("MyQue",1);  
! Write data into the queue.  
QueWrite(hQue,1,"Quetext");  
QueWrite(hQue,1,"Moretext");  
! Read back data from the queue.  
QueRead(hQue,Type,Str,0);
```

## See Also

[Task Functions](#)

## QuePeek

Searches a queue for a queue element. You can search for the element by specifying a string, an integer, or both. You can remove the element from the queue by adding 8 to the *Mode*.

---

**Note:** This function may modify the arguments *Type* and *Str* depending on the Mode. Therefore, these arguments need to be variables. You should consider that they may be impacted by the setting for Mode when calling the function.

---

## Syntax

**QuePeek(*hQue*, *Type*, *Str*, *Mode*)**

*hQue*:

The queue handle, returned from the [QueOpen\(\)](#) function. The queue handle identifies the table where all data on the associated queue is stored.

*nType*:

The number to search for (if using the search mode for a matching number). If you are using a matching string mode, the number found is returned in *Type*. Must be a Long type variable.

*Str*:

The string to search for (if using the search mode for a matching string). If you are using a matching number mode, the string found is returned in *Str*. Must be a String type variable.

*Mode*:

The mode of the search:

- 1 - Search for a matching string.
- 2 - Search for a matching number.
- 4 - Search for a matching string and use a case-sensitive search.
- 8 - If the element is found, remove it from the queue.

16 Search the queue, in order, for the element at the offset specified by Type.

Use mode 16 when you know the location of the element you want. For example if you set Type = 0, QuePeek will return the first element in the queue, type = 2, will return the 3rd element in the queue, etc. If you specify an offset which is greater than the length of the queue, the "queue empty" error (296) is returned.

You can extend the search by adding modes. For example, set Mode to 3 to search for a matching string and matching number, or set Mode to 11 to also remove the string and number from the queue.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[QueClose](#), [QueLength](#), [QueOpen](#), [QueRead](#), [QueWrite](#)

## Example

```
STRING Str;
INT Type;
! search for 'mystring' in queue, don't remove if found
Str = "mystring";
status=QuePeek(hQue,Type,Str,1);
IF Status = 0 THEN
    ! Now use found Type
...
END
```

## See Also

[Task Functions](#)

## QueRead

Reads data from a queue, starting from the head of the queue. Data is returned in the same order as it was written onto the queue and is removed from the queue when read. If the *Mode* is 0 (non-blocking) and the queue is empty, the function returns with an error. If the *Mode* is 1 (blocking) the function does not return until another Cicode task writes data onto the queue.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

## Syntax

**QueRead(*hQue*, *Type*, *Str*, *Mode*)**

*hQue*:

The queue handle, returned from the QueOpen() function. The queue handle identifies the table where all data on the associated queue is stored.

*nType*:

The integer to read from the queue (written to the queue as Type by the QueWrite() function). Must be an Integer type variable.

*Str:*

The string to read from the queue (written to the queue as Str by the QueWrite() function). Must be a String type variable.

*Mode:*

The mode of the read:

0 - Non-blocking.

1 - Wait for element.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[QueClose](#), [QueLength](#), [QueOpen](#), [QueWrite](#), [QuePeek](#)

## Example

```
Status=QueRead(hQue,Type,Str,0);
IF Status = 0 THEN
    ! Now use Type and Str.
    ...
END
```

## See Also

[Task Functions](#)

## QueWrite

Writes an integer and string onto the end of a queue. The integer and string have no meaning to the queue system, they are just passed from QueWrite() to QueRead(). Queue data is written to the end of the queue. When the data is later read from the queue, it is returned on a first-in-first-out basis.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

## Syntax

**QueWrite(*hQue, Type, Str*)**

*hQue:*

The queue handle, returned from the QueOpen() function. The queue handle identifies the table where all data on the associated queue is stored.

*nType:*

The integer to put into the queue.

*Str:*

The string to put into the queue.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[QueClose](#), [QueLength](#), [QueOpen](#), [QueRead](#), [QuePeek](#)

## Example

```
QueWrite(hQue,2,"Hello there");
QueWrite(hQue,4,"Help");
```

## See Also

[Task Functions](#)

## ReRead

ReRead is deprecated in this version of Plant SCADA.

Tags are now subscribed at the start of a function and updated tag values are sent to the subscribing function. Tag subscriptions are made at the update rate of:

- the graphics page if called from a page
- the default subscription rate as determined by [\[Code\]TimeData](#) if called from Cicode (default 250ms)
- the update rate requested of a task created using [TaskNewEx](#)
- the update rate requested of a subscription created using [TagSubscribe](#).

You will want to verify that the subscription update rate matches the requirements of your system.

After removing ReRead from looping code you may need to extend the period of the [Sleep](#) function. This is to replace the pause ReRead created while it read all the tag values.

## Syntax

**ReRead(*Mode*)**

*Mode:*

The mode of the read:

0 - Read only if data is stale.

1 - Read anyway.

## Return Value

No value (void).

## See Also

[Task Functions](#)

## SemClose

Closes a semaphore opened with [SemOpen\(\)](#). You should close all semaphores when they are no longer required, because they consume memory. If any Cicode tasks are waiting on this semaphore, the tasks are released with an error.

---

**Note:** This function is process-based, not computer-based, and so will only prevent access to an important section within a single process. This function only works between Cicode tasks within the same process.

---

## Syntax

**SemClose(*hSem*)**

*hSem*:

The semaphore handle, returned from the [SemOpen\(\)](#) function. The semaphore handle identifies the table where all data on the associated semaphore is stored.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[SemOpen](#), [SemSignal](#), [SemWait](#)

## Example

```
SemClose(hSem);
```

## See Also

[Task Functions](#)

## SemOpen

Opens a semaphore for access control. When the semaphore is opened, it is initially signaled. Use a semaphore for controlling access to a restricted device, for example, to stop another Cicode task accessing a device while it is in use. You might require semaphores for some Cicode operations, because they can access a device that is critical. (Cicode is a multi-tasking system.)

---

**Note:** This function is process-based, not computer-based, and so will only prevent access to a critical section within a single process. This function only works between Cicode tasks within the same process.

---

## Syntax

**SemOpen(*Name*, *Mode*)**

*sName*:

The name of the semaphore. Follow the naming convention outlined in the topic **Tag Name Syntax** in the main help.

*Mode*:

The mode of the open:

0 - Open existing semaphore.

1 - Create new semaphore.

2 - Attempts to open an existing semaphore. If the semaphore does not exist, it will create it.

## Return Value

The semaphore handle, or -1 if the semaphore was not opened successfully. The semaphore handle identifies the table where all data on the associated semaphore is stored.

## Related Functions

[SemClose](#), [SemSignal](#), [SemWait](#)

## Example

```
hSem=SemOpen("MySem",1);
```

## See Also

[Task Functions](#)

## SemSignal

Signals a semaphore. If several Cicode tasks are waiting on this semaphore, the first task is released. This function is a blocking function. It will block the calling Cicode task until the operation is complete.

---

**Note:** This function is process-based, not computer-based, and so will only prevent access to a critical section within a single process. This function only works between Cicode tasks within the same process.

---

## Syntax

**SemSignal(*hSem*)**

*hSem*:

The semaphore handle, returned from the SemOpen() function. The semaphore handle identifies the table where all data on the associated semaphore is stored.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[SemClose](#), [SemOpen](#), [SemWait](#)

## Example

```
SemSignal(hSem);
```

## See Also

[Task Functions](#)

## SemWait

Waits on a semaphore to be signaled. This function is a blocking function. It will block the calling Cicode task until the operation is complete.

**Note:** This function is process-based, not computer-based, and so will only prevent access to a critical section within a single process. This function only works between Cicode tasks within the same process.

## Syntax

**SemWait(*hSem, Timeout*)**

*hSem*:

The semaphore handle, returned from the SemOpen() function. The semaphore handle identifies the table where all data on the associated semaphore is stored.

*Timeout*:

Semaphore time-out time:

-1 - Wait until semaphore is clear (regardless of how long).

0 - Do not wait - return immediately. (This timeout can be used to check the state.)

> 0 - The number of seconds to wait if semaphore is not signalling, then return.

## Return Value

0 (zero) if the semaphore has been gained, otherwise an error code is returned.

## Related Functions

[SemClose](#), [SemOpen](#), [SemSignal](#)

## Example

```
Status=SemWait(hSem,10);
IF Status=0 THEN
    ...
ELSE
    Prompt("Could not get semaphore");
END
```

## See Also

[Task Functions](#)

## ServerRPC

Calls a remote procedure on the Plant SCADA server specified by the *ServerName* argument. You can call any of the built-in Cicode functions remotely, or your own functions. You pass the Name of the function as a string and pass the arguments for that function in Arg.

You can call the function in synchronous or asynchronous mode. In synchronous mode, ServerRPC() does not return until the function call has completed on the server and the result is returned. In asynchronous mode, ServerRPC() returns before the function is called, an empty string is returned as the result cannot be returned.

This means this function is a blocking function if *iMode* is set to zero (0). A logged on user is required for this function to be successful.

**Note:** The **Allow RPC** property for a Role determines if a user or group of users can perform remote MsgRPC and ServerRPC calls. This functionality now requires a user-specific permission. To allow existing users to use MsgRPC and ServerRPC, you need to manually change the value of **Allow RPC** to "TRUE" in Plant SCADA Studio's **Security | Roles** view. You also need to set the **Allow RPC** property to "TRUE" for each server process that these functions will access.

## Syntax

**ServerRPC(*sServerName*, *sName*, *sArg*, *iMode* [, *sClusterName*])**

*sServerName*:

Plant SCADA server name where the Cicode function needs to be executed. You can optionally specify this name in <ClusterName>.<ServerName> syntax.

*sName*:

The name of the Cicode function to call remotely as string.

*sArg*:

The arguments to pass to the function, separated by commas (,). Enclose string arguments in quotes "" and use the string escape character (^) around strings enclosed within a string. If you forget to enclose the string in quotes, then the string is only the first tag found.

***iMode:***

The mode of the call:

0 - Blocking mode - synchronous

1 - Non-blocking mode - asynchronous

***sClusterName:***

The name of the cluster that the server resides in. This argument is optional, as in several situations it may not be required. In single cluster systems, it is not required, or if the current Cicode task already has the correct cluster context for the server you may omit this argument.

## Return Value

The result of the remote function call (as a string). If the function is called in asynchronous mode the result of the remote function cannot be returned, so an empty string is returned. If the function cannot work due to an error, empty string is also returned and the error can be obtained by calling the IsError function. If current user has view-only access on client side, the function will return an error code.

**Note:** The error code 513 will be returned if the **Allow RPC** property is not set to TRUE in an associated Role or server process.

---

## Related Functions

[TaskNew](#), [TaskNewEx](#)

## See Also

[Task Functions](#)

## Sleep

Suspends the current Cicode task for a specified number of seconds. After the time delay, the Cicode task wakes and continues execution. If the sleep time is 0, the Cicode task is pre-empted for 1 time slice only.

This function does not affect any other Cicode tasks - only the task calling Sleep() is suspended. If you have Cicode that runs continuously in a loop, you should call the Sleep() function somewhere within the loop, to pause the loop and allow other tasks to run.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

## Syntax

**Sleep(Seconds)**

***Seconds:***

The number of seconds. Set to 0 to pre-empt the task for one time-slice.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TaskNew](#), [ReRead](#), [SleepMS](#)

## Example

Buttons

Text	Step
Command	PLCBit=1;Sleep(2);PLCBit=0;
Comment	Switch Bit ON and then OFF 2 seconds later

```
! Display "Hello" 10 times at 60 second intervals.  
WHILE I < 10 DO  
    Sleep(60);  
    Prompt("Hello");  
    I = I + 1;  
END  
! Sleep a while in polling loops  
WHILE < waiting for event or time> DO  
    ! do what ever here  
    ...  
    Sleep(10); ! sleep a while to give other tasks a go.  
    ! the longer the sleep the better for other tasks.  
END
```

## See Also

[Task Functions](#)

## SleepMS

Suspends the current Cicode task for a specified number of milliseconds. After the time delay, the Cicode task wakes and continues execution. This function is similar to the [Sleep](#) function but with greater resolution.

Although a value of 0 milliseconds is accepted, it is not recommended. Try to use at least a value of 1.

This function does not affect any other Cicode tasks; only the task calling SleepMS() is suspended. If you have Cicode that runs continuously in a loop, you should call the SleepMS() or Sleep() function somewhere within the loop, to pause the loop and allow other tasks to run.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

## Syntax

**SleepMS(*Milliseconds*)**

*Milliseconds*:

The number of milliseconds (1000 milliseconds per second). Set to 0 to pre-empt the task for one time-slice. Be careful not to use a value that is too small. Setting the value to 0 would generally have no desirable effect.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TaskNew](#), [ReRead](#), [Sleep](#)

## Example

Buttons

Text	Step
Command	PLCBit=1;SleepMS(500);PLCBit=0;
Comment	Switch Bit ON and then OFF 500 milliseconds later.

```
! Increment a memory variable by ten, 120 times over one minute
(twice a second).
I = 0;
WHILE I < 180 DO
    SleepMS(500);
    iRamp = iRamp + 10;
    I = I + 1;
END
! sleep a while in polling loops
WHILE < waiting for event or time> DO
    ! do what ever here
    ...
    SleepMS(200); ! sleep a while to give other tasks a go.
    ! the longer the sleep the better for other tasks.
END
```

## See Also

[Task Functions](#)

## TaskCall

Calls a Cicode function by specifying the function name and providing an arguments string.

The function will be executed in a new Cicode task with the same cluster context as the current task. The current task will be blocked until the new task completes and a value can be returned.

This function cannot be called from page foreground animation code. If this is attempted, a hardware alarm will be raised and IsError() will return 282 (foreground Cicode cannot block).

You can use this function to call a private function, but only from within the file in which the private function is declared. See [Function Scope](#).

TaskCall allows the function to be called and the arguments provided to be specified dynamically by the Cicode logic. This may be useful in some cases where the function needed is not known until runtime.

Plant SCADA requests the required I/O device data and waits for the data to be returned before starting the function.

## Syntax

**TaskCall(*sName*, *sArgs*)**

*sName*:

The name of the function to call, as a string.

*sArgs*;

The arguments to pass to the function, separated by commas (,). Enclose string arguments in quotes "" and use the string escape character (^) around strings enclosed within a string.

## Return Value

The result of the function call (as a string). If a void function is called, an empty string is returned. To see if an error occurred (such as an invalid function name or invalid arguments) call [IsError\(\)](#).

## Related Functions

[TaskNew](#), [TaskNewEx](#)

## Example

```
STRING result;
result = TaskCall("StrFill", "^"abc^",10");
// result will be set to "abcabcabca"
```

## See Also

[Task Functions](#)

## TaskCluster

Gets the name of the cluster context in which the current task is executing.

## Syntax

**TaskCluster()**

## Return Value

The cluster name of the current context or an empty string if the task is executing without a cluster context.

## Related Functions

[ClusterActivate](#), [ClusterDeactivate](#), [ClusterFirst](#), [ClusterGetName](#), [ClusterIsActive](#), [ClusterNext](#),  
[ClusterServerTypes](#), [ClusterSetName](#), [ClusterStatus](#), [ClusterSwapActive](#), [TaskNew](#), [TaskNewEx](#)

## Example

```
! Get the cluster context of the current task
sCluster = TaskCluster();
```

## See Also

[Task Functions](#)

[Cluster Functions](#)

[Clusters](#)

## TaskGetSignal

Retrieves a value that indicates the signal that is currently set for a specific task. This function can be used to check the value of the current signal before using [TaskSetSignal](#) to apply a new signal.

## Syntax

**TaskGetSignal(*Hnd*)**

*Hnd*:

The task's handle. To retrieve this use the function [TaskHnd\(\)](#).

## Return Value

The value of the current signal. (0 (zero) represents normal operation, 1 indicates the task is stopped).

## Related Functions

[TaskSetSignal](#), [TaskHnd](#), [TaskKill](#), [TaskNew](#), [TaskResume](#)

## See Also

[Task Functions](#)

## TaskHnd

Gets the task handle of a specific task. You can then use the task handle with other task functions to control the task. If you do not specify a thread name, it will default to that of the current task.

## Syntax

**TaskHnd( [sName] )**

*sName:*

The thread name of the task. The thread name is the name of the function that was passed to the [TaskNew\(\)](#) function. For example, if . . .

TaskNew("MyTask", "", 0);

then:

hTask=TaskHnd("MyTask");

will return the handle of this task.

If you do not specify a thread name, it will default to that of the current task.

## Return Value

The task handle, identifying the table where all data on the task is stored.

## Related Functions

[TaskKill](#), [TaskNew](#), [TaskResume](#), [TaskSuspend](#)

## Example

```
! Get the task handle of the current task and then kill it.  
hTask=TaskHnd();  
TaskKill(hTask);  
! Get the task handle of MyTask and then kill it.  
hTask=TaskHnd("MyTask");  
TaskKill(hTask);
```

## See Also

[Task Functions](#)

## TaskKill

Kills a task. The Cicode task will be stopped and will not run again.

## Syntax

**TaskKill(*hTask*)**

*hTask:*

The task handle, returned from the [TaskNew\(\)](#) or [TaskHnd\(\)](#) function. The task handle identifies the table where all data on the associated task is stored.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

**Note:** TaskKill is an abrupt way to stop a Cicode task. It can cause system errors and may have unintended consequences. Whenever possible, use [TaskGetSignal](#) and [TaskSetSignal](#) to stop Cicode tasks. Use TaskKill as a last resort and after observing the following:



### UNINTENDED EQUIPMENT OPERATION

- Do not use TaskKill to stop Cicode tasks until you have attempted the alternative methods stated above.
- Place the processes and devices controlled by Plant SCADA into a state preventing unintended operation before using TaskKill to stop a Cicode task.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Related Functions

[TaskGetSignal](#), [TaskSetSignal](#), [TaskHnd](#), [TaskNew](#), [TaskResume](#), [TaskSuspend](#)

## Example

```
! Create a task, run it for 10 seconds and then kill it.  
hTask=TaskNew("MyFunc","",0);  
Sleep(10);  
TaskKill(hTask);  
FUNCTION  
MyFunc()  
    INT Count;  
    WHILE 1 DO  
        Prompt("Hello "+Count:###);  
        Count=Count+1;  
    END  
END
```

## See Also

[Task Functions](#)

## TaskNew

Creates a new Cicode task and returns the task handle. You pass the *sName* of the function (that creates the task) as a string, not as the function tag, and pass the arguments for that function in *Arg*. After the task is created, it runs in parallel to the calling task. The new task will run forever unless it returns from the function or is killed by another task.

By default, Plant SCADA requests necessary I/O device data and waits for the data to be returned before starting

the task - the task is provided with the correct data, but there will be a delay in starting the task. If you add 16 to the Mode, Plant SCADA starts the task immediately, without waiting for any data from the I/O devices - any I/O device variable that you use will either contain 0 (zero) or the last value read.

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

When mode 16 is used, ensure either appropriate delays are applied before processing references to tags or check qualities of tags. Otherwise, the execution system may process invalid data and returns incorrect results.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

You can specify that if the task is already running, the function will exit without launching a new task and an error will display. This is useful when you want only a single instance of the function running at any point in time.

It is also possible to run the task within the context of a particular cluster in a multi cluster system by setting the *ClusterName* parameter. If a cluster is not specified, the task will use the cluster context of the caller, be it a page or Cicode task. Please be aware that the cluster context cannot be changed once the code is running.

Any animation output for the new task is displayed in the window where it was created. If you want to send output to other windows, use the [WinSelect\(\)](#) function.

**Note:** You can use this function to call a private function, but only from within the file in which the private function is declared. See [Function Scope](#).

## Syntax

**TaskNew(*sName*, *sArg*, *Mode* [, *sClusterName*] )**

*sName*:

The name of the function to create the task, as a string.

*sArg*:

The set of arguments to be passed to the function. Individual arguments need to be separated by commas (,). Enclose string arguments in quotes "" and use the string escape character (^) around strings enclosed within a string. If the string in quotes is not enclosed, then the string is only the first tag found. The entire set of arguments need to be enclosed in quotes ("").

*Mode*:

The mode of the task:

0 - Task runs forever.

1 - Task runs until the current page is changed.

2 - Task runs until the current window is freed.

4 - This mode is deprecated and not active. Currently, by default, task requests I/O device data before starting.

8 - If the task already exists, the function will exit without launching the new task.

16- Task doesn't wait for necessary I/O device data and starts immediately.

You can select any one of modes 0, 1 or 2 and may add mode 4 and/or mode 8 and/or mode 16. For example, set Mode to 6 to request I/O device data before starting the task, and to run the task until the current window is freed.

*sClusterName*:

The name of the cluster context to be applied to the new Cicode task. This is optional if you have one cluster or are resolving the task via the current cluster context. The argument is enclosed in quotation marks "". You may pass "-" as the *ClusterName* argument to run the requested Cicode task without a cluster context.

## Return Value

The task handle, or -1 if the task cannot be successfully created. The task handle identifies the table where data on the associated task is stored.

## Related Functions

[,TaskHnd](#), [TaskKill](#), [TaskNewEx](#), [TaskResume](#), [TaskSuspend](#), [ReRead](#), [WinSelect](#)

## Example

```
! Create a task that displays a message for 10 seconds.  
hTask=TaskNew("MyFunc","Data",0);  
! Continue to run while task runs.  
FUNCTION  
MyFunc(STRING Msg)  
    FOR I=0 TO 10 DO  
        Prompt(Msg);  
        Sleep(1);  
    END  
END  
...  
! Call a function which expects more complex argument  
hTask=TaskNew("ArgFunc","^"string one^","^"string two^",1,2",0);  
hTask=TaskNew("ArgFunc","^""+sOne+"^",^""+sTwo+"^","+iOne:##+", "+  
iTwo:##,0);  
FUNCTION  
ArgFunc(STRING sOne, STRING sTwo, INT iOne, INT iTwo)  
    ...  
END
```

## See Also

[Task Functions](#)

## TaskNewEx

Creates a new Cicode task with an individual subscription rate and returns the task handle. You pass the *sName* of the function (that creates the task) as a string, not as the function tag, and pass the arguments for that function in *Arg*. After the task is created, it runs in parallel to the calling task. The new task will run forever with tags updated at the specified time interval unless it returns from the function or is killed by another task.

By default, Plant SCADA requests necessary I/O device data and waits for the data to be returned before starting the task - the task is provided with the correct data, but there will be a delay in starting the task. If you add 16 to the Mode, Plant SCADA starts the task immediately, without waiting for any data from the I/O devices - any I/O device variable that you use will either contain 0 (zero) or the last value read. Use Mode 16 when the task has to

start immediately.

## WARNING

### UNINTENDED EQUIPMENT OPERATION

When mode 16 is used, ensure either appropriate delays are applied before processing references to tags or check qualities of tags. Otherwise, the execution system may process invalid data and return incorrect results.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

You can specify that if the task is already running, the function will exit without launching the new task and an error will display. This is useful when you want only a single instance of the function running at any point in time.

It is also possible to run the task within the context of a particular cluster in a multi cluster system by setting the *ClusterName* parameter. If a cluster is not specified, the task will use the cluster context of the caller, be it a page or Cicode task. Please be aware that the cluster context cannot be changed once the code is running.

Any animation output for the new task is displayed in the window where it was created. If you want to send output to other windows, use the [WinSelect\(\)](#) function.

**Note:** You can use this function to call a private function, but only from within the file in which the private function is declared. See [Function Scope](#).

## Syntax

**TaskNewEx(*sName*, *sArg*, *Mode*, *SubscriptionRate* [, *sClusterName*] )**

*sName*:

The name of the function to create the task, as a string.

*sArg*:

The set of arguments to be passed to the function. Individual arguments need to be separated by commas (,). Enclose string arguments in quotes "" and use the string escape character (^) around strings enclosed within a string. If you do not enclose the string in quotes, then the string is only the first tag found. The entire set of arguments need to be enclosed in quotes ("").

*Mode*:

The mode of the task:

0 - Task runs forever.

1 - Task runs until the current page is changed.

2 - Task runs until the current window is freed.

4 -This mode is deprecated and not active. Currently, by default, task requests all I/O device data before starting.

8 - If the task already exists, the function will exit without launching the new task.

16- Task doesn't wait for necessary I/O device data and starts immediately.

You can select any one of modes 0, 1 or 2 and may add mode 4 and/or mode 8, and/or mode 16. For example, set Mode to 6 to request I/O device data before starting the task, and to run the task until the current window is freed.

*SubscriptionRate*

The subscription rate for the task, between 0 and 60000 milliseconds, which determines the frequency at which the I / O Server reads I/O device data in order to provide tags with up-to-date Cicode variables. A value of -1 may

be passed which will use the current task's subscription rate or the default value as set by the existing parameter [\[Code\]TimeData](#) which defaults to 250.

*sClusterName*:

The name of the cluster context to be applied to the new Cicode task. This is optional if you have one cluster or are resolving the task via the current cluster context. The argument is enclosed in quotation marks "". You may pass "-" as the *ClusterName* argument to run the requested Cicode task without a cluster context.

## Return Value

The task handle, or -1 if the task cannot be successfully created. The task handle identifies the table where data on the associated task is stored.

## Related Functions

[TaskHnd](#), [TaskKill](#), [TaskNew](#), [TaskResume](#), [TaskSuspend](#), [ReRead](#), [WinSelect](#)

## Example

```
! Create a task that calls a function every half second.  
hTask=TaskNewEx("MyFunc","Data",0,500);
```

## See Also

[Task Functions](#)

## TaskResume

Resumes a task that was suspended by the [TaskSuspend\(\)](#) function. After a task is resumed, it runs on the next time-slice.

## Syntax

**TaskResume(*hTask*)**

*hTask*:

The task handle, returned from the [TaskNew\(\)](#) or [TaskHnd\(\)](#) function. The task handle identifies the table where all data on the associated task is stored.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TaskHnd](#), [TaskKill](#), [TaskSuspend](#), [TaskNew](#)

## Example

```
TaskResume(hTask);
```

## See Also

[Task Functions](#)

### TaskSetSignal

Manually applies a signal to a specified task.

## Syntax

**TaskSetSignal(Hnd, nSignal)**

*Hnd:*

The task's handle. To retrieve this use the function TaskHnd().

*nSignal:*

Allows you to signal a specified task. Set to 0 (zero) for normal operation, 1 to stop the task or any other number that represents a defined signal.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TaskGetSignal](#), [TaskHnd](#), [TaskKill](#), [TaskSuspend](#), [TaskNew](#), [TaskResume](#)

## See Also

[Task Functions](#)

### TaskSuspend

Suspends a task. The task will stop running and will start again only when [TaskResume\(\)](#) is called.

## Syntax

**TaskSuspend(hTask)**

*hTask:*

The task handle, returned from the TaskNew() or TaskHnd() function. The task handle identifies the table where all data on the associated task is stored.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TaskHnd](#), [TaskKill](#), [TaskNew](#), [TaskResume](#)

## Example

```
TaskSuspend(hTask);
...
TaskResume(hTask);
```

## See Also

[Task Functions](#)

## Time and Date Functions

Time/date functions manipulate time and date variables. Plant SCADA stores time/date-related variables as a single integer. This integer represents the number of seconds since 01/01/1970. It is in GMT, but it has an offset that updates it to local time (determined by the timezone the application is in). The Time/date functions convert this integer into time and date variables.

---

**Note:** The Time/date functions can only be used with dates between 1980 and 2035.

---

Following are functions relating to Time and Date:

<a href="#">Date</a>	Gets the current system date in string format.
<a href="#">DateAdd</a>	Adds time to a date.
<a href="#">DateDay</a>	Gets the day from a date.
<a href="#">DateInfo</a>	Returns the date format currently in effect on the Plant SCADA Server.
<a href="#">DateMonth</a>	Gets the month from a date.
<a href="#">DateSub</a>	Subtracts two dates.
<a href="#">DateWeekDay</a>	Gets the day of week from a date.
<a href="#">DateYear</a>	Gets the year from a date.
<a href="#">OLEDateToTime</a>	Converts an OLE DATE value to a Plant SCADA time/date value.
<a href="#">SysTime</a>	Marks the start of an event.

<a href="#">SysTimeDelta</a>	Calculates the time-span of an event.
<a href="#">Time</a>	Gets the current system time in string format.
<a href="#">TimeCurrent</a>	Gets the current time/date value.
<a href="#">TimeHour</a>	Gets hours from a time.
<a href="#">TimeInfo</a>	Returns the time format currently in effect on the Plant SCADA Server.
<a href="#">TimeMidNight</a>	Converts a time variable into the time at midnight.
<a href="#">TimeMin</a>	Gets minutes from a time.
<a href="#">TimeSec</a>	Gets seconds from a time.
<a href="#">TimeSet</a>	Sets the new system time.
<a href="#">TimeToOLEDate</a>	Converts a time/date value to an OLE DATE value.
<a href="#">TimeToStr</a>	Converts a time/date variable into a string.
<a href="#">TimeUTCOffset</a>	Determines the local time bias from UTC at a specified time.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

## Date

Gets the current date in string format.

**Note:** Time/Date functions can only be used with dates between 1980 and 2035. You should check that the date you are using is valid with Cicode similar to the following:

```
IF StrToDate(Arg1)>0 THEN
  ...
ELSE
  ...
END
```

## Syntax

**Date( [Format] )**

*Format:*

Format of the string:

0 - Short time format, hh:mm AM/PM.

1 - Long time format, hh:mm:ss AM/PM.  
2 - Short date format, dd/mm/yy.  
3 - Long date format, day month year.  
4 - Time and date, weekday month day year hh:mm:ss AM/PM.  
5 - Long time period, hh:mm:ss. Time needs to be in seconds.  
6 - Millisecond time period, hh:mm:ss.xxx ("xxx" represents milliseconds). Time needs to be in milliseconds.  
7 - Short time period, hh:mm. Time needs to be in seconds.  
8 - Long time period, "xxxxx Days hh Hours mm min ss sec where xxxx = number of days since 1/1/1970". Time needs to be in seconds.  
9 - Extended date format, dd/mm/yyyy.  
10 - Local TimeDate format, yyyy-mm-dd hh:mm:ss  
11 - Time of Day, hh:mm:ss tt format with no date  
*UTC:*  
Coordinated Universal Time (optional)  
0 - Display the string as a local date/time (default).  
1 - Display the string as a UTC date/time (valid for formats 0-4 and 9).  
If omitted, the default Format is 2. These formats follow the Regional Settings found in the Windows Control Panel.

## Return Value

The current date (in string format).

## Related Functions

[Time](#), [TimeToStr](#), [TimeCurrent](#)

## Example

```
/* If the current system date is 3rd November 1991 and the Windows
date format is dd/mm/yy; */
str = Date();
! Sets str to "3/11/91".
str = Date(2);
! Sets str to "3/11/91".
str = Date(3);
! Sets str to "3rd November 1991".
```

## See Also

[Time and Date Functions](#)

## DateAdd

Adds time (in seconds) to a time/date value. The return value is in time/date variable format. Use this function for time and date calculations.

## Syntax

**DateAdd**(*Time*, *AddTime*)

*Time*:

The time/date to which the *AddTime* will be added.

*AddTime*:

The time to add, in seconds.

## Return Value

The date as a time/date variable.

## Related Functions

[TimeToStr](#), [DateSub](#)

## Example

```
DateVariable=DateAdd(StrToDate("3/11/91"),86400);
! Adds 24 hours to 3/11/91.
NewDate=TimeToStr(DateVariable);
! Sets NewDate to 4/11/91.
```

## See Also

[Time and Date Functions](#)

## DateDay

Gets the day of the month from a time/date variable.

**Note:** Time/Date functions can only be used with dates between 1980 and 2035. You should check that the date you are using is valid with Cicode similar to the following:

```
IF StrToDate(Arg1)>0 THEN
...
ELSE
...
END
```

## Syntax

**DateDay(*Time*)**

*Time*:

The time/date variable.

## Return Value

The day of the month as an integer.

## Related Functions

[Date](#)

## Example

```
! If the current system date is 3rd November 1991;  
Variable=DateDay(TimeCurrent());  
! Sets Variable to 3.
```

## See Also

[Time and Date Functions](#)

## DateInfo

Returns the date format currently used on the Plant SCADA Server.

## Syntax

**DateInfo(*nInfo*, *nExtra*)**

*nInfo*:

Determines the contents of the string returned by the DateInfo() function. Valid values and resulting strings are:

1 - The current date order:

- "0" - MMDDYY
- "1" - DDMMYY
- "2" - YYMMDD.

2 - The current date delimiter.

3 - The current short date format.

4 - The current long date format.

5 - The current extended date format.

6 - The short weekday string. The particular weekday returned is determined by the *nExtra* argument.

7 - The long weekday string. The particular weekday returned is determined by the *nExtra* argument.

8 - The short month string. The particular month returned is determined by the *nExtra* argument.

9 - The long month string. The particular month returned is determined by the *nExtra* argument.

*nExtra*:

When an *nInfo* argument of 6 or 7 is specified, the *nExtra* argument determines which weekday (1-7) is returned by the DateInfo() function.

When an *nInfo* argument of 8 or 9 is specified, the *nExtra* argument determines which month (1-12) is returned by the DateInfo() function.

The *nExtra* argument is ignored if any other *nInfo* value is passed to the function.

## Return Value

A string containing one of the following:

- Current date order ("0" for MMDDYY, "1" for DDMMYY, "2" for YYMMDD);
- Current date delimiter;
- Current short date format;
- Current long date format;
- Current extended date format;
- Short weekday string;
- Long weekday string;
- Short month string;
- Long month string;

This depends on the *nInfo* and *nExtra* arguments passed to the function.

## Related Functions

[TimeInfo](#)

## Example

```
! If the current system date is the fourth of December 2002;  
TwelfthMonth=DateInfo(9,12);  
! Sets TwelfthMonth to "December".
```

## See Also

[Time and Date Functions](#)

## DateMonth

Gets the month from a time/date variable.

**Note:** Time/Date functions can only be used with dates between 1980 and 2035. You should check that the date

---

you are using is valid with Cicode similar to the following:

```
IF StrToDate(Arg1)>0 THEN
...
ELSE
...
END
```

---

## Syntax

**DateMonth(*Time*)**

*Time*:

The time/date variable.

## Return Value

The month of the year as an integer.

## Related Functions

[Date](#)

## Example

```
! If the current system date is 3rd November 1991;
Variable=DateMonth(TimeCurrent());
! Sets Variable to 11.
```

## See Also

[Time and Date Functions](#)

## DateSub

Subtracts time (in seconds) from a time/date value. The return value is in time/date variable format. Use this function for time and date calculations.

## Syntax

**DateSub(*Time*, *SubTime*)**

*Time*:

The time/date from which the SubTime will be subtracted.

*SubTime*:

The time to subtract, in seconds.

## Return Value

The time difference (in seconds) as an integer.

## Related Functions

[Date](#), [DateAdd](#)

## Example

```
Variable=DateSub(StrToDate("05/11/91"),StrToDate("03/11/91"));
! Sets Variable to number of seconds between 2 date/times.
Str=TimeToStr(Variable,5);
! Sets Str to "48:00:00".
```

## See Also

[Time and Date Functions](#)

## DateWeekDay

Gets the day of the week from a time/date variable.

**Note:** Time/Date functions can only be used with dates between 1980 and 2035. You should check that the date you are using is valid with Cicode similar to the following:

```
IF StrToDate(Arg1)>0 THEN
...
ELSE
...
END
```

## Syntax

**DateWeekDay(*Time*)**

*Time:*

The time/date variable.

## Return Value

An integer representing the day of the week as follows:

- 1 - Sunday
- 2 - Monday
- 3 - Tuesday
- 4 - Wednesday
- 5 - Thursday

6 - Friday  
7 - Saturday

## Related Functions

[Date](#), [TimeCurrent](#)

## Example

```
! If the current system date is Sunday, 3rd November 1991;  
Variable=DateWeekDay(TimeCurrent());  
! Sets Variable to 1.
```

## See Also

[Time and Date Functions](#)

## DateYear

Gets the year from a time/date variable.

**Note:** Time/Date functions can only be used with dates between 1980 and 2035. You should check that the date you are using is valid with Cicode similar to the following:

```
IF StrToDate(Arg1)>0 THEN  
...  
ELSE  
...  
END
```

## Syntax

**DateYear**(*Time* [, *Mode*] )

*Time*:

The time/date variable.

*Mode*:

The format required:

0 - Short year, yy. If you use this mode during the year 2000, 0 (zero) will be returned.

1 - Long year, yyyy

If omitted, the default Mode is 0.

## Return Value

The year as an integer.

**Note:** In the year 2000, this function will return 0 (zero) if mode 0 (zero) is used.

## Related Functions

[Date](#)

## Example

```
! If the current system date is 3rd November 1991;  
Variable=DateYear(TimeCurrent(),0);  
! Sets Variable to 91.  
! If the current system date is 18th October 2000;  
Variable=DateYear(TimeCurrent(),0);  
! Sets Variable to 0.  
Variable=DateYear(TimeCurrent(),1);  
! Sets Variable to 1991.
```

## See Also

[Time and Date Functions](#)

## OLEDateToTime

Converts an OLE DATE value (stored in a REAL) to a Plant SCADA time/date value.

**Note:** An OLE DATE representing a local time in the daylight savings(DST) to standard time(Std) transition period will be converted to the DST value internally. For example, if the DST transition is 30/3/2003 2:00:00 Std, the local time will behave in the following manner: 2:00:00 DST -> 2:59:59 DST -> 2:00:00 Std. Because of this, a value representing the period between 2:00:00 and 2:59:59 on that date will be interpreted as 2:00:00 DST, not Std.

## Syntax

**OLEDateToTime(OLEDate, Local)**

*OLEDate:*

The OLE DATE value to convert (stored as a REAL).

*Local:*

*0* - OleDate represents a UTC time.

*1* - OleDate represents a Local time.

## Return Value

Returns a Plant SCADA time/date value.

## Related Functions

[TimeCurrent](#), [TimeToOLEDate](#)

## Example

```
Real = TimeToOLEDate(TimeCurrent(), 1);
! Sets Real to the local date/time value
TimeVariable = OLEDateToTime(Real, 1);
! Sets TimeVariable to the value of Real when interpreted as Local
time.
```

## See Also

[Time and Date Functions](#)

## SysTime

Gets the Plant SCADA internal system millisecond counter. The counter is not based on time, but counts from 0 up to the maximum unsigned integer value and then back to 0.

As Cicode integer handling is signed, values above 24 days will appear negative. For users needing the full 49 day range, please use the [Timestamp Functions](#).

---

**Note:** The delta time between two timestamps will be correct and positive provided the difference between the two is less than 49 days. If the gap exceeds this, the timestamp functions should be used.

You can use this function to time events down to the millisecond level, either by subtracting the current SysTime from the SysTime at the start of the event, or by using the SysTimeDelta() function (which will give the same result).

The SysTime() function does not return the time of day. Use the Time() or TimeCurrent() function to obtain the time of day.

---

**Note:** Time/Date functions can only be used with dates between 1980 and 2035. You should check that the date you are using is valid with Cicode similar to the following:

```
IF StrToDate(Arg1)>0 THEN
...
ELSE
...
END
```

---

## Syntax

**SysTime()**

## Return Value

The Plant SCADA internal system millisecond counter (as an integer).

## Related Functions

[SysTimeDelta](#), [Time](#), [TimeCurrent](#)

## Example

```
Start=SysTime();
! Gets the current time.
...
Delay=SysTime()-Start;
! Sets Delay to the time difference, in milliseconds.
```

## See Also

[Time and Date Functions](#)

## SysTimeDelta

Calculates the time difference between a start time and the current time, and updates the start time to the current time. You can time continuous events in a single operation. See the SysTime() function for information on its use.

**Note:** Time/Date functions can only be used with dates between 1980 and 2035. You should check that the date you are using is valid with Cicode similar to the following:

```
IF StrToDate(Arg1)>0 THEN
...
ELSE
...
END
```

## Syntax

**SysTimeDelta(Start)**

*Start:*

The start time returned from the SysTime() function. Must be a Long type variable.

## Return Value

The time difference from a start time and the current time.

## Related Functions

[SysTime](#)

## Example

```
Start=SysTime();
! Gets the current time.
...
Delay1=SystimeDelta(Start);
! Sets Delay1 to the time difference from Start.
...
```

```
Delay2=SysTimeDelta(Start);
! Sets Delay2 to the time difference from the last SysTimeDelta()
call.
```

## See Also

[Time and Date Functions](#)

### Time

Gets the current time in string format.

**Note:** Time/Date functions can only be used with dates between 1980 and 2035. You should check that the date you are using is valid with Cicode similar to the following:

```
IF StrToDate(Arg1)>0 THEN
...
ELSE
...
END
```

## Syntax

**Time( [Format] )**

*Format:*

The format of the time:

0 - Short time format, hh:mm AM/PM

1 - Long time format., hh:mm:ss AM/PM

If omitted, the default *Format* is 0.

## Return Value

The current time (as a string).

## Related Functions

[Date](#), [TimeToStr](#)

## Example

```
! If the current time is 10:45:30;
Variable=Time();
! Sets Variable to "10:45".
Variable=Time(0);
! Sets Variable to "10:45 AM".
Variable=Time(1);
! Sets Variable to "10:45:30 AM".
```

## See Also

[Time and Date Functions](#)

## TimeCurrent

Gets the current system time/date in time/date variable format. Please be aware that Plant SCADA stores time as the number of seconds since 01/01/1970. You can convert this value into usable date and time variables by using the various Date and Time functions.

**Note:** Time/Date functions can only be used with dates between 1980 and 2035. You should check that the date you are using is valid with Cicode similar to the following:

```
IF StrToDate(Arg1)>0 THEN
...
ELSE
...
END
```

## Syntax

**TimeCurrent()**

## Return Value

A time/date variable.

## Related Functions

[StrToDate](#), [StrToTime](#)

## Example

```
! If the current system time is 11:43:10 a.m.;
TimeVariable=TimeToStr(TimeCurrent(),0);
! Sets TimeVariable to "11:43".
```

## See Also

[Time and Date Functions](#)

## TimeHour

Gets the hour value from a time/date variable.

**Note:** Time/Date functions can only be used with dates between 1980 and 2035. You should check that the date you are using is valid with Cicode similar to the following:

```
IF StrToDate(Arg1)>0 THEN
```

---

```
...
ELSE
...
END
```

---

## Syntax

**TimeHour(*Time*)**

*Time*:

The time/date variable.

## Return Value

The hour (as an integer).

## Related Functions

[TimeCurrent](#)

## Example

```
! If the current system time is 11:43:10 a.m.;
HoursVariable=TimeHour(TimeCurrent());
! Sets HoursVariable to 11.
```

## See Also

[Time and Date Functions](#)

## TimeInfo

Returns the time format currently used on the Plant SCADA Server.

## Syntax

**TimeInfo(*nInfo*)**

*nInfo*:

Determines the contents of the string returned by the TimeInfo() function. Valid values and resulting strings are:

1- The current time hour format:

- "0" - 12 hour
- "1" - 24 hour

2- The current time delimiter.

3- The current morning time extension.

4- The current evening time extension.

## Return Value

Depending on the *nInfo* argument passed to the function, a string containing:

- Current time hour format ("0" for 12 hour, "1" for 24 hour)
- Current time delimiter
- Current morning time extension
- Current evening time extension

## Related Functions

[DateInfo](#)

## Example

```
! If the current system time is 15:43:10.;  
ClockType=TimeInfo(1);  
! Sets ClockType to "1".
```

## See Also

[Time and Date Functions](#)

## TimeMidNight

Returns the number of seconds between midnight on January 1, 1970, and the midnight immediately prior to the specified time/date. This function is useful for performing calculations with the time and date.

**Note:** Time/Date functions can only be used with dates between 1980 and 2035. You should check that the date you are using is valid with Cicode similar to the following:

```
IF StrToDate(Arg1)>0 THEN  
...  
ELSE  
...  
END
```

## Syntax

**TimeMidNight(*Time*)**

*Time*:

The time/date variable.

## Return Value

A time/date variable.

## Related Functions

[TimeCurrent](#)

## Example

```
timeNow = TimeCurrent();
! get the time variable at 7am today
time7am = TimeMidNight(timeNow) + 7*60*60;
IF timeNow > time7am AND timeNow < time7am + 10 THEN
    Beep();
    Prompt("Wake Up!");
END
```

## See Also

[Time and Date Functions](#)

## TimeMin

Gets the minutes value from a time/date variable.

---

**Note:** Time/Date functions can only be used with dates between 1980 and 2035. You should check that the date you are using is valid with Cicode similar to the following:

```
IF StrToDate(Arg1)>0 THEN
    ...
ELSE
    ...
END
```

---

## Syntax

**TimeMin(*Time*)**

*Time*:

The time/date variable.

## Return Value

The minute (as an integer).

## Related Functions

[TimeCurrent](#)

## Example

```
! If the current system time is 11:43:10 a.m.
```

```
MinutesVariable=TimeMin(TimeCurrent());  
! Sets MinutesVariable to 43.
```

## See Also

[Time and Date Functions](#)

## TimeSec

Gets the seconds value from a time/date variable.

**Note:** Time/Date functions can only be used with dates between 1980 and 2035. You should check that the date you are using is valid with Cicode similar to the following:

```
IF StrToDate(Arg1)>0 THEN  
...  
ELSE  
...  
END
```

## Syntax

**TimeSec(*Time*)**

*Time*:

The time/date variable.

## Return Value

The second (as an integer).

## Example

```
! If the current system time is 11:43:10 a.m.;  
SecondsVariable=TimeSec(TimeCurrent());  
! Sets SecondsVariable to 10.
```

## See Also

[Time and Date Functions](#)

## TimeSet

Sets the new system time. You can set the time only on the computer which this function is called.

Time/date functions can only be used with dates from 1980 to 2035.

If you call TimeSet without the required privileges to change the system time you will receive a hardware alarm indicating this.

## Syntax

**TimeSet(*Time*)**

*Time*:

The time/date variable to which the new time is set. Sets the time on this computer only.

## Return Value

The error status of the set

## Related Functions

[DateInfo](#)

## Example

```
! set the time to 11:43 on June 23 1993
time = StrToTime("11:43:00") + StrToDate("23/6/93");
TimeSet(time);.
```

## See Also

[Time and Date Functions](#)

## TimeToOLEDate

Converts a Plant SCADA time/date value to an OLE DATE value (this should be stored in a REAL).

## Syntax

**TimeToOLEDate(*Time, Local*)**

*Time*:

Time/date variable.

*Local*:

0 - The return value is output as UTC time.

1 - The return value is output as Local time.

## Return Value

Returns an OLE date value.

## Related Functions

[TimeCurrent, OLEDATETIME](#)

## Example

```
Real = TimeToOLEDate(TimeCurrent(), 1);
! Sets Real to the local date/time value
```

## See Also

[Time and Date Functions](#)

### TimeToStr

Converts a time/date variable into a string. Use this function for calculating time differences or run times, and so on. Set *Format* to 6 to convert time periods that are in milliseconds, such as the times that are returned from the *SysTime()* and *SysTimeDelta()* functions.

**Note:** Once a date/time is retrieved as UTC, the string cannot be used by the Cicode functions [StrToDate](#) and [StrToTime](#) to synthesize a date/time value as these functions support local time only.

Time/date functions can only be used with dates from 1980 to 2035. You should check that the date you are using is valid with Cicode similar to the following:

```
IF StrToDate(Arg1)>0 THEN
...
ELSE
...
END
```

## Syntax

**TimeToStr**(*Time*, *Format* [, *UTC*] )

*Time*:

The time/date variable.

*Format*:

Format of the string:

0 - Short time format, hh:mm AM/PM.

1 - Long time format, hh:mm:ss AM/PM.

2 - Short date format, based on the system locale setting. For example, dd/mm/yy for English (Australia); mm/dd/yy for English (United States); or dd.mm.yy for German (Germany).

3 - Long date format, day month year.

4 - Time and date, weekday month day year hh:mm:ss AM/PM.

5 - Long time period, hh:mm:ss. Time needs to be in seconds.

6 - Millisecond time period, hh:mm:ss.xxx ("xxx" represents milliseconds). Time needs to be in milliseconds.

7 - Short time period, hh:mm. Time needs to be in seconds.

8 - Long time period, "xxxxx Days hh Hours mm min ss sec". Time needs to be in seconds.

9 - Extended date format, based on the system locale setting. For example, dd/mm/yyyy for English (Australia); mm/dd/yyyy for English (United States); or dd.mm.yyyy for German (Germany).

10 - Local TimeDate format, yyyy-mm-dd hh:mm:ss

11 - Time of Day, hh:mm:ss tt format with no date

*UTC:*

Coordinated Universal Time (optional)

0 - Display the string as a local date/time (default).

1 - Display the string as a UTC date/time (valid for formats 0-4 and 9).

## Return Value

A string containing the converted time/date or period variable, or an empty string if invalid.

## Related Functions

[Time](#), [TimeCurrent](#), [Date](#)

## Example

```
! If the current system time is 11:50:00 a.m.  
String=TimeToStr(TimeCurrent(),0);  
! Sets String to "11:50 AM".  
String=TimeToStr(125 + TimeCurrent(),5);  
! Sets String to "11:52:05" (the current time + 2 minutes and 5  
seconds).
```

## See Also

[Time and Date Functions](#)

## TimeUTCOffset

Determines the local time bias from UTC that was in force at a specified time. For example, US Pacific Standard Time is -8 hrs from UTC, so -28800 would be returned (-8 hours x 60 minutes x 60 seconds). However, if the specified time occurred during daylight saving, the returned value would be -7 hours (or -25200 seconds).

## Syntax

**TimeUTCOffset(*Time*)**

*Time:*

The time/date variable.

## Return Value

The local time bias in seconds.

## Related Functions

[TimeCurrent](#)

## See Also

[Time and Date Functions](#)

## Timestamp Functions

The Timestamp functions enable you to programmatically read and write the Timestamp values of tag elements and access the timestamp information associated with a tag value.

No function taking either Timestamp or Quality as an argument can be called from the Cicode Kernel Window or through a CtCicode CtAPI function.

The following functions are used to interface with the TIMESTAMP data type.

<a href="#">StrToTimestamp</a>	Converts timestamp in a STRING format into a TIMESTAMP format.
<a href="#">TimestampAdd</a>	Adds time (in part of) to a TIMESTAMP variable.
<a href="#">TimestampCreate</a>	Returns a timestamp variable created from the parts.
<a href="#">TimestampToStr</a>	Converts a TIMESTAMP variable into a string.
<a href="#">TimestampDifference</a>	Returns a difference between two TIMESTAMP variables as a number of milliseconds.
<a href="#">TimestampCurrent</a>	Returns the current system date and time as a TIMESTAMP variable.
<a href="#">TimestampFormat</a>	Format a TIMESTAMP variable into a string.
<a href="#">TimestampGetPart</a>	Returns one part (year, month, day, etc) of the timestamp variable.
<a href="#">TimeIntToTimestamp</a>	Converts a time INTEGER which is represented as a number of seconds since 01/01/1970 to a TIMESTAMP.
<a href="#">TimestampSub</a>	Subtracts time (in part of) from a TIMESTAMP variable.
<a href="#">TimestampToInt</a>	Converts a TIMESTAMP variable into a time INTEGER which is represented as a number of seconds since 01/01/1970.
<a href="#">VariableTimestamp</a>	Extracts the timestamp from a given variable.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

### StrToTimestamp

This function converts timestamp in a STRING format into a TIMESTAMP format. This function can be used in browsing functions to convert timestamp fields returned as a string.

## Syntax

`TIMESTAMP StrToTimestamp(STRING String [, INT Format [, INT UTC]])`

*String:*

The date and time as a STRING or TIMESTAMP

*Format:*

The format of the string. Should be 15. Other types reserved for future use.

*UTC:*

If 1, the date is considered to be in UTC. Otherwise, local time is used. This field is only applicable for format STRING formats (i.e. format is not equal 15).

## Return Value

This function returns a date in TIMESTAMP format if successful, otherwise returns TIMESTAMP of January 1, 1601, 00:00:00 (UTC). If the function is unsuccessful you can call the IsError() function to get the actual error code.

## Related Functions

[SchdSpecialItemAdd](#), [SchdSpecialItemDelete](#), [SchdSpecialItemGetField](#), [SchdConfigGetField](#), [SchdGetField](#),  
[SchdOpen](#), [SchdSpecialItemAdd](#), [ScheduleItemAdd](#), [ScheduleItemModify](#), [ScheduleItemSetRepeat](#)

## See Also

[Scheduler Functions](#)

[Timestamp Functions](#)

### TimeIntToTimestamp

Converts a time INTEGER which is represented as a number of seconds since 01/01/1970 to a TIMESTAMP.

## Syntax

**TimeIntToTimestamp(INT TimeInt [, INT Millisecond [, INT UTC]])**

*TimeInt*:

The number of seconds since 01/01/1970.

*Millisecond*:

The number of milliseconds since last second (optional).

*UTC*:

Coordinated Universal Time (optional):

0 – The given time INTEGER is a local date/time.

1 – The given time INTEGER is a UTC date/time (default).

## Return Value

A TIMESTAMP variable or INVALID\_TIMESTAMP if invalid.

## Related Functions

[TimestampCurrent](#), [TimestampDifference](#), [TimestampSub](#), [TimestampToStr](#), [TimestampAdd](#), [TimestampCreate](#), [TimestampFormat](#), [TimestampGetPart](#), [TimestampToInt](#)

## Example

```
INT TimeInt = TimeCurrent();
TIMESTAMP t1 = TimeIntToTimestamp(TimeInt);
STRING sTimestamp = TimestampToStr(t1, 0, 0);
// sTimestamp equals current time in the short time format
// i.e. 'HH:MM AM/PM'
```

## See Also

[Timestamp Functions](#)

## TimestampAdd

Adds an offset to a TIMESTAMP variable.

## Syntax

**TimestampAdd(TIMESTAMP *Timestamp*, INT *Offset* [, INT *Part*])**

*Timestamp*:

The timestamp to which Offset will be added

*Offset*:

The offset to add, expressed in units of the part parameter

*Part:*

Indicates which part to add:

- 0 – Offset is in years.
- 1 – Offset is in months.
- 2 – Offset is in days.
- 3 - Offset is in hours.
- 4 - Offset is in minutes.
- 5 - Offset is in seconds (default)
- 6 - Offset is in milliseconds

## Return Value

The TIMESTAMP variable, or INVALID\_TIMESTAMP if invalid.

## Related Functions

[TimestampCurrent](#), [TimestampDifference](#), [TimestampSub](#), [TimestampToStr](#), [TimeIntToTimestamp](#),  
[TimestampCreate](#), [TimestampFormat](#), [TimestampGetPart](#), [TimestampToTimeInt](#)

## Example

```
TIMESTAMP t1 = TimestampAdd(Tag1.T, 100); // 100 seconds
TIMESTAMP t2 = TimestampAdd(Tag1.T, 1, 0); // 1 year
```

## See Also

[Timestamp Functions](#)

## TimestampCurrent

Return the current system date and time as a TIMESTAMP variable.

## Syntax

**TimestampCurrent()**

## Return Value

A TIMESTAMP variable containing the current system date and time.

## Related Functions

[TimestampAdd](#), [TimestampDifference](#), [TimestampSub](#), [TimestampToStr](#), [TimestampCreate](#), [TimestampFormat](#),  
[TimestampGetPart](#), [TimestampToInt](#), [TimeIntToTimestamp](#)

## Example

```
TIMESTAMP t1 = TimestampCurrent();
```

## See Also

[Timestamp Functions](#)

## TimestampCreate

Returns a TIMESTAMP variable created from the parts.

## Syntax

**TimestampCreate(INT Year, INT Month, INT Day, INT Hour, INT Minute, INT Second, INT Millisecond [, INT bUtc])**

*Timestamp:*

The timestamp from which the part will be extracted.

*Year:*

The year part.

*Month:*

The month part.

*Day:*

The day part.

*Hour:*

The hour part.

*Minute:*

The minute part.

*Second:*

The second part.

*Millisecond:*

The millisecond part.

*UTC:*

Coordinated Universal Time (optional):

0 – The given time INTEGER is a local date/time.

1 – The given time INTEGER is a UTC date/time (default).

## Return Value

The composed TIMESTAMP variable, or INVALID\_TIMESTAMP if invalid

## Related Functions

[TimestampAdd](#), [TimestampCurrent](#), [TimestampDifference](#), [TimestampSub](#), [TimestampToStr](#), [TimestampFormat](#),  
[TimestampGetPart](#), [TimestampToInt](#), [TimeIntToTimestamp](#)

## Example

```
TIMESTAMP timestamp = TimestampCreate(2009, 6, 29, 11, 2, 10, 468);
STRING sTimestamp = TimestampFormat(t1, "dd/MM/yyyy hh:mm:ss.fff");
// sTimestamp equals '29/06/2009 11:02:10.468'
```

## See Also

[Timestamp Functions](#)

## TimestampDifference

Returns the difference between two TIMESTAMP variables as a number of milliseconds.

## Syntax

**TimestampDifference**(TIMESTAMP *Timestamp1*, TIMESTAMP *Timestamp2* [, INT *Part* [, INT *bCumulative*]])

*Timestamp 1:*

The TIMESTAMP variable 1.

*Timestamp 2:*

The TIMESTAMP variable 2.

*Part:*

The type of time units being used for the result:

0 – Result is in years.

1 – Result is in months.

2 – Result is in days.

3 - Result is in hours

4 - Result is in minutes.

5 - Result is in seconds (default).

6 - Result is in milliseconds.

*bCumulative:*

Defines how to pass results which values are greater than their time units (see example below).

0 – Non-cumulative

1 – Cumulative mode (default).

## Return Value

The time period between Timestamp1 and Timestamp2. The value is equal or greater than zero. If error, returns 0 with an error code.

## Related Functions

[TimestampAdd](#), [TimestampCurrent](#), [TimestampSub](#), [TimestampToStr](#), [TimestampCreate](#), [TimestampFormat](#),  
[TimestampGetPart](#), [TimestampToInt](#), [TimeIntToTimestamp](#)

## Example

```
TIMESTAMP t1 = TimestampCreate(2008, 11, 28, 09, 01, 30);
TIMESTAMP t2 = TimestampCreate(2008, 11, 28, 09, 00, 00);
INT nTimespanCumulative = TimestampDifference(t1, t2, 5, 1);
// nTimespanCumulative is equal 90 (seconds)
INT nTimespanNonCumulative = TimestampDifference(t1, t2, 5, 0);
// nTimespanNonCumulative is equal 30 (seconds)
```

## See Also

[Timestamp Functions](#)

## TimestampFormat

Format a TIMESTAMP variable into a string.

## Syntax

**TimestampFormat**(TIMESTAMP *Timestamp*, STRING *Format* [, INT *UTC*])

*Timestamp*:

The timestamp variable.

*Format*:

The format of the string is the same as .NET Framework DateTime format. Specifically be reminded that the format is case sensitive. For example 'MM' is the zero padded month number, whereas 'mm' is the zero padded current minute within the hour. Therefore no Year, Day or Seconds will be displayed if they are specified in uppercase as: YYYY, DD, SS. The correct display will only occur when they are specified in lowercase as: yyyy, dd, ss.

A short list of formats are included below extracted from the "Custom Date and Time Format Strings" in the Microsoft .NET Framework Developer's Guide from the MSDN Library. It is recommended that you confirm its currency by consulting the source information periodically.

Use the following single-character format strings by themselves to choose predefined standard date and time formats:

- d Short date pattern. Ex. "6/15/2009"
- D Long date pattern. Ex. "Monday, June 15, 2009"
- f Full date/time pattern (short time). Ex. "Monday, June 15, 2009 1:45 PM"
- F Full date/time pattern (long time). Ex. "Monday, June 15, 2009 1:45:30 PM"
- g General date/time pattern (short time). Ex. "6/15/2009 1:45 PM"
- G General date/time pattern (long time). Ex. "6/15/2009 1:45:30 PM"•
- M or m Month/day pattern. "June 15"
- O or o Round-trip date/time pattern. Ex. "2009-06-15T13:45:30.0900000"
- R or r RFC1123 pattern. Ex. "Mon, 15 Jun 2009 20:45:30 GMT"
- s Sortable date/time pattern. Ex. "2009-06-15T13:45:30"
- t Short time pattern. Ex. "1:45 PM"
- T Long time pattern. Ex. "1:45:30 PM"
- u Universal sortable date/time pattern. Ex. "2009-06-15 20:45:30Z"
- U Universal full date/time pattern. Ex. "Monday, June 15, 2009 8:45:30 PM"
- Y or y Year month pattern. Ex. "June, 2009"

For all the examples above, the input time/date is assumed to be 6/15/2009 1:45:30 PM with Windows set to the en-US locale. Many of the formats will change according to the current Windows locale.

You may combine the following format specifiers to make up custom format specifications:

- d The day of the month, from 1 through 31.
- dd The day of the month, from 01 through 31.
- ddd The abbreviated name of the day of the week.
- dddd The full name of the day of the week.
- f Fraction of a second. Returns one decimal place for each 'f', up to 'fffffff'
- F Fraction of a second, if non-zero. Returns one decimal place for each 'f', up to 'fffffff'
- g, gg The period or era.
- h The hour, using a 12-hour clock from 1 to 12.
- hh The hour, using a 12-hour clock from 01 to 12.
- H The hour, using a 24-hour clock from 0 to 23.
- HH The hour, using a 24-hour clock from 00 to 23.
- K Time zone information.
- m The minute, from 0 through 59.
- mm The minute, from 00 through 59.
- M The month, from 1 through 12.
- MM The month, from 01 through 12.
- MMM The abbreviated name of the month.
- MMMM The full name of the month.
- s The second, from 0 through 59.
- ss The second, from 00 through 59.
- t The first character of the AM/PM designator.

- tt The AM/PM designator.
- y The year, from 0 to 99.
- yy The year, from 00 to 99.
- yyy The year, with a minimum of three digits.
- yyyy The year as a four-digit number.
- yyyy The year as a five-digit number.
- z Hours offset from UTC, with no leading zeros.
- zz Hours offset from UTC, with a leading zero for a single-digit value.
- zzz Hours and minutes offset from UTC.
- : The time separator.
- / The date separator.
- "string" or 'string' Literal string delimiter.
- % Defines the following character as a custom format specifier.
- \ The escape character.
- Any other character The character is copied to the result string unchanged.

*UTC:*

Coordinated Universal Time (optional):

0 - Returns the time as a local date/time (default).

1 - Returns the time as a UTC date/time.

## Return Value

A string containing the converted time/date, or an empty string if invalid.

## Related Functions

[TimestampAdd](#), [TimestampCurrent](#), [TimestampDifference](#), [TimestampSub](#), [TimestampToStr](#), [TimestampCreate](#),  
[TimestampGetPart](#), [TimestampToInt](#), [TimeIntToTimestamp](#)

## Example

```
TIMESTAMP t1 = TimestampCreate(2009,07,11,09,27,34,123);
STRING sTimestamp = TimestampFormat(t1, "dd/MM/yyyy hh:mm:ss.ffff");
// sTimestamp equals "11/07/2009 09:27:34.123"
```

## See Also

[Timestamp Functions](#)

## TimestampGetPart

Returns one part (year, month, day, etc) of the TIMESTAMP variable.

## Syntax

**TimestampGetPart**(TIMESTAMP *Timestamp*, INT *Part* [, INT *bUtc*])

*Timestamp*:

The timestamp from which the part will be extracted.

*Part*:

Indicates which part to extract:

0 – The year part

1 – The month part.

2 – The day part.

3 – The hour part.

4 – The minute part.

5 – The second part.

6 – The millisecond part.

7 - The number of milliseconds since midnight last occurred.

*bUtc*:

Coordinated Universal Time (optional):

0 – The given time INTEGER is a local date/time.(default).

1 – The given time INTEGER is a UTC date/time.

## Return Value

The required part of the TIMESTAMP variable.

## Related Functions

[TimestampAdd](#), [TimestampCurrent](#), [TimestampDifference](#), [TimestampSub](#), [TimestampToStr](#), [TimestampCreate](#),  
[TimestampFormat](#), [TimestampToInt](#), [TimeIntToTimestamp](#)

## Example

```
TIMESTAMP t1;  
    // insert code here  
INT year = TimestampGetPart(t1, 0);  
INT second = TimestampGetPart(t1, 5);
```

## See Also

[Timestamp Functions](#)

## TimestampSub

Subtracts an offset from a TIMESTAMP variable.

## Syntax

**TimestampSub**(TIMESTAMP *Timestamp*, INT *Offset* [, INT *Part*])

*Timestamp*:

The timestamp to which Offset will be subtracted

*Offset*:

The offset to subtract, expressed in units of the part parameter

*Part*:

Indicates which part to subtract:

0 – Offset is in years.

1 – Offset is in months.

2 – Offset is in days.

3 - Offset is in hours.

4 - Offset is in minutes.

5 - Offset is in seconds (default)

6 - Offset is in milliseconds

## Return Value

The TIMESTAMP variable, or INVALID\_TIMESTAMP if invalid.

## Related Functions

[TimestampAdd](#), [TimestampCurrent](#), [TimestampDifference](#), [TimestampToStr](#), [TimestampCreate](#),  
[TimestampFormat](#), [TimestampGetPart](#), [TimestampToInt](#), [TimeIntToTimestamp](#)

## Example

```
TIMESTAMP t1 = TimestampSub(Tag1.T, 1, 0); // 1 year;
```

## See Also

[Timestamp Functions](#)

## TimestampToStr

Converts a TIMESTAMP variable into a string.

## Syntax

**TimestampToStr**(*Timestamp*, INT *Format* [, INT *UTC*])

*Timestamp*:

Specifies the TIMESTAMP variable.

*Format:*

The format number determines which of date/time patterns are used for formatting returned string. Date/time patterns are defined in regional settings on a particular computer and can vary depend on national or individual preferences. The possible format numbers together with examples based on en-US regional settings are listed below:

- 0 – Short time format, hh:mm.
- 1 – Long time format, hh:mm:ss.
- 2 – Short date format, dd/MM/yyyy.
- 3 – Long date format, dddd, dd MMMM yyyy.
- 4 – Short date & short time format, dd/MM/yyyy hh:mm.
- 5 – Short date & long time format, dd/MM/yyyy hh:mm:ss.
- 6 – Long date & short time format, dddd, dd MMMM yyyy hh:mm.
- 7 – Long date & long time format, dddd, dd MMMM yyyy hh:mm:ss.
- 8 – Month day format, dd MMMM.
- 9 – Year month format, MMMM yyyy.
- 10 – RFC1123 format, ddd, dd MMM yyyy hh:mm:ss GMT
- 11 – Sortable date time format, yyyy-MM-ddThh:mm:ss
- 12 – Short universal format, yyyy-MM-dd hh:mm:ssZ
- 13 – Long universal format, dddd, dd MMMM yyyy hh:mm:ss
- 14 – Long Time & millisecond, hh:mm:ss.fff
- 15 - Date and time as a 64-bit value specified by the number of 100-nanosecond intervals since January 1, 1601 .

*UTC (optional)*

Determines if Universal Time Coordinate is used..

- 0 - Display the string as a local date/time (default).
- 1 - Display the string as a UTC date/time.

## Return Value

A string containing the converted time/date or period variable, or an empty string if invalid.

## Related Functions

[TimestampAdd](#), [TimestampCurrent](#), [TimestampDifference](#), [TimestampCreate](#), [TimestampFormat](#),  
[TimestampGetPart](#), [TimestampToStr](#), [TimeIntToTimestamp](#)

## Example

```
TIMESTAMP t1 = TimestampCreate(2009,07,11,09,27,34,123);
STRING sTimestamp = TimestampToStr(t1, 0, 0);
// sTimestamp equals '9:27 AM'
```

## See Also

[Timestamp Functions](#)

### TimestampToInt

Converts a TIMESTAMP variable into a time INTEGER which is represented as a number of seconds since 01/01/1970.

## Syntax

**TimestampToInt(TIMESTAMP *Timestamp* [, INT *UTC*])**

*Timestamp*:

The timestamp variable.

*UTC*:

Coordinated Universal Time (optional):

0 – Returns the time as a local date/time.

1 – Returns the time as a UTC date/time (default)

## Return Value

Time as a number of seconds since 01/01/1970 in UTC or local time depending on the last input parameter,-1 if invalid.

## Related Functions

[TimestampAdd](#), [TimestampCurrent](#), [TimestampDifference](#), [TimestampSub](#), [TimestampToStr](#), [TimestampCreate](#), [TimestampFormat](#), [TimestampGetPart](#), [TimeIntToTimestamp](#)

## Example

```
TIMESTAMP t1 = TimestampCreate(2009,07,11,09,27,34,123);
INT TimeInt = TimestampToInt(t1);
STRING sTimestamp = TimeToStr(t1, 0, 0);
// sTimestamp equals '9:27 AM'
```

## See Also

[Timestamp Functions](#)

### VariableTimestamp

Extracts the timestamp from a given variable.

**Note:** This function is designed to be used within Cicode; using it on graphical pages may result in displaying an

---

error message instead of an expected timestamp message when either its argument has not good quality or an execution error is set.

---

## Syntax

**VariableTimestamp(Variable, INT Type)**

*Variable*:

The variable from which the timestamp will be extracted.

*nType*:

The type of timestamp:

- 0 – The element's date/time (default)
- 1 – The element's quality date/time
- 2 – The element's value date/time

## Return Value

A TIMESTAMP of the given variable depending on the type. If Variable is NULL, returns INVALID\_TIMESTAMP.

Timestamps of uninitialized stack variables, uninitialized code variables and constants are equal to 0 - invalid timestamp, while their qualities are GOOD.

## Related Functions

[TimestampAdd](#), [TimestampCurrent](#), [TimestampDifference](#), [TimestampSub](#), [TimestampToStr](#), [TimestampCreate](#),  
[TimestampFormat](#), [TimestampGetPart](#), [TimeIntToTimestamp](#)

## Example

```
INT codeVariable = 1;
INT
FUNCTION
MyFunction(REAL arg1)
STRING str = "My string";
TIMESTAMP ts;
ts = VariableTimestamp(codeVariable, 0); //code variable
ts = VariableTimestamp(arg1, 0); //function argument
ts = VariableTimestamp(str, 0); //stack variable
ts = VariableTimestamp(Tag1, 0); //any tag/local variable
RETURN 1;
END
```

## See Also

[Timestamp Functions](#)

## Trend Functions

You can control a trend's operation by using the trend functions. Plant SCADA has standard trend pages, so you would not normally use these low-level functions unless you want to define your own trend displays. You can control the movement of the trend cursor, trend scaling, and the definition of trend attributes (such as the trend starting time and sampling period). You can also create, and subsequently delete trends.

Following are functions relating to Trends:

<a href="#">TrnAddHistory</a>	Restores an old history file to the trend system.
<a href="#">TrnBrowseClose</a>	Closes a trend browse session.
<a href="#">TrnBrowseFirst</a>	Gets the oldest trend entry.
<a href="#">TrnBrowseGetField</a>	Gets the field indicated by the cursor position in the browse session.
<a href="#">TrnBrowseNext</a>	Gets the next trend entry in the browse session.
<a href="#">TrnBrowseNumRecords</a>	Returns the number of records in the current browse session.
<a href="#">TrnBrowseOpen</a>	Opens a trend browse session.
<a href="#">TrnBrowsePrev</a>	Gets the previous trend entry in the browse session.
<a href="#">TrnClientInfo</a>	Gets information about the trend that is being displayed at the AN point.
<a href="#">TrnComparePlot</a>	Prints two trends (one overlaid on the other), each with up to four trend tags.
<a href="#">TrnDelete</a>	Deletes a trend created by the TrnNew() function.
<a href="#">TrnDelHistory</a>	Deletes an old history file from the trend system.
<a href="#">TrnEcho</a>	Enables and disables the echo on the trend display.
<a href="#">TrnEventGetTable</a>	Stores event trend data and the associated time stamp in an event table and time table, for a specified trend tag.
<a href="#">TrnEventGetTableMS</a>	Stores event trend data and time data (including milliseconds) for a specified trend tag.
<a href="#">TrnEventSetTable</a>	Sets trend data from a table, for a specified trend tag.
<a href="#">TrnEventSetTableMS</a>	Sets event trend data and time data (including milliseconds) for a specified trend tag.
<a href="#">TrnExportClip</a>	Exports trend data to the clipboard.

TrnExportCSV	Exports trend data to a file in CSV (comma separated values) format.
TrnExportDBF	Exports trend data to a file in dBASE III format.
TrnExportDDE	Exports trend data to applications via DDE.
TrnFlush	Flushes the trend to disk.
TrnGetBufEvent	Gets the event number of a trend at an offset for a pen.
TrnGetBufTime	Gets the trend time at an offset for a pen.
TrnGetBufValue	Gets the trend value at an offset for a pen.
TrnGetCluster	Gets the name of the cluster the trend graph is associated with.
TrnGetCursorEvent	Gets the event number of a trend at the trend cursor.
TrnGetCursorMSTime	Gets the time (in milliseconds from the previous midnight) at a trend cursor for a specified pen.
TrnGetCursorPos	Gets the position of the trend cursor.
TrnGetCursorTime	Gets the time/date at the trend cursor.
TrnGetCursorValue	Gets the current trend cursor value of a pen.
TrnGetCursorValueStr	Gets the current trend cursor value of a pen as a formatted string.
TrnGetDefScale	Gets the default engineering zero and full scales of a trend tag.
TrnGetDisplayMode	Gets the display mode of a trend.
TrnGetEvent	Gets the event number of a trend at a percentage along the trend.
TrnGetFormat	Gets the format of a pen.
TrnGetGatedValue	Returns the internally stored value for <GATED>.
TrnGetInvalidValue	Returns the internally stored value for <TRN_NO_VALUES>.
TrnGetMode	Gets the display mode of trends (historical or real-time).
TrnGetMSTime	Gets the time (in milliseconds from the previous

	midnight) of the trend (plotted by a specified pen) at a percentage along the trend, using the time and date of the latest sample displayed.
TrnGetPen	Gets the trend tag of a pen.
TrnGetPenComment	Gets the comment of a trend pen.
TrnGetPenFocus	Gets the number of the pen currently in focus.
TrnGetPenNo	Gets the pen number of a pen name.
TrnGetPeriod	Gets the display period of a trend.
TrnGetScale	Gets the scale of a pen.
TrnGetScaleStr	Gets the scale of a pen as a formatted string.
TrnGetSpan	Gets the span time of a trend.
TrnGetTable	Stores trend data in an array.
TrnGetTime	Gets the time/date of a pen.
TrnGetUnits	Gets the data units of a trend pen.
TrnInfo	Gets the configured values of a trend tag.
TrnIsValidValue	Determines whether a logged trend value is <VALID>, <GATED>, or invalid <TRN_NO_VALUES>.
TrnNew	Creates a new trend at run time.
TrnPlot	Prints a plot of one or more trend tags.
TrnPrint	Prints a trend that is displayed on the screen.
TrnSamplesConfigured	Gets the number of samples configured for the currently displayed trend.
TrnScroll	Scrolls a trend pen.
TrnSelect	Sets up a page for a trend.
TrnSetCursor	Moves the trend cursor a specified number of samples.
TrnSetCursorPos	Moves the trend cursor to the given x-axis position.
TrnSetDisplayMode	Specifies how trend samples will be displayed on the screen.
TrnSetEvent	Sets the start event of a trend pen.

<a href="#">TrnSetPen</a>	Sets a trend pen to a new trend tag.
<a href="#">TrnSetPenFocus</a>	Sets the pen focus.
<a href="#">TrnSetPeriod</a>	Sets the display period (time base) of a trend.
<a href="#">TrnSetScale</a>	Re-scales a pen.
<a href="#">TrnSetSpan</a>	Sets the span time of a trend.
<a href="#">TrnSetTable</a>	Sets trend data from an array.
<a href="#">TrnSetTime</a>	Sets the starting time/date of a pen.

The following trend functions are used on standard trend templates. Use these functions only if you create your own trend templates.

(These functions are written in Cicode and can be found in the include project.)

<a href="#">TrendDspCursorComment</a>	Displays the Trend Comment for the currently selected pen.
<a href="#">TrendDspCursorScale</a>	Displays a scale value for the current pen.
<a href="#">TrendDspCursorTag</a>	Displays the tag name of the current pen.
<a href="#">TrendDspCursorTime</a>	Displays the cursor time of the current pen.
<a href="#">TrendDspCursorValue</a>	Displays the cursor value of the current pen.
<a href="#">TrendGetAn</a>	Gets the AN number of the trend under the mouse position.
<a href="#">TrendPopUp</a>	Displays a pop-up trend with the specified trend pens.
<a href="#">TrendRun</a>	Initializes the cursor and rubber-band features on a trend page.
<a href="#">TrendSetDate</a>	Sets the starting date for the pens on a trend.
<a href="#">TrendSetScale</a>	Sets the scale of one or more pens on a trend.
<a href="#">TrendSetSpan</a>	Sets the span time of a trend.
<a href="#">TrendSetTime</a>	Sets the starting time for the pens on a trend.
<a href="#">TrendSetTimebase</a>	Sets a new sampling period for a trend.
<a href="#">TrendWin</a>	Displays a trend page (in a new window) with the specified trend pens.
<a href="#">TrendZoom</a>	Zooms a trend in either one or both axes.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

### TrendDspCursorComment

Displays the Trend Comment for the currently selected pen on the displayed trend graph.

## Syntax

**TrendDspCursorComment(*nAN*)**

*nAN*:

The AN of the trend.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrendDspCursorScale](#), [TrendDspCursorTag](#), [TrendDspCursorTime](#), [TrendDspCursorValue](#), [TrendGetAn](#), [TrendRun](#),  
[TrendSetDate](#), [TrendSetScale](#), [TrendSetSpan](#), [TrendSetTime](#), [TrendSetTimebase](#), [TrendZoom](#)

## Example

See built-in trend templates.

## See Also

[Trend Functions](#)

### TrendDspCursorScale

Displays a scale value for the current pen in the current pen font.

## Syntax

**TrendDspCursorScale(*nAN, Percent*)**

*nAN*:

The AN of the trend.

*Percent*:

The percentage of full scale to display for the current pen, as an integer.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrendDspCursorComment](#), [TrendDspCursorTag](#), [TrendDspCursorTime](#), [TrendDspCursorValue](#), [TrendGetAn](#),  
[TrendRun](#), [TrendSetDate](#), [TrendSetScale](#), [TrendSetSpan](#), [TrendSetTime](#), [TrendSetTimebase](#), [TrendZoom](#)

## Example

See built-in trend templates.

## See Also

[Trend Functions](#)

## TrendDspCursorTag

Displays the trend tag name of the current pen in the pen font.

## Syntax

**TrendDspCursorTag(AN [, Mode] )**

*AN:*

The AN of the trend.

*Mode:*

An optional argument used to specify whether the trend tag name is displayed with a cluster prefix. Set:

0 display tag without cluster prefix (default)

1 display tag with cluster prefix.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrendDspCursorComment](#), [TrendDspCursorScale](#), [TrendDspCursorTime](#), [TrendDspCursorValue](#), [TrendGetAn](#),  
[TrendRun](#), [TrendSetDate](#), [TrendSetScale](#), [TrendSetSpan](#), [TrendSetTime](#), [TrendSetTimebase](#), [TrendZoom](#)

## Example

See built-in trend templates.

## See Also

[Trend Functions](#)

### TrendDspCursorTime

Displays the cursor time of the current pen in the current pen font.

## Syntax

**TrendDspCursorTime(*nAN*, *Format*)**

*nAN*:

The AN number of the trend.

*Format*:

Format of the string:

0 - Short time format, hh:mm AM/PM.

1 - Long time format, hh:mm:ss AM/PM.

2 - Short date format, dd/mm/yy.

3 - Long date format, day month year.

4 - Time and date, weekday month day year hh:mm:ss AM/PM.

5 - Long time period, hh:mm:ss. Time needs to be in seconds.

6 - Millisecond time period, hh:mm:ss:xxx ("xxx" represents milliseconds). Time needs to be in milliseconds.

7 - Short time period, hh:mm. Time needs to be in seconds.

8 - Long time period, days:hh:mm:sec. Time needs to be in seconds.

9 - Extended date format, dd/mm/yyyy.

---

**Note:** If *Format* is set to 1, 2, or 3, the mode will be ignored when the sampling period of the trend changes to less than 1 second. Instead, the time is displayed as hh:min:ss.xxx, that is, displayed as mode 6.

---

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrendDspCursorComment](#), [TrendDspCursorScale](#), [TrendDspCursorTag](#), [TrendDspCursorValue](#), [TrendGetAn](#),  
[TrendRun](#), [TrendSetDate](#), [TrendSetScale](#), [TrendSetSpan](#), [TrendSetTime](#), [TrendSetTimebase](#), [TrendZoom](#)

## Example

See built-in trend templates.

## See Also

[Trend Functions](#)

### TrendDspCursorValue

Display the cursor value of the current pen in the current pen font.

## Syntax

**TrendDspCursorValue(*nAN*)**

*nAN*:

The AN of the trend.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrendDspCursorComment](#), [TrendDspCursorScale](#), [TrendDspCursorTag](#), [TrendDspCursorTime](#), [TrendGetAn](#),  
[TrendRun](#), [TrendSetDate](#), [TrendSetScale](#), [TrendSetSpan](#), [TrendSetTime](#), [TrendSetTimebase](#), [TrendZoom](#)

## Example

See built-in trend templates.

## See Also

[Trend Functions](#)

### TrendGetAn

Gets the AN number of the trend beneath the current mouse position.

## Syntax

**TrendGetAn()**

## Return Value

The AN of the trend, or 0 (zero) if the mouse is not positioned over a valid trend.

## Related Functions

[TrendDspCursorComment](#), [TrendDspCursorScale](#), [TrendDspCursorTag](#), [TrendDspCursorTime](#),  
[TrendDspCursorValue](#), [TrendRun](#), [TrendSetDate](#), [TrendSetScale](#), [TrendSetSpan](#), [TrendSetTime](#), [TrendSetTimebase](#),  
[TrendZoom](#)

## Example

See built-in trend templates.

## See Also

[Trend Functions](#)

## TrendPopUp

Displays a pop-up trend with the specified trend pens. You need to create the trend page with the graphic builder and set the pen names to blank.

## Syntax

**TrendPopUp(*sPage*, *sTag1* [, *sTag2..sTag8*] )**

*sPage*:

The name of the trend page (drawn with the Graphics Builder).

*sTag1*:

The First trend tag to display on the page.

*sTag2..sTag8*:

The trend tags to display on the page.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[PageTrend](#), [TrendWin](#), [WinNewAt](#)

## Example

Buttons

Text	Popup Trend
Command	TrendPopUp("MyPop", "PV1", "PV2", "PV3")

Comment	Display a popup trend with three trend pens
---------	---

## See Also

[Trend Functions](#)

## TrendRun

Initializes the cursor and rubber-band features on a trend page. This function is included as a Cicode Object on all new trend pages. Only use this function when configuring a trend template that requires this functionality.

## Syntax

**TrendRun(*iPageType*)**

*iPageType*:

The type of the page:

0 - Normal trend page template

1 - Compare trend page template

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrendDspCursorComment](#), [TrendDspCursorScale](#), [TrendDspCursorTag](#), [TrendDspCursorTime](#),  
[TrendDspCursorValue](#), [TrendGetAn](#), [TrendSetDate](#), [TrendSetScale](#), [TrendSetSpan](#), [TrendSetTime](#),  
[TrendSetTimebase](#), [TrendZoom](#)

## Example

See built-in trend templates.

## See Also

[Trend Functions](#)

## TrendSetDate

Sets the end date for all pens on a trend. Samples taken after this date will not be displayed. You can enter the date in the *Value* argument, or leave the *Value* blank - a form is then displayed in run time for the operator to enter an end date.

## Syntax

**TrendSetDate(*nAN*, *Value*)**

*nAN*:

The AN of the trend.

*Value*:

The new date, as a string. Samples taken after date will not be displayed. This argument is optional.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrendDspCursorComment](#), [TrendDspCursorScale](#), [TrendDspCursorTag](#), [TrendDspCursorTime](#),  
[TrendDspCursorValue](#), [TrendGetAn](#), [TrendRun](#), [TrendSetScale](#), [TrendSetSpan](#), [TrendSetTime](#), [TrendSetTimebase](#),  
[TrendZoom](#)

## Example

See built-in trend templates.

## See Also

[Trend Functions](#)

## TrendSetScale

Sets the scale of the current pen or of all pens on a trend. You can enter a scale in the *Value* argument, or leave the *Value* blank. A form is then displayed in run time for the operator to enter a value for the scale.

## Syntax

**TrendSetScale(*nAN*, *Percent* [, *Value*] )**

*nAN*:

The AN of the trend.

*Percent*:

The scale to be set:

0 - Zero scale

100 - Full scale

*Value*:

An optional value for the scale, as a string.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrendDspCursorComment](#), [TrendDspCursorScale](#), [TrendDspCursorTag](#), [TrendDspCursorTime](#),  
[TrendDspCursorValue](#), [TrendGetAn](#), [TrendRun](#), [TrendSetDate](#), [TrendSetSpan](#), [TrendSetTime](#), [TrendSetTimebase](#),  
[TrendZoom](#)

## Example

See built-in trend templates.

## See Also

[Trend Functions](#)

## TrendSetSpan

Sets the span time of the trend. The span time is the time period covered in the trend window. You can enter a span time in the *Value* argument, or leave the *Value* blank - a form is then displayed in run time for the operator to enter a value for the span time.

## Syntax

**TrendSetSpan(AN [, Value] )**

*nAN*:

The AN of the trend.

*Value*:

An optional value for the span time, as a string.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrendDspCursorComment](#), [TrendDspCursorScale](#), [TrendDspCursorTag](#), [TrendDspCursorTime](#),  
[TrendDspCursorValue](#), [TrendGetAn](#), [TrendRun](#), [TrendSetDate](#), [TrendSetScale](#), [TrendSetTime](#), [TrendSetTimebase](#),  
[TrendZoom](#)

## Example

See built-in trend templates.

## See Also

[Trend Functions](#)

### TrendSetTime

Sets the end time for all the pens on a trend. Samples taken after this time will not be displayed. You can enter an end time in the *Value* argument, or leave the *Value* blank - a form is then displayed in run time for the operator to enter a value for the end time.

## Syntax

**TrendSetTime(AN [, Value] )**

*nAN*:

The AN of the trend.

*Value*:

An optional value for the end time, as a string. Samples taken after this time will not be displayed.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrendDspCursorComment](#), [TrendDspCursorScale](#), [TrendDspCursorTag](#), [TrendDspCursorTime](#),  
[TrendDspCursorValue](#), [TrendGetAn](#), [TrendRun](#), [TrendSetDate](#), [TrendSetScale](#), [TrendSetSpan](#), [TrendSetTimebase](#),  
[TrendZoom](#)

## Example

See built-in trend templates.

## See Also

[Trend Functions](#)

### TrendSetTimebase

Sets a new sampling period for a trend. You can enter a sampling period in the *Value* argument, or leave the *Value* blank. A form is then displayed in run time for the operator to enter a value for the sampling period.

## Syntax

**TrendSetTimebase(AN [, Value] )**

*nAN:*

The AN of the trend.

*Value:*

An optional value for the sampling period, as a string.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrendDspCursorComment](#), [TrendDspCursorScale](#), [TrendDspCursorTag](#), [TrendDspCursorTime](#),  
[TrendDspCursorValue](#), [TrendGetAn](#), [TrendRun](#), [TrendSetDate](#), [TrendSetScale](#), [TrendSetSpan](#), [TrendSetTime](#),  
[TrendZoom](#)

## Example

See built-in trend templates.

## See Also

[Trend Functions](#)

## TrendWin

Displays a trend page (in a new window) with the specified trend pens. You need to create the trend page with the graphic builder and set the pen names to blank. You then display that page by calling this function and pass the required trend tags. The function will create a new window with the specified window mode.

## Syntax

**TrendWin(sPage, X, Y, Mode, sTag1 [, sTag2..sTag8] )**

*sPage:*

The name of the trend page (drawn with the Graphics Builder).

*X:*

The x pixel coordinate of the top left corner of the window.

*Y:*

The y pixel coordinate of the top left corner of the window.

*Mode:*

The mode of the window:

- 0 - Normal page.
- 1 - Page child window. The window is closed when a new page is displayed, for example, when the [PageDisplay\(\)](#) or [PageGoto\(\)](#) function is called. The parent is the current active window.
- 2 - Window child window. The window is closed automatically when the parent window is freed with the [WinFree\(\)](#) function. The parent is the current active window.
- 4 - No re-size. The window is displayed with thin borders and no maximize/minimize icons. The window cannot be re-sized.
- 8 - No icons. The window is displayed with thin borders and no maximize/minimize or system menu icons. The window cannot be re-sized.
- 16 - No caption. The window is displayed with thin borders, no caption, and no maximize/minimize or system menu icons. The window cannot be re-sized.
- 32 - Echo enabled. When enabled, keyboard echo, prompts, and error messages are displayed on the parent window. This mode should only be used with child windows (for example, Mode 1 and 2).
- 64 - Always on top.
- 128 - Open a unique window. This mode helps prevent this window from being opened more than once.
- 256 - Display the entire window. This mode commands that no parts of the window will appear off the screen
- 512 - Open a unique Super Genie. This mode helps prevent a Super Genie from being opened more than once (at the same time). However, the same Super Genie with different associations can be opened.
- 1024 - Disables dynamic resizing of new window, overriding the setting of the [\[Page\]DynamicSizing](#) parameter.

You can select multiple modes by adding modes together (for example, set Mode to 9 to open a page child window without maximize, minimize, or system menu icons).

*sTag1:*

The first trend tag to display on the page.

*sTag2..sTag8:*

The trend tags to display on the page.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[PageTrend](#), [TrendPopUp](#), [WinNew](#)

## Example

Buttons

Text	Trend Window
Command	<code>TrendWin("MyTrend", 0, 0, 4, "PV1", "PV2", "PV3")</code>
Comment	Display a trend page in a new window with no maximize and minimize icons

## See Also

[Trend Functions](#)

### TrendZoom

"Zooms" a specified trend in either one or both axes. Set the zoom values (*TimeZoom* and/or *ScaleZoom*) to greater than one to "zoom in" or to less than one to "zoom out".

If you specify a destination AN, you can zoom one trend (at *SourceAn*) onto another (at *DestAn*), in the same way as on the standard zoom trend page.

## Syntax

**TrendZoom(*SourceAN*, *TimeZoom*, *ScaleZoom* [, *DestAn*] )**

*SourceAN*:

The AN on which the source trend is located.

*TimeZoom*:

The scale by which the Time axis will be changed (as a real number).

*ScaleZoom*:

The scale by which the Scale axis will be changed (as a real number).

*DestAn*:

The AN on which the destination or target trend is located. If you do not enter a DestAn, it is set to the same AN as SourceAn.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrendDspCursorComment](#), [TrendDspCursorScale](#), [TrendDspCursorTag](#), [TrendDspCursorTime](#),  
[TrendDspCursorValue](#), [TrendGetAn](#), [TrendRun](#), [TrendSetDate](#), [TrendSetScale](#), [TrendSetSpan](#), [TrendSetTime](#),  
[TrendSetTimebase](#)

## Example

```
TrendZoom(30, 2.0, 2.0);
/* Zoom in by a factor of 2 on both the time and scale axes. */
TrendZoom(30, 0.5, 0.5);
/* Zoom out by a factor of 2 on both the time and scale axes. */
```

## See Also

[Trend Functions](#)

## TrnAddHistory

Adds an old (backed up) trend history file to the trend system so that its data can be used. When you back-up a trend file, change its extension so that it indicates the age of the file's trend data (for example, the month and year).

An archived trend file does not need to reside in the same directory as existing active trends. Plant SCADA retrieves the trend name from the header of the specified file and adds its data to the trend history. Please be aware that only a reference to the archived file, and not the file itself, is kept in the trend history. Therefore, care needs to be taken if using this function to access archived files residing on removable storage media. When you remove the media, the archived data is no longer available for display.

This function can only be used if the Trend Server is on the current machine. When the Trend Server is not in the calling process, this function will become blocking and cannot be called from a foreground task. In this case, the return value will be undefined and a Cicode hardware alarm will be raised.

## Syntax

**TrnAddHistory(FileName [, sClusterName] )**

*FileName:*

The file name and directory path of an old history file. The old file does not need to reside in the same directory as existing active trends to be restored.

*sClusterName:*

Specifies the name of the cluster in which the Trend Server resides. This is optional if you have one cluster or are resolving the trend server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnDelHistory](#), [MsgRPC](#)

## Example

```
TrnAddHistory("C:\Plant SCADAx\DATA\TR1.MAY91");
! Adds the file TR1.MAY91 to the trend system.
```

## See Also

[Trend Functions](#)

## TrnBrowseClose

The TrnBrowseClose function terminates an active data browse session and cleans up all resources associated with the session.

This function is a non-blocking function. It does not block the calling Cicode task.

#### **TrnBrowseClose(*iSession*)**

*iSession*

The handle to a browse session previously returned by a [TrnBrowseOpen](#) call.

### **Return Value**

0 (zero) if the trend browse session exists, otherwise an error code is returned.

### **Related Functions**

[TrnBrowseFirst](#), [TrnBrowseGetField](#), [TrnBrowseNext](#), [TrnBrowseNumRecords](#), [TrnBrowsePrev](#)

### **See Also**

[Trend Functions](#)

#### **TrnBrowseFirst**

The TrnBrowseFirst function places the data browse cursor at the first record.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

### **Syntax**

#### **TrnBrowseFirst(*iSession*)**

*iSession*

The handle to a browse session previously returned by a [TrnBrowseOpen](#) call.

### **Return Value**

0 (zero) if the trend browse session exists, otherwise an error code is returned.

### **Related Functions**

[TrnBrowseClose](#), [TrnBrowseGetField](#), [TrnBrowseNext](#), [TrnBrowseNumRecords](#), [TrnBrowseOpen](#), [TrnBrowsePrev](#)

### **See Also**

[Trend Functions](#)

#### **TrnBrowseGetField**

The TrnBrowseGetField function retrieves the value of the specified field from the record the data browse cursor is currently referencing.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

```
STRING TrnBrowseGetField(INT Session, STRING FieldName)
```

*Session*

The handle to a browse session previously returned by a [TrnBrowseOpen](#) call.

*FieldName*

The name of the field that references the value to be returned. Supported fields are:

ACQERROR, AREA, EQUIPMENT, EXPRESSION, FILENAME, FILES, ITEM, LSL, PRIV, RANGE, SDEVIATION, SPCFLAG, STORMETHOD, SUBGRPSIZE, TIME, TRIGGER, USL, XDOUBLEBAR.

See [Browse Function Field Reference](#) for information about fields.

## Return Value

The value of the specified field as a string. An empty string may or may not be an indication that an error has been detected. This should be checked in this instance to determine if an error has actually occurred.

## Related Functions

[TrnBrowseClose](#), [TrnBrowseFirst](#), [TrnBrowseNext](#), [TrnBrowseNumRecords](#), [TrnBrowseOpen](#), [TrnBrowsePrev](#)

## Example

```
STRING fieldValue = "";
STRING fieldName = "TYPE";
INT errorCode = 0;
...
fieldValue = TrnBrowseGetField(iSession, sFieldName);
IF fieldValue <> "" THEN
    // Successful case
ELSE
    // Function did not succeed
END
...
```

## See Also

[Trend Functions](#)

## TrnBrowseNext

The TrnBrowseNext function moves the data browse cursor forward one record. If you call this function after you have reached the end of the records, error 412 is returned (Databrowse session EOF).

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**TrnBrowseNext(*iSession*)**

*iSession*

The handle to a browse session previously returned by a [TrnBrowseOpen](#) call.

## Return Value

0 (zero) if the trend browse session exists, otherwise an error code is returned.

## Related Functions

[TrnBrowseClose](#), [TrnBrowseFirst](#), [TrnBrowseGetField](#), [TrnBrowseNumRecords](#), [TrnBrowseOpen](#), [TrnBrowsePrev](#)

## See Also

[Trend Functions](#)

## TrnBrowseNumRecords

The TrnBrowseNumRecords function returns the number of records that match the filter criteria.

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

**TrnBrowseNumRecords(*iSession*)**

*iSession*

The handle to a browse session previously returned by a [TrnBrowseOpen](#) call.

## Return Value

The number of records that have matched the filter criteria. A value of 0 denotes that no records have matched. A value of -1 denotes that the browse session is unable to provide a fixed number. This may be the case if the data being browsed changed during the browse session.

## Related Functions

[TrnBrowseClose](#), [TrnBrowseFirst](#), [TrnBrowseGetField](#), [TrnBrowseNext](#), [TrnBrowseOpen](#), [TrnBrowsePrev](#)

## Example

```
INT numRecords = 0;  
...  
numRecords = TrnBrowseNumRecords(iSession);
```

```
IF numRecords <> 0 THEN
    // Have records
ELSE
    // No records
END
...
```

## See Also

[Trend Functions](#)

## TrnBrowseOpen

The TrnBrowseOpen function initiates a new browse session and returns a handle to the new session that can be used in subsequent data browse function calls.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**INT TrnBrowseOpen( STRING *Filter*, STRING *Fields* [, STRING *Clusters*] )**

### *Filter*

A filter expression specifying the records to return during the browse. An empty string indicates that all records will be returned. Where a fieldname is not specified in the filter, it is assumed to be tagname. For example, the filter "AAA" is equivalent to "name=AAA".

The following regular expressions are supported: \*expr, expr\*, and \*expr\*. To specify an exclusion filtering condition, use the NOT keyword after the = operator.

### *Fields*

Specifies via a comma delimited string the columns to be returned during the browse. An empty string indicates that the server will return all available columns. Supported fields are:

ACQERROR, CLUSTER, COMMENT, DEADBAND, ENG\_UNITS, EQUIPMENT, EXPRESSION, FILENAME, FILES, FORMAT, HISTORIAN, ITEM, LSL, NAME, PRIV, SAMPLEPER, SDEVIATION, SPCFLAG, STORMETHOD, SUBGRPSIZE, TAGGENLINK, TIME, TRIGGER, TYPE, USL, XDOUBLEBAR.

See [Browse Function Field Reference](#) for information about fields.

### *Clusters*

An optional parameter that specifies via a comma delimited string the subset of the clusters to browse. An empty string indicates that the connected clusters will be browsed.

## Return Value

Returns an integer handle to the browse session. Returns -1 if unsuccessful.

The returned entries will be ordered alphabetically by name.

## Related Functions

[TrnBrowseClose](#), [TrnBrowseFirst](#), [TrnBrowseGetField](#), [TrnBrowseNext](#), [TrnBrowseNumRecords](#), [TrnBrowsePrev](#)

## Example 1

```
INT iSession;  
...  
iSession = TrnBrowseOpen("NAME=ABC*", "NAME,TYPE",  
"ClusterA,ClusterB");  
IF iSession <> -1 THEN  
    // Successful case  
ELSE  
    // Function did not succeed  
END  
...
```

## Example 2 - Filters

The following trend tags have been defined: Trend01, Trend02 and Trend03.

```
// Example Filter  
FUNCTION TrnBrowseTest()  
    STRING sFilter = "TAG=Trend01 OR Trend03";  
    STRING sField = "TAG";  
    STRING sTag = "";  
    INT iStatus = -1;  
    INT hTagBrowse = TrnBrowseOpen(sFilter,sField);  
    IF (hTagBrowse <> -1) THEN  
        iStatus = TrnBrowseFirst(hTagBrowse);  
        WHILE iStatus = 0 DO  
            sTag = TrnBrowseGetField(hTagBrowse,sField);  
            Message(sTag,sTag,64); // Trend01, Trend03  
            iStatus = TrnBrowseNext(hTagBrowse);  
    END  
    TrnBrowseClose(hTagBrowse);  
END  
END
```

### Results

Good: STRING sFilter = "TAG=Trend01 OR Trend03";  
Good: STRING sFilter = "TAG=Trend01|Trend03";  
Fail: STRING sFilter = "TAG=Trend01 OR TAG=Trend03";

## See Also

[Trend Functions](#)

### TrnBrowsePrev

The TrnBrowsePrev function moves the data browse cursor back one record. If you call this function after you have reached the beginning of the records, error 412 is returned (Databrowse session EOF).

This function is a non-blocking function. It does not block the calling Cicode task.

## Syntax

**TrnBrowsePrev(*iSession*)**

*iSession*

The handle to a browse session previously returned by a [TrnBrowseOpen](#) call.

## Return Value

0 (zero) if the trend browse session exists, otherwise an error code is returned.

## Related Functions

[TrnBrowseClose](#), [TrnBrowseFirst](#), [TrnBrowseGetField](#), [TrnBrowseNext](#), [TrnBrowseNumRecords](#), [TrnBrowseOpen](#)

## See Also

[Trend Functions](#)

## TrnClientInfo

Gets information about the trend that is being displayed at the AN point.

## Syntax

**TrnClientInfo(*nAN, Pen, Type, Data, Error*)**

*nAN*:

The AN point number at which the trend is displayed.

*Pen*:

The trend pen number:

0 - The pen currently in focus

1...8 - Pen1 . . . Pen8

*nType*:

The type of information you would like returned.

1 - The number of samples configured for the trend. If you select Type 1, the *Data* argument is ignored.

*Data*:

For future enhancements; is currently ignored.

*Error*:

Output Parameter: If the function is successful, the error is set to 0 (zero). If unsuccessful, an error value is set, and a hardware alarm is triggered. Must be a Long type variable.

## Return Value

The requested information (as a string) if successful, otherwise a hardware alarm is generated.

## Related Functions

[TrnInfo](#)

## Example

```
INT nError;
INT nSamples;
INT nTime;
!Gets the number of samples configured for the current pen of the
trend displayed at AN 30.
nSamples = TrnClientInfo(30, 0, 1, "", nError);
IF nError = 0 THEN
    nTime = nSamples * 50;
ELSE
    nTime = 0;
END
```

## See Also

[Trend Functions](#)

## TrnComparePlot

Prints two trends, one overlaid on the other. Each trend can have up to four tags configured on it. The significance of this type of plot is that the two trends show different times and display periods. It is possible to compare a trend tag or tags over different time slots. Each trend line is drawn with a different pen style and marker as appropriate. The trend plot includes a comment and a legend, and you can specify the vertical high and low scales.

For more advanced trend plotting, you can use the low-level plot functions.

## Syntax

**TrnComparePlot(*sPort*, *sTitle*, *sComment*, *AN*, *iMode*, *nSamples*, *iTime1*, *rPeriod1*, *iTime2*, *rPeriod2*,  
*Tag1.....Tag8*, *rLoScale1*, *rHiScale1*,.....*rLoScale8*, *rHiScale8*)**

***sPort*:**

The name of the printer port to which the plot will be printed. This name needs to be enclosed within quotation marks. For example LPT1:, to print to the local printer, or \\Pserver\canon1 using UNC to print to a network printer.

***sTitle*:**

The title of the trend plot.

***sComment*:**

The comment that is to display beneath the title of the trend plot. You do not have to enter a comment.

*nAN:*

Sets the display mode. A value of 0 causes the default display mode to be used. Otherwise, the display mode of the specified AN is set.

*iMode:*

The color mode of the printer.

0 - black and white (default)

1 - Color

*nSamples:*

The number of data points on the plot.

*iTime1:*

The end point in time (the most recent point) for the first trend.

*rPeriod1:*

The period (in seconds) of the first trend. This can differ from the actual trend period. If you do not enter a period, it defaults to the sample period of Tag1.

*iTime2:*

The end point in time (the most recent point) for the second trend.

*rPeriod2:*

The period (in seconds) of the second trend. This can differ from the actual trend period. If you do not enter a period, it defaults to the sample period of Tag5.

*Tag1 . . . Tag8:*

The trend tags for the plot. Tags 1 to 4 needs to be for the first trend, and tags 5 to 8 needs to be for the second.

*rLoScale1, HiScale1,...,LoScale8, HiScale8*

The minimum and maximum on the vertical scale for the trend line of each of the tags (Tag1 . . . Tag8)

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnPlot](#), [TrnPrint](#), [PlotOpen](#), [SPCPlot](#)

## Example

```
/* Prints two black and white trends (one overlaid on the other)
to LPT1, comparing the trend lines of one trend tag (Feed_Flow) at
different times. The first trend line has a starting time of
12 noon, on 11/12/96, and the second has a starting time of 9am, on
11/10/96. Both contain 200 sample points, and have a period of 2
seconds. Both trend lines will be on a vertical scale of 10-100.
*/
INT Time;
```

```
INT RefTime;
Time = StrToDate("11/12/96") + StrToTime("12:00:00");
RefTime = StrToDate("11/10/96") + StrToTime("09:00:00");
TrnComparePlot("LPT1:","Citect Flow Comparison Plot","Comparison
with Reference",0,
0,200,Time,2,RefTime,2,"Feed_Flow","","","","Feed_Flow","","","","",
10,100,0,0,0,0,0,0,10,100);
```

## See Also

[Trend Functions](#)

### TrnDelete

Deletes a trend that is displayed on a current page. This trend may have been created by the [TrnNew\(\)](#) function or by a trend object.

## Syntax

**TrnDelete(*nAN*)**

*nAN*:

The AN where the trend is located.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnNew](#)

## Example

```
TrnDelete(20);
! Deletes the trend at AN20.
```

## See Also

[Trend Functions](#)

### TrnDelHistory

Removes a trend history file that has been added to the trend system by the [TrnAddHistory\(\)](#) function. This function does not delete the file completely, it only disconnects it from the historical trend system.

This function can only be used if the Trend Server is on the current machine. When the Trend Server is not in the calling process, this function will become blocking and cannot be called from a foreground task. In this case, the

return value will be undefined and a Cicode hardware alarm will be raised.

## Syntax

**TrnDelHistory(FileName [, sClusterName] )**

*FileName*:

The trend history file to disconnect from the historical trend system.

*sClusterName*:

Specifies the name of the cluster in which the Trend Server resides. This is optional if you have one cluster or are resolving the trend server via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnAddHistory](#), [MsgRPC](#)

## Example

```
TrnDelHistory("C:\Plant SCADA\DATA\TR1_91.MAY");
! Disconnects the file from the trend system.
```

## See Also

[Trend Functions](#)

## TrnEcho

Enables and disables the echo of the trend display. Use this function when you need to make many changes to a trend display, so that the display updates only once. You should first disable the trend echo, do all the trend changes, and then enable the echo to show the changes.

## Syntax

**TrnEcho(*nAN*, *nMode*)**

*nAN*:

The animation number of the trend.

*nMode*:

The mode of the echo:

0 - Disable echo of the trend display.

1 - Enable echo of the trend display, to show changes.

## Return Value

The current echo mode, otherwise 0 (zero) is returned, and an error code is set. You can call the `IsError()` function to get the actual error code.

## Related Functions

[TrnSetScale](#), [TrnSetPeriod](#)

## Example

```
! Disable echo of trend display at AN40
TrnEcho(40,0);
! Change the scales on pens 1 and 2
TrnSetScale(40,1,0,-1000);
TrnSetScale(40,1,100,-1000);
TrnSetScale(40,2,0,-1000);
TrnSetScale(40,2,100,-1000);
! Enable echo to show changes to the display
TrnEcho(40,1);
```

## See Also

[Trend Functions](#)

## TrnEventGetTable

Stores event trend data in an event table and the associated time stamp in a time table, for a specified trend tag. Data is stored at the specified *Period*, working backwards from the starting point specified by *EventNo*. If *Period* differs from the trend period configured in the Trend Tags database, the values to be stored are calculated from the trend data. Values are either averaged or interpolated.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**TrnEventGetTable**(*Tag*, *EventNo*, *Period*, *Length*, *Table*, *TimeTable*, *Mode* [, *sClusterName*] )

*Tag*:

The trend tag enclosed in quotation marks "" (can be prefixed by the name of the cluster that is ClusterName.Tag).

*EventNo*:

The starting event number for entries in the trend table.

*Period*:

The time difference between tabulated trend values (in seconds). For example, if you set this period to 30 seconds, Plant SCADA will get the last trend value (sampled at the end of the trend section), then get the trend value that was sampled 30 seconds before that, and so on until it reaches the start time of the trend section.

If this period is different to the trend's sampling period, the trend values will be averaged or interpolated. Set to

0 (zero) to default to the actual trend period.

*Length:*

The number of trend values to store in the trend table, from 1 to the maximum number of items in the table.

*Table:*

The Cicode array in which the trend data is stored. You can enter the name of an array here (see the example). Must be a Real type variable.

*TimeTable:*

The table of integer values in which the time stamp is stored. Must be a Long type variable.

*Mode:*

The Display Mode parameters allow you to enter a single integer to specify the display options for a trend (for a maximum of eight trends).

To calculate the integer that you should enter for a particular trend, select the options you want from the list below, adding their associated numbers together. The resulting integer is the DisplayMode parameter for that trend.

**Invalid/Gated trend options:**

0 - Convert invalid/gated trend samples to zero.

1 - Leave invalid/gated trend samples as they are.

**Ordering trend sample options:**

0 - Order returned trend samples from oldest to newest.

2 - Order returned trend samples from newest to oldest.

**Condense method options:**

0 - Set the condense method to use the mean of the samples.

4 - Set the condense method to use the minimum of the samples.

8 - Set the condense method to use the maximum of the samples.

**Stretch method options:**

0 - Set the stretch method to step.

128 - Set the stretch method to use a ratio.

256 - Set the stretch method to use raw samples.

**Gap Fill Constant option:**

*n* - the number of missed samples that the user wants to gap fill) x 4096.

The options listed in each group are mutually exclusive. The default value for each Display Mode is 258 (0 + 2 + 256).

*sClusterName:*

The name of the cluster in which the trend tag resides. This is optional if you have one cluster or are resolving the trend via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

The actual number of samples read. The return value is 0 if an error is detected. You can call the IsError() function to get the actual error code. If EventNo is 0 (zero) then the EventNo will be set to the current event number.

---

**Note:** A maximum of 32,000 samples can be retrieved in a single call to TrnEventGetTable.

---

## Related Functions

[TrnEventSetTable](#), [TrnGetEvent](#), [TrnGetDisplayMode](#)

## See Also

[Trend Functions](#)

### TrnEventGetTableMS

Stores event trend data and time data (including milliseconds) for a specified trend tag. The event trend data is stored in an event table, and the time stamp in time tables. Data is stored at the specified *Period*, working backwards from the starting point specified by *EventNo*. If *Period* differs from the trend period configured in the Trend Tags database, the values to be stored are calculated from the trend data. Values are either averaged or interpolated.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**TrnEventGetTableMS(*Tag*, *EventNo*, *Period*, *Length*, *Table*, *TimeTable*, *Mode*, *MSTimeTable* [, *sClusterName* ] )**

*Tag*:

The trend tag enclosed in quotation marks "" (can be prefixed by the name of the cluster that is ClusterName.Tag).

*EventNo*:

The starting event number for entries in the trend table.

*Period*:

The time difference between tabulated trend values (in seconds). For example, if you set this period to 30 seconds, Plant SCADA will get the last trend value (sampled at the end of the trend section), then get the trend value that was sampled 30 seconds before that, and so on until it reaches the start time of the trend section.

If this period is different to the trend's sampling period, the trend values will be averaged or interpolated. Set to 0 (zero) to default to the actual trend period.

*Length*:

The number of trend values to store in the trend table, from 1 to the maximum number of items in the table.

*Table*:

The Cicode array in which the trend data is stored. You can enter the name of an array here (see the example). Must be a Real type variable.

*TimeTable*:

The table of integer values in which the time stamp is stored. Must be a Long type variable.

*Mode*:

The Display Mode parameters allow you to enter a single integer to specify the display options for a trend (for a maximum of eight trends).

To calculate the integer that you should enter for a particular trend, select the options you wish to use from those listed below, adding their associated numbers together. The resulting integer is the DisplayMode parameter for that trend.

**Invalid/Gated trend options:**

- 0 - Convert invalid/gated trend samples to zero.
- 1 - Leave invalid/gated trend samples as they are.

**Ordering trend sample options:**

- 0 - Order returned trend samples from oldest to newest.
- 2 - Order returned trend samples from newest to oldest.

**Condense method options:**

- 0 - Set the condense method to use the mean of the samples.
- 4 - Set the condense method to use the minimum of the samples.
- 8 - Set the condense method to use the maximum of the samples.

**Stretch method options:**

- 0 - Set the stretch method to step.
- 128 - Set the stretch method to use a ratio.
- 256 - Set the stretch method to use raw samples.

**Gap Fill Constant option:**

*n* - the number of missed samples that the user wants to gap fill) x 4096.

Options listed in each group are mutually exclusive. The default value for each Display Mode is 258 (0 + 2 + 256).

**MSTimeTable:**

The table of integer values in which the millisecond component of the time stamp is stored. Must be a Long type variable.

**sClusterName:**

The name of the cluster in which the trend tag resides. This is optional if you have one cluster or are resolving the trend via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

The actual number of samples read. The return value is 0 if an error is detected. You can call the IsError() function to get the actual error code.

## Related Functions

[TrnEventSetTableMS](#), [TrnGetEvent](#), [TrnEventGetTable](#)

## See Also

[Trend Functions](#)

## TrnEventSetTable

Sets event trend data and time data for a specified trend tag. The event trend data is set in an event table, and the time stamp in time tables. Data is set at the period specified, working backwards from the starting point specified by *EventNo*.

If the period of setting data differs from the trend period (defined in the Trend Tags database), the values to be set are calculated from the trend data, either averaged or interpolated. The user needs to have the correct privilege (as specified in the Trend Tag settings), otherwise the data is not written.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**TrnEventSetTable**(*Tag*, *EventNo*, *Period*, *Length*, *Table*, *TimeTable* [, *sClusterName*] )

*Tag*:

The trend tag enclosed in quotation marks "" (can be prefixed by the name of the cluster that is ClusterName.Tag).

*EventNo*:

Event Number:

When adding new events to a trend, set *EventNo* to 0.

When overwriting existing values, set *EventNo* to the last event number to be overwritten. For example, if overwriting events 100 to 110, set *EventNo* to 110.

*Period*:

This will be the interval (in seconds) between trend values when they are set (that is, it will be the perceived sampling period for the trend). This period can differ from the actual trend period. Set to 0 (zero) to default to the actual trend period.

*Length*:

The number of trend values in the trend table.

*Table*:

The table of floating-point values in which the trend data is stored. You can enter the name of an array here (see the example). Must be a Real type variable.

*TimeTable*:

The table of integer values in which the time stamp is stored. If you would like events to stay in correct time-stamp order, sort the values in this table in ascending order. When *EventNo* is non-zero the values in this table may be zero. This will result in the values of the requested events being overwritten but the time-stamps staying the same. Must be a Long type variable.

*sClusterName*:

The name of the cluster in which the trend tag resides. This is optional if you have one cluster or are resolving the trend via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

The actual number of samples written. The return value is 0 if an error is detected. You can call the [IsError\(\)](#) function to get the actual error code.

## Related Functions

[TrnEventGetTable](#)

## Example

```
REAL TrendTable1[100];
INT TimeTable[100];
/* Defines an array of a maximum of 100 entries. Assume that
TrendTable1 has been storing data from a source. */
TrnEventSetTable("OP1",nEventNo, 1,10,TrendTable1[0],
TimeTable[0], "ClusterXYZ");
/* A set of 10 trend data values are set for the OP1 trend tag. */
```

## See Also

[Trend Functions](#)

## TrnEventSetTableMS

Sets event trend data and time data (including milliseconds) for a specified trend tag. The event trend data is set in an event table, and the time stamp in time tables. Data is set at the period specified, working backwards from the starting point specified by *EventNo*.

If the period of setting data differs from the trend period (defined in the Trend Tags database), the values to be set are calculated from the trend data, either averaged or interpolated. The user needs to have the correct privilege (as specified in the Trend Tags settings), otherwise the data is not written.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**TrnEventSetTableMS(*Tag*, *EventNo*, *Period*, *Length*, *Table*, *TimeTable*, *MSTimeTable* [, *sClusterName*] )**

*Tag*:

The trend tag enclosed in quotation marks "" (can be prefixed by the name of the cluster that is ClusterName.Tag).

*EventNo*:

Event Number:

When adding new events to a trend, set EventNo to 0.

When overwriting existing values, set EventNo to the last event number to be overwritten. For example, if overwriting events 100 to 110, set EventNo to 110.

*Period*:

This will be the interval (in seconds) between trend values when they are set (that is it will be the perceived sampling period for the trend). This period can differ from the actual trend period. Set to 0 (zero) to default to the actual trend period.

*Length*:

Number of trend values in the trend table.

**Table:**

Table of floating-point values in which the trend data is stored. You can enter the name of an array here (see example). Must be a Real type variable.

**TimeTable:**

Table of integer values in which the time stamp is stored. If you would like events to stay in correct time-stamp order, sort the values in this table in ascending order. When *EventNo* is non-zero the values in this table may be zero. This will result in the values of the requested events being overwritten but the timestamps staying the same. Must be a Long type variable.

**MSTimeTable:**

The table of integer values in which the millisecond component of the time stamp is stored. Must be a Long type variable.

**sClusterName:**

The name of the cluster in which the trend tag resides. This is optional if you have one cluster or are resolving the trend via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

The actual number of samples written. The return value is 0 if an error is detected. You can call the IsError() function to get the actual error code.

## Related Functions

[TrnEventGetTable](#)

## Example

```
// Arrays for trend data
REAL garSingleValue[1];
INT ganSingleTime[1];
INT ganSingleMS[1];
// Push the data in the trend system
INT
FUNCTION LogTrend(STRING sTagName, REAL rValue, INT nDateTime, INT
nMS)
    INT nSamplesWritten;
    garSingleValue[0] = rValue;
    ganSingleTime[0] = nDateTime;
    ganSingleMS[0] = nMS;
    nSamplesWritten = TrnEventSetTableMS
(sTagName, 0, 0, 1, garSingleValue[0], ganSingleTime[0], ganSingleMS[0], "ClusterXYZ");
    RETURN nSamplesWritten;
END
```

## See Also

[Trend Functions](#)

## TrnExportClip

Exports trend data to the Windows Clipboard. The data is set at the specified *Time* and *Period*, and listed from earliest to latest. Any gated or invalid data is written as 0.0.

Data is stored as a grid, with each row time-stamped. The first column/field is the date, followed by the time, followed by the tags 1 to 8.

You can use the *ClipMode* argument to make the output more useful. For example, to paste the data into Excel, use *ClipMode* 2 for CSV format. If you use *ClipMode* 1 or 3, the default paste menu option causes data to be pasted into the user's spreadsheet as text. If you use *ClipMode* 3, use the Paste Special option to paste the required format. Please be aware that not all packages support multiple clipboard formats in this way.

## Syntax

**TrnExportClip**(*Time*, *Period*, *Length*, *Mode*, *ClipMode*, *sTag1* ... *sTag8*, *iDisplayMode1* ... *iDisplayMode 8*)

*Time*:

The starting time for the data being exported.

*Period*:

The period (in seconds) of the entries being exported. (This period can differ from the actual trend period.)

*Length*:

The length of the data table, that is The number of rows of samples to be exported. for example if you put the length as 12, and you declare two tags to be exported, you get a grid with 12 rows of samples. Each row has values for each of the two tags making a total of 24 samples.

*Mode*:

The format mode to be used:

### Periodic trends

- 1 - Export the Date and Time, followed by the tags.
- 2 - Export the Time only, followed by the tags.
- 4 - Ignore any invalid or gated values. (Only supported for periodic trends.)
- 8 - The time returned will have millisecond accuracy.

### Event trends

- 1 - Export the Time, Date and Event Number, followed by the tags.
- 2 - Export the Time and Event Number, followed by the tags.
- 8 - The time returned will have millisecond accuracy.

*ClipMode*:

The format for the data being exported.

- 1 - Text
- 2 - CSV

You can add these modes for a combination of formats.

*sTag1* ... *sTag8*:

The trend tag names for the data being exported.

*iDisplayMode1* ... *iDisplayMode8*:

The Display Mode parameters allow you to enter a single integer to specify the display options for a trend (for a maximum of eight trends).

To calculate the integer that you should enter for a particular trend, select the options you wish to use from those listed below, adding their associated numbers together. The resulting integer is the `DisplayMode` parameter for that trend.

By default, this argument is set to 3 (see the details for options 1 and 2 below).

**Invalid/Gated trend options:**

0 - Convert invalid/gated trend samples to zero.

1 - Leave invalid/gated trend samples as they are.

Invalid and gated samples that are not converted to zero will appear in the destination file as the string "na" (for invalid) or "gated".

**Ordering trend sample options:**

0 - Order returned trend samples from newest to oldest.

2 - Order returned trend samples from oldest to newest.

**Condense method options:**

0 - Set the condense method to use the mean of the samples.

4 - Set the condense method to use the minimum of the samples.

8 - Set the condense method to use the maximum of the samples.

12 - Set the condense method to use the newest of the samples.

**Stretch method options:**

0 - Set the stretch method to step.

128 - Set the stretch method to use a ratio.

256 - Set the stretch method to use raw samples.

**Gap Fill Constant option:**

*n* - the number of missed samples that the user wants to gap fill) x 4096.

**Display as Periodic options:**

0 - Display according to trend type.

1048576 - Display as periodic regardless of trend type.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

---

**Note:** If using this function to export trends by using event numbers, you need to specify a valid event number in the `Time` argument, rather than a time.

## Related Functions

[ClipSetMode](#), [TrnExportCSV](#)

## Example

```
TrnExportClip(TimeCurrent(), 2, 60 * 60/2, 2, 3, "Feed", "Weight");
/* Export the last hour of data from the trend tags Feed and Weight to the clipboard in
both Text and CSV formats. Be aware that the 60 * 60/2 is a decomposed way of writing 1800,
which is the number of 2 second samples in 1 hour. */
```

## See Also

[Trend Functions](#)

### TrnExportCSV

Exports trend data to a file in CSV (Comma Separated Variable) format. The data is set at the specified *Time* and *Period*, and listed from earliest to latest. Any gated or invalid data is written as 0.0.

Data is stored as a grid, with each row time-stamped. The first column/field is the date, followed by the time, followed by the tags 1 to 8.

You can view the CSV file with a text editor, and import the file directly into other packages such as Excel for data analysis and presentation.

If you are using this function to export trends by using event numbers, you need to specify a valid event number in the Time argument, rather than a time.

## Syntax

**TrnExportCSV(Filename, Time, Period, Length, Mode, sTag1 ... sTag8, iDisplayMode1 ... iDisplayMode 8)**

*Filename:*

The name of the destination path and file.

*Time:*

The starting time for the data being exported.

*Period:*

The period (in seconds) of the entries being exported. (This period can differ from the actual trend period.)

*Length:*

The length of the data table, that is, The number of rows of samples to be exported. for example if you put the length as 12, and you declare two tags to be exported, you get a grid with 12 rows of samples. Each row has values for each of the two tags making a total of 24 samples.

*Mode:*

The format mode to be used:

#### Periodic trends

1 - Export the Date and Time, followed by the tags.

2 - Export the Time only, followed by the tags.

4 - Ignore any invalid or gated values. (This mode is only supported for periodic trends.)

8 - The time returned will have millisecond accuracy.

#### Event trends

1 - Export the Time, Date and Event Number, followed by the tags.

2 - Export the Time and Event Number, followed by the tags.

8 - The time returned will have millisecond accuracy.

*sTag1 ... sTag8:*

The trend tag names for the data being exported.

*iDisplayMode1 ... iDisplayMode8:*

The Display Mode parameters allow you to enter a single integer to specify the display options for a trend (for a maximum of eight trends).

To calculate the integer that you should enter for a particular trend, select the options you wish to use from those listed below, adding their associated numbers together. The resulting integer is the DisplayMode parameter for that trend. By default, this argument is set to 3 (see the details for options 1 and 2 below).

#### **Invalid/Gated trend options:**

0 - Convert invalid/gated trend samples to zero.

1 - Leave invalid/gated trend samples as they are.

Invalid and gated samples that are not converted to zero will appear in the destination file as the string "na" (for invalid) or "gated".

#### **Ordering trend sample options:**

0 - Order returned trend samples from newest to oldest.

2 - Order returned trend samples from oldest to newest.

#### **Condense method options:**

0 - Set the condense method to use the mean of the samples.

4 - Set the condense method to use the minimum of the samples.

8 - Set the condense method to use the maximum of the samples.

12 - Set the condense method to use the newest of the samples.

#### **Stretch method options:**

0 - Set the stretch method to step.

128 - Set the stretch method to use a ratio.

256 - Set the stretch method to use raw samples.

#### **Gap Fill Constant option:**

n - the number of missed samples that the user wants to gap fill) x 4096.

Options listed in each group are mutually exclusive.

## **Return Value**

0 (zero) if successful, otherwise an error code is returned.

## **Related Functions**

[TrnExportDBF](#), [TrnPrint](#)

## Example

```
TrnExportCSV("c:\TrnData.CSV", TimeCurrent(), 2, 60 * 60/2, 2,  
"Feed", "Weight");  
/* Export the last hour of data from the trend tags Feed and  
Weight.  
The 60 * 60/2 is a decomposed way of writing 1800, which is the  
number of 2 second samples in 1 hour. */
```

## See Also

[Trend Functions](#)

### TrnExportDBF

Exports trend data to a file in dBASE III format. The data is set at the specified *Time* and *Period*, and listed from earliest to latest. Any gated or invalid data is written as 0.0.

Data is stored as a grid, with each row time-stamped. The first column/field is the date, followed by the time, followed by the tags 1 to 8.

You can import the DBF file directly into other packages such as Excel, for data analysis and presentation.

## Syntax

**TrnExportDBF(*Filename*, *Time*, *Period*, *Length*, *Mode*, *sTag1* ... *sTag8*, *iDisplayMode1* ... *iDisplayMode 8*)**

*Filename*:

The name of the destination path and file.

*Time*:

The starting time for the data being exported.

*Period*:

The period (in seconds) of the entries being exported. (This period can differ from the actual trend period.)

*Length*:

The length of the data table, that is The number of rows of samples to be exported. for example if you put the length as 12, and you declare two tags to be exported, you get a grid with 12 rows of samples. Each row has values for each of the two tags making a total of 24 samples.

*Mode*:

The format mode to be used:

#### Periodic trends

1 - Export the Date and Time, followed by the tags.

2 - Export the Time only, followed by the tags.

4 - Ignore any invalid or gated values. (This mode is only supported for periodic trends.)

8 - The time returned will have millisecond accuracy.

#### Event trends

1 - Export the Time, Date and Event Number, followed by the tags.

2 - Export the Time and Event Number, followed by the tags.

8 - The time returned will have millisecond accuracy.

*sTag1 ... sTag8:*

The trend tag names for the data being exported. Tag names longer than 10 characters will be truncated, as the standard DBF field format is 10 characters.

*iDisplayMode1 ... iDisplayMode8:*

The Display Mode parameters allow you to enter a single integer to specify the display options for a trend (for a maximum of eight trends).

To calculate the integer that you should enter for a particular trend, select the options you wish to use from those listed below, adding their associated numbers together. The resulting integer is the DisplayMode parameter for that trend. By default, this argument is set to 3 (see the details for options 1 and 2 below).

**Invalid/Gated trend options:**

0 - Convert invalid/gated trend samples to zero.

1 - Leave invalid/gated trend samples as they are.

Invalid and gated samples that are not converted to zero will appear in the destination file as the string "na" (for invalid) or "gated".

**Ordering trend sample options:**

0 - Order returned trend samples from newest to oldest.

2 - Order returned trend samples from oldest to newest.

**Condense method options:**

0 - Set the condense method to use the mean of the samples.

4 - Set the condense method to use the minimum of the samples.

8 - Set the condense method to use the maximum of the samples.

12 - Set the condense method to use the newest of the samples.

**Stretch method options:**

0 - Set the stretch method to step.

128 - Set the stretch method to use a ratio.

256 - Set the stretch method to use raw samples.

**Gap Fill Constant option:**

n - the number of missed samples that the user wants to gap fill) x 4096.

Options listed in each group are mutually exclusive.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnExportCSV](#), [TrnPrint](#)

## Example

```
TrnExportDBF("c:\TrnData.DBF", TimeCurrent(), 2, 60 * 60/2, 2,  
"Feed", "Weight");  
/* Export the last hour of data from the trend tags Feed and  
Weight. Be aware that the 60 * 60/2 is a decomposed way of writing  
1800, which is the number of 2 second samples in 1 hour. */
```

## See Also

[Trend Functions](#)

### TrnExportDDE

Exports trend data via DDE. The data is set at the specified Time and Period, and listed from earliest to latest. Any gated or invalid data is written as 0.0. Data is stored as a grid, with each row time-stamped. The first column/field is the date, followed by the time, followed by the tags 1 to 8.

You can use the DDEMMode argument to make the output more useful. For example; to paste the data into Excel, use DDEMMode 2 for CSV format. If you use DDEMMode 1, data will be put into the user's spreadsheet as text.

**Note:** If you are using this function to export trends by using event numbers, you need to specify a valid event number in the Time argument, rather than a time.

## Syntax

**TrnExportDDE**(*sApplication*, *sDocument*, *sTopic*, *Time*, *Period*, *Length*, *Mode*, *DDEMMode*, *sTag1* ... *sTag8*,  
*iDisplayMode1* ... *iDisplayMode8*)

*sApplication*:

The application name to export the data.

*sDocument*:

The document in the application to export the data.

*sTopic*:

The topic in the application to export the data. Be aware you may have to use a special topic format to make the data export correctly. See your application documentation for details; For example with Excel you need to specify the matrix of rows and columns as "R1C1:R8C50" depending on the size of the data.

*Filename*:

The name of the destination path and file.

*Time*:

The starting time for the data being exported.

*Period*:

The period (in seconds) of the entries being exported. (This period can differ from the actual trend period.)

*Length*:

The length of the data table, that is The number of rows of samples to be exported. for example if you put the length as 12, and you declare two tags to be exported, you get a grid with 12 rows of samples. Each row has values for each of the two tags making a total of 24 samples.

**Mode:**

The format mode to be used:

**Periodic trends**

- 1 - Export the Date and Time, followed by the tags.
- 2 - Export the Time only, followed by the tags.
- 4 - Ignore any invalid or gated values. (This mode is only supported for periodic trends.)
- 8 - The time returned will have millisecond accuracy.

**Event trends**

- 1 - Export the Time, Date and Event Number, followed by the tags.
- 2 - Export the Time and Event Number, followed by the tags.
- 8 - The time returned will have millisecond accuracy.

**DDEMode:**

The format for the data being exported. CSV format allows the application to separate the data into each individual element, however not every application will support this mode. See your applications documentation for details.

- 1 - Text (default)
- 2 - CSV

**sTag1 ... sTag8:**

The trend tag names for the data being exported. Tag names longer than 10 characters will be truncated, as the standard DBF field format is 10 characters.

**iDisplayMode1 ... iDisplayMode8:**

The Display Mode parameters allow you to enter a single integer to specify the display options for a trend (for a maximum of eight trends).

To calculate the integer that you should enter for a particular trend, select the options you wish to use from those listed below, adding their associated numbers together. The resulting integer is the DisplayMode parameter for that trend.

By default, this argument is set to 3 (see the details for options 1 and 2 below).

**Invalid/Gated trend options:**

- 0 - Convert invalid/gated trend samples to zero.
- 1 - Leave invalid/gated trend samples as they are.

Invalid and gated samples that are not converted to zero will appear in the destination file as the string "na" (for invalid) or "gated".

**Ordering trend sample options:**

- 0 - Order returned trend samples from newest to oldest.
- 2 - Order returned trend samples from oldest to newest.

**Condense method options:**

- 0 - Set the condense method to use the mean of the samples.
- 4 - Set the condense method to use the minimum of the samples.
- 8 - Set the condense method to use the maximum of the samples.
- 12 - Set the condense method to use the newest of the samples.

**Stretch method options:**

- 0 - Set the stretch method to step.
- 128 - Set the stretch method to use a ratio.
- 256 - Set the stretch method to use raw samples.

**Gap Fill Constant option:**

- $n$  - the number of missed samples that the user wants to gap fill)  $\times 4096$ .
- Options listed in each group are mutually exclusive.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnExportCSV](#), [TrnExportClip](#), [TrnExportDBF](#)

## Example

```
TrnExportDDE("Excel", "data.xls", "R1C1:R61C4", TimeCurrent(), 1,  
60, 2, 2, "Feed", "Weight");  
/* Export the last 60 seconds of data from the trend tags Feed and  
Weight into Excel at R1C1:R61C4 in CSV formats */
```

## See Also

[Trend Functions](#)

## TrnFlush

Writes acquired trend data to disk without waiting for the trend buffer to be filled. Plant SCADA normally buffers the trend data in memory and only writes to disk when required, to give optimum performance. Because this function reduces the performance of the Trends Server, use it only when necessary.

## Syntax

**TrnFlush(*sName* [, *sClusterName*] )**

*sName*:

The name of the logging tag (can be prefixed by the name of the cluster that is ClusterName.Tag). Set to "\*" to flush all trend data.

*sClusterName*:

The name of the cluster in which the trend tag resides. This is optional if you have one cluster or are resolving the trend via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnSetTable](#)

## Example

```
TrnFlush("Trend1", "ClusterXYZ");
! Forces the Trend1 data to be written to disk.
```

## See Also

[Trend Functions](#)

## TrnGetBufEvent

Gets the event number of a trend at an offset for a specified pen. This function only operates on an event-based trend.

## Syntax

**TrnGetBufEvent(*nAN*, *Pen*, *Offset*)**

*nAN*:

The AN where the trend is located.

*Pen*:

The trend pen number:

0 - The pen currently in focus

1...8 - Pen1...Pen8

*Offset*:

The trend buffer offset, in samples. The number of samples can range from 0 to the maximum number of samples that can display on the trend - 1.

## Return Value

The event number. If *Offset* is not within boundaries, 0 (zero) is returned. If *AN* or *Pen* is invalid, 0 (zero) is returned and an error code is set.

## Related Functions

[TrnGetEvent](#), [TrnSetEvent](#), [TrnGetCursorEvent](#)

## Example

```
! For the trend at AN20
DspText(31,0,TrnGetBufEvent(20,0,10));
/* Displays the trend event at offset 10 for the pen currently in
focus. The event will display at AN31. */
```

## See Also

[Trend Functions](#)

### TrnGetBufTime

Gets the time and date of a trend at an offset for a specified pen. The *Offset* should be a value between 0 (zero) and the number of samples displayed on the trend.

## Syntax

**TrnGetBufTime(*nAN*, *Pen*, *Offset*)**

*nAN*:

The AN where the trend is located.

*Pen*:

The trend pen number:

0 - The pen currently in focus

1...8 - Pen1...Pen8

*Offset*:

The trend buffer offset, in samples. The number of samples can range from 0 to the maximum number of samples that can display on the trend - 1.

## Return Value

A time/date variable. If *Offset* is not within boundaries, 0 (zero) is returned. If *nAN* or *Pen* is invalid, 0(zero) is returned and an error code is set.

## Related Functions

[TrnGetCursorTime](#)

## Example

```
! For the trend at AN20
INT time;
time = TrnGetBufTime(20,0,10);
IF time <> 0 THEN
    DspText(31,0,TimeToStr(time,2));
```

```
END
/* Displays the trend date at offset 10 for the pen currently in
focus. The time will display at AN31. */
```

## See Also

[Trend Functions](#)

### TrnGetBufValue

Gets the value of a trend at an offset for a specified pen. The offset should be a value between -1 and the number of samples displayed on the trend.

## Syntax

**TrnGetBufValue(*nAN*, *Pen*, *Offset*)**

*nAN*:

The AN where the trend is located.

*Pen*:

The trend pen number:

0 - The pen currently in focus

1...8 - Pen1...Pen8

*Offset*:

The trend buffer offset, in samples. The number of samples can range from -1 to the maximum number of samples that can display on the trend minus 1.

-1 means get the last valid value in the display (provided it is less than 1.5 sample periods old).

If there is no invalid or gated sample within the last 1.5 sample periods, it is assumed that a sample has been missed and an invalid trend value is returned. See the [TrnIsHeaderValue](#) function.

## Return Value

The trend value. If the actual value is gated or invalid, the standard invalid or gated values are returned (no error is set). You can check this return value using [TrnIsHeaderValue\(\)](#).

## Related Functions

[TrnGetCursorValue](#), [TrnIsHeaderValue](#)

## Example

```
! For the trend at AN20
DspText(31,0,TrnGetBufValue(20,0,10):###.#);
/* Displays the trend value at offset 10 for the pen currently in
focus. */
```

## See Also

[Trend Functions](#)

### TrnGetCluster

Gets the cluster name of a trend graph on a page.

## Syntax

**TrnGetCluster(*nAN*)**

*nAN*:

The AN of the chosen trend graph.

## Return Value

The name of the cluster the trend graph is associated with.

## See Also

[Trend Functions](#)

### TrnGetCursorEvent

Gets the event number of a trend, at the trend cursor position for a specified pen. This function only operates on an event-based trend.

## Syntax

**TrnGetCursorEvent(*nAN*, *Pen*)**

*nAN*:

The AN where the trend is located.

*Pen*:

The trend pen number:

0 - The pen currently in focus

1...8 - Pen1...Pen8

## Return Value

The event number, or 0 (zero) if the trend cursor is disabled.

## Related Functions

[TrnSetEvent](#), [TrnGetCluster](#), [TrnGetEvent](#)

## Example

```
! For the trend at AN20
DspText(31,0,TrnGetCursorEvent(20,0));
/* Displays the trend event at the cursor for the pen currently in
focus. The event will display at AN31. */
```

## See Also

[Trend Functions](#)

## TrnGetCursorMSTime

Gets the time (in milliseconds from the previous midnight) at a trend cursor for a specified pen.

## Syntax

**TrnGetCursorMSTime(*nAN*, *Pen*)**

*nAN*:

The AN where the trend is located.

*Pen*:

The trend pen number:

0 - The pen currently in focus

1...8 - Pen1..Pen8

## Return Value

The number of milliseconds since the previous midnight. If the trend cursor is disabled, 0 (zero) is returned. If *nAN* or *Pen* is invalid, 0 (zero) is returned and an error code is set.

## Related Functions

[TrnGetCursorTime](#)

## Example

```
! For the trend at AN20
STRING timeStr;
STRING msecStr;
timeStr = TimeToString(TrnGetCursorTime(20,1),2) + " ";
msecStr = TimeToString(TrnGetCursorMSTime(20,1),6);
```

```
DspText(31,0,timeStr + msecStr);
! Returns "23/02/01 10:53:22.717"
```

## See Also

[Trend Functions](#)

### TrnGetCursorPos

Gets the offset of a trend cursor from its origin, in samples.

## Syntax

**TrnGetCursorPos(*nAN*)**

*nAN*:

The AN where the trend is located.

## Return Value

The offset of a trend cursor from its origin, in samples, or -1 if the trend cursor is disabled.

## Related Functions

[TrnSetCursorPos](#)

## Example

```
! For the trend at AN20
! If the trend cursor is disabled
Offset=TrnGetCursorPos(20);
! Sets Offset to -1.
! If the trend cursor is 50 samples from the origin
Offset=TrnGetCursorPos(20);
! Sets Offset to 50.
```

## See Also

[Trend Functions](#)

### TrnGetCursorTime

Gets the time and date at a trend cursor for a specified pen.

## Syntax

**TrnGetCursorTime(*nAN, Pen*)**

*nAN:*

The AN where the trend is located.

*Pen:*

The trend pen number:

0 - The pen currently in focus

1...8 - Pen1 . . . Pen8

## Return Value

A time/date variable. If the trend cursor is disabled, 0 (zero) is returned. If *nAN* or *Pen* is invalid, 0 (zero) is returned and an error code is set.

## Related Functions

[TrnGetBufTime](#)

## Example

```
! For the trend at AN20
INT time;
time = TrnGetCursorTime(20,1);
DspText(31,0,TimeToStr(time,2));
! Displays the trend cursor date for Pen1.
DspText(32,0,TimeToStr(time,1));
! Displays the trend cursor time for Pen1.
```

## See Also

[Trend Functions](#)

## TrnGetCursorValue

Gets the value at a trend cursor for a specified pen.

## Syntax

**TrnGetCursorValue(*nAN*, *Pen*)**

*nAN:*

The AN where the trend is located.

*Pen:*

The trend pen number:

0 - The pen currently in focus

1...8 - Pen1 . . . Pen8

## Return Value

The trend value. If the actual value is gated or invalid, the standard invalid or gated values are returned (no error is set). You can check this return value using [TrnIsValidValue\(\)](#).

## Related Functions

[TrnGetBufValue](#)

## Example

```
! For the trend at AN20
DspText(31,0,TrnGetCursorValue(20,0));
! Displays the value at the trend cursor for the focus pen.
```

## See Also

[Trend Functions](#)

## TrnGetCursorValueStr

Gets the value at a trend cursor for a specified pen. The value is returned as a formatted string using the pen's format specification and (optionally) the engineering units.

## Syntax

**TrnGetCursorValueStr(*nAN*, *Pen*, *EngUnits*)**

*nAN*:

The AN where the trend is located.

*Pen*:

The trend pen number:

0 - The pen currently in focus

1...8 - Pen1...Pen8

*EngUnits*:

Engineering units mode:

0 - Do not include the engineering units at the end of the formatted string.

1 - Include the engineering units at the end of the formatted string.

## Return Value

The trend value (as a string). If trend data is invalid, or an argument passed to the function is invalid "<na>" is returned. If the actual value is gated (not triggered) "<gated>" is returned. If the trend cursor is disabled, an empty string is returned.

## Related Functions

[TrnGetCursorValue](#)

## Example

```
! For the trend at AN20
DspText(31,0,TrnGetCursorValueStr(20,0,1));
/* Displays the value at the trend cursor for the focus pen. The
value will display as a formatted string (including the
engineering units).*/
```

## See Also

[Trend Functions](#)

## TrnGetDefScale

Gets the default engineering zero and full scales of a trend tag.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

## Syntax

**TrnGetDefScale(*Tag*, *LoScale*, *HiScale* [, *sClusterName*] )**

*Tag*:

The trend tag enclosed in quotation marks "" (can be prefixed by the name of the cluster that is "ClusterName.Tag").

*LoScale*:

The engineering zero scale. Must be a Real type variable.

*HiScale*:

The engineering full scale. Must be a Real type variable.

*sClusterName*:

The name of the cluster in which the trend tag resides. This is optional if you have one cluster or are resolving the trend via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnGetScale](#), [TrnInfo](#)

## Example

```
REAL LoScale;  
REAL HiScale;  
TrnGetDefScale("PV1",LoScale,HiScale,"ClusterXYZ");  
/* Returns engineering zero and full scales of the trend tag  
"PV1". */
```

## See Also

[Trend Functions](#)

### TrnGetDisplayMode

Returns the display mode of the selected trend pen. The display mode is set using [TrnSetDisplayMode](#).

## Syntax

**TrnGetDisplayMode(*nAN*, *PenNumber*)**

*nAN*:

The AN of the chosen trend.

*PenNumber*:

The trend pen number:

0 - The pen currently in focus.

1...8 - Pen1...Pen8.

## Return Value

An integer representing the trend's display mode:

**Invalid/Gated trend options:**

0 - Convert invalid/gated trend samples to zero.

1 - Leave invalid/gated trend samples as they are.

**Ordering trend sample options:**

0 - Order returned trend samples from oldest to newest.

2 - Order returned trend samples from newest to oldest.

**Condense method options:**

0 - Set the condense method to use the mean of the samples.

4 - Set the condense method to use the minimum of the samples.

8 - Set the condense method to use the maximum of the samples.

12 - Set the condense method to use the newest of the samples.

**Stretch method options:**

0 - Set the stretch method to step.

128 - Set the stretch method to use a ratio.

256 - Set the stretch method to use raw samples.

**Gap Fill Constant option:**

*n* - the number of missed samples that the user wants to gap fill) x 4096.

**Display as Periodic options:**

0 - Display according to trend type.

1048576 - display as periodic regardless of trend type.

## Related Functions

[TrnSetDisplayMode](#)

## Example

```
int DisplayMode = TrnGetDisplayMode (10, 7)
/* Returns The Display Mode of pen 7 for the trend at AN 10.*/
```

## See Also

[Trend Functions](#)

## TrnGetEvent

Gets the event number of the trend at a percentage along the trend, using the current event as the base point. This function only operates on an event-based trend. The first recorded event (the start event) would be event number 1 and the highest number would be the latest event. The event number is stored in a LONG and would eventually wrap around if you have enough events.

## Syntax

**TrnGetEvent(*nAN*, *Pen*, *Percent*)**

*nAN*:

The AN of the chosen trend.

*Pen*:

The trend pen number:

0 - The pen currently in focus

1...8 - Pen1..Pen8

*Percent*:

The percentage of the trend from the starting event, from 0 (the start event) to 100 (the end event).

## Return Value

The event number.

## Related Functions

[TrnSetEvent](#), [TrnGetCluster](#), [TrnGetCursorEvent](#)

## Example

```
/* Display the start event for the current pen of the trend at
AN20. */
DspText(31,0,TrnGetEvent(20,0,0));
```

## See Also

[Trend Functions](#)

## TrnGetFormat

Gets the format of a trend tag being plotted by a specified pen.

## Syntax

**TrnGetFormat**(*nAN*, *Pen*, *Width*, *DecPlaces*)

*nAN*:

The AN of the chosen trend.

*Pen*:

The trend pen number:

0 - The pen currently in focus

1...8 - Pen1...Pen8

*Width*:

Variable denoting the width of the format.

*DecPlaces*:

Variable denoting the number of decimal places in the format.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnGetScale](#), [TrnGetUnits](#)

## Example

```
/* If the trend tag being plotted by Pen1 of the trend at AN20 has
a format of "###.#" */
```

```
TrnGetFormat(20,1,Width,DecPlaces);
! Sets Width to 5 and DecPlaces to 1.
```

## See Also

[Trend Functions](#)

### TrnGetGatedValue

Returns the internally stored value for <GATED>. If the internally stored value changes in the future, you will not need to modify your Cicode, as this function will return the correct value.

## Syntax

```
TrnGetGatedValue()
```

## Return Value

The internally stored value for <GATED>.

## Related Functions

[TrnGetInvalidValue](#), [TrnIsValidValue](#)

## Example

```
REAL MyTrendValue;
IF MyTrendValue = TrnGetGatedValue() THEN
    Prompt ("This value is <GATED>")
ELSE
    IF MyTrendValue = TrnGetInvalidValue() THEN
        Prompt("This value is <TRN_NO_VALUES>")
    ELSE
        Prompt("Trend value is = " + RealToStr(MyTrendValue, 10, 1));
    END
END
```

## See Also

[Trend Functions](#)

### TrnGetInvalidValue

Returns the internally stored value for <INVALID>. If the internally stored value changes in the future, you will not need to modify your Cicode, as this function will return the correct value.

## Syntax

```
TrnGetInvalidValue()
```

## Return Value

The internally stored value for <INVALID>.

## Related Functions

[TrnGetGatedValue](#), [TrnIsHeaderValue](#)

## Example

```
REAL newArray[100];
REAL oldArray[90];
INT trigger;
INT
FUNCTION
DoubleArray()
    INT i;
    FOR i = 0 TO 99 DO
        IF TrnIsValidValue(oldArray[i]) = 1 OR trigger = 0 THEN
            newArray[i] = TrnGetGatedValue();
        ELSE
            IF i >= 90 OR TrnIsValidValue(oldArray[i]) = 2 THEN
                newArray[i] = TrnGetHeaderValue();
            ELSE
                newArray[i] = oldArray[i] * 2;
            END
        END
    END
    RETURN i;
END
```

## See Also

[Trend Functions](#)

## TrnGetMode

Gets the mode (real-time or historical trending) of the trend pen.

## Syntax

```
TrnGetMode(nAN, Pen)
```

*nAN*:

The AN of the chosen trend.

*Pen:*

The trend pen number:

0 - The pen currently in focus

1...8 - Pen1...Pen8

## Return Value

The current mode, 0 for real-time or 1 for historical.

## Related Functions

[TrnScroll](#)

## Example

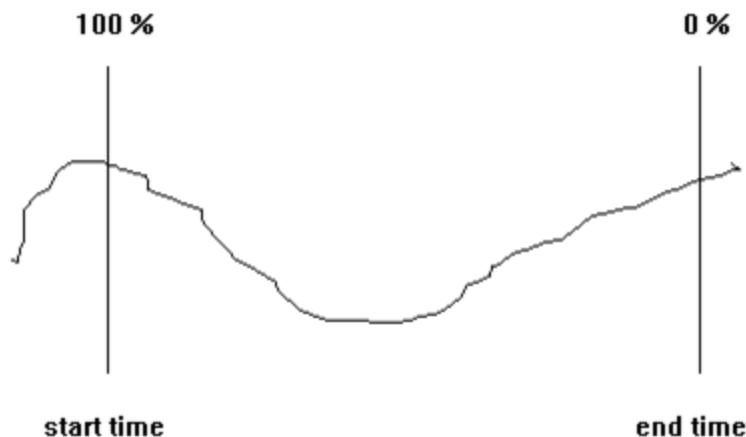
```
! For the trend at AN20
INT Mode;
Mode=TrnGetMode(20,0);
! Gets the current mode of the pen in focus.
IF Mode=0 THEN
    DspText(31,0,"Real Time Trending");
ELSE
    DspText(31,0,"Historical Trending");
END
```

## See Also

[Trend Functions](#)

## TrnGetMSTime

Gets the time (in milliseconds from the previous midnight) of the trend (plotted by a specified pen) at a percentage along the trend, using the time and date of the right-most sample displayed. The time associated with the right-most sample displayed is known as the end time. The start time is the time of the left-most sample displayed. Percent 0 (zero) will correspond to the end time, and Percent 100 will correspond to the start time



## Syntax

**TrnGetMSTime(*nAN*, *Pen*, *Percent*)**

*nAN*:

The AN of the chosen trend.

*Pen*:

The trend pen number:

0 - The pen currently in focus

1...8 - Pen1...Pen8

*Percent*:

The percentage of the trend from the time and date of the right-most sample displayed (end time), from 0 to 100.

## Return Value

The number of milliseconds since the previous midnight. Zero (0) is returned if an error is detected.

## Related Functions

[TrnGetTime](#)

## Example

```
! For Pen 1 at AN20
STRING timeStr;
STRING msecStr;
timeStr = TimeToString(TrnGetTime(20,1,100),2) + " ";
msecStr = TimeToString(TrnGetMSTime(20,1,100),6);
DspText(31,0,timeStr + msecStr);
```

Returns:

```
"23/02/01 10:53:22.717"
```

## See Also

[Trend Functions](#)

### TrnGetPen

Gets the trend tag being plotted by a specified pen.

## Syntax

**TrnGetPen(*nAN*, *Pen* [, *Mode*] )**

*nAN*:

The AN of the chosen trend.

*Pen*:

The trend pen number:

0 - The pen currently in focus

1...8 - Pen1 . . . Pen8

*Mode*:

An optional argument used to specify whether the trend tag name is returned with a cluster prefix. Set:

0 return tag without cluster prefix (default)

1 return tag with cluster prefix.

## Return Value

The trend tag (as a string) being plotted by *Pen*. If *nAN* or *Pen* is invalid, an empty string is returned, and an error code is set. You can call the *IsError()* function to get the actual error code.

## Related Functions

[TrnSetPen](#)

## Example

```
! For the trend at AN20
DspText(31,0,TrnGetPen(20,0,1));
! Displays the trend tag with cluster prefix of the focus pen.
```

## See Also

[Trend Functions](#)

## TrnGetPenComment

Retrieves the comment of a pen.

### Syntax

**TrnGetPenComment(*nAN*, *Pen*)**

*AN*

The AN of the chosen trend.

*Pen*:

The trend pen number:

0 - The pen currently in focus

1...8 - Pen1...Pen8

### Return Value

The comment of the pen as a string.

### Related Functions

[TrnGetPen](#)

### Example

```
! For the trend at AN18
DspText(31,0,TrnGetPenComment(18,0));
! Displays the trend comment of the focus pen.
```

### See Also

[Trend Functions](#)

## TrnGetPenFocus

Gets the number of the pen currently in focus.

### Syntax

**TrnGetPenFocus(*nAN*)**

*nAN*:

The AN of the chosen trend.

## Return Value

The pen currently in focus (between 1 and 8). If *nAN* is invalid, -1 is returned and an error code is set.

## Related Functions

[TrnSetPenFocus](#)

## Example

```
! For the trend at AN20
DspText(31,0,TrnGetPenFocus(20));
! Displays the pen currently in focus.
```

## See Also

[Trend Functions](#)

## TrnGetPenNo

Gets the pen number of a pen name. The pens on a trend are either defined in the Page Trends database or set by the [TrnSetPen\(\)](#) function.

## Syntax

**TrnGetPenNo(*nAN*, *Tag*)**

*AN*:

The AN of the chosen trend.

*Tag*:

The trend tag.

## Return Value

The pen number, or 0 (zero) if an error is detected.

## Related Functions

[TrnSetPen](#)

## Example

```
/* Assume that 8 trend fonts, Pen1TrendFont ... Pen8TrendFont are
defined in the Fonts database. The following code will display the
trend tag using the matching font for that pen. */
! For the trend at AN20
```

```
STRING sFont;
INT iPen;
iPen = TrnGetPenNo(20,"PV1");
IF 0 < iPen AND iPen < 9 THEN
    sFont = "Pen" + IntToStr(iPen) + "TrendFont";
    DspStr(31,sFont,"PV1");
END
```

## See Also

[Trend Functions](#)

### TrnGetPeriod

Gets the current display period of a trend. (To obtain the sampling period, use the [TrnInfo](#) function.)

## Syntax

**TrnGetPeriod(*nAN*)**

*nAN*:

The AN of the chosen trend.

## Return Value

The current display period of a trend (in seconds), or 0 (zero) if an error code is detected.

## Related Functions

[TrnSetPeriod](#), [TrnInfo](#)

## Example

```
/* For the trend at AN20, get and display the current display
period. */
! If the period is 10 seconds
INT Period;
STRING Str;
Period=TrnGetPeriod(20);
Str=TimeToStr(Period,5);
DspStr(31,"",Str);
```

## See Also

[Trend Functions](#)

## TrnGetScale

Gets the display scale of the trend tag being plotted by a specified pen.

## Syntax

**TrnGetScale(*nAN*, *Pen*, *Percent*)**

*AN*:

The AN of the chosen trend.

*Pen*:

The trend pen number:

0 - The pen currently in focus

1...8 - Pen1 . . . Pen8

*Percent*:

The percentage of the full scale, from 0 to 100.

## Return Value

The scale of the trend tag being plotted by *Pen*. If *nAN* or *Pen* is invalid, 0 (zero) is returned and an error code is set.

## Related Functions

[TrnSetScale](#), [TrnGetDefScale](#)

## Example

```
! For the trend at AN20
DspText(31,0,TrnGetScale(20,0,0));
! Displays the zero scale of the focus pen.
DspText(32,0,TrnGetScale(20,0,50));
! Displays the 50% scale of the focus pen.
DspText(33,0,TrnGetScale(20,0,100));
! Displays the full scale of the focus pen.
```

## See Also

[Trend Functions](#)

## TrnGetScaleStr

Gets the scale of the trend tag being plotted by a specified pen. The value is returned as a formatted string using the pen's format specification and (optionally) the engineering units.

## Syntax

**TrnGetScaleStr(*nAN*, *Pen*, *Percent*, *EngUnits*)**

*AN*:

The AN of the chosen trend.

*Pen*:

The trend pen number:

0 - The pen currently in focus

1...8 - Pen1..Pen8

*Percent*:

The percentage of the full scale, from 0 to 100.

*EngUnits*:

Engineering units mode:

0 - Do not include the engineering units at the end of the formatted string.

1 - Include the engineering units at the end of the formatted string.

## Return Value

The scale of the trend tag being plotted by *Pen* (as a string). If *nAN* or *Pen* is invalid, <na> is returned and an error code is set.

## Related Functions

[TrnGetScale](#)

## Example

```
! For the trend at AN20
DspText(31,0,TrnGetScaleStr(20,0,0,1));
/* Displays the zero scale of the focus pen. The scale displays as
a formatted string (including the engineering units). */
DspText(32,0,TrnGetScaleStr(20,2,50,1));
/* Displays the 50% scale of Pen2. The scale displays as a
formatted string (including the engineering units). */
DspText(33,0,TrnGetScaleStr(20,0,100,0));
/* Displays the full scale of the trend tag being plotted by the
focus pen. The scale displays as a formatted string (excluding the
engineering units). */
```

## See Also

[Trend Functions](#)

## TrnGetSpan

Gets the span time of a trend (if the span was set by the [TrnSetSpan\(\)](#) function). The span time is the total time displayed in the trend window.

**Note:** If you call the [TrnSetPeriod\(\)](#) function after the [TrnSetSpan\(\)](#) function, the span is automatically set to 0 (zero).

## Syntax

**TrnGetSpan(*nAN*)**

*nAN*:

The AN of the chosen trend.

## Return Value

The span time, in seconds. 0(zero) is returned if the AN is invalid or if the span was not set by the [TrnSetSpan\(\)](#) function.

## Related Functions

[TrnSetSpan](#), [TrnGetPeriod](#), [TrnSetPeriod](#)

## Example

```
// Use a keyboard command or button to set a span of 2 hours.  
TrnSetSpan(40,StrToTime("2:00:00"));  
// Then use TrnGetSpan function to display the span  
Time = TrnGetSpan(40)  
DspText(31,0,TimeToStr(Time,5));
```

## See Also

[Trend Functions](#)

## TrnGetTable

This function allows you to tabulate values from a specific section of trend. The values in the table (possibly an array variable) are arranged by time.

If the period (*Period*) is different to the trend's sampling period (configured in the Trend Tags database), the returned values are determined by *DisplayMode*.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

## Syntax

**TrnGetTable(*Tag, Time, Period, Length, Table, DisplayMode [, Milliseconds] [, sClusterName]*)**

**Tag:**

The trend tag enclosed in quotation marks "" (can be prefixed by the name of the cluster that is ClusterName.Tag).

**Time:**

The end time and date (long integer) of the desired trend section. Once you have entered the end time and date (Time), period (Period), and number of trend tag values collected (Length), the start time and date will be calculated automatically. For example, if Time = StrToDate("21/07/20") + StrToTime("09:00"), Period = 30, and Length = 60, the start time would be 08:30. In other words, the trend values for the period between 8.30am and 9am (on July 21, 2020) would be tabulated.

If this argument is set to 0 (zero), the time used will be the current time.

**Period:**

The time difference between tabulated trend values (in seconds). For example, if you set this period to 30 seconds, Plant SCADA will get the last trend value (sampled at the end of the trend section), then get the trend value that was sampled 30 seconds before that, and so on until it reaches the start time of the trend section.

If this period is different to the trend's sampling period, the trend values will be averaged or interpolated. Set to 0 (zero) to default to the actual trend period.

**Length:**

The number of trend values to store in the trend table, from 1 to the maximum number of items in the table. This argument has a max of 4000 (in v6). Specifying a length of greater than 4000 results in a return value of 0 and IsError()=274 (INVALID\_ARGUMENT). This limit can be configured using the citect.ini parameter [Trend]MaxRequestLength.

**Table:**

Variable containing the Cicode array in which the trend data is stored. You can enter the name of an array here (see the example).

**DisplayMode:**

The Display Mode parameters allow you to enter a single integer to specify the display options for a trend (for a maximum of eight trends).

To calculate the integer that you should enter for a particular trend, select the options you want to use from those listed below, adding their associated numbers together. The resulting integer is the DisplayMode parameter for that trend.

**Invalid/Gated trend options:**

0 - Convert invalid/gated trend samples to zero.

1 - Leave invalid/gated trend samples as they are.

**Ordering trend sample options:**

0 - Order returned trend samples from newest to oldest.

2 - Order returned trend samples from oldest to newest.

**Condense method options:**

0 - Set the condense method to use the mean of the samples.

4 - Set the condense method to use the minimum of the samples.

8 - Set the condense method to use the maximum of the samples.

12 - Set the condense method to use the newest of the samples.

**Stretch method options:**

- 0 - Set the stretch method to step.
- 128 - Set the stretch method to use a ratio.
- 256 - Set the stretch method to use raw samples.

**Gap Fill Constant option:**

*n* - the number of missed samples that the user wants to gap fill) x 4096.

**Synchronize to Display Period options:**

- 0 - Synchronize the returned samples to the nearest display period.
- 16777216 - Do not synchronize samples to the nearest display period.

Options listed in each group are mutually exclusive. The default value for each Display Mode is 258 (0 + 2 + 256).

**Milliseconds:**

This argument allows you to set your sample request time with millisecond precision. After defining the time and date in seconds with the Time argument, you can then use this argument to define the milliseconds component of the time.

For example, if you wanted to request data from the 21/07/20, at 9am, 13 seconds, and 250 milliseconds you could set the Time and Milliseconds arguments as follows:

```
Time = StrToDate("21/07/20") + StrToTime("09:00:13")
Milliseconds = 250
```

If you don't enter a Milliseconds value, it defaults to 0 (zero). There is no range constraint, but as there are only 1000 milliseconds in a second, you should keep your entry between 0 (zero) and 999.

**sClusterName:**

Name of the cluster in which the trend tag resides. This is optional if you have one cluster or are resolving the trend via the current cluster context. The argument is enclosed in quotation marks "".

---

**Note:** If you are putting a value for "sClusterName" in the TrnGetTable(...), then you have to provide a value for "Milliseconds" parameter.

---

## Return Value

The actual number of samples read. 0(zero) is returned if an error occurs. You can call the IsError() function to get the actual error code.

## Related Functions

[TrnSetTable](#), [TrnGetDisplayMode](#)

## Example

```
REAL TrendTable1[100];
/* Defines an array of a maximum of 100 entries in which the trend
data is stored. */
TrnGetTable("OP1",StrToDate("17/07/20")
+StrToTime("09:00"),2,10,TrendTable1[0],0,0,"ClusterXYZ");
/* Stores the values of trend tag "OP1" in Table TrendTable1. Data
is stored at the following times:
17/07/20 09:00:00 TrendTable1[0]
```

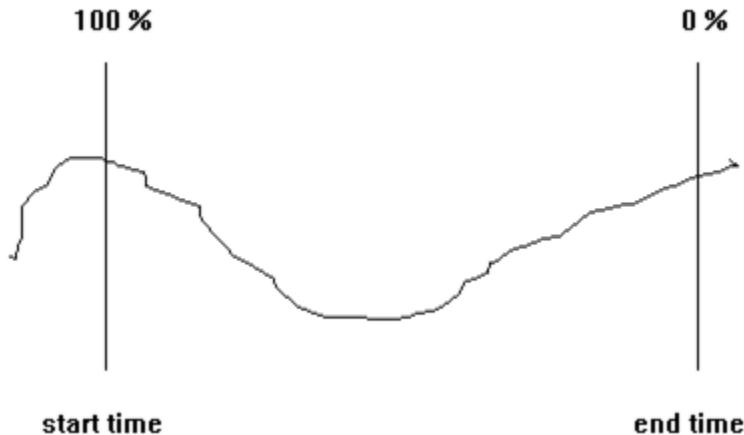
```
08:59:58 TrendTable1[1]
08:59:56 TrendTable1[2]
...
17/07/20 08:59:42 TrendTable1[9] */
Average=TableMath(TrendTable1[0],100,2);
/* Gets the average of the trend data. */
```

## See Also

[Trend Functions](#)

### TrnGetTime

Gets the time and date of the trend (plotted by a specified pen) at a percentage along the trend, using the time and date of the right-most sample displayed. The time associated with the rightmost sample displayed is known as the end time. The start time is the time of the left-most sample displayed. Percent 0 (zero) will correspond to the end time, and Percent 100 will correspond to the start time.



## Syntax

**TrnGetTime(*nAN*, *Pen*, *Percent*)**

*AN*:

The AN of the chosen trend.

*Pen*:

The trend pen number:

0 - The pen currently in focus

1..8 - Pen1..Pen8

*Percent*:

The percentage of the trend from the time and date of the right-most sample displayed (end time), from 0 to 100.

## Return Value

A time/date variable. 0 (zero) is returned if an error is detected.

## Related Functions

[TrnSetTime](#)

## Example

```
! For the trend at AN20
DspText(31,0,TimeToStr(TrnGetTime(20,0,0),2));
! Displays the trend current date for the focus pen.
DspText(32,0,TimeToStr(TrnGetTime(20,0,0),1));
! Displays the trend current time for the focus pen.
DspText(33,0,TimeToStr(TrnGetTime(20,0,50),1));
! Displays the time 50% along the trend for the focus pen.
```

## See Also

[Trend Functions](#)

## TrnGetUnits

Gets the data units for the trend tag plotted by a specified *Pen*.

## Syntax

**TrnGetUnits(*nAN*, *Pen*)**

*AN*:

The AN of the chosen trend.

*Pen*:

The trend pen number:

0 - The pen currently in focus

1...8 - Pen1...Pen8

## Return Value

The data units for the trend tag plotted by *Pen*, otherwise an empty string is returned, and an error code is set.  
You can call the [IsError\(\)](#) function to get the actual error code.

## Related Functions

[TrnGetFormat](#), [TrnGetScale](#)

## Example

```
! For the trend at AN20
DspText(31,0,TrnGetUnits(20,0));
! Displays the data units for the focus pen.
```

## See Also

[Trend Functions](#)

### TrnInfo

Gets the configured values of a trend tag.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**TrnInfo(*Tag*, *Type* [, *sClusterName*] )**

*Tag*:

The name of the trend tag enclosed in quotation marks "" (can be prefixed by the name of the cluster that is ClusterName.Tag).

*nType*:

The type of information required:

1 - Trend Type

2 - Sample Period (to obtain the Display Period, use the TrnGetPeriod function)

3 - Trend File Name (without file extension)

4 - Area

5 - Privilege

6 - Current Event Number. Valid only for event type trends

7 - Engineering Units

8 - The storage method used for the tag. A returned value of 2 represents two byte storage (scaled), 8 represents eight byte storage (floating point).

9 - The file period of the tag in seconds. If the file period is set to monthly or yearly, a file period cannot be calculated as months and years vary in length. Therefore, a file period of 0 will be returned for trends with such file periods.

*sClusterName*:

The name of the cluster in which the trend tag resides. This is optional if you have one cluster or are resolving the trend via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

The value (as a string), otherwise an empty string is returned, and an error code is set. You can call the [IsError\(\)](#) function to get the actual error code.

## Example

```
! Get the file name of trend tag LT131
sFileName = TrnInfo("LT131", 3, "ClusterXYZ");
```

## See Also

[Trend Functions](#)

### TrnIsValidValue

Determines whether a logged trend value is:

- <VALID> - an actual trend value;
- <GATED> - if a periodic trend has a trigger condition, and that condition is FALSE, a standard substitute (or GATED) value is logged instead of the actual value; or
- <INVALID> - for some reason, no value was logged.

## Syntax

**TrnIsValidValue(*TrendValue*)**

*TrendValue*:

A trend value (of type REAL).

## Return Value

0 for <VALID>  
1 for <GATED>  
2 for <INVALID>

## Related Functions

[TrnGetGatedValue](#), [TrnGetInvalidValue](#)

## Example

```
INT
FUNCTION
DoubleArray()
    INT i;
    FOR i = 0 TO 99 DO
        IF TrnIsValidValue(oldArray[i]) = 1 OR trigger = 0 THEN
            newArray[i] = TrnGetGatedValue();
            Prompt ("This value is <GATED>");
        ELSE
            IF i >= 90 OR TrnIsValidValue(oldArray[i]) = 2 THEN
```

```
        newArray[i] = TrnGetInvalidValue();
ELSE
    newArray[i] = oldArray[i] * 2;
    Prompt ("This value is <TRN_NO_VALUES>");
END
END
END
RETURN i;
END
```

## See Also

[Trend Functions](#)

## TrnNew

Creates a new trend at run time. This function performs the same operation as an entry in the Page Trends database. After the trend is created by the TrnNew() function, all the other trend functions can access and control the trend.

## Syntax

**TrnNew(*nAN*, *Trend* [, *Tag1* ... *Tag8*] [, *sClusterName*] )**

*AN*:

The AN where the bottom right-hand corner of the trend is located.

*Trend*:

The trend definition number (as a STRING).

*Tag1* ... *Tag8*:

The trend tags. (These tags cannot be prefixed with cluster name, cluster should be specified with the ClusterName argument).

*sClusterName*:

The name of the cluster in which all the trend tags reside. This is optional if you have one cluster or are resolving the trends via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnDelete](#)

## Example

```
TrnNew(20, "trn002", "PV1", "OP1", "ClusterXYZ");
/* Creates a new trend at AN20 using trend definition 2, plotting
```

```
"PV1" on Pen1 and "OP1" on Pen2. */
```

## See Also

[Trend Functions](#)

### TrnPlot

Prints the trend line of one or more trend tags. Each trend line is drawn with a different pen style and marker as appropriate. The trend plot includes a comment and a legend, and you can specify the vertical high and low scales. The *Mode* defines the color mode of the printer. The default mode is black and white.

---

**Note:** The TrnPlot() function is used only for sending trends to a printer, and cannot be used for embedding a trend plot in a report.

---

For more advanced trend plotting, you can use the low-level plot functions.

## Syntax

```
TrnPlot(sPort, nSamples, iTime, rPeriod, sTitle, AN, Tag1.....Tag8, iMode, sComment, rLoScale1, rHiScale1,  
.....rLoScale8, rHiScale8)
```

*sPort*:

The name of the printer port to which the plot will be printed. This name needs to be enclosed within quotation marks. For example LPT1:, to print to the local printer, or \\Pserver\canon1 using UNC to print to a network printer.

*nSamples*:

The number of data points on the plot.

*iTime*:

For periodic trend or event trends displayed as periodic:

The end point in time (the latest point) for the trend plot.

For event trend type:

The event sample number (e.g. using [TrnGetEvent](#))

*rPeriod*:

The period (in seconds) of the trend plot. This can differ from the actual trend period.

If you omit the period, it defaults to the sample period of Tag1.

*sTitle*:

The title of the trend plot.

*Tag1..Tag8*:

The trend tags.

*AN*:

The AN of the chosen trend. If you enter 0 (zero), the display mode will default to 258. (This is the display mode that is passed into [TrnGetTable](#)() when it is called internally by TrnPlot().) If you call TrnPlot() from a report, you need to enter 0 (zero) here.

*iMode*:

The color mode of the printer.

0 - Black and White

1 - Color

*sComment:*

The comment that is to display beneath the title of the trend plot. You may pass an empty string if no comment is required.

*rLoScale1, HiScale1,.....LoScale8, HiScale8:*

The minimum and maximum on the vertical scale for the trend line of each of the tags (Tag1... Tag8).

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnComparePlot](#), [TrnPrint](#), [PlotOpen](#), [SPCPlot](#)

## Example

```
/* Prints a black and white plot to LPT1, containing the trend
lines of two variable tags (PV1 & PV2). The trend lines have a
starting time of 9am, on 11/10/96, 200 sample points, and a period
of 2 seconds. The trend line of PV1 will be on a vertical scale of
0-200, and PV2 will be on a vertical scale of 0-400. */
INT time;
Time = StrToDate("11/10/96") + StrToTime("09:00:00");
TrnPlot("LPT1:",200,Time,2,"Citect Trend
Plot","PV1","PV2","","","","","","",0,"Process variable operation
at shutdown",0,200,0,400);
```

## See Also

[Trend Functions](#)

## TrnPrint

Prints the trend that is displayed on the screen (at *nAN*) using the current display mode for each trend. You can specify the trend title, the target printer, whether to print in color or black and white, and whether to display the Plot Setup form when the function is called.

## Syntax

**TrnPrint(*sPort*, *sTitle*, *AN*, *iModeColor*, *iDisplayForm*)**

*sPort:*

The name of the printer port to which the plot will be printed. This name needs to be enclosed within quotation marks "". For example "LPT1:", to print to the local printer, or "\\\Pserver\canon1" using UNC to print to a

network printer.

It is not necessary to enter a printer port. The first time the printer port is omitted, you will be prompted to select one at the Printer Setup form. The selection you make will then be used as the default.

*sTitle:*

The title to print at the top of the trend plot. If you omit the title in *sTitle*, the page title will be used.

*AN:*

The AN where the trend plot is located.

*iModeColor:*

The color mode of the printer.

-1 - Color to be decided (Default). Plant SCADA refers to the [General]PrinterColourMode parameter to determine print color. If there is no setting for this parameter, it will default to black and white.

0 - Black and White

1 - Color

*DisplayForm:*

Defines whether or not the Plot Setup form will display when the function is called. This form allows you to enter the color mode of the printer, and define the printer setup etc. (See Printing Trend Data for more information on this form.)

-1 - Plant SCADA refers to the [General]DisablePlotSetupForm parameter to determine if the form will display.

0 - Do not display form

1 - Display form

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnPlot](#), [TrnComparePlot](#), [WinPrint](#), [SPCPlot](#)

## Example

```
TrnPrint("LPT1:","Test Print",40,0,0);
/* Prints the trend plot displayed at AN40, without prompting for setup details.*/
```

## See Also

[Trend Functions](#)

## TrnSamplesConfigured

Gets the number of samples configured for the currently displayed trend.

## Syntax

**TrnSamplesConfigured(*nAN*)**

*AN*:

The AN where the trend is located.

## Return Value

The number of samples configured for the trend, or 0 (zero) if an error is detected. You can call the [IsError\(\)](#) function to get the actual error code.

## Example

```
/* For the trend at AN20, get and display the number of samples */  
INT nSamples;  
nSamples=TrnSamplesConfigured(20);  
DspStr(31,"",IntToStr(nSamples));
```

## See Also

[Trend Functions](#)

## TrnScroll

Scrolls the trend pen by a specified percentage (of span), or number of samples.

## Syntax

**TrnScroll(*nAN, Pen, nScroll [, nMode]* )**

*AN*:

The AN where the trend is located. Set to -1 for all trends on the current page.

*Pen*:

The trend pen number. Set to -1 for all pens.

*nScroll*:

The amount by which the trend will be scrolled. Use *nMode* to specify whether the trend will be scrolled by percentage or by number of samples.

Because the resolution of Client requests is 1 second, requests of millisecond accuracy are rounded to 1 second. For example, if requested to scroll 2 samples of 400 milliseconds (a total of 0.8 seconds), the trend will actually scroll 1 second.

*nMode*

The type of scrolling to be performed.

1 - The trend will be scrolled by a percentage of span. Default.

2 - The trend will be scrolled by a number of samples. This mode is not available if the user puts the trend into the 'trend span' mode by setting the span. In this case no scrolling would take place; the user needs to use

*nMode* 1.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnSetTime](#)

## Example

```
! Scroll all pens (of the trend at an20) 100% forwards.  
TrnScroll(20,-1,100); or TrnScroll(20,-1,100,1);  
! Scrolls all pens (of all trends on the current trend page) 300% backwards.  
TrnScroll(-1, -1, -300); or TrnScroll(20,-1,-300,1);  
!Scrolls all pens (of all trends on the current trend page) 3 samples forwards.  
TrnScroll(20,-1,3,2);  
!Scrolls all pens (of all trends on the current trend page) 1 sample backwards.  
TrnScroll(20,-1,-1,2);
```

## See Also

[Trend Functions](#)

## TrnSelect

Sets up a page for a trend. This function allows you to set up a trend before the trend page is displayed. You can therefore use a single trend page to display any trend in the project by selecting the trend first, and then displaying the trend page. The [PageTrend\(\)](#) function uses this function to display the standard trend pages.

Call this function and a set of [TrnSetPen\(\)](#) functions before you display a trend page. When the trend page is displayed, all pens set by the TrnSetPen() functions are displayed. You can use the TrnSelect() function to configure different set of pens to be displayed on one generic trend page. The pen settings in the Page Trend database are overridden.

---

**Note:** Trend functions used after the TrnSelect() function needs to use the special value -2 as their AN. (See the example below).

---

## Syntax

**TrnSelect(*Window*, *Page*, *AN* [, *sClusterName*] )**

*Window*:

The window number (returned from the WinNumber function).

-3 - for the current window.

-2 - for the next window displayed.

*Page*:

The name of the page that displays the trend.

*AN:*

The AN where the trend displays, or -3 for the first trend on the page.

*sClusterName:*

The name of the cluster that is associated with any trend tag for this trend graph. This is optional if you have one cluster or are resolving the trend via the current cluster context. The argument is enclosed in quotation marks "".

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnSetPen](#), [PageTrend](#), [WinNumber](#)

## Example

```
TrnSelect(WinNumber(), "TrendPage", 40, "ClusterXYZ");
TrnSetPen(-2,1,"PV1");
TrnSetPen(-2,2,"PV2");
TrnSetPen(-2,3,"PV3");
TrnSetPen(-2,4,"PV4");
PageDisplay("TrendPage");
```

## See Also

[Trend Functions](#)

## TrnSetCursor

Moves the trend cursor by a specified number of samples. If the trend cursor is disabled, this function enables it. If the cursor is enabled and the number of samples is 0 (zero), the cursor is disabled. If the cursor is moved off the current trend frame, the trend scrolls.

## Syntax

**TrnSetCursor(*nAN, Samples*)**

*AN:*

The AN where the trend is located. Set to -1 for all trends on the current page.

*Samples:*

The number of samples to move the cursor.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnGetCursorTime](#), [TrnGetCursorValue](#), [TrnGetCursorValueStr](#), [TrnSetCursorPos](#)

## Example

```
! For the trend at AN20
TrnSetCursor(20,1);
! Moves the trend cursor forwards 1 sample.
TrnSetCursor(-1,-40);
! Moves the trend cursor (of all trends on the current trend page)
backwards 40 samples.
```

## See Also

[Trend Functions](#)

## TrnSetCursorPos

Moves the trend cursor to a specified x-axis point, offset from the trend cursor origin. If the trend cursor is disabled, this function enables it. If the position is outside of the trend frame, it sets the trend cursor to half of the frame.

## Syntax

**TrnSetCursorPos(*nAN*, *Position*)**

*AN*:

The AN where the trend is located. Set to -1 for all trends on the current page.

*Position*:

The x-axis point at which to position the trend cursor, offset from the trend cursor origin.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnGetCursorPos](#), [TrnSetCursor](#)

## Example

```
! For the trend at AN20, if the trend frame is 400 points
TrnSetCursorPos(20,0);
! Moves the trend cursor to its origin.
TrnSetCursorPos(20,200);
! Moves the trend cursor to half of its frame size (200 points).
```

## See Also

[Trend Functions](#)

### TrnSetDisplayMode

Specifies how raw trend samples are displayed on the screen.

## Syntax

**TrnSetDisplayMode(*nAN*, *PenNumber*, *DisplayMode*)**

*AN*:

The animation number of the chosen trend.

*PenNumber*:

The pen number of the chosen trend. Specify:

0 - The current pen

1-8 - Pens 1 through 8

-1 - All pens

*DisplayMode*:

The Display Mode parameters allow you to enter a single integer to specify the display options for a trend (for a maximum of eight trends).

To calculate the integer you should enter, select the options you want to use from the list below, adding their associated numbers together. The resulting integer is the *DisplayMode* parameter for that trend.

**Invalid/Gated trend options:**

0 - Convert invalid/gated trend samples to zero.

1 - Leave invalid/gated trend samples as they are.

**Ordering trend sample options:**

0 - Order returned trend samples from oldest to newest.

2 - Order returned trend samples from newest to oldest.

**Condense method options:**

0 - Set the condense method to use the mean of the samples.

4 - Set the condense method to use the minimum of the samples.

8 - Set the condense method to use the maximum of the samples.

12 - Set the condense method to use the newest of the samples.

**Stretch method options:**

0 - Set the stretch method to step.

128 - Set the stretch method to use a ratio.

256 - Set the stretch method to use raw samples.

**Gap Fill Constant option:**

*n* - the number of missed samples that the user wants to gap fill) x 4096.

**Display as Periodic options:**

0 - Display according to trend type.

1048576 - display as periodic regardless of trend type.

Since the Display as Periodic options are read-only, they cannot be set using TrnSetDisplayMode. They can be retrieved using [TrnGetDisplayMode\(\)](#) and also used with the TrnExport group of functions.

**Synchronize to Display Period options:**

0 - Synchronize the returned samples to the nearest display period.

16777216 - Do not synchronize samples to the nearest display period.

**Note:** Options listed in each group are mutually exclusive. The default value for each Display Mode is 258 (0 + 2 + 256).

---

**Return Value**

0 (zero) if successful, otherwise an error code is returned.

**Related Functions**

[TrnGetDisplayMode](#), [TrnGetTable](#)

**See Also**

[Trend Functions](#)

**TrnSetEvent**

Sets the start event of a trend pen. This function only operates on an event-based trend.

**Syntax**

**TrnSetEvent(*nAN*, *Pen*, *Event*)**

*AN*:

The AN of the chosen trend.

*Pen*:

The trend pen number:

0 - The pen currently in focus

1...8 - Pen1 . . . Pen8

*Event*:

The number of the start event.

**Return Value**

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnGetEvent](#), [TrnGetCluster](#), [TrnGetCursorEvent](#)

## Example

```
! Sets pen1 to event number 123456
TrnSetEvent(20,1,123456);
! Scrolls pen1 back by 100 events
TrnSetEvent(20,1,TrnGetBufEvent(20,1,0)-100);
```

## See Also

[Trend Functions](#)

### TrnSetPen

Sets the trend tag of a trend pen. The trend pen changes to the specified tag and the trend is refreshed. The trend pen needs to be in the operator's area to be displayed. If outside of the operator's area, data is not displayed. You cannot mix periodic trends and event trends in the same trend window.

This function may sometimes return before the pen is actually set when called on a PC which is not the trend server. This may create difficulties for following functions such as [TrnSetScale](#). A wrapper function can be used to confirm the pen is set before returning. See example 2 below.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

## Syntax

**TrnSetPen(*nAN*, *Pen*, *Tag*)**

*AN*:

The AN where the trend is located.

-1 - All trends on the current trend page.

-2 - The function being called is using the special AN setup by the TrnSelect() function.

*Pen*:

The pen for which the trend tag will be changed.

-2 - The first available pen (This value is automatically changed to 0 for SPC trends because they have only one pen per trend.)

-1 - All pens on the trend. (Not allowed for SPC trends.)

0 - The pen currently in focus.

1...8 - Pen1....Pen8

*Tag*:

The trend tag. If Tag = ! the pen is deleted.

## Return Value

0 (zero) if successful, otherwise an error code is returned. Be aware that if a mixture of periodic and event trends is detected, the return value is 0 (zero), but the hardware alarm #329 is set.

## Related Functions

[TrnGetPen](#), [TrnSelect](#)

### Example 1

```
! For the trend at AN20
TrnSetPen(20,1,"PV1");
! Sets the trend tag of Pen1 to "PV1".
```

### Example 2

```
INT
FUNCTION
BlockedTrnSetPen(INT hAN, INT nPen, STRING sTrend)
    INT timeout = 5000
    INT sleepTime = 10
    INT error = -1
    INT elapsed = 0
    INT currentTime
    STRING sPenName
    error = TrnSetPen(hAN, nPen, sTrend)
    IF error = 0 THEN
        error = -1
        currentTime = SysTime()
        WHILE error <> 0 AND elapsed < timeout DO
            sPenName = TrnGetPen(hAN, nPen)
            IF sPenName = sTrend THEN
                error = 0
            ELSE
                SleepMS(sleepTime)
                elapsed = elapsed + SysTimeDelta(currentTime)
            END
        END
    END
    RETURN error
END
```

## See Also

[Trend Functions](#)

### TrnSetPenFocus

Sets the focus to a specified pen. After the focus is set, the focus pen is used with other trend functions.

## Syntax

**TrnSetPenFocus(*nAN*, *Pen*)**

*AN*:

The AN of the chosen trend.

*Pen*:

The trend pen:

- 4 - Make the next pen the focus pen; without skipping blank pens.
- 3 - Make the previous pen the focus pen; without skipping blank pens.
- 2 - Make the next pen the focus pen; skip blank pens.
- 1 - Make the previous pen the focus pen; skip blank pens.
- 0 - Keep the current focus.
- 1...8 - Change Pen1. . .8 to be the focus pen.

## Return Value

The old pen focus number, or -1 if an error is detected. You can call the [IsError\(\)](#) function to get the actual error code.

## Related Functions

[TrnGetPenFocus](#)

## Example

System Keyboard

Key Sequence	NextPen
Command	<code>TrnSetPenFocus(20, -2)</code>
Comment	For the trend at AN20, make the next pen the focus pen

## See Also

[Trend Functions](#)

## TrnSetPeriod

Sets the display period (time base) of a trend. When the period is changed, Plant SCADA reads the historical data to reconstruct the trend data, and refreshes the trend. Every pen has the same display period.

This function clears the span set by the [TrnSetSpan\(\)](#) function.

## Syntax

**TrnSetPeriod(*nAN*, *Period*)**

*AN*:

The AN where the trend is located. Set to -1 for every trend on the current page.

*Period*:

The new sampling period (in seconds) of the trend. To set the display period to the sampling period, set this argument to 0 (zero),

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnGetPeriod](#), [TrnEcho](#), [TrendSetTimebase](#), [TrendSetSpan](#)

## Example

System Keyboard

Key Sequence	## Enter
Command	TrnSetPeriod(20, Arg1)
Comment	Set a new sampling period for the trend at AN20

## See Also

[Trend Functions](#)

## TrnSetScale

Sets a new scale for a trend pen. In the automatic scaling mode, the zero and full scales are automatically generated.

## Syntax

**TrnSetScale(*nAN*, *Pen*, *Percent*, *Scale*)**

*AN*:

The AN where the trend is located. Set to -1 for all trends on the current page.

*Pen*:

The trend pen number:

-1 - All pens

0 - The pen currently in focus

1...8 - Pen1...Pen8

*Percent:*

The scale mode:

-2 - Set both zero and full scales to the default scales.

-1 - Place the trend into automatic scale mode.

0 - Set the zero scale.

100 - Set the full scale.

*Scale:*

The new value of the scale. Scale is ignored if Percent is -2.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnGetScale](#), [TrnEcho](#), [TrnSetScale](#)

## Example

```
! For the trend at AN20
TrnSetScale(20,-1,100,5000.0);
! Sets the full scale of all pens to 5000.0
```

## See Also

[Trend Functions](#)

## TrnSetSpan

Sets the span time of a trend. The span time is the total time displayed in the trend window. You can set the period to contain fractions of a second. For example, if you set a trend with 240 samples to a span of 10 minutes, then each sample would be 2.5 seconds. Choose a span long enough to provide a sufficient sample rate to capture accurate real time data.

## Syntax

**TrnSetSpan(*nAN*, *Span*)**

*AN:*

The AN of the chosen trend.

*Span:*

The span time (in seconds).

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnSetPeriod](#), [TrnGetSpan](#), [TrnSetSpan](#)

## Example

```
// Set a span of 2 hours.  
TrnSetSpan(40,StrToTime("2:00:00"));  
// Then use TrnGetSpan function to display the span  
Time = TrnGetSpan(40);  
DspText(31,0,TimeToStr(Time,5));
```

## See Also

[Trend Functions](#)

## TrnSetTable

Writes trend tag data from a table to the trend logging system (starting at the top of the table, and continuing to the bottom). Each value is written with a time and date, as specified by *Period*. If *Period* differs from the trend sampling period (defined in the Trend Tags database), the trend's sample values will be calculated (averaged or interpolated) from the tabulated trend data.

The user needs to have the correct privilege (as specified in the database), otherwise the data is not written.

This function is a blocking function. It will block the calling Cicode task until the operation is complete.

## Syntax

**TrnSetTable**(*Tag*, *Time*, *Period*, *Length*, *Table* [, *Milliseconds*] [, *sClusterName*] )

*Tag*:

The trend tag enclosed in quotation marks "" (can be prefixed by the name of the cluster that is ClusterName.Tag).

*Time*:

The time and date (long integer) to be associated with the first value in the table when it is set. Once you have entered the end time and date (Time), set period (Period), and number of trend tag values to be set (Length), the start time and date will be calculated automatically. For example, if Time = StrToDate("22/07/20") + StrToTime("09:00"), Period = 30, and Length = 60, the start time would be 08:30. In other words, the first value from the table would be set with time 9am, and the last would be set with time 8.30am (on July 22, 2020).

If this argument is set to 0 (zero), the time used will be the current time.

*Period*:

This will be the interval (in seconds) between trend values when they are set (that is it will be the perceived sampling period for the trend). This period can differ from the actual trend period. Set to 0 (zero) to default to

the actual trend period.

*Length:*

The number of trend values in the trend table.

*Table:*

Variable containing the table of floating-point values in which the trend data is stored. You can enter the name of an array here (see the example). Must be a Real type variable.

*Milliseconds:*

This argument allows you to set the time of the first sample in the table with millisecond precision. After defining the time and date in seconds with the Time argument, you can then use this argument to define the milliseconds component of the time.

For example, if you wanted to set data from the 22/07/20, at 9am, 13 seconds, and 250 milliseconds you could set the Time and Milliseconds arguments as follows:

```
Time = StrToDate("22/07/20") + StrToTime("09:00:13")
Milliseconds = 250
```

If you don't enter a milliseconds value, it defaults to 0 (zero). There is no range constraint, but as there are only 1000 milliseconds in a second, you should keep your entry between 0 (zero) and 999.

*sClusterName:*

The name of the cluster in which the trend tag resides. This is optional if you have one cluster or are resolving the trend via the current cluster context. The argument is enclosed in quotation marks "".

---

**Note:** If you are putting a value for "sClusterName" in the TrnSetTable(...), then you have to provide a value for "Milliseconds" parameter.

---

## Return Value

The actual number of samples written. The return value is 0 if an error is detected. You can call the IsError() function to get the actual error code.

## Related Functions

[TrnGetTable](#)

## Example

```
REAL TrendTable1[100];
/* Defines an array of a maximum of 100 entries. Assume that
TrendTable1 has been storing data from a source. */
TrnSetTable("OP1",StrToDate("22/07/20")
+StrToTime("09:00"),2,10,TrendTable1,0,"ClusterXYZ");
/* A set of 10 trend data values are set for the OP1 trend tag. */
```

## See Also

[Trend Functions](#)

## TrnSetTime

Sets the end time and date of a trend pen.

If you set a time less than the current time, the trend display is set to historical mode and samples taken after this time and date will not be displayed.

If you set the time to the current time, for example by using the [TimeCurrent](#) or [TrendZoom](#) Cicode functions, the trend is displayed in real-time mode and samples after this date and time will display.

**Note:** The end time specified for a trend object may automatically align with the display period that is used. To stop this occurring, you can use the [TrnSetDisplayMode](#) function by setting the *DisplayMode* parameter to 16777216 (samples will no longer synchronize to the nearest display period). You can set this as a default for a page by configuring the function as an **On page shown** event in the Page Properties (see [Page Properties - Events](#)).

## Syntax

**TrnSetTime(***nAN, Pen, Time***)**

*AN*:

The AN where the trend is located, or:

-1 - All trends on the current page

0 - The trend where the cursor is positioned

*Pen*:

The trend pen number:

-1 - All pens

0 - The pen currently in focus

1...8 - Pen1...Pen8

*Time*:

The end time and date of the trend. Samples taken after this time and date will not be displayed. Set to 0 (zero) to set the trend to the current time (real-time mode).

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[TrnGetTime](#)

## Example

```
TrnSetTime(20,1,TimeCurrent()-60*30);
/* Sets Pen1 to 30 minutes before the current time (30 minutes ago). */
TrnSetTime(20,1,0);
/* Sets the trend to real-time mode. */
```

## See Also

[Trend Functions](#)

## Window Functions

Window functions control the display of windows. You can open, move, size, activate, and de-activate windows. You can also specify titles for your windows.

Following are functions relating to Windows:

<a href="#">GetWinTitle</a>	Returns the name of the active window as a string.
<a href="#">HtmlHelp</a>	Display a specific topic from an HTML Help file (.chm).
<a href="#">MultiMonitorStart</a>	Displays a Plant SCADA window on each of the configured monitors when a display client starts up.
<a href="#">WinCopy</a>	Copies the active window to the Windows clipboard.
<a href="#">WinFile</a>	Writes the active window to a file.
<a href="#">WinFree</a>	Removes a display window.
<a href="#">WinGetClicked</a>	Gets the number of the Plant SCADA window that has most recently been clicked on using the left mouse button.
<a href="#">WinGetFirstChild</a>	Gets the window number of the first child of a parent window.
<a href="#">WinGetFocus</a>	Gets the number of the Plant SCADA window that has the keyboard focus.
<a href="#">WinGetName</a>	Gets the name previously associated with a particular window number using WinSetName.
<a href="#">WinGetNextChild</a>	Gets the window number of the next child in a child link.
<a href="#">WinGetParent</a>	Retrieves the Window Number of the specified window's parent or root window.
<a href="#">WinGetWndHnd</a>	Gets the window handle for the current window.
<a href="#">WinGoto</a>	Changes the active window.
<a href="#">WinMode</a>	Sets the display mode of the active window.
<a href="#">WinMove</a>	Moves the active window.
<a href="#">WinNew</a>	Opens a display window.

<a href="#">WinNewAt</a>	Opens a display window at specified coordinates.
<a href="#">WinNewPinAt</a>	Opens a new display window at a specified location, relative to the current active window, with a selected page displayed.
<a href="#">WinNext</a>	Makes the next window active.
<a href="#">WinNumber</a>	Gets the window number of the active Plant SCADA window.
<a href="#">WinPos</a>	Positions a window on the screen.
<a href="#">WinPrev</a>	Makes the previous window active.
<a href="#">WinPrint</a>	Prints the active window.
<a href="#">WinPrintFile</a>	Prints a file to the printer.
<a href="#">WinSelect</a>	Selects a window for Cicode output.
<a href="#">WinSetName</a>	Associates a name with a particular window by its window number.
<a href="#">WinSize</a>	Sizes a window.
<a href="#">WinStyle</a>	Switches on and off scrolling and scroll bar features for existing windows.
<a href="#">WinTitle</a>	Sets the title of the active window.
<a href="#">WndFind</a>	Gets the Windows number of any window in any application.
<a href="#">WndGetFileProfile</a>	Gets a profile string from any .INI file.
<a href="#">WndGetProfile</a>	This function is now obsolete.
<a href="#">WndHelp</a>	This function is obsolete.
<a href="#">WndInfo</a>	Gets the Windows system metrics information.
<a href="#">WndMonitorInfo</a>	Returns information about a particular monitor.
<a href="#">WndMonitorInfoEx</a>	Returns information about a particular monitor using the screen name.
<a href="#">WndPutFileProfile</a>	Puts a profile string into any .INI file.
<a href="#">WndPutProfile</a>	This function is now obsolete.
<a href="#">WndShow</a>	Sets the display mode of any window of any

	application.
WndViewer	This function is obsolete.

## See Also

[Cicode Function Categories](#)

[Using Cicode Functions](#)

[WndPutFileProfile](#)

## GetWinTitle

Returns the name of the active window as a string.

**Note:** This function does not work with pinned windows.

## Syntax

`GetWinTitle()`

## Return Value

The title of the active window as a string if successful; otherwise, an error is returned.

## Related Functions

[WinTitle](#)

## See Also

[Window Functions](#)

## HtmlHelp

Invokes the Microsoft HTML Help application (hh.exe) to display a specific topic from an HTML help file (.chm).

## Syntax

`HtmlHelp(sHelpFile, nCommand, sData)`

*sHelpFile*:

The help file to display. For example, "C:\Program Files (x86)\AVEVA Plant SCADA\Bin\ProcessAnalyst.chm".

This argument supports path substitutions. For example, "[BIN]:ProcessAnalyst.chm".

*nCommand*:

The type of help:

0 - Display a topic identified by an internal file name in the *sData* field. This is the name of the file within the .chm file. For example: "1126187.htm".

1 - Display a topic identified by the Mapped Topic ID in the *sData* field. For example: "1000".

2 - Terminate the help application.

3 - Display the index.

*sData*:

Optional data, depending on the value of *nCommand*.

## Return Value

0 (zero) if successful, otherwise an error is returned.

## See Also

[Window Functions](#)

## MultiMonitorStart

Displays a Plant SCADA window on each of the configured monitors when a display client starts up. It sets up the windows according to the Multi-Monitor Parameters [\[MultiMonitors\]Monitors](#) and [\[MultiMonitors\]StartupPage<n>](#).

## Syntax

**MultiMonitorStart()**

## Return Value

None.

## See Also

[Window Functions](#)

## WinCopy

Copies the graphics image of the active window to the Windows Clipboard. You can paste this Clipboard image into other applications.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

---

### Note:

- This function might not work as expected if called directly from the Kernel; instead, this function should be called from a graphics page.
  - This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.
-

## Syntax

**WinCopy( [xScale] [, yScale] [, bSwapBlackWhite] [, sMap] )**

*xScale*:

The x scaling factor for the item being copied. This argument is optional, as a default setting of 1 is used to maintain the current horizontal scaling of the image. The outcome is proportional to 1; for example, 0.5 halves the width of the image.

*yScale*:

The y scaling factor for the item being copied. This argument is optional, as a default setting of 1 is used to maintain the current vertical scaling of the image. The outcome is proportional to 1; for example, 0.5 halves the height of the image.

*bSwapBlackWhite*:

Swaps the colors black and white for the purpose of printing pages with a lot of black content. Use the default value of 1 to swap black and white (optional).

*sMap*:

The file name or path of a text based map file used to specify colors to swap when printing. By default Plant SCADA will look in the bin directory for the map file. The format for the map file is:

RRR GGG BBB RRR GGG BBB [HHH] [SSS] [LLL]//  
RRR GGG BBB RRR GGG BBB [HHH] [SSS] [LLL]//

Where:

*RRR* is a decimal red intensity value between 000 and 255.

*GGG* is a decimal green intensity value between 000 to 255.

*BBB* is a decimal blue intensity value between 000 to 255.

*HHH*, *SSS* and *LLL* are optional tolerance values to enable swapping a range of colors. Default values are shown below.

*HHH* is a decimal value representing the Hue tolerance. Valid range is 0 to 360. Default = 0.

*SSS* is a decimal value representing the Saturation tolerance. Valid range is 0 to 255. Default = 2 \* Hue tolerance.

*LLL* is a decimal value representing the Luminance tolerance. Valid range is 0 to 255. Default = 2 \* Hue tolerance.

Leading zeros are not required, however they aid readability.

The first three RGB values specify the FROM color, and the second three RGB values specify the TO color. The tolerance values are applied to the FROM color when replacing individual pixels in the image.

Comments may be placed at the end of each line, or on individual lines, and needs to be proceeded with // or !.

## Example

```
043 043 255 155 205 255 025 000 025
//Change dark blue to light blue using Hue and Luminance tolerance of 25.
000 000 000 255 255 255 //Swap black to white with no tolerance
```

**Note:** If swap color ranges overlap, the behavior is undefined (i.e. a color that falls in both ranges may end up as either color)

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[WinPrint](#)

## Example

```
WinCopy();  
! Copies the active window to the Windows Clipboard.  
WinCopy(0.5,0.5);  
! Copies the active window to the Windows Clipboard at half the  
current size.
```

## See Also

[Window Functions](#)

## WinFile

Writes the graphics image of the active window to a file. The file is saved in the Plant SCADA compressed .bmp format.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

---

**Note:**

- This function might not work as expected if called directly from the Kernel; instead, this function should be called from a graphics page.
  - This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.
- 

## Syntax

**WinFile(*sFile* [, *xScale*] [, *yScale*] [, *bSwapBlackWhite*] [, *sMap*] )**

*sFile*:

The name of the file to be created. If no path information is specified, the function will try to create the file in the bin directory (the current directory of the runtime process). Write permission is required.

Path operators such as [data] and [run] are also supported. For example to save the file in the [data] directory, you can specify:

[data]:screendump.bmp

The specified path needs to point to a writable folder.

*xScale*:

The x scaling factor for the item being printed. This argument is optional, as a default setting of 1 is used to maintain the horizontal scaling of the image. The outcome is proportional to 1; for example, 0.5 halves the width of the image .

**yScale:**

The y scaling factor for the item being printed. This argument is optional, as a default setting of 1 is used to maintain the current vertical scaling of the image. The outcome is proportional to 1; for example, 0.5 halves the height of the image.

**bSwapBlackWhite:**

Swaps the colors black and white for the purpose of printing pages with a lot of black content. Use the default value of 1 to swap black and white (optional).

**sMap:**

The file name or path of a text based map file used to specify colors to swap when printing. By default Plant SCADAWill look in the bin directory for the map file. The format for the map file is:

```
RRR GGG BBB RRR GGG BBB [HHH] [SSS] [LLL]//  
RRR GGG BBB RRR GGG BBB [HHH] [SSS] [LLL]//
```

Where:

*RRR* is a decimal red intensity value between 000 and 255.

*GGG* is a decimal green intensity value between 000 to 255.

*BBB* is a decimal blue intensity value between 000 to 255.

*HHH*, *SSS* and *LLL* are optional tolerance values to enable swapping a range of colors. Default values are shown below.

*HHH* is a decimal value representing the Hue tolerance. Valid range is 0 to 360. Default = 0.

*SSS* is a decimal value representing the Saturation tolerance. Valid range is 0 to 255. Default = 2 \* Hue tolerance.

*LLL* is a decimal value representing the Luminance tolerance. Valid range is 0 to 255. Default = 2 \* Hue tolerance.

Leading zeros are not required, however they aid readability.

The first three RGB values specify the FROM color, and the second three RGB values specify the TO color. The tolerance values are applied to the FROM color when replacing individual pixels in the image.

Comments may be placed at the end of each line, or on individual lines, and needs to be proceeded with // or !.

Example map file:

```
043 043 255 155 205 255 025 000 025  
//Change dark blue to light blue using Hue and Luminance tolerance of 25.  
000 000 000 255 255 255 //Swap black to white with no tolerance
```

**Note:** If swap color ranges overlap, the behavior is undefined. This means a color that falls in both ranges may end up as either color.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[WinPrint](#)

## Example

```
WinFile("DUMP");
```

```
/* Writes the active window to a file named DUMP in the current
directory. */
```

## See Also

[Window Functions](#)

### WinGetFirstChild

Gets the window number of the first child of a parent window. This number can be used with other functions to control the window.

## Syntax

**WinGetFirstChild(*Window*)**

*Window*:

The window number of the parent window.

## Return Value

The window number associated with the first child of the specified parent window. If a child window does not exist, -1 is returned.

## Related Functions

[WinGetNextChild](#), [WinNew](#), [WinNewAt](#), [WinNumber](#)

## Example

```
...
// Get the active window number
iParentWindowNum = WinNumber();
iFirstChild = WinGetFirstChild(iParentWindowNum);
iSecondChild = WinGetNextChild(iFirstChild);
iThirdChild = WinGetNextChild(iSecondChild);
...
```

## See Also

[Window Functions](#)

### WinFree

Removes the active display window. Be aware that the last window (and any child windows owned by the last window) cannot be removed. You cannot call this function as an exit command or from a Cicode Object.

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function

---

from the server process results in a hardware alarm being raised.

---

## Syntax

**WinFree()**

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[WinNew](#), [WinNewAt](#)

## Example

```
WinFree();  
! Removes the active display window.
```

## See Also

[Window Functions](#)

## WinFreeEx

Removes the active display window and allows you to pass a Windows handle to the function for closing the window. Note that the last window (and any child windows owned by the last window) cannot be removed. You cannot call this function as an exit command or from a Cicode Object.

## Syntax

**WinFreeEx(*Window*)**

*Window*:

The window number (returned from the [WinNumber\(\)](#) function). Note that this is not the same as the window handle, returned from the [WndFind\(\)](#) function.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[WinFree](#), [WinNew](#), [WinNewAt](#)

## Example

```
FUNCTION Activate_Pinned()
pinnedWindow = WinNewPinAt("Pinned", 1000, 300, 0);
WinGoto(pinnedWindow);
END
FUNCTION Deactivate_Pinned()
WinFreeEx(pinnedWindow);
END
```

## See Also

[Window Functions](#)

### WinGetClicked

Gets the number of the Plant SCADA window that has most recently been clicked on using the left mouse button.

## Syntax

**WinGetClicked()**

## Return Value

The window number of the Plant SCADA window that has most recently been clicked on using the left mouse button. Be aware that this is not the same as the window handle, returned from the WndFind() function.

## Related Functions

[WndFind](#)

## Example

```
nCitectWin=WinGetClicked();
! Gets the number of the Plant SCADA window that has most recently been
clicked on using the left mouse button.
```

## See Also

[Window Functions](#)

### WinGetFocus

Gets the number of the Plant SCADA window that has the keyboard focus.

## Syntax

**WinGetFocus()**

## Return Value

The window number of the Plant SCADA window that has the keyboard focus. Be aware that this is not the same as the window handle, returned from the WndFind() function.

## Related Functions

[WndFind](#)

## Example

```
nCitectWin=WinGetFocus();
! Gets the number of the Plant SCADA window that has
the keyboard focus
```

## See Also

[Window Functions](#)

## WinGetName

Gets the name previously associated with a particular window number using [WinSetName](#). An association with a window is removed when the window is free-ed.

## Syntax

**WinGetName([*iWinNum*])**

*iWinNum*

An optional parameter which specifies the window number to lookup for the associated Name. If no number is specified, the name returned is the one associated with the currently selected window number.

## Return Value

String name associated with window number *iWinNum*, otherwise an empty string is returned.

## Related Functions

[WinSetName](#), [WinNumber](#)

## See Also

[Window Functions](#)

### WinGetNextChild

Gets the window number of the next child in a child link. This number can be used with other functions to control the window.

## Syntax

**WinGetNextChild(*Window*)**

*Window*:

The window number of the previous child in the child link.

## Return Value

The window number associated with the next child in the child link. If a window does not exist, -1 is returned.

## Related Functions

[WinGetFirstChild](#), [WinNew](#), [WinNewAt](#), [WinNumber](#)

## Example

```
...
// Get the active window number
iParentWindowNum = WinNumber();
iFirstChild = WinGetFirstChild(iParentWindowNum);
iSecondChild = WinGetNextChild(iFirstChild);
iTThirdChild = WinGetNextChild(iSecondChild);
...
```

## See Also

[Window Functions](#)

### WinGetParent

Retrieves the window number of the specified window's parent or root window.

## Syntax

**WinGetParent([*WinNum*], [*GetRoot*])**

*WinNum*

An optional parameter which specifies the window number of the child window. If no number is specified, it retrieves the window number of the currently selected window's parent or root window.

#### *GetRoot*

An optional parameter which determines whether to get the parent or root window. If false, it will retrieve the immediate parent window number (default). If true, it retrieves the root window number by walking the chain of parent windows.

### Return Value

The window number of the parent or root window.

### Related Functions

[WinGetName](#), [WinGetFocus](#)

### Examples

```
nParentWin=WinGetParent();
! Gets the number of the parent window for the currently selected Plant SCADA window

nRootWin=WinGetParent(WinGetFocus(), TRUE);
! Gets the number of the root window for the window that has keyboard focus
```

### See Also

[Window Functions](#)

### WinGetWndHnd

Gets the window handle for the current window. The window handle may be used by 'C' programs and Plant SCADA Wnd... functions. You may pass the windows handle to a 'C' program by using the DLL functions.

### Syntax

**WinGetWndHnd()**

### Return Value

The window handle if successful, otherwise 0 (zero) is returned. Be aware that this is not the same as a Plant SCADA window number returned from the [WinNumber\(\)](#) function.

### Related Functions

[DLLCall](#), [WinNew](#), [WndFind](#), [WndShow](#)

## Example

```
INT hWnd;
hWnd = WinGetWndHnd();
WinShow(hWnd,6); //iconize the window
```

## See Also

[Window Functions](#)

## WinGoto

Changes the active window. The specified window is placed in front of all other windows and all keyboard commands will apply to this window. You cannot call this function as an exit command or from a Cicode Object.

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.

## Syntax

**WinGoto(*Window*)**

*Window*:

The window number (returned from the [WinNumber\(\)](#) function). Be aware that this is not the same as the window handle, returned from the [WndFind\(\)](#) function.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[WinNew](#)

## Example

```
! If two windows are displayed;
WinGoto(1);
! Changes the active window to Window 1.
WinGoto(0);
! Changes the active window to Window 0.
```

## See Also

[Window Functions](#)

## WinMode

Sets the display mode of the active Plant SCADA window.

**Note:**

- This function does not work with pinned windows.
- This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.

## Syntax

**WinMode(*Mode*)**

*Mode*:

The mode:

- 0 - Hide the window.
- 2 - Activate the window in an iconized state.
- 3 - Activate the window in a maximized state.
- 4 - Display the window in its previous state without activating it.
- 5 - Activate the window in its current state.
- 6 - Iconize the window.
- 7 - Display the window in an iconized state without activating it.
- 8 - Display the window in its current state without activating it.
- 9 - Activate the window in its previous state.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[WinNew](#)

## Example

```
! Iconize the active Plant SCADA window.  
WinMode(7);
```

## See Also

[Window Functions](#)

## WinMove

Moves the active window to a new location and sizes the window in a single operation. This is the same as calling

the WinPos() and the WinSize() functions. You use PageInfo to get the current window position.

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.

## Syntax

**WinMove(*X*, *Y*, *Width*, *Height*)**

*X*, *Y*:

The new x and y pixel coordinates for the top-left corner of the active window.

If the window is a pinned window, the new coordinates will specify a location for the window that is based on the unscaled version of the page that hosts it.

*Width*:

The width of the window, in pixels.

If the target is a pinned window, the new width will be measured against the original window size as it is on the unscaled version of the page that hosts it.

*Height*:

The height of the window, in pixels.

If the target is a pinned window, the new height will be measured against the original window size as it is on the unscaled version of the page that hosts it.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[WinSize](#), [WinPos](#), [PageInfo](#)

## Example

```
WinMove(100,50,500,300);
/* Moves the top-left corner of the active window to the pixel
coordinate 100,50 and size the window to 500 x 300 pixels. */
```

## See Also

[Window Functions](#)

## WinNew

Opens a new display window, with a specified page displayed. The window can later be destroyed with the [WinFree\(\)](#) function.

You can also specify if the displayed page operates within the context of a particular cluster in a multiple cluster project. When the page is displayed during runtime, the *ClusterName* argument is used to resolve any tags that

do not have a cluster explicitly defined.

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.

## Syntax

**WinNew(*Page*, *ClusterName*)**

*Page*:

The name or page number of the page to display (in quotation marks ""). Can be prefixed by the name of a host cluster, that is "ClusterName.Page". This will take precedence over the use of the *ClusterName* parameter if the two differ.

*sClusterName*:

The name of the cluster that will accommodate the page at runtime. This is optional if you have one cluster or are resolving the page via the current cluster context. The argument is enclosed in quotation marks "". If the *Page* parameter is prefixed with the name of a cluster, this parameter will not be used.

## Return Value

The window number of the window, or -1 if the window cannot be opened. Be aware that this is not the same as the window handle returned from the WndFind() function.

## Related Functions

[WinFree](#), [WinNewAt](#)

## Example

```
! If the display window being opened is window number 2:  
Window=WinNew("Alarm");  
! Displays the Alarm page and sets Window to 2.
```

## See Also

[Window Functions](#)

## WinNewAt

Opens a new display window at a specified location, with a selected page displayed. The window can later be removed with the [WinFree\(\)](#) function.

You can also specify if the displayed page operates within the context of a particular cluster in a multiple cluster project. When the page is displayed during runtime, the *ClusterName* argument is used to resolve any tags that have a cluster omitted.

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.

## Syntax

**WinNewAt(*Page*, *X*, *Y*, *Mode*[, *sClusterName*])**

*Page*:

The name or page number of the page to display (in quotation marks ""). Can be prefixed by the name of a host cluster, that is "ClusterName.Page". This will take precedence over the use of the *sClusterName* parameter if the two differ.

*X*:

The x pixel coordinate of the top left corner of the window.

*Y*:

The y pixel coordinate of the top left corner of the window.

*Mode*:

The mode of the window:

0 - Normal page.

1 - Page child window. The window is closed when a new page is displayed, for example, when the [PageDisplay\(\)](#) or [PageGoto\(\)](#) function is called. The parent is the current active window.

2 - Window child window. The window is closed automatically when the parent window is freed with the [WinFree\(\)](#) function. The parent is the current active window.

4 - No re-size. The window is displayed with thin borders and no maximize/minimize icons. The window cannot be re-sized.

8 - No icons. The window is displayed with thin borders and no maximize/minimize or system menu icons. The window cannot be re-sized.

16 - No caption. The window is displayed with thin borders, no caption, and no maximize/minimize or system menu icons. The window cannot be re-sized.

32 - Echo enabled. When enabled, keyboard echo, prompts, and error messages are displayed on the parent window. This mode should only be used with child windows (for example, Mode 1 and 2).

64 - Always on top.

128 - Open a unique window. This mode helps to prevent this window from being opened more than once.

256 - Display the entire window. This mode commands that no parts of the window will appear off the screen

512 - Open a unique Super Genie. This mode helps to prevent a Super Genie from being opened more than once (at the same time). However, the same Super Genie with different associations can be opened.

1024 - Disables dynamic resizing of the new window, overriding the setting of the [\[Page\]DynamicSizing](#) parameter.

4096 - Allows the window to be resized without maintaining the current aspect ratio. The aspect ratio defines the relationship between the width and the height of the window, which means this setting allows you to stretch or compress the window to any proportions. This option overrides the setting of the [\[Page\]MaintainAspectRatio](#) parameter.

8192 - Text on a page will be resized in proportion with the maximum scale change for a resized window. For example, consider a page that is resized to three times the original width, and half the original height. If this mode is set, the font size of the text on the page will be tripled (in proportion with the maximum scale). This option overrides the setting of the [\[Page\]ScaleTextToMax](#) parameter.

16384 - Hide the horizontal scroll bar.

32768 - Hide the vertical scroll bar.

65536 - Disable horizontal scrolling.

131072 - Disable vertical scrolling.

You can select multiple modes by adding modes together (for example, set Mode to 9 to open a page child window without maximize, minimize, or system menu icons).

*sClusterName*:

The name of the cluster that will accommodate the page at runtime. This is optional if you have one cluster or are resolving the page via the current cluster context. The argument is enclosed in quotation marks "". If the Page parameter is prefixed with the name of a cluster, this parameter will not be used.

## Return Value

The window number of the window, or -1 if the window cannot be opened. Be aware that this is not the same as the window handle returned from the WndFind() function.

## Related Functions

[WinFree](#), [WinNew](#) [WinNewPinAt](#)

## Examples

### Buttons

Text	Mimic Page
Command	WinNewAt("Mimic", 100, 20, 0)
Comment	Display the mimic page in a new window at coordinate 100, 20.

### Buttons

Text	Pop Page
Command	WinNewAt("Popup", 100, 200, 2)
Comment	Display the popup page in a child window at coordinate 100, 200

### Buttons

Text	Pop Page
Command	WinNewAt("Popup", 100, 200, 4)
Comment	Display the popup page in a new window with no maximize and minimize icons

### System Keyboard

Key Sequence	Pop ##### Enter
Command	WinNewAt(Arg1, 100, 200, 2)
Comment	Display a specified popup page in a child window at coordinate 100, 200

**System Keyboard**

Key Sequence	Pop ##### Enter
Command	WinNewAt(Arg1, 100, 200, 4)
Comment	Display a specified popup page in a new window with no maximize and minimize icons

**See Also**

[Window Functions](#)

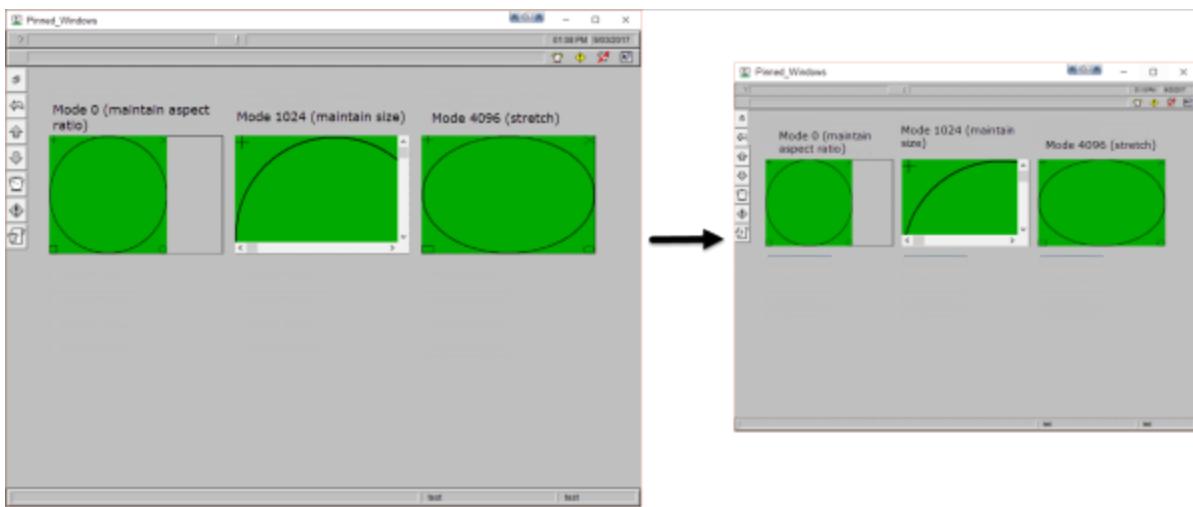
**WinNewPinAt**

Allows for windows to be "pinned" at specified locations within a main window or within other pinned windows. This function opens a new display window at a specified location, relative to the current active window, with a selected page displayed. The window can later be removed with the WinFree() function.

The pinned windows become child windows of the parent window, and maintain their size and position relative to the parent window.

Pinned windows do not have a border, title bar or title. They cannot be maximized or minimized. Pinned windows are automatically closed when the main window is closed. They can be created in one of the following modes (as detailed in the Syntax section below).

- As the window is sized, the page inside maintains its original aspect ratio. The aspect ratio defines the relationship between the width and the height of the window. In this mode, the page is aligned to the top left. Blank areas are filled in using the background color of the main window.
- As the window is sized, the page inside maintains its original size. Scroll bars are displayed to allow the user to navigate the window if the content cannot be accommodated in the window. The scroll bars can be disabled using the Cicode function WinStyle OR by passing in various modes into the winnewpinat mode parameter.
- the page is stretched to fit in the window. For example, consider a page that is resized to three times the original width, and half the original height. If this mode is set, the font size of the text on the page will be tripled (in proportion with the maximum scale).



**Note:** Pinned windows cannot overlap. If you use this function to create a pinned window that overlaps an existing one, the window will not appear and the "Pinned window overlaps" hardware alarm will be generated.

The content of pinned windows can be manipulated using most Windows Cicode functions **except** the following:

- GetWindowTitle()
- WinMode()
- WinTitle()
- WndShow()

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.

You can also specify if the displayed page operates within the context of a particular cluster in a multiple cluster project. When the page is displayed during runtime, the *ClusterName* argument is used to resolve any tags that have a cluster omitted.

**Note:** When executing a keyboard command while the focus is in a pinned window, the keyboard command will first be matched against the list of keyboard commands associated with the graphics object that has current focus, and then the list of keyboard commands associated with the pinned window page itself. The keyboard command will then be matched against the list of keyboard commands associated with the parent of the pinned window, and so on. Command execution will stop at the first object or page that matches the keyboard command sequence.

## Syntax

**WinNewPinAt(*Page*, *X*, *Y*[, *Mode*][, *Width*][, *Height*][, *sClusterName*])**

*Page*:

The name or page number of the page to display (in quotation marks ""). Can be prefixed by the name of a host cluster, that is "ClusterName.Page". This will take precedence over the use of the ClusterName parameter if the two differ.

*X*:

The x pixel coordinate of the top left corner of the window.

*Y:*

The y pixel coordinate of the top left corner of the window.

*Mode:*

The mode of the window:

0 - Set by default. Normal page, maintain aspect ratio when resized. The aspect ratio defines the relationship between the width and the height of the window.

1024 - Disables dynamic resizing of the new window.

4096 - Allows the window to be resized without maintaining the current aspect ratio. The aspect ratio defines the relationship between the width and the height of the window, which means this setting allows you to stretch or compress the window to any proportions.

8192 - Text on a page will be resized in proportion with the maximum scale change for a resized window. This option overrides the setting of the [\[Page\]ScaleTextToMax](#) parameter.

You can select multiple modes by adding modes together.

*Width:*

The pixel width of the window, scaled relative to the current active window.

Default value - The width of the page associated with argument 'Page'.

*Height:*

The pixel height of the window, scaled relative to the current active window.

Default value - The height of the page associated with argument 'Page'.

*sClusterName:*

The name of the cluster that will accommodate the page at runtime. This is optional if you have one cluster or are resolving the page via the current cluster context. The argument is enclosed in quotation marks "". If the Page parameter is prefixed with the name of a cluster, this parameter will not be used.

## Return Value

The window number of the window, or -1 if the window cannot be opened. Be aware that this is not the same as the window handle returned from the WndFind() function.

---

**Note:** A common cause of the window not being created is because you have reached the maximum allowable windows, which can be changed via the parameter [\[Page\]Windows](#). Also, a window will not be created if it overlaps with an existing one. Check for the "Pinned window overlaps" hardware alarm.

---

## Related Functions

[WinFree](#), [WinNew](#), [WinNewAt](#)

## Example

Here is a Cicode example that creates three pinned windows using the current window as the parent:

- window 1 : coordinates - 0,0, size - 100, 100, resize mode - maintain aspect ratio
- window 2 : coordinates - 100,0, size - 100, 100, resize mode - stretch content to fit window bounds
- window 3 : coordinates - 0,100, size - 200, 100, resize mode - maintain content size

```
INT windowPin1 = -1;
INT windowPin2 = -1;
INT windowPin3 = -1;
INT windowParent = -1;
FUNCTION MyPinnedLayout()
    IF windowParent = -1 THEN
        windowParent = WinNumber();
    END
    IF windowParent <> -1 THEN
        WinGoto(windowParent);
        windowPin1 = WinNewPinAt("page1", 0, 0, 0, 100, 100);
        IF -1 = windowPin1 THEN
            Prompt("WinNewPinAt failed with error: " + ErrMsg(IsError()));
        END
        WinGoto(windowParent);
        windowPin2 = WinNewPinAt("page2", 100, 0, 4096, 100, 100);
        IF -1 = windowPin2 THEN
            Prompt("WinNewPinAt failed with error: " + ErrMsg(IsError()));
        END
        WinGoto(windowParent);
        windowPin3 = WinNewPinAt("page3", 0, 100, 1024, 200, 100);
        IF -1 = windowPin3 THEN
            Prompt("WinNewPinAt failed with error: " + ErrMsg(IsError()));
        END
        WinGoto(windowParent);
    END
END
```

## See Also

[Window Functions](#)

## WinNext

Makes the next window (in order of creation) active.

## Syntax

**WinNext()**

## Return Value

The window number of the window, or -1 if there is no next window. Be aware that this is not the same as the window handle returned from the WndFind() function.

## Related Functions

[WinNew](#), [WinPrev](#)

## Example

```
! If the display window being made active is window number 2:  
Window=WinNext();  
! Makes the next window active and sets Window to 2.
```

## See Also

[Window Functions](#)

### WinNumber

Gets the window number of the active Plant SCADA window. This number can be used with other functions to control the window.

## Syntax

**WinNumber([*sName*])**

*sName*:

String name previously associated with a window number using WinSetName().

## Return Value

Window Number associated with the Name provided. If no Name is specified then the active window number is returned. If there isn't a valid window number associated with the name provided then -1 is returned.

## Related Functions

[WinNew](#), [WinGoto](#), [WinSetName](#)

## Example

```
! Create a new window, but keep the active window the same:  
Window=WinNumber();  
WinNew("Alarm");  
WinGoto(Window);
```

## See Also

[Window Functions](#)

### WinPos

Moves the active window to a new location. You use [PageInfo\(\)](#) to get the current window position.

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function

---

from the server process results in a hardware alarm being raised.

---

## Syntax

**WinPos(*X*, *Y*)**

*X*, *Y*:

The new x and y pixel coordinates of the top-left corner of the active window.

If the window is a pinned window, the new coordinates will specify a location for the window that is based on the unscaled version of the page that hosts it.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[WinSize](#), [WinMove](#), [PageInfo](#)

## Example

```
WinPos(100,50);
/* Moves the top-left corner of the active window to the pixel coordinate 100,50. */
```

## See Also

[Window Functions](#)

## WinPrev

Makes the previous window (in order of creation) active.

## Syntax

**WinPrev()**

## Return Value

The window number of the window, or -1 if there is no next window. Be aware that this is not the same as the window handle returned from the WndFind() function.

## Related Functions

[WinNext](#)

## Example

```
! If the display window being made active is window number 2:  
Window=WinPrev();  
! Makes the previous window active and sets Window to 2.
```

## See Also

[Window Functions](#)

## WinPrint

Sends the graphics image of the active window to a printer.

This function is a blocking function. It blocks the calling Cicode task until the operation is complete.

### Note:

- This function might not work as expected if called directly from the Kernel; instead, this function should be called from a graphics page.
- This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.

## Syntax

**WinPrint(*sPort* [, *xScale*] [, *yScale*] [, *bSwapBlackWhite*] [, *sMap*] )**

*sPort*:

The name of the printer port to which the window will be printed. This name needs to be enclosed within quotation marks "". For example "LPT1:", to print to the local printer, or "\\Pserver\canon1" using UNC to print to a network printer. *sPort* may not contain spaces.

*xScale*:

The x scaling factor for the print. The default value of 0 (zero) automatically scales the print to fit the page. A value of 1 is used to maintain the current horizontal scaling of the image. The outcome is proportional to 1; for example, 0.5 halves the width of the image (optional).

*yScale*:

The y scaling factor for the print. The default value of 0 (zero) automatically scales the print to fit the page. A value of 1 is used to maintain the current horizontal scaling of the image. The outcome is proportional to 1; for example, 0.5 halves the width of the image (optional).

*bSwapBlackWhite*:

Swaps the colors black and white for the purpose of printing pages with a lot of black content. Use the default value of 1 to swap black and white (optional).

*sMap*:

The file name or path of a text based map file used to specify colors to swap when printing. By default Plant SCADA will look in the bin directory for the map file. The format for the map file is:

RRR GGG BBB RRR GGG BBB [HHH] [SSS] [LLL]//  
RRR GGG BBB RRR GGG BBB [HHH] [SSS] [LLL]//

Where:

*RRR* is a decimal red intensity value between 000 and 255.

*GGG* is a decimal green intensity value between 000 to 255.

*BBB* is a decimal blue intensity value between 000 to 255.

*HHH*, *SSS* and *LLL* are optional tolerance values to enable swapping a range of colors. Default values are shown below.

*HHH* is a decimal value representing the Hue tolerance. Valid range is 0 to 360. Default = 0.

*SSS* is a decimal value representing the Saturation tolerance. Valid range is 0 to 255. Default = 2 \* Hue tolerance.

*LLL* is a decimal value representing the Luminance tolerance. Valid range is 0 to 255. Default = 2 \* Hue tolerance.

Leading zeros are not required, however they aid readability.

The first three RGB values specify the FROM color, and the second three RGB values specify the TO color. The tolerance values are applied to the FROM color when replacing individual pixels in the image.

Comments may be placed at the end of each line, or on individual lines, and needs to be proceeded with // or !.

Example map file:

```
043 043 255 155 205 255 025 000 025
//Change dark blue to light blue using Hue and Luminance tolerance of 25.
000 000 000 255 255 255 //Swap black to white with no tolerance
```

**Note:** If swap color ranges overlap, the behavior is undefined (i.e. a color that falls in both ranges may end up as either color).

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[WinPrintFile](#)

## Example

```
WinPrint("LPT3:",0,0,0);
! Prints the active window on printer "LPT3". The print will be
scaled to fit the largest possible page area and will retain the
orientation of the printer, aspect ratio and colors that are
displayed on screen.
WinPrint("LPT3:");
!Prints the page as in the first example, but swaps black and
white on the printout.
```

## See Also

[Window Functions](#)

## WinPrintFile

Prints a file to the system printer. The file needs to be saved with the WinFile() function.

---

**Note:** This function might not work as expected if called directly from the Kernel; if the WinFile function does not succeed, WinPrintFile either doesn't print anything or prints a previously saved page. This function should be called from a graphics page.

---

## Syntax

**WinPrintFile(*sFile*, *sPort* [, *xScale*] [, *yScale*] [, *bSwapBlackWhite*] [, *fromColor*] [, *toColor*] )**

*sFile*:

The file name.

*sPort*:

The name of the printer port to which the window will be printed. This name needs to be enclosed within quotation marks "". For example "LPT1:", to print to the local printer, or "\\Pserver\canon1" using UNC to print to a network printer. *sPort* may not contain spaces

*xScale*:

The x scaling factor for the print. The default value of 0 (zero) automatically scales the print to fit the page. A value of 1 is used to maintain the current horizontal scaling of the image. The outcome is proportional to 1; for example, 0.5 halves the width of the image (optional).

*yScale*:

The y scaling factor for the print. The default value 0 (zero) automatically scales the print to fit the page. A value of 1 is used to maintain the current horizontal scaling of the image. The outcome is proportional to 1; for example, 0.5 halves the width of the image (optional).

*bSwapBlackWhite*:

Swaps the colors black and white for the purpose of printing pages with a lot of black content. Use the default value of 1 to swap black and white (optional).

*fromColor*

The hex RGB color value (0xRRGGBB) to change into *toColor*. The default value of -1 results in no color change (optional).

*toColor*

The hex RGB color value (0xRRGGBB) into which *fromColor* is changed. The default value of -1 results in no color change (optional).

---

**Note:** To change a color, a value needs to be specified for both *fromColor* and *toColor*.

---

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[WinPrint](#)

## Example

```
! Save image to disk then print.
```

```
WinFile("temp");
WinPrintFile("temp", "LPT3:",0,0,0);
! Prints the file "temp" on printer "LPT3". The print will be
scaled to fit the largest possible page area and will retain the
orientation of the printer, aspect ratio and colors that are
displayed on screen.
WinPrintFile("temp","LPT3:");
! Prints the page as in the first example, but swaps black and
white on the printout.
WinPrintFile("temp","LPT3:",0,0,0,0x00FF00,0xFF0000);
! Changes green to red.
WinPrintFile("temp","LPT3:",0,0,1,0x00FF00,0xFF0000);
! Changes green to red and black to white.
```

## See Also

[Window Functions](#)

### WinSelect

Selects a window to make active. This function only affects the output of Cicode functions. It does not change the screen focus of the windows, or move a background window to the foreground.

Always re-select the original window if it is called from a Page database (Page Numbers, Page Symbols, and so on), because other Cicode tasks will assume it is the correct window. This function only changes the active window for the Cicode task that called it.

---

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.

---

## Syntax

**WinSelect(*Window*)**

*Window*:

The window number to select. Be aware that this is not the same as the window handle returned from the WndFind() function.

## Return Value

The old window number.

## Related Functions

[WinGoto](#), [WinNumber](#)

## Example

```
OldWindow=WinSelect(1);
! Selects window number 1.
```

```
Prompt("Message to Window 1");
! Sends message to window number 1.
WinSelect(2);
! Selects window number 2.
Prompt("Message to Window 2");
! Sends message to window number 2.
WinSelect(OldWindow);
! Selects original window.
```

## See Also

[Window Functions](#)

### WinSetName

Associates a name with a particular window by its window number. An association with a window is removed when the window is free.

## Syntax

**WinSetName**(*sName* [, *iWinNum*] )

*sName*:

String name to associate with window number *iWinNum*.

*iWinNum*:

An optional parameter which specifies the window number with which to associate Name. If no number is specified the name is associated with the currently selected window number.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[WinNumber](#)

## See Also

[Window Functions](#)

### WinSize

Sizes the active window. The origin of the window does not move.

**Note:** This function is not supported in the server process in a multiprocessor environment. Calling this function from the server process results in a hardware alarm being raised.

## Syntax

**WinSize(Width, Height, Mode)**

*Width, Height:*

The new width and height of the window, in pixels.

If the target is a pinned window, the new width and height will be measured against the original window size as it is on the unscaled version of the page that hosts it.

*Mode:*

Specifies if the size refers to the viewable area or the total window.

0 - the size to refer to the viewable area.

1 - the size to refer to the total window size.

Default - 1

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[WinMove](#), [WinPos](#)

## Example

```
WinSize(200,100);
! Sizes the active window to 200 pixels wide x 100 pixels high.
```

## See Also

[Window Functions](#)

## WinStyle

Switches on and off scrolling and scrollbar features for existing windows.

## Syntax

**WinStyle(Style, Mode)**

*Style:*

One of the following:

- 1 - Hide horizontal scroll bars.
- 2 - Hide vertical scroll bars.
- 3 - Disable horizontal scrolling.
- 4 - Disable vertical scrolling.

**Mode:**

The mode of the option:

0 - Turn option off.

1 - Turn option on.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[WinNewAt](#)

## Example

To turn horizontal and vertical scroll bars off for the current window:

```
WinStyle(1, 1);  
WinStyle(2, 1);
```

## See Also

[Window Functions](#)

## WinTitle

Sets the title of the active window.

If a window title has been set with the [\[Page\]WinTitle](#) parameter, Plant SCADA uses this title when it refreshes the page (overriding the window title set with the WinTitle() function). To minimize the likelihood of Plant SCADA from overriding the title, set the parameter [\[Page\]WinTitle](#) to \*.

---

**Note:** This function does not work with pinned windows.

---

## Syntax

**WinTitle(*sTitle*)**

*sTitle*:

The new title for the window.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[WinNew](#), [GetWinTitle](#)

## Example

```
WinTitle(Time()+" "+Date());  
! Places the current time and date into the window title.
```

## See Also

[Window Functions](#)

## WndFind

Gets the Windows handle of any window of any application, so that the window can be manipulated. The window handle is not the same as the Plant SCADA window number and cannot be used with functions that expect the Plant SCADA window number (the *Win...* functions).

The window title (caption) needs to be an exact match of the window name (including any blank spaces) for this function to find the window. You should therefore check that the other application does not change the title of the window during execution.

Be aware that if the title banner of a Plant SCADA window is set with the Plant SCADA parameter [\[Page\]WinTitle](#), you should not specify justification (for example, use {TITLE,32,N}). If justification is not disabled (that is the N is omitted), you need to pass the full title of the window (including leading and trailing blanks) to this function.

## Syntax

**WndFind(*sTitle*)**

*sTitle*

The title (caption) of the window.

## Return Value

The window handle. Be aware that this is not the same as a Plant SCADA window number returned from the WinNumber() function.

## Related Functions

[WinNew](#)

## Example

```
hWndExcel=WndFind("Microsoft Excel - Book1");  
! Gets the Windows number of the window titled "Microsoft Excel -  
Book1"
```

## See Also

[Window Functions](#)

## WndGetFileProfile

Gets a profile string from any .ini file.

## Syntax

**WndGetFileProfile(*sGroup*, *sName*, *sDefault*, *sFile*)**

*sGroup*:

The name of the [group].

*sName*:

The name of the variable.

*sDefault*:

The default value.

*sFile*:

The .ini file name.

## Return Value

The profile string from *sFile*.

## Related Functions

[WndPutFileProfile](#)

## Example

```
! get this user startup page from USER.INI File
sStartup =
WndGetFileProfile(Name(), "Startup", "menu", "[Run]:\USER.INI");
PageDisplay(sStartup);
```

## See Also

[Window Functions](#)

## WndInfo

Gets information on the window system (such as the width and height of the various elements displayed by Windows). WndInfo() can also return flags that indicate whether the current version of the Windows operating system is a debugging version, whether a mouse is present, or whether the functions of the left and right mouse buttons have been exchanged.

## Syntax

**WndInfo(*iType*)**

*iType*:

The system measurement to be retrieved. Measurements are in pixels. The system measurement needs to be one of the following values:

0 - Width of the screen.

1 - Height of the screen.

2 - Width of the arrow bitmap on a vertical scroll bar.

3 - Height of the arrow bitmap on a horizontal scroll bar.

4 - Height of the window title. This is the title height plus the height of the window frame that cannot be sized (SM\_CYBORDER).

5 - Width of the window frame that cannot be sized.

6 - Height of the window frame that cannot be sized.

7 - Width of the frame when the window has the WS\_DLFRAME style.

8 - Height of the frame when the window has the WS\_DLFRAME style.

9 - Height of the scroll box on vertical scroll bar.

10 - Width of the scroll box (thumb) on horizontal scroll bar.

12 - Height of the icon.

13 - Width of the cursor.

14 - Height of the cursor.

15 - Height of a single-line menu bar. This is the menu height minus the height of the window frame that cannot be sized (SM\_CYBORDER).

16 - Width of the window client area for a fullscreen window.

17 - Height of the window client area for a fullscreen window (equivalent to the height of the screen minus the height of the window title).

18 - Height of a Kanji window.

19 - Non-zero if the mouse hardware is installed.

20 - Height of arrow bitmap on a vertical scroll bar.

21 - Width of arrow bitmap on a horizontal scroll bar.

22 - Non-zero if the Windows version is a debugging version.

23 - Non-zero if the left and right mouse buttons are swapped.

24-27 - Not Used

28 - Minimum width of the window.

29 - Minimum height of the window.

30 - Width of bitmaps contained in the title bar.

31 - Height of bitmaps contained in the title bar.

32 - Width of the window frame that can be sized.

33 - Height of the window frame that can be sized.

34 - Minimum tracking width of the window.

- 35 - Minimum tracking height of the window.
- 76 - x co-ordinate of upper left corner of virtual screen
- 77 - y co-ordinate of upper left corner of virtual screen
- 78 - width of virtual screen
- 79 - height of virtual screen
- 80 - number of monitors available

## Return Value

The system metric information.

## Example

```
width = WndInfo(0); ! get width of screen
height = WndInfo(1); ! get height of screen
WinPos(width/2, height/2); ! move window to centre of screen
monitors = WndInfo(80); ! get number of monitors available
```

## See Also

[Window Functions](#)

## WndMonitorInfo

Returns information about a particular monitor.

## Syntax

**WndMonitorInfo(*iMonitor*, *iType*)**

*iMonitor*:

Monitor Number 1 to n, where n is the number of monitors returned from WndInfo(80). Using 0 will return the value for the virtual screen. Using an unsupported monitor number (0 or n) will return -1 for all values.

*iType*:

Type The monitor measurement to be retrieved:

0 - x co-ordinate of upper left corner of monitor.

1 - y co-ordinate of upper left corner of monitor.

2 - width of monitor.

3 - height of monitor.

4 - x co-ordinate of upper left corner of monitor working area.

5 - y co-ordinate of upper left corner of monitor working area.

6 - width of monitor working area.

7 - height of monitor working area.

## Return Value

Requested information about the selected monitor.

## See Also

[Window Functions](#)

### WndMonitorInfoEx

Returns information about a particular monitor using the screen name of the screen profile.

## Syntax

**WndMonitorInfoEx(*sMonitor*, *iType*)**

*sMonitor*:

Screen profile's screen name.

*iType*:

Type The monitor measurement to be retrieved:

0 - x co-ordinate of upper left corner of monitor.

1 - y co-ordinate of upper left corner of monitor.

2 - width of monitor.

3 - height of monitor.

4 - x co-ordinate of upper left corner of monitor working area.

5 - y co-ordinate of upper left corner of monitor working area.

6 - width of monitor working area.

7 - height of monitor working area.

## Return Value

Requested information about the selected monitor.

## See Also

[Window Functions](#)

### WndPutFileProfile

Puts a profile string into any .INI file.

## Syntax

**WndPutFileProfile(*sGroup*, *sName*, *sData*, *sFile*)**

*sGroup*:

The name of the [group].

*sName*:

The name of the variable.

*sData*:

The variable data.

*sFile*:

The .INI file name.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[WndGetFileProfile](#)

## Example

```
WndPutFileProfile(Name(), "What", "100", "USER.INI");
```

## See Also

[Window Functions](#)

## WndShow

Sets the display mode of any window of any application.

---

**Note:** This function does not work with pinned windows.

---

## Syntax

**WndShow(*hWnd*, *nMode*)**

*hWnd*:

The Windows handle of the window (returned from the WndFind() function). Be aware that this is not the same as a Plant SCADA window number returned from the WinNumber() function.

*nMode*:

The window mode:

0 - Hide the window.

- 1 - Activate the window in normal mode.
- 2 - Activate the window in an iconized state.
- 3 - Activate the window in a maximized state.
- 4 - Display the window in its previous state without activating it.
- 5 - Activate the window in its current state.
- 6 - Iconize the window.
- 7 - Display the window in an iconized state without activating it.
- 8 - Display the window in its current state without activating it.
- 9 - Activate the window in its previous state.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[WndFind](#)

## Example

```
WndShow(WndFind("Microsoft Excel"), 0);
! Hides the "Microsoft Excel" window.
```

## See Also

[Window Functions](#)

## XML Functions

Following are functions related to XML:

<a href="#">XMLClose</a>	Deletes an XML document in memory.
<a href="#">XMLCreate</a>	Creates a new XML document in memory.
<a href="#">XMLGetAttribute</a>	Retrieves the attribute value of the node from an XML document in memory.
<a href="#">XMLGetAttributeCount</a>	Retrieves the number of attributes (properties of a node. Each attribute has a name and a value) within an XML document in memory.
<a href="#">XMLGetAttributeName</a>	Retrieves the name of an attribute (property of a node. Each attribute has a name and a value) within an XML document in memory.

<a href="#">XMLGetAttributeValue</a>	Retrieves the value of an attribute (property of a node. Each attribute has a name and a value) within an XML document in memory.
<a href="#">XMLGetChild</a>	Retrieves the child node for the specified parent node in XML document in memory.
<a href="#">XMLGetChildCount</a>	Retrieves the total number of child nodes for the specified parent node in an XML document in memory.
<a href="#">XMLGetParent</a>	Retrieves the parent node within the contents of an XML document in memory.
<a href="#">XMLGetRoot</a>	Retrieves the root node of an XML document in memory
<a href="#">XMLNodeAddChild</a>	Creates an element node with the specified Name and Namespace and appends the node to the end of the list of child nodes of specified parent node in the XML document.
<a href="#">XMLNodeFind</a>	Selects a single node from the contents of an XML document in memory.
<a href="#">XMLNodeGetName</a>	Retrieves the name of the specified node.
<a href="#">XMLNodeGetValue</a>	Retrieves the value of a node from the contents of an XML document in memory.
<a href="#">XMLNodeRemove</a>	Removes specified XML node from its parent and XML document.
<a href="#">XMLNodeSetValue</a>	Sets the value of the specified node.
<a href="#">XMLOpen</a>	Loads an XML file from disk.
<a href="#">XMLSave</a>	Saves an XML file to disk.
<a href="#">XMLSetAttribute</a>	Sets the value of specified attribute of the node in the XML document. If the attribute does not exist, it will be created.

## See Also

[Cicode Function Categories](#)[Using Cicode Functions](#)

## XMLClose

Use this function to delete an XML document in memory.

## Syntax

```
INT XMLClose(INT hDoc)
```

*hDoc*

Handle of the XML document. Returned by [XMLCreate](#) or [XMLOpen](#).

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[XMLCreate](#), [XMLGetAttribute](#), [XMLGetAttributeCount](#), [XMLGetAttributeName](#), [XMLGetAttributeValue](#),  
[XMLGetChild](#), [XMLGetChildCount](#), [XMLGetParent](#), [XMLGetRoot](#), [XMLNodeAddChild](#), [XMLNodeFind](#),  
[XMLNodeGetName](#), [XMLNodeGetValue](#), [XMLNodeRemove](#), [XMLNodeSetValue](#), [XMLOpen](#), [XMLSave](#),  
[XMLSetAttribute](#)

## Example

Following example creates an empty XML document and sets root element to shapes and namespace of root element to "http://mycompany.com/shapes/v1", then destroys the XML document.

```
INT hDoc=XMLCreate("shapes","http://mycompany.com/shapes/v1");  
XMLClose(hDoc);
```

## See Also

[XML Functions](#)

## XMLCreate

Use this function to create a new XML document in memory.

## Syntax

```
INT XMLCreate(STRING sRootElement, STRING sNamespace = "")
```

*sRootElement*:

Root element of the XML document.

*sNamespace*:

Optional. Namespace URI of root element.

## Return Value

Handle of XML document, or returns -1 on error

## Related Functions

[XMLClose](#), [XMLGetAttribute](#), [XMLGetAttributeCount](#), [XMLGetAttributeName](#), [XMLGetAttributeValue](#),  
[XMLGetChild](#), [XMLGetChildCount](#), [XMLGetParent](#), [XMLGetRoot](#), [XMLNodeAddChild](#), [XMLNodeFind](#),  
[XMLNodeGetName](#), [XMLNodeGetValue](#), [XMLNodeRemove](#), [XMLNodeSetValue](#), [XMLOpen](#), [XMLSave](#),  
[XMLSetAttribute](#)

## Example

Following example creates an empty XML document and sets root element to shapes and namespace of root element to "http://mycompany.com/shapes/v1".

```
INT hDoc=XMLCreate("shapes","http://mycompany.com/shapes/v1");
```

## See Also

[XML Functions](#)

## XMLGetAttribute

Retrieves the attribute value of the node from an XML document in memory.

## Syntax

```
INT XMLGetAttribute(INT hDoc, INT hNode, STRING sAttributeName)
```

*hDoc*

Handle of the XML document. Returned by [XMLCreate](#) or [XMLOpen](#).

*hNode*

Handle of the node. Returned by [XMLGetRoot](#), [XMLGetChild](#), [XMLGetParent](#) or [XMLNodeFind](#).

*sAttributeName*

Name of attribute.

## Return Value

Value of specified node's attribute

## Related Functions

[XMLClose](#), [XMLCreate](#), [XMLGetAttributeCount](#), [XMLGetAttributeName](#), [XMLGetAttributeValue](#), [XMLGetChild](#),  
[XMLGetChildCount](#), [XMLGetParent](#), [XMLGetRoot](#), [XMLNodeAddChild](#), [XMLNodeFind](#), [XMLNodeGetName](#),  
[XMLNodeGetValue](#), [XMLNodeRemove](#), [XMLNodeSetValue](#), [XMLOpen](#), [XMLSave](#), [XMLSetAttribute](#)

## Example

Following example creates an XML document from local file "H:\Data\books.xml", then selects the first node that matches XPath expression "/bookstore/book/", then gets value of the node's attribute "Language".

```
INT hDoc, hNode;
STRING sValue;
hDoc=XMLOpen("H:\Data\books.xml");
IF hDoc <> -1 THEN
    hNode = XMLNodeFind(hDoc, "/bookstore/book/");
    IF hDoc <> -1 THEN
        sValue=XMLGetAttribute(hDoc,hNode,"Language");
    End
End
```

## See Also

[XML Functions](#)

### XMLGetAttributeCount

Gets number of attribute of specified XML node.

## Syntax

INT **XMLGetAttributeCount**(INT *hDoc*, INT *hNode*)

*hDoc*

Handle of the XML document. Returned by or [XMLOpen](#).

*hNode*

Handle of the node. Returned by [XMLGetRoot](#), [XMLGetChild](#), [XMLGetParent](#) or [XMLNodeFind](#).

## Return Value

Number of attribute of specified node.

## Related Functions

[XMLClose](#), [XMLCreate](#), [XMLGetAttribute](#), [XMLGetAttributeName](#), [XMLGetAttributeValue](#), [XMLGetChild](#),  
[XMLGetChildCount](#), [XMLGetParent](#), [XMLGetRoot](#), [XMLNodeAddChild](#), [XMLNodeFind](#), [XMLNodeGetName](#),  
[XMLNodeGetValue](#), [XMLNodeRemove](#), [XMLNodeSetValue](#), [XMLOpen](#), [XMLSave](#), [XMLSetAttribute](#)

## Example

Following example creates an XML document from local file "H:\Data\books.xml", then selects the first node that matches XPath expression "/bookstore/book/", then gets number of attribute of the node.

```
INT hDoc, hNode;
INT nCount;
```

```
hDoc=XMLOpen("H:\Data\books.xml");
IF hDoc <> -1 THEN
    hNode = XMLNodeFind(hDoc, "/bookstore/book/");
    IF hDoc <> -1 THEN
        nCount=XMLGetAttributeCount(hDoc, hNode);
    End
End
```

## See Also

[XML Functions](#)

### XMLGetAttributeName

Gets name of specified XML Node's attribute by using attribute index.

## Syntax

STRING **XMLGetAttributeName**(INT *hDoc*, INT *hNode*, INT *iAttribute*)

*hDoc*

Handle of the XML document. Returned by [XMLCreate](#) or [XMLOpen](#).

*hNode*

Handle of the node. Returned by [XMLGetRoot](#), [XMLGetChild](#), [XMLGetParent](#) or [XMLNodeFind](#).

*iAttribute*

The attribute index.

## Return Value

Name of specified node's attribute

## Related Functions

[XMLClose](#), [XMLCreate](#), [XMLGetAttribute](#), [XMLGetAttributeCount](#), [XMLGetAttributeValue](#), [XMLGetChild](#),  
[XMLGetChildCount](#), [XMLGetParent](#), [XMLGetRoot](#), [XMLNodeAddChild](#), [XMLNodeFind](#), [XMLNodeGetName](#),  
[XMLNodeGetValue](#), [XMLNodeRemove](#), [XMLNodeSetValue](#), [XMLOpen](#), [XMLSave](#), [XMLSetAttribute](#)

## Example

Following example creates an XML document from local file "H:\Data\books.xml", then selects the first node that matches XPath expression "/bookstore/book/", then gets name of the first attribute of the node.

```
INT hDoc, hNode;
STRING sName;
hDoc=XMLOpen("H:\Data\books.xml");
IF hDoc <> -1 THEN
    hNode = XMLNodeFind(hDoc, "/bookstore/book/");
    IF hNode <> -1 THEN
```

```
sName=XMLGetAttributeName(hDoc,hNode,0);
End
End
```

## See Also

[XML Functions](#)

### XMLGetAttributeValue

Gets the value of the specified XML node's attribute by using attribute index.

## Syntax

```
STRING XMLGetAttributeValue(INT hDoc, INT hNode, INT iAttribute)
```

*hDoc*

Handle of the XML document. Returned by [XMLCreate](#) or [XMLOpen](#).

*hNode*

Handle of the node. Returned by [XMLGetRoot](#), [XMLGetChild](#), [XMLGetParent](#) or [XMLNodeFind](#).

*iAttribute*

The attribute index.

## Return Value

Value of specified node's attribute.

## Related Functions

[XMLClose](#), [XMLCreate](#), [XMLGetAttribute](#), [XMLGetAttributeCount](#), [XMLGetAttributeName](#), [XMLGetChild](#),  
[XMLGetChildCount](#), [XMLGetParent](#), [XMLGetRoot](#), [XMLNodeAddChild](#), [XMLNodeFind](#), [XMLNodeGetName](#),  
[XMLNodeGetValue](#), [XMLNodeRemove](#), [XMLNodeSetValue](#), [XMLOpen](#), [XMLSave](#), [XMLSetAttribute](#)

## Example

Following example creates an XML document from local file "H:\Data\books.xml", then selects the first node that matches XPath expression "/bookstore/book/", then gets value of the first attribute of the node.

```
INT hDoc, hNode;
STRING sValue;
hDoc=XMLOpen("H:\Data\books.xml");
IF hDoc <> -1 THEN
    hNode = XMLNodeFind(hDoc,"/bookstore/book/");
    IF hNode <> -1 THEN
        sValue=XMLGetAttributeValue(hDoc,hNode,0);
    End
End
```

## See Also

[XML Functions](#)

### XMLGetChild

Retrieves the child node for the specified parent node in XML document in memory.

## Syntax

`INT XMLGetChild(INT hDoc, INT hNode, INT iChild)`

*hDoc*

Handle of the XML document the parent node belongs to. Returned by [XMLCreate](#) or [XMLOpen](#).

*hNode*

Handle of the parent node. Returned by [XMLGetRoot](#), [XMLGetChild](#), [XMLGetParent](#) or [XMLNodeFind](#).

*iChild*

The child index.

## Return Value

Handle of child node, or -1 on error

## Related Functions

[XMLClose](#), [XMLCreate](#), [XMLGetAttribute](#), [XMLGetAttributeCount](#), [XMLGetAttributeName](#), [XMLGetAttributeValue](#),  
[XMLGetChildCount](#), [XMLGetParent](#), [XMLGetRoot](#), [XMLNodeAddChild](#), [XMLNodeFind](#), [XMLNodeGetName](#),  
[XMLNodeGetValue](#), [XMLNodeRemove](#), [XMLNodeSetValue](#), [XMLOpen](#), [XMLSave](#), [XMLSetAttribute](#)

## Example

Following example creates an XML document from local file "H:\Data\books.xml", then gets the root node, then gets the first child node of the root node.

```
INT hDoc, hRoot, hChild;  
hDoc = XMLOpen("H:\Data\books.xml");  
IF hDoc <> -1 THEN  
    hRoot = XMLGetRoot(hDoc);  
    IF hRoot <> -1 THEN  
        hChild = XMLGetChild(hDoc,hRoot,0);  
    End  
End
```

## See Also

[XML Functions](#)

## XMLGetChildCount

Retrieves the total number of child nodes for the specified parent node in an XML document in memory.

## Syntax

```
INT XMLGetChildCount(INT hDoc, INT hNode)
```

*hDoc*

Handle of the XML document the parent node belongs to. Returned by [XMLCreate](#) or [XMLOpen](#).

*hNode*

Handle of the parent node. Returned by [XMLGetRoot](#), [XMLGetChild](#), [XMLGetParent](#) or [XMLNodeFind](#).

## Return Value

Number of child nodes, or -1 on error

## Related Functions

[XMLClose](#), [XMLCreate](#), [XMLGetAttribute](#), [XMLGetAttributeCount](#), [XMLGetAttributeName](#), [XMLGetAttributeValue](#),  
[XMLGetChild](#), [XMLGetParent](#), [XMLGetRoot](#), [XMLNodeAddChild](#), [XMLNodeFind](#), [XMLNodeGetName](#),  
[XMLNodeGetValue](#), [XMLNodeRemove](#), [XMLNodeSetValue](#), [XMLOpen](#), [XMLSave](#), [XMLSetAttribute](#)

## Example

Following example creates an XML document from local file "H:\Data\books.xml", then selects the first node that matches XPath expression "/bookstore/book/", then gets the number of children of the node.

```
INT hDoc, hNode;
INT nCount;
hDoc=XMLOpen("H:\Data\books.xml");
IF hDoc <> -1 THEN
    hNode = XMLNodeFind(hDoc, "/bookstore/book/");
    IF hDoc <> -1 THEN
        nCount=XMLGetChildCount(hDoc,hNode);
    End
End
```

## See Also

[XML Functions](#)

## XMLGetParent

Gets the parent of specified node. If specified node is the root node in the tree, the parent is the document node.

## Syntax

```
INT XMLGetParent(INT hDoc, INT hNode)
```

*hDoc*

Handle of the XML document. Returned by [XMLCreate](#) or [XMLOpen](#).

*hNode*

Handle of the node. Returned by [XMLGetRoot](#), [XMLGetChild](#), [XMLGetParent](#) or [XMLNodeFind](#).

## Return Value

Handle of parent node, or error will be returned.

## Related Functions

[XMLClose](#), [XMLCreate](#), [XMLGetAttribute](#), [XMLGetAttributeCount](#), [XMLGetAttributeName](#), [XMLGetAttributeValue](#),  
[XMLGetChild](#), [XMLGetChildCount](#), [XMLGetRoot](#), [XMLNodeAddChild](#), [XMLNodeFind](#), [XMLNodeGetName](#),  
[XMLNodeGetValue](#), [XMLNodeRemove](#), [XMLNodeSetValue](#), [XMLOpen](#), [XMLSave](#), [XMLSetAttribute](#)

## Example

Following example creates an XML document from local file "H:\Data\books.xml", then selects the first node that matches XPath expression "/bookstore/book/", then gets the parent of the node.

```
INT hDoc, hNode, hParent;  
hDoc=XMLOpen("H:\Data\books.xml");  
If hDoc <> -1 THEN  
    hNode=XMLNodeFind(hDoc, "/bookstore/book/");  
    IF hNode <> -1 THEN  
        hParent=XMLGetParent(hDoc, hNode);  
    End  
End
```

## See Also

[XML Functions](#)

## [XMLGetRoot](#)

Gets the root element of the specified XML document.

## Syntax

```
INT XMLGetRoot(INT hDoc)
```

*hDoc*

Handle of the XML document. Returned by [XMLCreate](#) or [XMLOpen](#).

## Return Value

Handle of root element, or -1 on error.

## Related Functions

[XMLClose](#), [XMLCreate](#), [XMLGetAttribute](#), [XMLGetAttributeCount](#), [XMLGetAttributeName](#), [XMLGetAttributeValue](#),  
[XMLGetChild](#), [XMLGetChildCount](#), [XMLGetParent](#), [XMLNodeAddChild](#), [XMLNodeFind](#), [XMLNodeGetName](#),  
[XMLNodeGetValue](#), [XMLNodeRemove](#), [XMLNodeSetValue](#), [XMLOpen](#), [XMLSave](#), [XMLSetAttribute](#)

## Example

Following example creates an XML document from local file "H:\Data\books.xml", then gets the root element.

```
INT hDoc, hRoot;  
hDoc=XMLOpen( "H:\Data\books.xml" );  
If hDoc <> -1 THEN  
    hRoot=XMLGetRoot(hDoc);  
End
```

## See Also

[XML Functions](#)

### XMLNodeAddChild

Creates an element node with the specified Name and Namespace and appends the node to the end of the list of child nodes of specified parent node in the XML document.

## Syntax

INT **XMLNodeAddChild**(INT *hDoc*, INT *hNode*, STRING *sName*[, STRING *sNamespace* = "" ])

*hDoc*

Handle of the XML document the parent node belongs to. Returned by [XMLCreate](#) or [XMLOpen](#).

*hNode*

Handle of the parent node.e.g. <shapes>, <polygons>, <polygon>. Returned by [XMLGetRoot](#), [XMLGetChild](#), [XMLGetParent](#) or [XMLNodeFind](#).

*sName*

The qualified name of the new element node.

*sNamespace*

Optional parameter. The namespace URI of the new node. For example, <shapes xmlns="http://mycompany.com/shapes/v1">.

## Return Value

Handle of the newly created node, or -1 on error.

## Related Functions

[XMLClose](#), [XMLCreate](#), [XMLGetAttribute](#), [XMLGetAttributeCount](#), [XMLGetAttributeName](#), [XMLGetAttributeValue](#),  
[XMLGetChild](#), [XMLGetChildCount](#), [XMLGetParent](#), [XMLGetRoot](#), [XMLNodeFind](#), [XMLNodeGetName](#),  
[XMLNodeGetValue](#), [XMLNodeRemove](#), [XMLNodeSetValue](#), [XMLOpen](#), [XMLSave](#), [XMLSetAttribute](#)

## Example

The example adds a new element node to the root of XML document and then adds two new element nodes to the first created node.

```
INT hDoc, hRoot, hNode;
hDoc = XMLCreate("shapes", "http://mycompany.com/shapes/v1");
IF hDoc <> -1 THEN
    hRoot = XMLGetRoot(hDoc);
    IF hRoot <> -1 THEN
        hNode = XMLNodeAddChild(hDoc, hRoot, "polygons");
        IF hNode <> -1 THEN
            XMLNodeAddChild(hDoc, hNode, "polygon");
            XMLNodeAddChild(hDoc, hNode, "polygon");
        END
    END
END
```

The xml document created by the code above:

```
<?xml version="1.0" encoding="utf-8" ?>
<shapes xmlns="http://mycompany.com/shapes/v1">
<polygons>
<polygon/>
<polygon/>
</polygons>
</shapes>
```

## See Also

[XML Functions](#)

## XMLNodeFind

Use this function to select the first XML node that matches the XPath expression.

## Syntax

**INT XMLNodeFind(INT *hDoc*, STRING *sQuery*)**

*hDoc*

Handle of the XML document. Returned by [XMLCreate](#) or [XMLOpen](#).

*sQuery*

XPath Expression

## Return Value

Handle of the first XML node that matches the XPath expression, or BAD HANDLE on error.

## Related Functions

[XMLClose](#), [XMLCreate](#), [XMLGetAttribute](#), [XMLGetAttributeCount](#), [XMLGetAttributeName](#), [XMLGetAttributeValue](#),  
[XMLGetChild](#), [XMLGetChildCount](#), [XMLGetParent](#), [XMLGetRoot](#), [XMLNodeAddChild](#), [XMLNodeGetName](#),  
[XMLNodeGetValue](#), [XMLNodeRemove](#), [XMLNodeSetValue](#), [XMLOpen](#), [XMLSave](#), [XMLSetAttribute](#)

## Example

Following example creates an XML document from local file "H:\Data\books.xml", then selects the first node that matches XPath expression "/bookstore/book/"

```
INT hDoc, hNode;  
hDoc=XMLOpen("H:\data\books.xml");  
IF hDoc <> -1 THEN  
    hNode=XMLNodeFind(hDoc,"/bookstore/book/");  
End
```

## See Also

[XML Functions](#)

## XMLNodeGetName

Gets the name of the specified node.

## Syntax

STRING **XMLNodeGetName**(INT *hDoc*, INT *hNode*)

*hDoc*

Handle of the XML document. Returned by [XMLCreate](#) or [XMLOpen](#).

*hNode*

Handle of the node. Returned by [XMLGetRoot](#), [XMLGetChild](#), [XMLGetParent](#) or [XMLNodeFind](#).

## Return Value

Name of specified node.

## Related Functions

[XMLClose](#), [XMLCreate](#), [XMLGetAttribute](#), [XMLGetAttributeCount](#), [XMLGetAttributeName](#), [XMLGetAttributeValue](#),  
[XMLGetChild](#), [XMLGetChildCount](#), [XMLGetParent](#), [XMLGetRoot](#), [XMLNodeAddChild](#), [XMLNodeFind](#),  
[XMLNodeGetValue](#), [XMLNodeRemove](#), [XMLNodeSetValue](#), [XMLOpen](#), [XMLSave](#), [XMLSetAttribute](#)

## Example

Following example creates an XML document from local file "H:\Data\books.xml", then selects the first node that matches XPath expression "/bookstore/book/", then gets the name of the node.

```
INT hDoc, hNode;
STRING sName;
hDoc=XMLOpen("H:\Data\books.xml");
IF hDoc <> -1 THEN
    hNode = XMLNodeFind(hDoc, "/bookstore/book/");
    IF hDoc <> -1 THEN
        sName=XMLNodeGetName(hDoc, hNode);
    End
End
```

## See Also

[XML Functions](#)

### XMLNodeGetValue

Gets the value of the specified node.

## Syntax

STRING **XMLNodeGetValue**(INT *hDoc*, INT *hNode*)

*hDoc*

Handle of the XML document. Returned by [XMLCreate](#) or [XMLOpen](#).

*hNode*

Handle of the node. Returned by [XMLGetRoot](#), [XMLGetChild](#), [XMLGetParent](#) or [XMLNodeFind](#).

## Return Value

Value of specified node

## Related Functions

[XMLClose](#), [XMLCreate](#), [XMLGetAttribute](#), [XMLGetAttributeCount](#), [XMLGetAttributeName](#), [XMLGetAttributeValue](#),  
[XMLGetChild](#), [XMLGetChildCount](#), [XMLGetParent](#), [XMLGetRoot](#), [XMLNodeAddChild](#), [XMLNodeFind](#),  
[XMLNodeGetName](#), [XMLNodeRemove](#), [XMLNodeSetValue](#), [XMLOpen](#), [XMLSave](#), [XMLSetAttribute](#)

## Example

Following example creates an XML document from local file "H:\Data\books.xml", then selects the first node that matches XPath expression "/bookstore/book/", then gets the value of the node.

```
INT hDoc, hNode;
STRING sValue;
hDoc=XMLOpen("H:\Data\books.xml");
IF hDoc <> -1 THEN
    hNode = XMLNodeFind(hDoc, "/bookstore/book/");
    IF hDoc <> -1 THEN
        sValue=XMLNodeGetValue(hDoc,hNode);
    End
End
```

## See Also

[XML Functions](#)

### XMLNodeRemove

Removes the specified XML node from its parent and XML document.

## Syntax

INT **XMLNodeRemove**(INT *hDoc*, INT *hNode*)

*hDoc*

Handle of the XML document. Returned by [XMLCreate](#) or [XMLOpen](#).

*hNode*

Handle of the node. Returned by [XMLGetRoot](#), [XMLGetChild](#), [XMLGetParent](#) or [XMLNodeFind](#).

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[XMLClose](#), [XMLCreate](#), [XMLGetAttribute](#), [XMLGetAttributeCount](#), [XMLGetAttributeName](#), [XMLGetAttributeValue](#),  
[XMLGetChild](#), [XMLGetChildCount](#), [XMLGetParent](#), [XMLGetRoot](#), [XMLNodeAddChild](#), [XMLNodeFind](#),  
[XMLNodeGetName](#), [XMLNodeGetValue](#), [XMLNodeSetValue](#), [XMLOpen](#), [XMLSave](#), [XMLSetAttribute](#)

## Example

```
INT hDoc, hNode;
hDoc=XMLOpen("H:\Data\books.xml");
IF hDoc <> -1 THEN
    hNode = XMLNodeFind(hDoc, "/bookstore/book/");
    IF hDoc <> -1 THEN
```

```
    XMLNodeRemove(hDoc, hNode);
End
End
```

## See Also

[XML Functions](#)

### XMLNodeSetValue

Sets the value of the specified node.

## Syntax

```
INT XMLNodeSetValue(INT hDoc, INT hNode, STRING sValue)
```

*hDoc*

Handle of the XML document the parent node belongs to. Returned by [XMLCreate](#) or [XMLOpen](#).

*hNode*

Handle of the node. Returned by [XMLGetRoot](#), [XMLGetChild](#), [XMLGetParent](#), [XMLNodeFind](#).

*sValue*

Value of the element node.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[XMLClose](#), [XMLCreate](#), [XMLGetAttribute](#), [XMLGetAttributeCount](#), [XMLGetAttributeName](#), [XMLGetAttributeValue](#),  
[XMLGetChild](#), [XMLGetChildCount](#), [XMLGetParent](#), [XMLGetRoot](#), [XMLNodeAddChild](#), [XMLNodeFind](#),  
[XMLNodeGetName](#), [XMLNodeGetValue](#), [XMLNodeRemove](#), [XMLOpen](#), [XMLSave](#), [XMLSetAttribute](#)

## Example

The following example creates an XML document from local file "H:\Data\books.xml", then selects the first node that matches XPath expression "/bookstore/book/title" and sets the node's value to "Harry Potter".

```
INT hDoc, hNode;
hDoc = XMLOpen("H:\Data\books.xml");
IF hDoc <> -1 THEN
    hNode=XMLNodeFind(hDoc,"/bookstore/book/title");
    IF hNode <>-1 THEN
        XMLNodeSetValue(hDoc,hNode,"Harry Potter");
    End
End
```

## See Also

[XML Functions](#)

### XMLOpen

Reads an XML file from disk to create XML document.

## Syntax

**INT XMLOpen(STRING *sFilePath*)**

*sFilePath*

Absolute path of XML file

## Return Value

Handle of the XML document, or -1 on error.

## Related Functions

[XMLClose](#), [XMLCreate](#), [XMLGetAttribute](#), [XMLGetAttributeCount](#), [XMLGetAttributeName](#), [XMLGetAttributeValue](#),  
[XMLGetChild](#), [XMLGetChildCount](#), [XMLGetParent](#), [XMLGetRoot](#), [XMLNodeAddChild](#), [XMLNodeFind](#),  
[XMLNodeGetName](#), [XMLNodeGetValue](#), [XMLNodeRemove](#), [XMLNodeSetValue](#), [XMLSave](#), [XMLSetAttribute](#)

## Example

Following example reads disk file "H:\Data\books.xml" to create XML document.

```
INT hDoc, hNode;  
hDoc=XMLOpen("H:\Data\books.xml");
```

## See Also

[XML Functions](#)

### XMLSave

Use this function to save XML document on disk.

## Syntax

**INT XMLSave(INT *hDoc*, STRING *sFilePath*)**

*hDoc*

Handle of the XML document. Returned by XMLCreate or XMLOpen.

*sQuery*

Absolute path of the disk file that XML document is written to.

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[XMLClose](#), [XMLCreate](#), [XMLGetAttribute](#), [XMLGetAttributeCount](#), [XMLGetAttributeName](#), [XMLGetAttributeValue](#),  
[XMLGetChild](#), [XMLGetChildCount](#), [XMLGetParent](#), [XMLGetRoot](#), [XMLNodeAddChild](#), [XMLNodeFind](#),  
[XMLNodeGetName](#), [XMLNodeGetValue](#), [XMLNodeRemove](#), [XMLNodeSetValue](#), [XMLOpen](#), [XMLSetAttribute](#)

## Example

```
INT hDoc, hNode;
hdoc=XMLOpen("H:\Data_Backup\books.xml");
IF hDoc <> -1 THEN
    XMLSave(hDoc, "H:\Data_Backup\books.xml");
END
```

## See Also

[XML Functions](#)

## XMLSetAttribute

Use this function to set the value of specified attribute of the node in the XML document. If the attribute does not exist, it will be created.

## Syntax

INT **XMLSetAttribute**(INT *hDoc*, INT *hNode*, STRING *sName*, STRING *sValue*)

*hDoc*

Handle of the XML document. Returned by [XMLCreate](#) or [XMLOpen](#).

*hNode*

Handle of the node. Returned by [XMLGetRoot](#), [XMLGetChild](#), [XMLGetParent](#) or [XMLNodeFind](#).

*sName*

The name of the attribute.

*sValue*

The value of the attribute

## Return Value

0 (zero) if successful, otherwise an error code is returned.

## Related Functions

[XMLClose](#), [XMLCreate](#), [XMLGetAttribute](#), [XMLGetAttributeCount](#), [XMLGetAttributeName](#), [XMLGetAttributeValue](#),  
[XMLGetChild](#), [XMLGetChildCount](#), [XMLGetParent](#), [XMLGetRoot](#), [XMLNodeAddChild](#), [XMLNodeFind](#),  
[XMLNodeGetName](#), [XMLNodeGetValue](#), [XMLNodeRemove](#), [XMLNodeSetValue](#), [XMLOpen](#), [XMLSave](#)

## Example

Following example creates an XML document from local file "H:\Data\books.xml", then selects the first node that matches XPath expression "/bookstore/book/" and sets the value of the node's attribute "Language" to "English".

```
INT hDoc, hNode;
hDoc = XMLOpen("H:\Data\books.xml");
IF hDoc less -1 THEN
    hNode = XMLNodeFind(hDoc, "/bookstore/book/");
    IF hNode less -1 THEN
        XMLSetAttribute(hDoc, hNode, "Language", "English");
    END
END
```

## See Also

[XML Functions](#)

## Browse Function Field Reference

To view the browse function fields for a function, select the function from the list below.

### Accum Browse Functions

- [AccumBrowseOpen Fields](#)

### Alarm Browse Functions

- [AlarmGetDsp Fields](#)
- [AlarmGetFieldRec Fields](#)
- [AlmBrowseGetField Fields](#)
- [AlmBrowseOpen Fields](#)
- [AlmSummaryOpen Fields](#)
- [AlmTagsOpen Fields](#)

### Equipment Browse Functions

- [EquipBrowseOpen Fields](#)
- [EquipRefBrowseGetField Fields](#)

- [EquipRefBrowseOpen Fields](#)
- [EquipStateBrowseOpen Fields](#)

## Tag Browse Functions

- [TagBrowseOpen Fields](#)

## Trn BrowseFunctions

- [TrnBrowseOpen Fields](#)

## See Also

[Server Browse Function Fields](#)

## AccumBrowseOpen Fields

The fields in the following table can be used with the [AccumBrowseOpen](#) Cicode function.

Field Name	Description	Length	Values
AREA	Alarm area.	16	Numeric value (integer) indicating the security area configured for the accumulator.
CLUSTER	Cluster name.	32	The name of the cluster that runs the accumulator.
EQUIPMENT	Equipment name.	254	The name of the equipment associated with the accumulator.
ITEM	Name of an equipment item.	63	The name of the equipment item with which the accumulator is associated.
NAME	Accumulator name.	80	The name configured for the accumulator.
PRIV	Operator privilege.	16	Numeric value (integer) representing the privilege needed by an operator to perform operations on the accumulator (by using accumulator functions).

Field Name	Description	Length	Values
RUNNING	Running time in seconds.	16	Numeric value (real).
STARTS	Number of starts.	16	Numeric value (real) indicating the number of times the trigger has changed from FALSE to TRUE.
TOTALISER	Running total of value.	16	Numeric value (real).
TRIGGER (accumulator fields)	Last trigger value, used to check for rising edge of the trigger code.	5	0 or 1.
VALUE (accumulator fields)	The value to be added to the totalizer while trigger is true.	16	Numeric value (real).

## See Also

[Browse Function Field Reference](#)

### AlmBrowseGetField Fields

The fields in the following table can be used with the [AlmBrowseGetField](#) Cicode function.

Field Name	Description	Length	Values
ACKDATE	Event acknowledgement date (short format).	12	<p>Short date format (based on the system locale settings). For example, "dd-mm-yy". This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
ACKDATEEXT	Event acknowledgement date (extended format).	12	The date format used when

Field Name	Description	Length	Values
			<p>[Alarm]ExtendedDate is set to 1 in the Citect.ini file.</p> <p>For non-hardware alarms, extended date format (based on the system locale settings).</p> <p>For example, "dd-mm-yyyy".</p> <p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
ACKMILLI	Millisecond precision for the event acknowledgement time.	6	<p>For high resolution time stamped digital and time stamped analog alarms, this value represents the milliseconds part of the acknowledgement time.</p>
ACKTIME	Event acknowledgement time.	12	<p>This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.</p> <p><b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
ACKUTC	Event acknowledgment time in UTC format.	12	<p>Integer that represents a time value based on the number of seconds since Jan 1, 1970 (in UTC format).</p>
ACQDESC	Acquisition error description.	254	Textual representation of an alarm acquisition error.
ACQERROR	Acquisition error.	6	Numeric value (integer).
ALARMTYPE	Alarm type.	16	Text describing the

Field Name	Description	Length	Values
			following alarm types: "Digital", "Analog", "Advanced", "Multi-Digital", "Time Stamped", "Time Stamped Digital", "Time Stamped Analog".
ALMCOMMENT	Alarm comment.	254	A text-based comment added to an alarm during configuration.
AREA	Alarm area.	16	Numeric value (integer) representing area.
ARR_SIZE	Array size.	16	Integer (1 means it is not an array).
CATEGORY	Alarm category.	16	Numeric value (integer) representing the alarm category.
CAUSE1	The first cause defined for the alarm.	254	Up to eight causes can be defined for an alarm to help operators determine the most appropriate course of action.  See <a href="#">Add Cause and Response Information to Alarms</a> .
CAUSE2	The second cause defined for the alarm.	254	
CAUSE3	The third cause defined for the alarm.	254	
CAUSE4	The fourth cause defined for the alarm.	254	
CAUSE5	The fifth cause defined for the alarm.	254	
CAUSE6	The sixth cause defined for the alarm.	254	
CAUSE7	The seventh cause defined for the alarm.	254	
CAUSE8	The eighth cause defined for the alarm.	254	
CLASSIFICATION	The class of the event.	32	The classification applied to an event. For example: Action - The event was

Field Name	Description	Length	Values
			logged due to an action being performed.  Comment - The event was logged due to a comment being created.  Configuration - The event was logged due to configuration changes being made.  System - The event has been logged due to an occurrence that affects the entire system, such as the server being stopped and restarted.  Alarm type - Type of alarm (for example, digital, multi-digital, etc.).
CLUSTER	The cluster to which the tag belongs.	32	The cluster name.
COMMENT	Comment.	64	If hardware alarm = "". Or a configured event description or trend comment.
CONSEQUENCE1	The first consequence defined for an alarm.	254	Up to eight consequences can be defined for an alarm to help operators determine the most appropriate course of action.  See <a href="#">Add Cause and Response Information to Alarms</a> .
CONSEQUENCE2	The second consequence defined for an alarm.	254	
CONSEQUENCE3	The third consequence defined for an alarm.	254	
CONSEQUENCE4	The fourth consequence defined for an alarm.	254	
CONSEQUENCE5	The fifth consequence defined for an alarm.	254	
CONSEQUENCE6	The sixth consequence defined for an alarm.	254	
CONSEQUENCE7	The seventh consequence defined for an alarm.	254	

Field Name	Description	Length	Values
CONSEQUENCE8	The eighth consequence defined for an alarm.	254	
CUSTOM1	Alarm custom field #1	64	A user-defined string.
CUSTOM2	Alarm custom field #2	64	A user-defined string.
CUSTOM3	Alarm custom field #3	64	A user-defined string.
CUSTOM4	Alarm custom field #4	64	A user-defined string.
CUSTOM5	Alarm custom field #5	64	A user-defined string.
CUSTOM6	Alarm custom field #6	64	A user-defined string.
CUSTOM7	Alarm custom field #7	64	A user-defined string.
CUSTOM8	Alarm custom field #8	64	A user-defined string.
DATE	Alarm date in short format.	12	<p>Short date format (based on the system locale settings).  For example, "dd-mm-yy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
DATEEXT	Alarm date in extended format.	12	<p>The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file.  For non-hardware alarms, extended date format (based on the system locale settings).  For example, "dd-mm-yyyy".  This works between UTC</p>

Field Name	Description	Length	Values
			time January 1 1970 - January 18 2038, otherwise "".
DEADBAND	Alarm deadband.	12	For analog, and timestamped analog alarms, a numeric value (real).
DELAY	The delay configured for the alarm.	12	The delay period in the following format: hh:mm:ss The value will be between 0 seconds (00:00:00) and 24 hours (24:00:00).
DELTAMILLI	Millisecond precision for the delta time.	6	For high resolution time stamped digital and time stamped analog alarms, it is the milliseconds part of the delta time.
DELTATIME	The time difference between OnDate/OnTime and OffDate/OffTime.	12	The delta time in the following format: hh:mm:ss
DESC	Alarm description.	254	For analog alarms: "DEVIATION", "RATE OF CHANGE", "LOW", "HIGH", "LOW LOW", "HIGH HIGH". For other alarm types: Configured descriptions.
DEVDELAY	Deviation delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
DEVIATION	Alarm deviation.	12	For analog and time stamped analog alarms: Numeric value (real).
DISABLECOMMENT	A comment added by a user when an alarm is shelved.	254	A user defined text string.

Field Name	Description	Length	Values
DISABLEDDATE	The date an alarm was disabled.	12	<p>The date format is based on the system locale settings.</p> <p>Extended date format: (for example, dd-mm-yyyy).</p> <p>Short date format: (for example, dd-mm-yy).</p> <p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> Extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
DISABLEDTIME	The time an alarm was disabled.	12	<p>A time string that indicates when an alarm was disabled.</p> <p>This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.</p>
DISABLEENDDATE	The disable end date for a shelved alarm (short format).	12	<p>Short date format (based on the system locale settings).</p> <p>For example, "dd-mm-yy".</p> <p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date format is used by default. The short date format only applies when</p>

Field Name	Description	Length	Values
			[Alarm]ExtendedDate is set to 0 in the Citect.ini file.
DISABLEENDDATEEXT	The disable end date for a shelved alarm (extended format).	12	<p>The date format used when [Alarm]ExtendedDate is set to 1 in the Citect.ini file.</p> <p>For non-hardware alarms, extended date format (based on the system locale settings).</p> <p>For example, "dd-mm-yyyy".</p> <p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
DISABLEENDTIME	The disable end time for a shelved alarm.	12	<p>A time string that indicates when a shelved alarm will no longer be disabled.</p> <p>This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.</p>
ENG_ZERO	The engineering zero scale for this value.	11	Numeric value (real).
EQUIPMENT	The name of the equipment associated with the alarm.	254	This can be a name that represents a single piece of equipment, or a name that also reflects the location of the equipment within the equipment hierarchy (where a period (.) is used to indicate levels in the hierarchy).

Field Name	Description	Length	Values
FORMAT	Alarm format.	12	For analog, and time stamped analog alarms, a numeric value (real).
FULLNAME	Event user full name.	20	For non-hardware alarms, the configured full name of the event user if exists. Otherwise "System".
GROUP	Alarm group.	16	For multi-digital alarms, a numeric value
HDELAY	High delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
HELP	Alarm help.	254	The name of the help page.
HHDELAY	High high delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
HIGH	Alarm high	12	For analog and time stamped analog alarms, a numeric value (real).
HIGHHIGH	Alarm high high	12	For analog and time stamped analog alarms, a numeric value (real).
HISTORIAN	Determines if the alarm will be included in an automated configuration process within the Historian environment.	6	True or False (blank).
ITEM	Name of equipment item.	63	The name configured in the alarm's <b>Item Name</b> field.
LDELAY	Low delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
LLDELAY	Low low delay.	16	Delay value (in seconds)

Field Name	Description	Length	Values
			for analog alarms and time stamped analog alarms.
LOCALTIMEDATE	Alarm date and time.	24	yyyy-mm-dd hh:mm:ss[.ttt]. This works between: UTC time January 1 1970 - January 18 2038, otherwise "".
LOGSTATE	Log state text.	13	"ACTIVE", "INACTIVE", "DISABLED", "ENABLED", "ACKNOWLEDGED", "LOW", "HIGH", "LOW LOW", "HIGH HIGH", "RATE", "DEVIATION", "Cleared", "UNACKNOWLEDGED".
LOW	Alarm low.	12	For analog and time stamped analog alarms, a numeric value (real).
LOWLOW	Alarm low low.	12	For analog and time stamped analog alarms, a numeric value (real).
MESSAGE	The event message.	254	Text entered in the <b>Alarm Desc</b> field of digital, advanced and time-stamped alarms.
MILLISEC	Alarm milliseconds.	6	Numeric value (integer).
NAME	Alarm name.	80	A meaningful description of the alarm defined in the <b>Alarm Name</b> property.
NATIVE_COMMENT	Event comment.	64	A text-based comment added to an event on the SOE page at runtime.
NATIVE_DESC	Alarm description.	80	For analog alarms, the alarm state text. Otherwise, the configured alarm description.

Field Name	Description	Length	Values
NATIVE_NAME	Alarm name.	80	Configured alarm name.
NATIVE_SUMDESC	Event description text.	80	For non-hardware analog alarms: the event state string. For non-hardware other alarms: the event [or if it does not exist] the alarm description.
OFFDATE	Event off date (short format).	12	Short date format (based on the system locale settings). For example, "dd-mm-yy". This works between UTC time January 1 1970 - January 18 2038, otherwise "". <b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.
OFFDATEEXT	Extended event off date.	12	The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file. For non-hardware alarms, extended date format (based on the system locale settings). For example, "dd-mm-yyyy". This works between UTC time January 1 1970 - January 18 2038, otherwise "".
OFFMILLI	Millisecond precision for the off time.	6	For high resolution timestamped digital and timestamped analog

Field Name	Description	Length	Values
			alarms, it is the milliseconds part of the off time.
OFFTIME	Event off time.	12	<p>This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.</p> <p><b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
OFFTIMEDATE	Special SQL formatted date and time.	28	<p>For non-hardware alarms: HH:MI:SS.</p> <p><b>Note:</b> The format can be configured in the citect.ini file using the <a href="#">[Alarm]TimeDate</a> parameter.</p>
OFFUTC	Alarm off time in UTC format.	12	Integer that represents a "time" value based on the number of seconds since Jan 1, 1970 (in UTC format not Local).
OLD_DESC	Multi digital old state description.	8	"Invalid", "OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7".
ONDATE	Event on date (short format).	12	<p>Short date format (based on the system locale settings).</p> <p>For example, "dd-mm-yy".</p> <p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date</p>

Field Name	Description	Length	Values
			format is used by default. The short date format only applies when [Alarm]ExtendedDate is set to 0 in the Citect.ini file.
ONDATEEXT	Event on date (extended format).	12	<p>The date format used when [Alarm]ExtendedDate is set to 1 in the Citect.ini file.</p> <p>For non-hardware alarms, extended date format (based on the system locale settings).</p> <p>For example, "dd-mm-yyyy".</p> <p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
ONMILLI	Millisecond precision for the on time.	6	<p>For high resolution timestamped digital and timestamped analog alarms, it is the milliseconds part of the on time.</p>
ONTIME	Event on time.	12	<p>This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.</p> <p><b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
ONTIMEDATE	Special SQL formatted date and time.	28	<p>For non-hardware alarms: HH:MI:SS.</p> <p><b>Note:</b> The format can be</p>

Field Name	Description	Length	Values
			configured in the citect.ini file using the <a href="#">[Alarm]TimeDate</a> parameter.
ONUTC	Alarm on time in UTC format.	12	Integer that represents a "time" value based on the number of seconds since Jan 1, 1970 (in UTC format not Local).
PAGING	Alarm paged flag.	8	A flag to indicate that the alarm is going to be paged. Values are 1 (TRUE) or 0 (FALSE).
PAGINGGROUP	Paging group for alarm.	80	A freeform text field indicating the sequence of people to notify in the event the alarm occurred.
PRIORITY	Alarm category priority.	4	Numeric value (integer).
PRIV	Alarm privilege.	16	Numeric value (integer) that represents the level of privilege an operator requires to acknowledge or disable the alarm.
RATE	Alarm rate.	12	For analog and time stamped analog alarms, a numeric value (real).
RECEIPTLOCALTIMEDATE	The alarm's occurrence time.	28	Displays the alarm's occurrence time (if known), otherwise it displays the receipt time of the alarm.
RECEIPTDATE	The date the master station received the event (short format).  For example, the date the RTU received the event.	12	Short date format (based on the system locale settings).  For example, "dd-mm-yy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".

Field Name	Description	Length	Values
			<b>Note:</b> An extended date format is used by default. The short date format only applies when [Alarm]ExtendedDate is set to 0 in the Citect.ini file.
RECEIPTDATEEXT	The date the master station received the event (extended format).  For example, the date the RTU received the event.	12	The date format used when [Alarm]ExtendedDate is set to 1 in the Citect.ini file.  For non-hardware alarms, extended date format (based on the system locale settings).  For example, "dd-mm-yyyy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".
RECEIPTMILLISEC	Millisecond precision for the receipt time.	6	For high resolution timestamped digital and timestamped analog alarms, it is the milliseconds part of the receipt time.
RECEIPTTIME	The time the master station received the event.  For example, the time the RTU received the event.	16	This can be in 12 hour format (for example, h:mm:ss tt) or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.
RECEIPTTIMEINT	Receipt time in UTC format.	12	An integer that represents a "time" value based on the number of seconds since Jan 1, 1970 (in UTC format not Local).
RECEIPTTIMETICKS	Receipt time measured	20	Time expressed as the

Field Name	Description	Length	Values
	using ticks.		number of ticks elapsed since 12:00:00 midnight, January 1, 1601 in UTC format. Each tick is 100 nanoseconds.
RESPONSE1	The first response defined for the alarm.	254	Up to eight responses can be defined for an alarm to help operators determine the most appropriate course of action.  See <a href="#">Add Cause and Response Information to Alarms</a> .
RESPONSE2	The second response defined for the alarm.	254	
RESPONSE3	The third response defined for the alarm.	254	
RESPONSE4	The fourth response defined for the alarm.	254	
RESPONSE5	The fifth response defined for the alarm.	254	
RESPONSE6	The sixth response defined for the alarm.	254	
RESPONSE7	The seventh response defined for the alarm.	254	
RESPONSE8	The eighth response defined for the alarm.	254	
RESPONSENUM	The number of alarm responses that have been configured for an alarm tag.	8	0 to 8.
SETPOINT	The set point value. This is only applicable to analog alarms.	16	An analog variable tag, expression or base value that determines if a deviation alarm is to be triggered.
STATE	Alarm state string.	16	For analog and time stamped analog alarms: "OFF", "DEVIATION", "RATE OF CHANGE", "LOW", "HIGH", "LOW LOW", "HIGH HIGH".

Field Name	Description	Length	Values
			For multi-digital alarms: "OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7".  For double point status alarms: The configured state name.  For all other alarms: "OFF", "ON".
STATE_DESC	Multi-digital state description.	8	"OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7".
STATE_DESC0	Multi-digital state 0.	64	For multi-digital alarms: "OFF".  For others: "INVALID".
STATE_DESC1	Multi-digital state 1.	64	For multi-digital alarms: "ON".  For others: "INVALID".
STATE_DESC2	Multi-digital state 2.	64	For multi-digital alarms: "ON state 2".  For others: "INVALID".
STATE_DESC3	Multi-digital state 3.	64	For multi-digital alarms: "ON state 3".  For others: "INVALID".
STATE_DESC4	Multi-digital state 4.	64	For multi-digital alarms: "ON state 4".  For others: "INVALID".
STATE_DESC5	Multi-digital state 5.	64	For multi-digital alarms: "ON state 5".  For others: "INVALID".
STATE_DESC6	Multi-digital state 6.	64	For multi-digital alarms: "ON state 6".

Field Name	Description	Length	Values
			For others: "INVALID".
STATE_DESC7	Multi-digital state 7.	64	For multi-digital alarms: "ON state 7".  For others: "INVALID".
SUMDESC	Event description text.	80	Analog alarm - event state text.  Otherwise - configured event [or alarm in case event not defined] description.
SUMSTATE	Event state text.	16	Displays the last active state of the alarm.  "OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7", "DEVIATION", "RATE OF CHANGE", "LOW", "HIGH", "LOW LOW", "HIGH HIGH", "Invalid".
SUMTYPE	Event state.	16	For non-hardware alarms: "DISABLED", "UNACKNOWLEDGED", "ACKNOWLEDGED", "Cleared".
TAG	Tag name.	80	The configured tag name.
TAGEX	Extended alarm tag.	32	<cluster>.<tag> when a cluster is defined.  Otherwise <tag>.
TAGGENLINK	Indicates a tag was imported from an I/O device.	16	Name of the I/O device from which this tag was generated.
TIME	The time the event occurred.	12	This can be in 12 hour format (hh:mm:ss tt), or 24 hour format (HH:mm:ss) depending on the current date and time settings on the local computer.

Field Name	Description	Length	Values
			<b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".
TIMEDATE	Special SQL formatted time.	28	For non-hardware alarms: HH:MI:SS.  <b>Note:</b> The format can be configured in the citect.ini file using the [Alarm]TimeDate parameter.
TIMEINT	Event time in UTC format.	12	Integer that represents a time value based on the number of seconds since Jan 1, 1970 (in UTC format).
TIMETICKS	Event time measured using ticks.	20	Time expressed as the number of ticks elapsed since 12:00:00 midnight, January 1, 1601 in UTC format.  Each tick is 100 nanoseconds.
TSQUALITY	The quality of the alarm tag. This is passed to the alarm server by the I/O server when a value change occurs.	64	Expected time stamp quality strings are:  Time Good, Time Uncertain, Clock Not Synchronized, or empty string if no match is found.
TYPE	Alarm type.	16	"DISABLED", "UNACKNOWLEDGED", "ACKNOWLEDGED", "CLEARED"
TYPENUM	Alarm type represented as a numeric value.	4	Values are:  -1 - Invalid 0 - Digital 1 - Analog 2 - Advanced

Field Name	Description	Length	Values
			3 - Multi-Digital 4 - ArgAna 5 - User event 6 - Time stamped 7 - Hardware 8 - Time stamped digital 9 - Time stamped analog
USERDESC	The text related to a user event.	64	If you specify an asterisk '*' as the first character of the <i>sTag</i> argument in <a href="#">AlarmSumAppend</a> , the summary entry becomes a user event. This value is the text that follows the asterisk.
USERNAME	Event user name.	16	For non-hardware alarms, the configured name of the event user (if it exists). Otherwise "System".
USERLOCATION	The IP of the machine which last raised, modified, or performed an action on the alarm.	80	<b>Note:</b> If the last action performed is from a system machine, the IP will not over ride the last user machine IP address.
VALUE (alarm fields)	Formatted alarm value.	16	For analog alarms, a numeric value formatted according to configuration.

## See Also

[Browse Function Field Reference](#)

### AlarmGetDsp Fields

The fields in the following table can be used with the [AlarmGetDsp](#) Cicode function.

Field Name	Description	Length	Values
ACKDATE	Event acknowledgement date (short format).	12	<p>Short date format (based on the system locale settings). For example, "dd-mm-yy". This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
ACKDATEEXT	Event acknowledgement date (extended format).	12	<p>The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file. For non-hardware alarms, extended date format (based on the system locale settings). For example, "dd-mm-yyyy". This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
ACKMILLI	Millisecond precision for the event acknowledgement time.	6	For high resolution time stamped digital and time stamped analog alarms, this value represents the milliseconds part of the acknowledgement time.
ACKTIME	Event acknowledgement time.	12	This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and

Field Name	Description	Length	Values
			time settings on the local computer. <b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".
ACKUTC	Event acknowledgment time in UTC format.	12	Integer that represents a time value based on the number of seconds since Jan 1, 1970 (in UTC format).
ACQDESC	Acquisition error description.	254	Textual representation of an alarm acquisition error.
ACQERROR	Acquisition error.	6	Numeric value (integer).
ALARMTYPE	Alarm type.	16	Text describing the following alarm types: "Digital", "Analog", "Advanced", "Multi-Digital", "Time Stamped", "Time Stamped Digital", "Time Stamped Analog".
ALMCOMMENT	Alarm comment.	254	A text-based comment added to an alarm during configuration.
AREA	Alarm area.	16	Numeric value (integer) representing area.
ARR_SIZE	Array size.	16	Integer (1 means it is not an array).
CATEGORY	Alarm category.	16	Numeric value (integer) representing the alarm category.
CAUSE1	The first cause defined for the alarm.	254	Up to eight causes can be defined for an alarm to help operators determine the most appropriate course of action.  See <a href="#">Add Cause and Response Information to</a>
CAUSE2	The second cause defined for the alarm.	254	
CAUSE3	The third cause defined for the alarm.	254	

Field Name	Description	Length	Values
CAUSE4	The fourth cause defined for the alarm.	254	<a href="#">Alarms</a> .
CAUSE5	The fifth cause defined for the alarm.	254	
CAUSE6	The sixth cause defined for the alarm.	254	
CAUSE7	The seventh cause defined for the alarm.	254	
CAUSE8	The eighth cause defined for the alarm.	254	
CLASSIFICATION	The class of the event.	32	The classification applied to an event. For example:  Action - The event was logged due to an action being performed.  Comment - The event was logged due to a comment being created.  Configuration - The event was logged due to configuration changes being made.  System - The event has been logged due to an occurrence that affects the entire system, such as the server being stopped and restarted.  Alarm type - Type of alarm (for example, digital, multi-digital, etc.).
CLUSTER	The cluster to which the tag belongs.	32	The cluster name.
COMMENT	Comment.	64	If hardware alarm = "".  Or a configured event description or trend comment.
CONSEQUENCE1	The first consequence	254	Up to eight consequences

Field Name	Description	Length	Values
	defined for an alarm.		can be defined for an alarm to help operators determine the most appropriate course of action. See <a href="#">Add Cause and Response Information to Alarms</a> .
CONSEQUENCE2	The second consequence defined for an alarm.	254	
CONSEQUENCE3	The third consequence defined for an alarm.	254	
CONSEQUENCE4	The fourth consequence defined for an alarm.	254	
CONSEQUENCE5	The fifth consequence defined for an alarm.	254	
CONSEQUENCE6	The sixth consequence defined for an alarm.	254	
CONSEQUENCE7	The seventh consequence defined for an alarm.	254	
CONSEQUENCE8	The eighth consequence defined for an alarm.	254	
CUSTOM1	Alarm custom field #1	64	A user-defined string.
CUSTOM2	Alarm custom field #2	64	A user-defined string.
CUSTOM3	Alarm custom field #3	64	A user-defined string.
CUSTOM4	Alarm custom field #4	64	A user-defined string.
CUSTOM5	Alarm custom field #5	64	A user-defined string.
CUSTOM6	Alarm custom field #6	64	A user-defined string.
CUSTOM7	Alarm custom field #7	64	A user-defined string.
CUSTOM8	Alarm custom field #8	64	A user-defined string.
DATE	Alarm date in short format.	12	Short date format (based on the system locale settings). For example, "dd-mm-yy". This works between UTC time January 1 1970 - January 18 2038, otherwise "". <b>Note:</b> An extended date format is used by default.

Field Name	Description	Length	Values
			The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.
DATEEXT	Alarm date in extended format.	12	<p>The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file.</p> <p>For non-hardware alarms, extended date format (based on the system locale settings).</p> <p>For example, "dd-mm-yyyy".</p> <p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
DEADBAND	Alarm deadband.	12	For analog, and timestamped analog alarms, a numeric value (real).
DELAY	The delay configured for the alarm.	12	<p>The delay period in the following format:</p> <p>hh:mm:ss</p> <p>The value will be between 0 seconds (00:00:00) and 24 hours (24:00:00).</p>
DELTAMILLI	Millisecond precision for the delta time.	6	For high resolution time stamped digital and time stamped analog alarms, it is the milliseconds part of the delta time.
DELTATIME	The time difference between OnDate/OnTime and OffDate/OffTime.	12	The delta time in the following format:
DESC	Alarm description.	254	For analog alarms: "DEVIATION", "RATE OF"

Field Name	Description	Length	Values
			CHANGE", "LOW", "HIGH", "LOW LOW", "HIGH HIGH".  For other alarm types: Configured descriptions.
DEVDELAY	Deviation delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
DEVIATION	Alarm deviation.	12	For analog and time stamped analog alarms: Numeric value (real).
DISABLECOMMENT	A comment added by a user when an alarm is shelved.	254	A user defined text string.
DISABLEDDATE	The date an alarm was disabled.	12	The date format is based on the system locale settings.  Extended date format: (for example, dd-mm-yyyy).  Short date format: (for example, dd-mm-yy).  This works between UTC time January 1 1970 - January 18 2038, otherwise "".  <b>Note:</b> Extended date format is used by default. The short date format only applies when [Alarm]ExtendedDate is set to 0 in the Citect.ini file.
DISABLEDTIME	The time an alarm was disabled.	12	A time string that indicates when an alarm was disabled.  This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour

Field Name	Description	Length	Values
			format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.
DISABLEENDDATE	The disable end date for a shelved alarm (short format).	12	<p>Short date format (based on the system locale settings). For example, "dd-mm-yy". This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
DISABLEENDDATEEXT	The disable end date for a shelved alarm (extended format).	12	<p>The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file. For non-hardware alarms, extended date format (based on the system locale settings). For example, "dd-mm-yyyy". This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
DISABLEENDTIME	The disable end time for a shelved alarm.	12	<p>A time string that indicates when a shelved alarm will no longer be disabled. This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour</p>

Field Name	Description	Length	Values
			format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.
ENG_ZERO	The engineering zero scale for this value.	11	Numeric value (real).
EQUIPMENT	The name of the equipment associated with the alarm.	254	This can be a name that represents a single piece of equipment, or a name that also reflects the location of the equipment within the equipment hierarchy (where a period (.) is used to indicate levels in the hierarchy).
FORMAT	Alarm format.	12	For analog, and time stamped analog alarms, a numeric value (real).
FULLNAME	Event user full name.	20	For non-hardware alarms, the configured full name of the event user if exists. Otherwise "System".
GROUP	Alarm group.	16	For multi-digital alarms, a numeric value
HDELAY	High delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
HELP	Alarm help.	254	The name of the help page.
HHDELAY	High high delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
HIGH	Alarm high	12	For analog and time stamped analog alarms, a numeric value (real).

Field Name	Description	Length	Values
HIGHHIGH	Alarm high high	12	For analog and time stamped analog alarms, a numeric value (real).
HISTORIAN	Determines if the alarm will be included in an automated configuration process within the Historian environment.	6	True or False (blank).
ITEM	Name of equipment item.	63	The name configured in the alarm's <b>Item Name</b> field.
LDELAY	Low delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
LLDELAY	Low low delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
LOCALTIMEDATE	Alarm date and time.	24	yyyy-mm-dd hh:mm:ss[.ttt]. This works between: UTC time January 1 1970 - January 18 2038, otherwise "".
LOGSTATE	Log state text.	13	"ACTIVE", "INACTIVE", "DISABLED", "ENABLED", "ACKNOWLEDGED", "LOW", "HIGH", "LOW LOW", "HIGH HIGH", "RATE", "DEVIATION", "CLEARED", "UNACKNOWLEDGED".
LOW	Alarm low.	12	For analog and time stamped analog alarms, a numeric value (real).
LOWLOW	Alarm low low.	12	For analog and time stamped analog alarms, a numeric value (real).

Field Name	Description	Length	Values
MESSAGE	The event message.	254	Text entered in the <b>Alarm Desc</b> field of digital, advanced and time-stamped alarms.
MILLISEC	Alarm milliseconds.	6	Numeric value (integer).
NAME	Alarm name.	80	A meaningful description of the alarm defined in the <b>Alarm Name</b> property.
NATIVE_COMMENT	Event comment.	64	A text-based comment added to an event on the SOE page at runtime.
NATIVE_DESC	Alarm description.	80	For analog alarms, the alarm state text. Otherwise, the configured alarm description.
NATIVE_NAME	Alarm name.	80	Configured alarm name.
NATIVE_SUMDESC	Event description text.	80	For non-hardware analog alarms: the event state string. For non-hardware other alarms: the event [or if it does not exist] the alarm description.
OFFDATE	Event off date (short format).	12	Short date format (based on the system locale settings). For example, "dd-mm-yy". This works between UTC time January 1 1970 - January 18 2038, otherwise "". <b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.

Field Name	Description	Length	Values
OFFDATEEXT	Extended event off date.	12	<p>The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file.</p> <p>For non-hardware alarms, extended date format (based on the system locale settings).</p> <p>For example, "dd-mm-yyyy".</p> <p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
OFFMILLI	Millisecond precision for the off time.	6	<p>For high resolution timestamped digital and timestamped analog alarms, it is the milliseconds part of the off time.</p>
OFFTIME	Event off time.	12	<p>This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.</p> <p><b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
OFFTIMEDATE	Special SQL formatted date and time.	28	<p>For non-hardware alarms: HH:MI:SS.</p> <p><b>Note:</b> The format can be configured in the citect.ini file using the <a href="#">[Alarm]TimeDate</a> parameter.</p>
OFFUTC	Alarm off time in UTC format.	12	<p>Integer that represents a "time" value based on the</p>

Field Name	Description	Length	Values
			number of seconds since Jan 1, 1970 (in UTC format not Local).
OLD_DESC	Multi digital old state description.	8	"Invalid", "OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7".
ONDATE	Event on date (short format).	12	Short date format (based on the system locale settings). For example, "dd-mm-yy". This works between UTC time January 1 1970 - January 18 2038, otherwise "". <b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.
ONDATEEXT	Event on date (extended format).	12	The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file. For non-hardware alarms, extended date format (based on the system locale settings). For example, "dd-mm-yyyy". This works between UTC time January 1 1970 - January 18 2038, otherwise "".
ONMILLI	Millisecond precision for the on time.	6	For high resolution timestamped digital and timestamped analog alarms, it is the

Field Name	Description	Length	Values
			milliseconds part of the on time.
ONTIME	Event on time.	12	<p>This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.</p> <p><b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
ONTIMEDATE	Special SQL formatted date and time.	28	<p>For non-hardware alarms: HH:MI:SS.</p> <p><b>Note:</b> The format can be configured in the citect.ini file using the <a href="#">[Alarm]TimeDate</a> parameter.</p>
ONUTC	Alarm on time in UTC format.	12	Integer that represents a "time" value based on the number of seconds since Jan 1, 1970 (in UTC format not Local).
PAGING	Alarm paged flag.	8	A flag to indicate that the alarm is going to be paged. Values are 1 (TRUE) or 0 (FALSE).
PAGINGGROUP	Paging group for alarm.	80	A freeform text field indicating the sequence of people to notify in the event the alarm occurred.
PRIORITY	Alarm category priority.	4	Numeric value (integer).
PRIV	Alarm privilege.	16	Numeric value (integer) that represents the level of privilege an operator requires to acknowledge or disable the alarm.

Field Name	Description	Length	Values
RATE	Alarm rate.	12	For analog and time stamped analog alarms, a numeric value (real).
RECEIPTLOCAUTIMEDATE	The alarm's occurrence time.	28	Displays the alarm's occurrence time (if known), otherwise it displays the receipt time of the alarm.
RECEIPTDATE	The date the master station received the event (short format).  For example, the date the RTU received the event.	12	Short date format (based on the system locale settings).  For example, "dd-mm-yy". This works between UTC time January 1 1970 - January 18 2038, otherwise "".  <b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.
RECEIPTDATEEXT	The date the master station received the event (extended format).  For example, the date the RTU received the event.	12	The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file.  For non-hardware alarms, extended date format (based on the system locale settings).  For example, "dd-mm-yyyy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".
RECEIPTMILLISEC	Millisecond precision for the receipt time.	6	For high resolution timestamped digital and timestamped analog alarms, it is the

Field Name	Description	Length	Values
			milliseconds part of the receipt time.
RECEIPTTIME	The time the master station received the event.  For example, the time the RTU received the event.	16	This can be in 12 hour format (for example, h:mm:ss tt) or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.
RECEIPTTIMEINT	Receipt time in UTC format.	12	An integer that represents a "time" value based on the number of seconds since Jan 1, 1970 (in UTC format not Local).
RECEIPTTIMETICKS	Receipt time measured using ticks.	20	Time expressed as the number of ticks elapsed since 12:00:00 midnight, January 1, 1601 in UTC format.  Each tick is 100 nanoseconds.
RESPONSE1	The first response defined for the alarm.	254	Up to eight responses can be defined for an alarm to help operators determine the most appropriate course of action.  See <a href="#">Add Cause and Response Information to Alarms</a> .
RESPONSE2	The second response defined for the alarm.	254	
RESPONSE3	The third response defined for the alarm.	254	
RESPONSE4	The fourth response defined for the alarm.	254	
RESPONSE5	The fifth response defined for the alarm.	254	
RESPONSE6	The sixth response defined for the alarm.	254	
RESPONSE7	The seventh response defined for the alarm.	254	
RESPONSE8	The eighth response	254	

Field Name	Description	Length	Values
	defined for the alarm.		
RESPONSENUM	The number of alarm responses that have been configured for an alarm tag.	8	0 to 8.
SETPOINT	The set point value. This is only applicable to analog alarms.	16	An analog variable tag, expression or base value that determines if a deviation alarm is to be triggered.
STATE	Alarm state string.	16	For analog and time stamped analog alarms: "OFF", "DEVIATION", "RATE OF CHANGE", "LOW", "HIGH", "LOW LOW", "HIGH HIGH".  For multi-digital alarms: "OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7".  For double point status alarms: The configured state name.  For all other alarms: "OFF", "ON".
STATE_DESC	Multi-digital state description.	8	"OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7".
STATE_DESC0	Multi-digital state 0.	64	For multi-digital alarms: "OFF".  For others: "INVALID".
STATE_DESC1	Multi-digital state 1.	64	For multi-digital alarms: "ON".  For others: "INVALID".

Field Name	Description	Length	Values
STATE_DESC2	Multi-digital state 2.	64	For multi-digital alarms: "ON state 2". For others: "INVALID".
STATE_DESC3	Multi-digital state 3.	64	For multi-digital alarms: "ON state 3". For others: "INVALID".
STATE_DESC4	Multi-digital state 4.	64	For multi-digital alarms: "ON state 4". For others: "INVALID".
STATE_DESC5	Multi-digital state 5.	64	For multi-digital alarms: "ON state 5". For others: "INVALID".
STATE_DESC6	Multi-digital state 6.	64	For multi-digital alarms: "ON state 6". For others: "INVALID".
STATE_DESC7	Multi-digital state 7.	64	For multi-digital alarms: "ON state 7". For others: "INVALID".
SUMDESC	Event description text.	80	Analog alarm - event state text.  Otherwise - configured event [or alarm in case event not defined] description.
SUMSTATE	Event state text.	16	Displays the last active state of the alarm.  "OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7", "DEVIATION", "RATE OF CHANGE", "LOW", "HIGH", "LOW LOW", "HIGH HIGH", "Invalid".
SUMTYPE	Event state.	16	For non-hardware alarms: "DISABLED", "UNACKNOWLEDGED", "ACKNOWLEDGED",

Field Name	Description	Length	Values
			"CLEARED".
TAG	Tag name.	80	The configured tag name.
TAGEX	Extended alarm tag.	32	<cluster>.<tag> when a cluster is defined. Otherwise <tag>.
TAGGENLINK	Indicates a tag was imported from an I/O device.	16	Name of the I/O device from which this tag was generated.
TIME	The time the event occurred.	12	This can be in 12 hour format (hh:mm:ss tt), or 24 hour format (HH:mm:ss) depending on the current date and time settings on the local computer.  <b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".
TIMEDATE	Special SQL formatted time.	28	For non-hardware alarms: HH:MI:SS.  <b>Note:</b> The format can be configured in the citect.ini file using the [Alarm]TimeDate parameter.
TIMEINT	Event time in UTC format.	12	Integer that represents a time value based on the number of seconds since Jan 1, 1970 (in UTC format).
TIMETICKS	Event time measured using ticks.	20	Time expressed as the number of ticks elapsed since 12:00:00 midnight, January 1, 1601 in UTC format.  Each tick is 100 nanoseconds.

Field Name	Description	Length	Values
TSQUALITY	The quality of the alarm tag. This is passed to the alarm server by the I/O server when a value change occurs.	64	Expected time stamp quality strings are: Time Good, Time Uncertain, Clock Not Synchronized, or empty string if no match is found.
TYPE	Alarm type.	16	"DISABLED", "UNACKNOWLEDGED", "ACKNOWLEDGED", "Cleared"
TYPENUM	Alarm type represented as a numeric value.	4	Values are: -1 - Invalid 0 - Digital 1 - Analog 2 - Advanced 3 - Multi-Digital 4 - ArgAna 5 - User event 6 - Time stamped 7 - Hardware 8 - Time stamped digital 9 - Time stamped analog
USERDESC	The text related to a user event.	64	If you specify an asterisk '*' as the first character of the <i>sTag</i> argument in <a href="#">AlarmSumAppend</a> , the summary entry becomes a user event. This value is the text that follows the asterisk.
USERNAME	Event user name.	16	For non-hardware alarms, the configured name of the event user (if it exists). Otherwise "System".
USERLOCATION	The IP of the machine which last raised, modified, or performed	80	<b>Note:</b> If the last action performed is from a system machine, the IP

Field Name	Description	Length	Values
	an action on the alarm.		will not over ride the last user machine IP address.
VALUE (alarm fields)	Formatted alarm value.	16	For analog alarms, a numeric value formatted according to configuration.

## See Also

[Browse Function Field Reference](#)

### AlarmGetFieldRec Fields

The fields in the following table can be used with the [AlarmGetFieldRec](#) Cicode function.

Field Name	Description	Length	Values
ACKDATE	Event acknowledgement date (short format).	12	<p>Short date format (based on the system locale settings).            For example, "dd-mm-yy".            This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
ACKDATEEXT	Event acknowledgement date (extended format).	12	<p>The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file.            For non-hardware alarms, extended date format (based on the system locale settings).            For example, "dd-mm-yyyy".</p>

Field Name	Description	Length	Values
			This works between UTC time January 1 1970 - January 18 2038, otherwise "".
ACKMILLI	Millisecond precision for the event acknowledgement time.	6	For high resolution time stamped digital and time stamped analog alarms, this value represents the milliseconds part of the acknowledgement time.
ACKTIME	Event acknowledgement time.	12	This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.  <b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".
ACKUTC	Event acknowledgment time in UTC format.	12	Integer that represents a time value based on the number of seconds since Jan 1, 1970 (in UTC format).
ACQDESC	Acquisition error description.	254	Textual representation of an alarm acquisition error.
ACQERROR	Acquisition error.	6	Numeric value (integer).
ALARMTYPE	Alarm type.	16	Text describing the following alarm types: "Digital", "Analog", "Advanced", "Multi-Digital", "Time Stamped", "Time Stamped Digital", "Time Stamped Analog".
ALMCOMMENT	Alarm comment.	254	A text-based comment added to an alarm during configuration.

Field Name	Description	Length	Values
AREA	Alarm area.	16	Numeric value (integer) representing area.
ARR_SIZE	Array size.	16	Integer (1 means it is not an array).
CATEGORY	Alarm category.	16	Numeric value (integer) representing the alarm category.
CAUSE1	The first cause defined for the alarm.	254	Up to eight causes can be defined for an alarm to help operators determine the most appropriate course of action.  See <a href="#">Add Cause and Response Information to Alarms</a> .
CAUSE2	The second cause defined for the alarm.	254	
CAUSE3	The third cause defined for the alarm.	254	
CAUSE4	The fourth cause defined for the alarm.	254	
CAUSE5	The fifth cause defined for the alarm.	254	
CAUSE6	The sixth cause defined for the alarm.	254	
CAUSE7	The seventh cause defined for the alarm.	254	
CAUSE8	The eighth cause defined for the alarm.	254	
CLASSIFICATION	The class of the event.	32	The classification applied to an event. For example:  Action - The event was logged due to an action being performed.  Comment - The event was logged due to a comment being created.  Configuration - The event was logged due to configuration changes being made.  System - The event has

Field Name	Description	Length	Values
			been logged due to an occurrence that affects the entire system, such as the server being stopped and restarted. Alarm type - Type of alarm (for example, digital, multi-digital, etc.).
CLUSTER	The cluster to which the tag belongs.	32	The cluster name.
COMMENT	Comment.	64	If hardware alarm = "". Or a configured event description or trend comment.
CONSEQUENCE1	The first consequence defined for an alarm.	254	Up to eight consequences can be defined for an alarm to help operators determine the most appropriate course of action.  See <a href="#">Add Cause and Response Information to Alarms</a> .
CONSEQUENCE2	The second consequence defined for an alarm.	254	
CONSEQUENCE3	The third consequence defined for an alarm.	254	
CONSEQUENCE4	The fourth consequence defined for an alarm.	254	
CONSEQUENCE5	The fifth consequence defined for an alarm.	254	
CONSEQUENCE6	The sixth consequence defined for an alarm.	254	
CONSEQUENCE7	The seventh consequence defined for an alarm.	254	
CONSEQUENCE8	The eighth consequence defined for an alarm.	254	
CUSTOM1	Alarm custom field #1	64	A user-defined string.
CUSTOM2	Alarm custom field #2	64	A user-defined string.
CUSTOM3	Alarm custom field #3	64	A user-defined string.
CUSTOM4	Alarm custom field #4	64	A user-defined string.

Field Name	Description	Length	Values
CUSTOM5	Alarm custom field #5	64	A user-defined string.
CUSTOM6	Alarm custom field #6	64	A user-defined string.
CUSTOM7	Alarm custom field #7	64	A user-defined string.
CUSTOM8	Alarm custom field #8	64	A user-defined string.
DATE	Alarm date in short format.	12	<p>Short date format (based on the system locale settings).  For example, "dd-mm-yy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
DATEEXT	Alarm date in extended format.	12	<p>The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file.  For non-hardware alarms, extended date format (based on the system locale settings).  For example, "dd-mm-yyyy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
DEADBAND	Alarm deadband.	12	For analog, and timestamped analog alarms, a numeric value (real).
DELAY	The delay configured for	12	The delay period in the

Field Name	Description	Length	Values
	the alarm.		following format: hh:mm:ss  The value will be between 0 seconds (00:00:00) and 24 hours (24:00:00).
DELTAMILLI	Millisecond precision for the delta time.	6	For high resolution time stamped digital and time stamped analog alarms, it is the milliseconds part of the delta time.
DELTATIME	The time difference between OnDate/OnTime and OffDate/OffTime.	12	The delta time in the following format: hh:mm:ss
DESC	Alarm description.	254	For analog alarms: "DEVIATION", "RATE OF CHANGE", "LOW", "HIGH", "LOW LOW", "HIGH HIGH".  For other alarm types: Configured descriptions.
DEVDELAY	Deviation delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
DEVIATION	Alarm deviation.	12	For analog and time stamped analog alarms: Numeric value (real).
DISABLECOMMENT	A comment added by a user when an alarm is shelved.	254	A user defined text string.
DISABLEDDATE	The date an alarm was disabled.	12	The date format is based on the system locale settings.  Extended date format: (for example, dd-mm-yyyy).  Short date format: (for example, dd-mm-yy).

Field Name	Description	Length	Values
			<p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> Extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
DISABLETIME	The time an alarm was disabled.	12	<p>A time string that indicates when an alarm was disabled.</p> <p>This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.</p>
DISABLEENDDATE	The disable end date for a shelved alarm (short format).	12	<p>Short date format (based on the system locale settings).</p> <p>For example, "dd-mm-yy".</p> <p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
DISABLEENDDATEEXT	The disable end date for a shelved alarm (extended format).	12	<p>The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file.</p>

Field Name	Description	Length	Values
			For non-hardware alarms, extended date format (based on the system locale settings). For example, "dd-mm-yyyy". This works between UTC time January 1 1970 - January 18 2038, otherwise "".
DISABLEENDTIME	The disable end time for a shelved alarm.	12	A time string that indicates when a shelved alarm will no longer be disabled. This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.
ENG_ZERO	The engineering zero scale for this value.	11	Numeric value (real).
EQUIPMENT	The name of the equipment associated with the alarm.	254	This can be a name that represents a single piece of equipment, or a name that also reflects the location of the equipment within the equipment hierarchy (where a period (.) is used to indicate levels in the hierarchy).
FORMAT	Alarm format.	12	For analog, and time stamped analog alarms, a numeric value (real).
FULLNAME	Event user full name.	20	For non-hardware alarms, the configured full name of the event user if exists. Otherwise "System".
GROUP	Alarm group.	16	For multi-digital alarms, a

Field Name	Description	Length	Values
			numeric value
HDELAY	High delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
HELP	Alarm help.	254	The name of the help page.
HHDELAY	High high delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
HIGH	Alarm high	12	For analog and time stamped analog alarms, a numeric value (real).
HIGHHIGH	Alarm high high	12	For analog and time stamped analog alarms, a numeric value (real).
HISTORIAN	Determines if the alarm will be included in an automated configuration process within the Historian environment.	6	True or False (blank).
ITEM	Name of equipment item.	63	The name configured in the alarm's <b>Item Name</b> field.
LDELAY	Low delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
LLDELAY	Low low delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
LOCALTIMEDATE	Alarm date and time.	24	yyyy-mm-dd hh:mm:ss[.ttt]. This works between: UTC time January 1 1970 - January 18 2038,

Field Name	Description	Length	Values
			otherwise "".
LOGSTATE	Log state text.	13	"ACTIVE", "INACTIVE", "DISABLED", "ENABLED", "ACKNOWLEDGED", "LOW", "HIGH", "LOW LOW", "HIGH HIGH", "RATE", "DEVIATION", "CLEARED", "UNACKNOWLEDGED".
LOW	Alarm low.	12	For analog and time stamped analog alarms, a numeric value (real).
LOWLOW	Alarm low low.	12	For analog and time stamped analog alarms, a numeric value (real).
MESSAGE	The event message.	254	Text entered in the <b>Alarm Desc</b> field of digital, advanced and time-stamped alarms.
MILLISEC	Alarm milliseconds.	6	Numeric value (integer).
NAME	Alarm name.	80	A meaningful description of the alarm defined in the <b>Alarm Name</b> property.
NATIVE_COMMENT	Event comment.	64	A text-based comment added to an event on the SOE page at runtime.
NATIVE_DESC	Alarm description.	80	For analog alarms, the alarm state text. Otherwise, the configured alarm description.
NATIVE_NAME	Alarm name.	80	Configured alarm name.
NATIVE_SUMDESC	Event description text.	80	For non-hardware analog alarms: the event state string. For non-hardware other alarms: the event [or if it does not exist] the alarm

Field Name	Description	Length	Values
	description.		
OFFDATE	Event off date (short format).	12	<p>Short date format (based on the system locale settings).  For example, "dd-mm-yy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
OFFDATEEXT	Extended event off date.	12	<p>The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file.  For non-hardware alarms, extended date format (based on the system locale settings).  For example, "dd-mm-yyyy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
OFFMILLI	Millisecond precision for the off time.	6	For high resolution timestamped digital and timestamped analog alarms, it is the milliseconds part of the off time.
OFFTIME	Event off time.	12	This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on

Field Name	Description	Length	Values
			<p>the regional date and time settings on the local computer.</p> <p><b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
OFFTIMEDATE	Special SQL formatted date and time.	28	<p>For non-hardware alarms: HH:MI:SS.</p> <p><b>Note:</b> The format can be configured in the citect.ini file using the <a href="#">[Alarm]TimeDate</a> parameter.</p>
OFFUTC	Alarm off time in UTC format.	12	<p>Integer that represents a "time" value based on the number of seconds since Jan 1, 1970 (in UTC format not Local).</p>
OLD_DESC	Multi digital old state description.	8	<p>"Invalid", "OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7".</p>
ONDATE	Event on date (short format).	12	<p>Short date format (based on the system locale settings).</p> <p>For example, "dd-mm-yy".</p> <p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
ONDATEEXT	Event on date (extended format).	12	The date format used when

Field Name	Description	Length	Values
			<p>[Alarm]ExtendedDate is set to 1 in the Citect.ini file.</p> <p>For non-hardware alarms, extended date format (based on the system locale settings).</p> <p>For example, "dd-mm-yyyy".</p> <p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
ONMILLI	Millisecond precision for the on time.	6	<p>For high resolution timestamped digital and timestamped analog alarms, it is the milliseconds part of the on time.</p>
ONTIME	Event on time.	12	<p>This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.</p> <p><b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
ONTIMEDATE	Special SQL formatted date and time.	28	<p>For non-hardware alarms: HH:MI:SS.</p> <p><b>Note:</b> The format can be configured in the citect.ini file using the [Alarm]TimeDate parameter.</p>
ONUTC	Alarm on time in UTC format.	12	<p>Integer that represents a "time" value based on the number of seconds since Jan 1, 1970 (in UTC format)</p>

Field Name	Description	Length	Values
			not Local).
PAGING	Alarm paged flag.	8	A flag to indicate that the alarm is going to be paged. Values are 1 (TRUE) or 0 (FALSE).
PAGINGGROUP	Paging group for alarm.	80	A freeform text field indicating the sequence of people to notify in the event the alarm occurred.
PRIORITY	Alarm category priority.	4	Numeric value (integer).
PRIV	Alarm privilege.	16	Numeric value (integer) that represents the level of privilege an operator requires to acknowledge or disable the alarm.
RATE	Alarm rate.	12	For analog and time stamped analog alarms, a numeric value (real).
RECEIPTLOCALTIMEDATE	The alarm's occurrence time.	28	Displays the alarm's occurrence time (if known), otherwise it displays the receipt time of the alarm.
RECEIPTDATE	The date the master station received the event (short format).  For example, the date the RTU received the event.	12	Short date format (based on the system locale settings).  For example, "dd-mm-yy". This works between UTC time January 1 1970 - January 18 2038, otherwise "".  <b>Note:</b> An extended date format is used by default. The short date format only applies when [Alarm]ExtendedDate is set to 0 in the Citect.ini file.
RECEIPTDATEEXT	The date the master	12	The date format used

Field Name	Description	Length	Values
	<p>station received the event (extended format).</p> <p>For example, the date the RTU received the event.</p>		<p>when [Alarm]ExtendedDate is set to 1 in the Citect.ini file.</p> <p>For non-hardware alarms, extended date format (based on the system locale settings).</p> <p>For example, "dd-mm-yyyy".</p> <p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
RECEIPTMILLISEC	Millisecond precision for the receipt time.	6	For high resolution timestamped digital and timestamped analog alarms, it is the milliseconds part of the receipt time.
RECEIPTTIME	<p>The time the master station received the event.</p> <p>For example, the time the RTU received the event.</p>	16	This can be in 12 hour format (for example, h:mm:ss tt) or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.
RECEIPTTIMEINT	Receipt time in UTC format.	12	An integer that represents a "time" value based on the number of seconds since Jan 1, 1970 (in UTC format not Local).
RECEIPTTIMETICKS	Receipt time measured using ticks.	20	<p>Time expressed as the number of ticks elapsed since 12:00:00 midnight, January 1, 1601 in UTC format.</p> <p>Each tick is 100 nanoseconds.</p>
RESPONSE1	The first response defined	254	Up to eight responses can

Field Name	Description	Length	Values
	for the alarm.		be defined for an alarm to help operators determine the most appropriate course of action.
RESPONSE2	The second response defined for the alarm.	254	See <a href="#">Add Cause and Response Information to Alarms</a> .
RESPONSE3	The third response defined for the alarm.	254	
RESPONSE4	The fourth response defined for the alarm.	254	
RESPONSE5	The fifth response defined for the alarm.	254	
RESPONSE6	The sixth response defined for the alarm.	254	
RESPONSE7	The seventh response defined for the alarm.	254	
RESPONSE8	The eighth response defined for the alarm.	254	
RESPONSENUM	The number of alarm responses that have been configured for an alarm tag.	8	0 to 8.
SETPOINT	The set point value. This is only applicable to analog alarms.	16	An analog variable tag, expression or base value that determines if a deviation alarm is to be triggered.
STATE	Alarm state string.	16	For analog and time stamped analog alarms: "OFF", "DEVIATION", "RATE OF CHANGE", "LOW", "HIGH", "LOW LOW", "HIGH HIGH". For multi-digital alarms: "OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7". For double point status

Field Name	Description	Length	Values
			alarms: The configured state name. For all other alarms: "OFF", "ON".
STATE_DESC	Multi-digital state description.	8	"OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7".
STATE_DESC0	Multi-digital state 0.	64	For multi-digital alarms: "OFF". For others: "INVALID".
STATE_DESC1	Multi-digital state 1.	64	For multi-digital alarms: "ON". For others: "INVALID".
STATE_DESC2	Multi-digital state 2.	64	For multi-digital alarms: "ON state 2". For others: "INVALID".
STATE_DESC3	Multi-digital state 3.	64	For multi-digital alarms: "ON state 3". For others: "INVALID".
STATE_DESC4	Multi-digital state 4.	64	For multi-digital alarms: "ON state 4". For others: "INVALID".
STATE_DESC5	Multi-digital state 5.	64	For multi-digital alarms: "ON state 5". For others: "INVALID".
STATE_DESC6	Multi-digital state 6.	64	For multi-digital alarms: "ON state 6". For others: "INVALID".
STATE_DESC7	Multi-digital state 7.	64	For multi-digital alarms: "ON state 7". For others: "INVALID".
SUMDESC	Event description text.	80	Analog alarm - event state

Field Name	Description	Length	Values
			text. Otherwise - configured event [or alarm in case event not defined] description.
SUMSTATE	Event state text.	16	Displays the last active state of the alarm. "OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7", "DEVIATION", "RATE OF CHANGE", "LOW", "HIGH", "LOW LOW", "HIGH HIGH", "Invalid".
SUMTYPE	Event state.	16	For non-hardware alarms: "DISABLED", "UNACKNOWLEDGED", "ACKNOWLEDGED", "Cleared".
TAG	Tag name.	80	The configured tag name.
TAGEX	Extended alarm tag.	32	<cluster>.<tag> when a cluster is defined. Otherwise <tag>.
TAGGENLINK	Indicates a tag was imported from an I/O device.	16	Name of the I/O device from which this tag was generated.
TIME	The time the event occurred.	12	This can be in 12 hour format (hh:mm:ss tt), or 24 hour format (HH:mm:ss) depending on the current date and time settings on the local computer. <b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".
TIMEDATE	Special SQL formatted time.	28	For non-hardware alarms: HH:MI:SS.

Field Name	Description	Length	Values
			<b>Note:</b> The format can be configured in the citect.ini file using the [Alarm]TimeDate parameter.
TIMEINT	Event time in UTC format.	12	Integer that represents a time value based on the number of seconds since Jan 1, 1970 (in UTC format).
TIMETICKS	Event time measured using ticks.	20	Time expressed as the number of ticks elapsed since 12:00:00 midnight, January 1, 1601 in UTC format. Each tick is 100 nanoseconds.
TSQUALITY	The quality of the alarm tag. This is passed to the alarm server by the I/O server when a value change occurs.	64	Expected time stamp quality strings are: Time Good, Time Uncertain, Clock Not Synchronized, or empty string if no match is found.
TYPE	Alarm type.	16	"DISABLED", "UNACKNOWLEDGED", "ACKNOWLEDGED", "CLEARED"
TYPENUM	Alarm type represented as a numeric value.	4	Values are: -1 - Invalid 0 - Digital 1 - Analog 2 - Advanced 3 - Multi-Digital 4 - ArgAna 5 - User event 6 - Time stamped 7 - Hardware 8 - Time stamped digital

Field Name	Description	Length	Values
			9 - Time stamped analog
USERDESC	The text related to a user event.	64	If you specify an asterisk '*' as the first character of the <i>sTag</i> argument in <a href="#">AlarmSumAppend</a> , the summary entry becomes a user event. This value is the text that follows the asterisk.
USERNAME	Event user name.	16	For non-hardware alarms, the configured name of the event user (if it exists). Otherwise "System".
USERLOCATION	The IP of the machine which last raised, modified, or performed an action on the alarm.	80	<b>Note:</b> If the last action performed is from a system machine, the IP will not over ride the last user machine IP address.
VALUE (alarm fields)	Formatted alarm value.	16	For analog alarms, a numeric value formatted according to configuration.

## See Also

[Browse Function Field Reference](#)

## AlmBrowseOpen Fields

The fields in the following table can be used with the [AlmBrowseOpen](#) Cicode function.

Field Name	Description	Length	Values
ACKDATE	Event acknowledgement date (short format).	12	Short date format (based on the system locale settings). For example, "dd-mm-yy". This works between UTC time January 1 1970 - January 18 2038, otherwise "".

Field Name	Description	Length	Values
			<b>Note:</b> An extended date format is used by default. The short date format only applies when [Alarm]ExtendedDate is set to 0 in the Citect.ini file.
ACKDATEEXT	Event acknowledgement date (extended format).	12	The date format used when [Alarm]ExtendedDate is set to 1 in the Citect.ini file.  For non-hardware alarms, extended date format (based on the system locale settings).  For example, "dd-mm-yyyy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".
ACKMILLI	Millisecond precision for the event acknowledgement time.	6	For high resolution time stamped digital and time stamped analog alarms, this value represents the milliseconds part of the acknowledgement time.
ACKTIME	Event acknowledgement time.	12	This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.  <b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".
ACKUTC	Event acknowledgment time in UTC format.	12	Integer that represents a time value based on the

Field Name	Description	Length	Values
			number of seconds since Jan 1, 1970 (in UTC format).
ACQDESC	Acquisition error description.	254	Textual representation of an alarm acquisition error.
ACQERROR	Acquisition error.	6	Numeric value (integer).
ALARMTYPE	Alarm type.	16	Text describing the following alarm types: "Digital", "Analog", "Advanced", "Multi-Digital", "Time Stamped", "Time Stamped Digital", "Time Stamped Analog".
ALMCOMMENT	Alarm comment.	254	A text-based comment added to an alarm during configuration.
AREA	Alarm area.	16	Numeric value (integer) representing area.
ARR_SIZE	Array size.	16	Integer (1 means it is not an array).
CATEGORY	Alarm category.	16	Numeric value (integer) representing the alarm category.
CAUSE1	The first cause defined for the alarm.	254	Up to eight causes can be defined for an alarm to help operators determine the most appropriate course of action.  See <a href="#">Add Cause and Response Information to Alarms</a> .
CAUSE2	The second cause defined for the alarm.	254	
CAUSE3	The third cause defined for the alarm.	254	
CAUSE4	The fourth cause defined for the alarm.	254	
CAUSE5	The fifth cause defined for the alarm.	254	
CAUSE6	The sixth cause defined for the alarm.	254	

Field Name	Description	Length	Values
CAUSE7	The seventh cause defined for the alarm.	254	
CAUSE8	The eighth cause defined for the alarm.	254	
CLASSIFICATION	The class of the event.	32	The classification applied to an event. For example: Action - The event was logged due to an action being performed. Comment - The event was logged due to a comment being created. Configuration - The event was logged due to configuration changes being made. System - The event has been logged due to an occurrence that affects the entire system, such as the server being stopped and restarted. Alarm type - Type of alarm (for example, digital, multi-digital, etc.).
CLUSTER	The cluster to which the tag belongs.	32	The cluster name.
COMMENT	Comment.	64	If hardware alarm = "". Or a configured event description or trend comment.
CONSEQUENCE1	The first consequence defined for an alarm.	254	Up to eight consequences can be defined for an alarm to help operators determine the most appropriate course of action.  See <a href="#">Add Cause and Response Information to Alarms</a> .
CONSEQUENCE2	The second consequence defined for an alarm.	254	
CONSEQUENCE3	The third consequence defined for an alarm.	254	
CONSEQUENCE4	The fourth consequence	254	

Field Name	Description	Length	Values
	defined for an alarm.		
CONSEQUENCE5	The fifth consequence defined for an alarm.	254	
CONSEQUENCE6	The sixth consequence defined for an alarm.	254	
CONSEQUENCE7	The seventh consequence defined for an alarm.	254	
CONSEQUENCE8	The eighth consequence defined for an alarm.	254	
CUSTOM1	Alarm custom field #1	64	A user-defined string.
CUSTOM2	Alarm custom field #2	64	A user-defined string.
CUSTOM3	Alarm custom field #3	64	A user-defined string.
CUSTOM4	Alarm custom field #4	64	A user-defined string.
CUSTOM5	Alarm custom field #5	64	A user-defined string.
CUSTOM6	Alarm custom field #6	64	A user-defined string.
CUSTOM7	Alarm custom field #7	64	A user-defined string.
CUSTOM8	Alarm custom field #8	64	A user-defined string.
DATE	Alarm date in short format.	12	<p>Short date format (based on the system locale settings).  For example, "dd-mm-yy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
DATEEXT	Alarm date in extended format.	12	The date format used when

Field Name	Description	Length	Values
			<p>[Alarm]ExtendedDate is set to 1 in the Citect.ini file.</p> <p>For non-hardware alarms, extended date format (based on the system locale settings).</p> <p>For example, "dd-mm-yyyy".</p> <p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
DEADBAND	Alarm deadband.	12	For analog, and timestamped analog alarms, a numeric value (real).
DELAY	The delay configured for the alarm.	12	<p>The delay period in the following format:</p> <p>hh:mm:ss</p> <p>The value will be between 0 seconds (00:00:00) and 24 hours (24:00:00).</p>
DELTAMILLI	Millisecond precision for the delta time.	6	For high resolution time stamped digital and time stamped analog alarms, it is the milliseconds part of the delta time.
DELTATIME	The time difference between OnDate/OnTime and OffDate/OffTime.	12	<p>The delta time in the following format:</p> <p>hh:mm:ss</p>
DESC	Alarm description.	254	<p>For analog alarms:</p> <p>"DEVIATION", "RATE OF CHANGE", "LOW", "HIGH", "LOW LOW", "HIGH HIGH".</p> <p>For other alarm types:</p> <p>Configured descriptions.</p>
DEVDELAY	Deviation delay.	16	Delay value (in seconds)

Field Name	Description	Length	Values
			for analog alarms and time stamped analog alarms.
DEVIATION	Alarm deviation.	12	For analog and time stamped analog alarms: Numeric value (real).
DISABLECOMMENT	A comment added by a user when an alarm is shelved.	254	A user defined text string.
DISABLEDDATE	The date an alarm was disabled.	12	<p>The date format is based on the system locale settings.</p> <p>Extended date format: (for example, dd-mm-yyyy).</p> <p>Short date format: (for example, dd-mm-yy).</p> <p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> Extended date format is used by default. The short date format only applies when [Alarm]ExtendedDate is set to 0 in the Citect.ini file.</p>
DISABLEDTIME	The time an alarm was disabled.	12	<p>A time string that indicates when an alarm was disabled.</p> <p>This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.</p>
DISABLEENDDATE	The disable end date for a	12	Short date format (based

Field Name	Description	Length	Values
	shelved alarm (short format).		<p>on the system locale settings).</p> <p>For example, "dd-mm-yy". This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
DISABLEENDDATEEXT	The disable end date for a shelved alarm (extended format).	12	<p>The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file.</p> <p>For non-hardware alarms, extended date format (based on the system locale settings).</p> <p>For example, "dd-mm-yyyy".</p> <p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
DISABLEENDTIME	The disable end time for a shelved alarm.	12	<p>A time string that indicates when a shelved alarm will no longer be disabled.</p> <p>This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.</p>
ENG_ZERO	The engineering zero	11	Numeric value (real).

Field Name	Description	Length	Values
	scale for this value.		
EQUIPMENT	The name of the equipment associated with the alarm.	254	This can be a name that represents a single piece of equipment, or a name that also reflects the location of the equipment within the equipment hierarchy (where a period (.) is used to indicate levels in the hierarchy).
FORMAT	Alarm format.	12	For analog, and time stamped analog alarms, a numeric value (real).
FULLNAME	Event user full name.	20	For non-hardware alarms, the configured full name of the event user if exists. Otherwise "System".
GROUP	Alarm group.	16	For multi-digital alarms, a numeric value
HDELAY	High delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
HELP	Alarm help.	254	The name of the help page.
HHDELAY	High high delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
HIGH	Alarm high	12	For analog and time stamped analog alarms, a numeric value (real).
HIGHHIGH	Alarm high high	12	For analog and time stamped analog alarms, a numeric value (real).
HISTORIAN	Determines if the alarm will be included in an automated configuration	6	True or False (blank).

Field Name	Description	Length	Values
	process within the Historian environment.		
ITEM	Name of equipment item.	63	The name configured in the alarm's <b>Item Name</b> field.
LDELAY	Low delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
LLDELAY	Low low delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
LOCALTIMEDATE	Alarm date and time.	24	yyyy-mm-dd hh:mm:ss[.ttt]. This works between: UTC time January 1 1970 - January 18 2038, otherwise "".
LOGSTATE	Log state text.	13	"ACTIVE", "INACTIVE", "DISABLED", "ENABLED", "ACKNOWLEDGED", "LOW", "HIGH", "LOW LOW", "HIGH HIGH", "RATE", "DEVIATION", "CLEARED", "UNACKNOWLEDGED".
LOW	Alarm low.	12	For analog and time stamped analog alarms, a numeric value (real).
LOWLOW	Alarm low low.	12	For analog and time stamped analog alarms, a numeric value (real).
MESSAGE	The event message.	254	Text entered in the <b>Alarm Desc</b> field of digital, advanced and time-stamped alarms.
MILLISEC	Alarm milliseconds.	6	Numeric value (integer).

Field Name	Description	Length	Values
NAME	Alarm name.	80	A meaningful description of the alarm defined in the <b>Alarm Name</b> property.
NATIVE_COMMENT	Event comment.	64	A text-based comment added to an event on the SOE page at runtime.
NATIVE_DESC	Alarm description.	80	For analog alarms, the alarm state text. Otherwise, the configured alarm description.
NATIVE_NAME	Alarm name.	80	Configured alarm name.
NATIVE_SUMDESC	Event description text.	80	For non-hardware analog alarms: the event state string. For non-hardware other alarms: the event [or if it does not exist] the alarm description.
OFFDATE	Event off date (short format).	12	Short date format (based on the system locale settings). For example, "dd-mm-yy". This works between UTC time January 1 1970 - January 18 2038, otherwise "". <b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.
OFFDATEEXT	Extended event off date.	12	The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file. For non-hardware alarms,

Field Name	Description	Length	Values
			<p>extended date format (based on the system locale settings).  For example, "dd-mm-yyyy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
OFFMILLI	Millisecond precision for the off time.	6	<p>For high resolution timestamped digital and timestamped analog alarms, it is the milliseconds part of the off time.</p>
OFFTIME	Event off time.	12	<p>This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.  <b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
OFFTIMEDATE	Special SQL formatted date and time.	28	<p>For non-hardware alarms: HH:MI:SS.  <b>Note:</b> The format can be configured in the citect.ini file using the <a href="#">[Alarm]TimeDate</a> parameter.</p>
OFFUTC	Alarm off time in UTC format.	12	<p>Integer that represents a "time" value based on the number of seconds since Jan 1, 1970 (in UTC format not Local).</p>
OLD_DESC	Multi digital old state description.	8	"Invalid", "OFF", "ON", "ON State 2", "ON State

Field Name	Description	Length	Values
			3", "ON State 4", "ON State 5", "ON State 6", "ON State 7".
ONDATE	Event on date (short format).	12	<p>Short date format (based on the system locale settings).  For example, "dd-mm-yy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
ONDATEEXT	Event on date (extended format).	12	<p>The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file.  For non-hardware alarms, extended date format (based on the system locale settings).  For example, "dd-mm-yyyy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
ONMILLI	Millisecond precision for the on time.	6	For high resolution timestamped digital and timestamped analog alarms, it is the milliseconds part of the on time.
ONTIME	Event on time.	12	This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour

Field Name	Description	Length	Values
			format (for example, H:mm:ss) depending on the regional date and time settings on the local computer. <b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".
ONTIMEDATE	Special SQL formatted date and time.	28	For non-hardware alarms: HH:MI:SS. <b>Note:</b> The format can be configured in the citect.ini file using the <a href="#">[Alarm]TimeDate</a> parameter.
ONUTC	Alarm on time in UTC format.	12	Integer that represents a "time" value based on the number of seconds since Jan 1, 1970 (in UTC format not Local).
PAGING	Alarm paged flag.	8	A flag to indicate that the alarm is going to be paged. Values are 1 (TRUE) or 0 (FALSE).
PAGINGGROUP	Paging group for alarm.	80	A freeform text field indicating the sequence of people to notify in the event the alarm occurred.
PRIORITY	Alarm category priority.	4	Numeric value (integer).
PRIV	Alarm privilege.	16	Numeric value (integer) that represents the level of privilege an operator requires to acknowledge or disable the alarm.
RATE	Alarm rate.	12	For analog and time stamped analog alarms, a numeric value (real).
RECEIPTLOCALTIMEDATE	The alarm's occurrence	28	Displays the alarm's

Field Name	Description	Length	Values
	time.		occurrence time (if known), otherwise it displays the receipt time of the alarm.
RECEIPTDATE	The date the master station received the event (short format).  For example, the date the RTU received the event.	12	Short date format (based on the system locale settings).  For example, "dd-mm-yy". This works between UTC time January 1 1970 - January 18 2038, otherwise "".  <b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.
RECEIPTDATEEXT	The date the master station received the event (extended format).  For example, the date the RTU received the event.	12	The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file.  For non-hardware alarms, extended date format (based on the system locale settings).  For example, "dd-mm-yyyy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".
RECEIPTMILLISEC	Millisecond precision for the receipt time.	6	For high resolution timestamped digital and timestamped analog alarms, it is the milliseconds part of the receipt time.
RECEIPTTIME	The time the master station received the	16	This can be in 12 hour format (for example,

Field Name	Description	Length	Values
	event. For example, the time the RTU received the event.		h:mm:ss tt) or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.
RECEIPTTIMEINT	Receipt time in UTC format.	12	An integer that represents a "time" value based on the number of seconds since Jan 1, 1970 (in UTC format not Local).
RECEIPTTIMETICKS	Receipt time measured using ticks.	20	Time expressed as the number of ticks elapsed since 12:00:00 midnight, January 1, 1601 in UTC format. Each tick is 100 nanoseconds.
RESPONSE1	The first response defined for the alarm.	254	Up to eight responses can be defined for an alarm to help operators determine the most appropriate course of action.  See <a href="#">Add Cause and Response Information to Alarms</a> .
RESPONSE2	The second response defined for the alarm.	254	
RESPONSE3	The third response defined for the alarm.	254	
RESPONSE4	The fourth response defined for the alarm.	254	
RESPONSE5	The fifth response defined for the alarm.	254	
RESPONSE6	The sixth response defined for the alarm.	254	
RESPONSE7	The seventh response defined for the alarm.	254	
RESPONSE8	The eighth response defined for the alarm.	254	
RESPONSENUM	The number of alarm responses that have been configured for an alarm	8	0 to 8.

Field Name	Description	Length	Values
	tag.		
SETPOINT	The set point value. This is only applicable to analog alarms.	16	An analog variable tag, expression or base value that determines if a deviation alarm is to be triggered.
STATE	Alarm state string.	16	For analog and time stamped analog alarms: "OFF", "DEVIATION", "RATE OF CHANGE", "LOW", "HIGH", "LOW LOW", "HIGH HIGH".  For multi-digital alarms: "OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7".  For double point status alarms: The configured state name.  For all other alarms: "OFF", "ON".
STATE_DESC	Multi-digital state description.	8	"OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7".
STATE_DESC0	Multi-digital state 0.	64	For multi-digital alarms: "OFF".  For others: "INVALID".
STATE_DESC1	Multi-digital state 1.	64	For multi-digital alarms: "ON".  For others: "INVALID".
STATE_DESC2	Multi-digital state 2.	64	For multi-digital alarms: "ON state 2".  For others: "INVALID".

Field Name	Description	Length	Values
STATE_DESC3	Multi-digital state 3.	64	For multi-digital alarms: "ON state 3". For others: "INVALID".
STATE_DESC4	Multi-digital state 4.	64	For multi-digital alarms: "ON state 4". For others: "INVALID".
STATE_DESC5	Multi-digital state 5.	64	For multi-digital alarms: "ON state 5". For others: "INVALID".
STATE_DESC6	Multi-digital state 6.	64	For multi-digital alarms: "ON state 6". For others: "INVALID".
STATE_DESC7	Multi-digital state 7.	64	For multi-digital alarms: "ON state 7". For others: "INVALID".
SUMDESC	Event description text.	80	Analog alarm - event state text.  Otherwise - configured event [or alarm in case event not defined] description.
SUMSTATE	Event state text.	16	Displays the last active state of the alarm. "OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7", "DEVIATION", "RATE OF CHANGE", "LOW", "HIGH", "LOW LOW", "HIGH HIGH", "Invalid".
SUMTYPE	Event state.	16	For non-hardware alarms: "DISABLED", "UNACKNOWLEDGED", "ACKNOWLEDGED", "CLEARED".
TAG	Tag name.	80	The configured tag name.

Field Name	Description	Length	Values
TAGEX	Extended alarm tag.	32	<cluster>.<tag> when a cluster is defined. Otherwise <tag>.
TAGGENLINK	Indicates a tag was imported from an I/O device.	16	Name of the I/O device from which this tag was generated.
TIME	The time the event occurred.	12	This can be in 12 hour format (hh:mm:ss tt), or 24 hour format (HH:mm:ss) depending on the current date and time settings on the local computer. <b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".
TIMEDATE	Special SQL formatted time.	28	For non-hardware alarms: HH:MI:SS. <b>Note:</b> The format can be configured in the citect.ini file using the [Alarm]TimeDate parameter.
TIMEINT	Event time in UTC format.	12	Integer that represents a time value based on the number of seconds since Jan 1, 1970 (in UTC format).
TIMETICKS	Event time measured using ticks.	20	Time expressed as the number of ticks elapsed since 12:00:00 midnight, January 1, 1601 in UTC format. Each tick is 100 nanoseconds.

Field Name	Description	Length	Values
TSQUALITY	The quality of the alarm tag. This is passed to the alarm server by the I/O server when a value change occurs.	64	Expected time stamp quality strings are: Time Good, Time Uncertain, Clock Not Synchronized, or empty string if no match is found.
TYPE	Alarm type.	16	"DISABLED", "UNACKNOWLEDGED", "ACKNOWLEDGED", "Cleared"
TYPENUM	Alarm type represented as a numeric value.	4	Values are: -1 - Invalid 0 - Digital 1 - Analog 2 - Advanced 3 - Multi-Digital 4 - ArgAna 5 - User event 6 - Time stamped 7 - Hardware 8 - Time stamped digital 9 - Time stamped analog
USERDESC	The text related to a user event.	64	If you specify an asterisk '*' as the first character of the <i>sTag</i> argument in <a href="#">AlarmSumAppend</a> , the summary entry becomes a user event. This value is the text that follows the asterisk.
USERNAME	Event user name.	16	For non-hardware alarms, the configured name of the event user (if it exists). Otherwise "System".
USERLOCATION	The IP of the machine which last raised, modified, or performed	80	<b>Note:</b> If the last action performed is from a system machine, the IP

Field Name	Description	Length	Values
	an action on the alarm.		will not over ride the last user machine IP address.
VALUE (alarm fields)	Formatted alarm value.	16	For analog alarms, a numeric value formatted according to configuration.

## See Also

[Browse Function Field Reference](#)

### AlmSummaryOpen Fields

The fields in the following table can be used with the [AlmSummaryOpen](#) Cicode function.

Field Name	Description	Length	Values
ACKDATE	Event acknowledgement date (short format).	12	<p>Short date format (based on the system locale settings).  For example, "dd-mm-yy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
ACKDATEEXT	Event acknowledgement date (extended format).	12	<p>The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file.  For non-hardware alarms, extended date format (based on the system locale settings).  For example, "dd-mm-yyyy".</p>

Field Name	Description	Length	Values
			This works between UTC time January 1 1970 - January 18 2038, otherwise "".
ACKMILLI	Millisecond precision for the event acknowledgement time.	6	For high resolution time stamped digital and time stamped analog alarms, this value represents the milliseconds part of the acknowledgement time.
ACKTIME	Event acknowledgement time.	12	This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.  <b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".
ACKUTC	Event acknowledgment time in UTC format.	12	Integer that represents a time value based on the number of seconds since Jan 1, 1970 (in UTC format).
ACQDESC	Acquisition error description.	254	Textual representation of an alarm acquisition error.
ACQERROR	Acquisition error.	6	Numeric value (integer).
ALARMTYPE	Alarm type.	16	Text describing the following alarm types: "Digital", "Analog", "Advanced", "Multi-Digital", "Time Stamped", "Time Stamped Digital", "Time Stamped Analog".
ALMCOMMENT	Alarm comment.	254	A text-based comment added to an alarm during configuration.

Field Name	Description	Length	Values
AREA	Alarm area.	16	Numeric value (integer) representing area.
ARR_SIZE	Array size.	16	Integer (1 means it is not an array).
CATEGORY	Alarm category.	16	Numeric value (integer) representing the alarm category.
CAUSE1	The first cause defined for the alarm.	254	Up to eight causes can be defined for an alarm to help operators determine the most appropriate course of action.  See <a href="#">Add Cause and Response Information to Alarms</a> .
CAUSE2	The second cause defined for the alarm.	254	
CAUSE3	The third cause defined for the alarm.	254	
CAUSE4	The fourth cause defined for the alarm.	254	
CAUSE5	The fifth cause defined for the alarm.	254	
CAUSE6	The sixth cause defined for the alarm.	254	
CAUSE7	The seventh cause defined for the alarm.	254	
CAUSE8	The eighth cause defined for the alarm.	254	
CLASSIFICATION	The class of the event.	32	The classification applied to an event. For example:  Action - The event was logged due to an action being performed.  Comment - The event was logged due to a comment being created.  Configuration - The event was logged due to configuration changes being made.  System - The event has

Field Name	Description	Length	Values
			been logged due to an occurrence that affects the entire system, such as the server being stopped and restarted. Alarm type - Type of alarm (for example, digital, multi-digital, etc.).
CLUSTER	The cluster to which the tag belongs.	32	The cluster name.
COMMENT	Comment.	64	If hardware alarm = "". Or a configured event description or trend comment.
CONSEQUENCE1	The first consequence defined for an alarm.	254	Up to eight consequences can be defined for an alarm to help operators determine the most appropriate course of action.  See <a href="#">Add Cause and Response Information to Alarms</a> .
CONSEQUENCE2	The second consequence defined for an alarm.	254	
CONSEQUENCE3	The third consequence defined for an alarm.	254	
CONSEQUENCE4	The fourth consequence defined for an alarm.	254	
CONSEQUENCE5	The fifth consequence defined for an alarm.	254	
CONSEQUENCE6	The sixth consequence defined for an alarm.	254	
CONSEQUENCE7	The seventh consequence defined for an alarm.	254	
CONSEQUENCE8	The eighth consequence defined for an alarm.	254	
CUSTOM1	Alarm custom field #1	64	A user-defined string.
CUSTOM2	Alarm custom field #2	64	A user-defined string.
CUSTOM3	Alarm custom field #3	64	A user-defined string.
CUSTOM4	Alarm custom field #4	64	A user-defined string.

Field Name	Description	Length	Values
CUSTOM5	Alarm custom field #5	64	A user-defined string.
CUSTOM6	Alarm custom field #6	64	A user-defined string.
CUSTOM7	Alarm custom field #7	64	A user-defined string.
CUSTOM8	Alarm custom field #8	64	A user-defined string.
DATE	Alarm date in short format.	12	<p>Short date format (based on the system locale settings).  For example, "dd-mm-yy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
DATEEXT	Alarm date in extended format.	12	<p>The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file.  For non-hardware alarms, extended date format (based on the system locale settings).  For example, "dd-mm-yyyy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
DEADBAND	Alarm deadband.	12	For analog, and timestamped analog alarms, a numeric value (real).
DELAY	The delay configured for	12	The delay period in the

Field Name	Description	Length	Values
	the alarm.		following format: hh:mm:ss  The value will be between 0 seconds (00:00:00) and 24 hours (24:00:00).
DELTAMILLI	Millisecond precision for the delta time.	6	For high resolution time stamped digital and time stamped analog alarms, it is the milliseconds part of the delta time.
DELTATIME	The time difference between OnDate/OnTime and OffDate/OffTime.	12	The delta time in the following format: hh:mm:ss
DESC	Alarm description.	254	For analog alarms: "DEVIATION", "RATE OF CHANGE", "LOW", "HIGH", "LOW LOW", "HIGH HIGH".  For other alarm types: Configured descriptions.
DEVDELAY	Deviation delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
DEVIATION	Alarm deviation.	12	For analog and time stamped analog alarms: Numeric value (real).
DISABLECOMMENT	A comment added by a user when an alarm is shelved.	254	A user defined text string.
DISABLEDDATE	The date an alarm was disabled.	12	The date format is based on the system locale settings.  Extended date format: (for example, dd-mm-yyyy).  Short date format: (for example, dd-mm-yy).

Field Name	Description	Length	Values
			<p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> Extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
DISABLETIME	The time an alarm was disabled.	12	<p>A time string that indicates when an alarm was disabled.</p> <p>This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.</p>
DISABLEENDDATE	The disable end date for a shelved alarm (short format).	12	<p>Short date format (based on the system locale settings).</p> <p>For example, "dd-mm-yy".</p> <p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
DISABLEENDDATEEXT	The disable end date for a shelved alarm (extended format).	12	<p>The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file.</p>

Field Name	Description	Length	Values
			For non-hardware alarms, extended date format (based on the system locale settings). For example, "dd-mm-yyyy". This works between UTC time January 1 1970 - January 18 2038, otherwise "".
DISABLEENDTIME	The disable end time for a shelved alarm.	12	A time string that indicates when a shelved alarm will no longer be disabled. This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.
ENG_ZERO	The engineering zero scale for this value.	11	Numeric value (real).
EQUIPMENT	The name of the equipment associated with the alarm.	254	This can be a name that represents a single piece of equipment, or a name that also reflects the location of the equipment within the equipment hierarchy (where a period (.) is used to indicate levels in the hierarchy).
FORMAT	Alarm format.	12	For analog, and time stamped analog alarms, a numeric value (real).
FULLNAME	Event user full name.	20	For non-hardware alarms, the configured full name of the event user if exists. Otherwise "System".
GROUP	Alarm group.	16	For multi-digital alarms, a

Field Name	Description	Length	Values
			numeric value
HDELAY	High delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
HELP	Alarm help.	254	The name of the help page.
HHDELAY	High high delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
HIGH	Alarm high	12	For analog and time stamped analog alarms, a numeric value (real).
HIGHHIGH	Alarm high high	12	For analog and time stamped analog alarms, a numeric value (real).
HISTORIAN	Determines if the alarm will be included in an automated configuration process within the Historian environment.	6	True or False (blank).
ITEM	Name of equipment item.	63	The name configured in the alarm's <b>Item Name</b> field.
LDELAY	Low delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
LLDELAY	Low low delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
LOCALTIMEDATE	Alarm date and time.	24	yyyy-mm-dd hh:mm:ss[.ttt]. This works between: UTC time January 1 1970 - January 18 2038,

Field Name	Description	Length	Values
			otherwise "".
LOGSTATE	Log state text.	13	"ACTIVE", "INACTIVE", "DISABLED", "ENABLED", "ACKNOWLEDGED", "LOW", "HIGH", "LOW LOW", "HIGH HIGH", "RATE", "DEVIATION", "CLEARED", "UNACKNOWLEDGED".
LOW	Alarm low.	12	For analog and time stamped analog alarms, a numeric value (real).
LOWLOW	Alarm low low.	12	For analog and time stamped analog alarms, a numeric value (real).
MESSAGE	The event message.	254	Text entered in the <b>Alarm Desc</b> field of digital, advanced and time-stamped alarms.
MILLISEC	Alarm milliseconds.	6	Numeric value (integer).
NAME	Alarm name.	80	A meaningful description of the alarm defined in the <b>Alarm Name</b> property.
NATIVE_COMMENT	Event comment.	64	A text-based comment added to an event on the SOE page at runtime.
NATIVE_DESC	Alarm description.	80	For analog alarms, the alarm state text. Otherwise, the configured alarm description.
NATIVE_NAME	Alarm name.	80	Configured alarm name.
NATIVE_SUMDESC	Event description text.	80	For non-hardware analog alarms: the event state string. For non-hardware other alarms: the event [or if it does not exist] the alarm

Field Name	Description	Length	Values
			description.
OFFDATE	Event off date (short format).	12	<p>Short date format (based on the system locale settings).  For example, "dd-mm-yy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
OFFDATEEXT	Extended event off date.	12	<p>The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file.  For non-hardware alarms, extended date format (based on the system locale settings).  For example, "dd-mm-yyyy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
OFFMILLI	Millisecond precision for the off time.	6	For high resolution timestamped digital and timestamped analog alarms, it is the milliseconds part of the off time.
OFFTIME	Event off time.	12	This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on

Field Name	Description	Length	Values
			<p>the regional date and time settings on the local computer.</p> <p><b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
OFFTIMEDATE	Special SQL formatted date and time.	28	<p>For non-hardware alarms: HH:MI:SS.</p> <p><b>Note:</b> The format can be configured in the citect.ini file using the <a href="#">[Alarm]TimeDate</a> parameter.</p>
OFFUTC	Alarm off time in UTC format.	12	<p>Integer that represents a "time" value based on the number of seconds since Jan 1, 1970 (in UTC format not Local).</p>
OLD_DESC	Multi digital old state description.	8	<p>"Invalid", "OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7".</p>
ONDATE	Event on date (short format).	12	<p>Short date format (based on the system locale settings).</p> <p>For example, "dd-mm-yy".</p> <p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
ONDATEEXT	Event on date (extended format).	12	The date format used when

Field Name	Description	Length	Values
			<p>[Alarm]ExtendedDate is set to 1 in the Citect.ini file.</p> <p>For non-hardware alarms, extended date format (based on the system locale settings).</p> <p>For example, "dd-mm-yyyy".</p> <p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
ONMILLI	Millisecond precision for the on time.	6	<p>For high resolution timestamped digital and timestamped analog alarms, it is the milliseconds part of the on time.</p>
ONTIME	Event on time.	12	<p>This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.</p> <p><b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
ONTIMEDATE	Special SQL formatted date and time.	28	<p>For non-hardware alarms: HH:MI:SS.</p> <p><b>Note:</b> The format can be configured in the citect.ini file using the [Alarm]TimeDate parameter.</p>
ONUTC	Alarm on time in UTC format.	12	<p>Integer that represents a "time" value based on the number of seconds since Jan 1, 1970 (in UTC format)</p>

Field Name	Description	Length	Values
			not Local).
PAGING	Alarm paged flag.	8	A flag to indicate that the alarm is going to be paged. Values are 1 (TRUE) or 0 (FALSE).
PAGINGGROUP	Paging group for alarm.	80	A freeform text field indicating the sequence of people to notify in the event the alarm occurred.
PRIORITY	Alarm category priority.	4	Numeric value (integer).
PRIV	Alarm privilege.	16	Numeric value (integer) that represents the level of privilege an operator requires to acknowledge or disable the alarm.
RATE	Alarm rate.	12	For analog and time stamped analog alarms, a numeric value (real).
RECEIPTLOCALTIMEDATE	The alarm's occurrence time.	28	Displays the alarm's occurrence time (if known), otherwise it displays the receipt time of the alarm.
RECEIPTDATE	The date the master station received the event (short format).  For example, the date the RTU received the event.	12	Short date format (based on the system locale settings).  For example, "dd-mm-yy". This works between UTC time January 1 1970 - January 18 2038, otherwise "".  <b>Note:</b> An extended date format is used by default. The short date format only applies when [Alarm]ExtendedDate is set to 0 in the Citect.ini file.
RECEIPTDATEEXT	The date the master	12	The date format used

Field Name	Description	Length	Values
	<p>station received the event (extended format).</p> <p>For example, the date the RTU received the event.</p>		<p>when [Alarm]ExtendedDate is set to 1 in the Citect.ini file.</p> <p>For non-hardware alarms, extended date format (based on the system locale settings).</p> <p>For example, "dd-mm-yyyy".</p> <p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
RECEIPTMILLISEC	Millisecond precision for the receipt time.	6	For high resolution timestamped digital and timestamped analog alarms, it is the milliseconds part of the receipt time.
RECEIPTTIME	<p>The time the master station received the event.</p> <p>For example, the time the RTU received the event.</p>	16	This can be in 12 hour format (for example, h:mm:ss tt) or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.
RECEIPTTIMEINT	Receipt time in UTC format.	12	An integer that represents a "time" value based on the number of seconds since Jan 1, 1970 (in UTC format not Local).
RECEIPTTIMETICKS	Receipt time measured using ticks.	20	<p>Time expressed as the number of ticks elapsed since 12:00:00 midnight, January 1, 1601 in UTC format.</p> <p>Each tick is 100 nanoseconds.</p>
RESPONSE1	The first response defined	254	Up to eight responses can

Field Name	Description	Length	Values
	for the alarm.		be defined for an alarm to help operators determine the most appropriate course of action.
RESPONSE2	The second response defined for the alarm.	254	See <a href="#">Add Cause and Response Information to Alarms</a> .
RESPONSE3	The third response defined for the alarm.	254	
RESPONSE4	The fourth response defined for the alarm.	254	
RESPONSE5	The fifth response defined for the alarm.	254	
RESPONSE6	The sixth response defined for the alarm.	254	
RESPONSE7	The seventh response defined for the alarm.	254	
RESPONSE8	The eighth response defined for the alarm.	254	
RESPONSENUM	The number of alarm responses that have been configured for an alarm tag.	8	0 to 8.
SETPOINT	The set point value. This is only applicable to analog alarms.	16	An analog variable tag, expression or base value that determines if a deviation alarm is to be triggered.
STATE	Alarm state string.	16	For analog and time stamped analog alarms: "OFF", "DEVIATION", "RATE OF CHANGE", "LOW", "HIGH", "LOW LOW", "HIGH HIGH". For multi-digital alarms: "OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7". For double point status

Field Name	Description	Length	Values
			alarms: The configured state name. For all other alarms: "OFF", "ON".
STATE_DESC	Multi-digital state description.	8	"OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7".
STATE_DESC0	Multi-digital state 0.	64	For multi-digital alarms: "OFF". For others: "INVALID".
STATE_DESC1	Multi-digital state 1.	64	For multi-digital alarms: "ON". For others: "INVALID".
STATE_DESC2	Multi-digital state 2.	64	For multi-digital alarms: "ON state 2". For others: "INVALID".
STATE_DESC3	Multi-digital state 3.	64	For multi-digital alarms: "ON state 3". For others: "INVALID".
STATE_DESC4	Multi-digital state 4.	64	For multi-digital alarms: "ON state 4". For others: "INVALID".
STATE_DESC5	Multi-digital state 5.	64	For multi-digital alarms: "ON state 5". For others: "INVALID".
STATE_DESC6	Multi-digital state 6.	64	For multi-digital alarms: "ON state 6". For others: "INVALID".
STATE_DESC7	Multi-digital state 7.	64	For multi-digital alarms: "ON state 7". For others: "INVALID".
SUMDESC	Event description text.	80	Analog alarm - event state

Field Name	Description	Length	Values
			text. Otherwise - configured event [or alarm in case event not defined] description.
SUMSTATE	Event state text.	16	Displays the last active state of the alarm. "OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7", "DEVIATION", "RATE OF CHANGE", "LOW", "HIGH", "LOW LOW", "HIGH HIGH", "Invalid".
SUMTYPE	Event state.	16	For non-hardware alarms: "DISABLED", "UNACKNOWLEDGED", "ACKNOWLEDGED", "Cleared".
TAG	Tag name.	80	The configured tag name.
TAGEX	Extended alarm tag.	32	<cluster>.<tag> when a cluster is defined. Otherwise <tag>.
TAGGENLINK	Indicates a tag was imported from an I/O device.	16	Name of the I/O device from which this tag was generated.
TIME	The time the event occurred.	12	This can be in 12 hour format (hh:mm:ss tt), or 24 hour format (HH:mm:ss) depending on the current date and time settings on the local computer. <b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".
TIMEDATE	Special SQL formatted time.	28	For non-hardware alarms: HH:MI:SS.

Field Name	Description	Length	Values
			<b>Note:</b> The format can be configured in the citect.ini file using the [Alarm]TimeDate parameter.
TIMEINT	Event time in UTC format.	12	Integer that represents a time value based on the number of seconds since Jan 1, 1970 (in UTC format).
TIMETICKS	Event time measured using ticks.	20	Time expressed as the number of ticks elapsed since 12:00:00 midnight, January 1, 1601 in UTC format. Each tick is 100 nanoseconds.
TSQUALITY	The quality of the alarm tag. This is passed to the alarm server by the I/O server when a value change occurs.	64	Expected time stamp quality strings are: Time Good, Time Uncertain, Clock Not Synchronized, or empty string if no match is found.
TYPE	Alarm type.	16	"DISABLED", "UNACKNOWLEDGED", "ACKNOWLEDGED", "CLEARED"
TYPENUM	Alarm type represented as a numeric value.	4	Values are: -1 - Invalid 0 - Digital 1 - Analog 2 - Advanced 3 - Multi-Digital 4 - ArgAna 5 - User event 6 - Time stamped 7 - Hardware 8 - Time stamped digital

Field Name	Description	Length	Values
			9 - Time stamped analog
USERDESC	The text related to a user event.	64	If you specify an asterisk '*' as the first character of the <i>sTag</i> argument in <a href="#">AlarmSumAppend</a> , the summary entry becomes a user event. This value is the text that follows the asterisk.
USERNAME	Event user name.	16	For non-hardware alarms, the configured name of the event user (if it exists). Otherwise "System".
USERLOCATION	The IP of the machine which last raised, modified, or performed an action on the alarm.	80	<b>Note:</b> If the last action performed is from a system machine, the IP will not over ride the last user machine IP address.
VALUE (alarm fields)	Formatted alarm value.	16	For analog alarms, a numeric value formatted according to configuration.

## See Also

[Browse Function Field Reference](#)

## AlmTagsOpen Fields

The fields in the following table can be used with the [AlmTagsOpen](#) Cicode function.

Field Name	Description	Length	Values
ACKDATE	Event acknowledgement date (short format).	12	Short date format (based on the system locale settings). For example, "dd-mm-yy". This works between UTC time January 1 1970 - January 18 2038, otherwise "".

Field Name	Description	Length	Values
			<b>Note:</b> An extended date format is used by default. The short date format only applies when [Alarm]ExtendedDate is set to 0 in the Citect.ini file.
ACKDATEEXT	Event acknowledgement date (extended format).	12	The date format used when [Alarm]ExtendedDate is set to 1 in the Citect.ini file.  For non-hardware alarms, extended date format (based on the system locale settings).  For example, "dd-mm-yyyy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".
ACKMILLI	Millisecond precision for the event acknowledgement time.	6	For high resolution time stamped digital and time stamped analog alarms, this value represents the milliseconds part of the acknowledgement time.
ACKTIME	Event acknowledgement time.	12	This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.  <b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".
ACKUTC	Event acknowledgment time in UTC format.	12	Integer that represents a time value based on the

Field Name	Description	Length	Values
			number of seconds since Jan 1, 1970 (in UTC format).
ACQDESC	Acquisition error description.	254	Textual representation of an alarm acquisition error.
ACQERROR	Acquisition error.	6	Numeric value (integer).
ALARMTYPE	Alarm type.	16	Text describing the following alarm types: "Digital", "Analog", "Advanced", "Multi-Digital", "Time Stamped", "Time Stamped Digital", "Time Stamped Analog".
ALMCOMMENT	Alarm comment.	254	A text-based comment added to an alarm during configuration.
AREA	Alarm area.	16	Numeric value (integer) representing area.
ARR_SIZE	Array size.	16	Integer (1 means it is not an array).
CATEGORY	Alarm category.	16	Numeric value (integer) representing the alarm category.
CAUSE1	The first cause defined for the alarm.	254	Up to eight causes can be defined for an alarm to help operators determine the most appropriate course of action.  See <a href="#">Add Cause and Response Information to Alarms</a> .
CAUSE2	The second cause defined for the alarm.	254	
CAUSE3	The third cause defined for the alarm.	254	
CAUSE4	The fourth cause defined for the alarm.	254	
CAUSE5	The fifth cause defined for the alarm.	254	
CAUSE6	The sixth cause defined for the alarm.	254	

Field Name	Description	Length	Values
CAUSE7	The seventh cause defined for the alarm.	254	
CAUSE8	The eighth cause defined for the alarm.	254	
CLASSIFICATION	The class of the event.	32	The classification applied to an event. For example: Action - The event was logged due to an action being performed. Comment - The event was logged due to a comment being created. Configuration - The event was logged due to configuration changes being made. System - The event has been logged due to an occurrence that affects the entire system, such as the server being stopped and restarted. Alarm type - Type of alarm (for example, digital, multi-digital, etc.).
CLUSTER	The cluster to which the tag belongs.	32	The cluster name.
COMMENT	Comment.	64	If hardware alarm = "". Or a configured event description or trend comment.
CONSEQUENCE1	The first consequence defined for an alarm.	254	Up to eight consequences can be defined for an alarm to help operators determine the most appropriate course of action.  See <a href="#">Add Cause and Response Information to Alarms</a> .
CONSEQUENCE2	The second consequence defined for an alarm.	254	
CONSEQUENCE3	The third consequence defined for an alarm.	254	
CONSEQUENCE4	The fourth consequence	254	

Field Name	Description	Length	Values
	defined for an alarm.		
CONSEQUENCE5	The fifth consequence defined for an alarm.	254	
CONSEQUENCE6	The sixth consequence defined for an alarm.	254	
CONSEQUENCE7	The seventh consequence defined for an alarm.	254	
CONSEQUENCE8	The eighth consequence defined for an alarm.	254	
CUSTOM1	Alarm custom field #1	64	A user-defined string.
CUSTOM2	Alarm custom field #2	64	A user-defined string.
CUSTOM3	Alarm custom field #3	64	A user-defined string.
CUSTOM4	Alarm custom field #4	64	A user-defined string.
CUSTOM5	Alarm custom field #5	64	A user-defined string.
CUSTOM6	Alarm custom field #6	64	A user-defined string.
CUSTOM7	Alarm custom field #7	64	A user-defined string.
CUSTOM8	Alarm custom field #8	64	A user-defined string.
DATE	Alarm date in short format.	12	<p>Short date format (based on the system locale settings).  For example, "dd-mm-yy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
DATEEXT	Alarm date in extended format.	12	The date format used when

Field Name	Description	Length	Values
			[Alarm]ExtendedDate is set to 1 in the Citect.ini file.  For non-hardware alarms, extended date format (based on the system locale settings).  For example, "dd-mm-yyyy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".
DEADBAND	Alarm deadband.	12	For analog, and timestamped analog alarms, a numeric value (real).
DELAY	The delay configured for the alarm.	12	The delay period in the following format:  hh:mm:ss  The value will be between 0 seconds (00:00:00) and 24 hours (24:00:00).
DELTAMILLI	Millisecond precision for the delta time.	6	For high resolution time stamped digital and time stamped analog alarms, it is the milliseconds part of the delta time.
DELTATIME	The time difference between OnDate/OnTime and OffDate/OffTime.	12	The delta time in the following format:  hh:mm:ss
DESC	Alarm description.	254	For analog alarms: "DEVIATION", "RATE OF CHANGE", "LOW", "HIGH", "LOW LOW", "HIGH HIGH".  For other alarm types: Configured descriptions.
DEVDELAY	Deviation delay.	16	Delay value (in seconds)

Field Name	Description	Length	Values
			for analog alarms and time stamped analog alarms.
DEVIATION	Alarm deviation.	12	For analog and time stamped analog alarms: Numeric value (real).
DISABLECOMMENT	A comment added by a user when an alarm is shelved.	254	A user defined text string.
DISABLEDDATE	The date an alarm was disabled.	12	<p>The date format is based on the system locale settings.</p> <p>Extended date format: (for example, dd-mm-yyyy).</p> <p>Short date format: (for example, dd-mm-yy).</p> <p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> Extended date format is used by default. The short date format only applies when [Alarm]ExtendedDate is set to 0 in the Citect.ini file.</p>
DISABLEDTIME	The time an alarm was disabled.	12	<p>A time string that indicates when an alarm was disabled.</p> <p>This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.</p>
DISABLEENDDATE	The disable end date for a	12	Short date format (based

Field Name	Description	Length	Values
	shelved alarm (short format).		<p>on the system locale settings).</p> <p>For example, "dd-mm-yy". This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
DISABLEENDDATEEXT	The disable end date for a shelved alarm (extended format).	12	<p>The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file.</p> <p>For non-hardware alarms, extended date format (based on the system locale settings).</p> <p>For example, "dd-mm-yyyy".</p> <p>This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
DISABLEENDTIME	The disable end time for a shelved alarm.	12	<p>A time string that indicates when a shelved alarm will no longer be disabled.</p> <p>This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.</p>
ENG_ZERO	The engineering zero	11	Numeric value (real).

Field Name	Description	Length	Values
	scale for this value.		
EQUIPMENT	The name of the equipment associated with the alarm.	254	This can be a name that represents a single piece of equipment, or a name that also reflects the location of the equipment within the equipment hierarchy (where a period (.) is used to indicate levels in the hierarchy).
FORMAT	Alarm format.	12	For analog, and time stamped analog alarms, a numeric value (real).
FULLNAME	Event user full name.	20	For non-hardware alarms, the configured full name of the event user if exists. Otherwise "System".
GROUP	Alarm group.	16	For multi-digital alarms, a numeric value
HDELAY	High delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
HELP	Alarm help.	254	The name of the help page.
HHDELAY	High high delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
HIGH	Alarm high	12	For analog and time stamped analog alarms, a numeric value (real).
HIGHHIGH	Alarm high high	12	For analog and time stamped analog alarms, a numeric value (real).
HISTORIAN	Determines if the alarm will be included in an automated configuration	6	True or False (blank).

Field Name	Description	Length	Values
	process within the Historian environment.		
ITEM	Name of equipment item.	63	The name configured in the alarm's <b>Item Name</b> field.
LDELAY	Low delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
LLDELAY	Low low delay.	16	Delay value (in seconds) for analog alarms and time stamped analog alarms.
LOCALTIMEDATE	Alarm date and time.	24	yyyy-mm-dd hh:mm:ss[.ttt]. This works between: UTC time January 1 1970 - January 18 2038, otherwise "".
LOGSTATE	Log state text.	13	"ACTIVE", "INACTIVE", "DISABLED", "ENABLED", "ACKNOWLEDGED", "LOW", "HIGH", "LOW LOW", "HIGH HIGH", "RATE", "DEVIATION", "CLEARED", "UNACKNOWLEDGED".
LOW	Alarm low.	12	For analog and time stamped analog alarms, a numeric value (real).
LOWLOW	Alarm low low.	12	For analog and time stamped analog alarms, a numeric value (real).
MESSAGE	The event message.	254	Text entered in the <b>Alarm Desc</b> field of digital, advanced and time-stamped alarms.
MILLISEC	Alarm milliseconds.	6	Numeric value (integer).

Field Name	Description	Length	Values
NAME	Alarm name.	80	A meaningful description of the alarm defined in the <b>Alarm Name</b> property.
NATIVE_COMMENT	Event comment.	64	A text-based comment added to an event on the SOE page at runtime.
NATIVE_DESC	Alarm description.	80	For analog alarms, the alarm state text. Otherwise, the configured alarm description.
NATIVE_NAME	Alarm name.	80	Configured alarm name.
NATIVE_SUMDESC	Event description text.	80	For non-hardware analog alarms: the event state string. For non-hardware other alarms: the event [or if it does not exist] the alarm description.
OFFDATE	Event off date (short format).	12	Short date format (based on the system locale settings). For example, "dd-mm-yy". This works between UTC time January 1 1970 - January 18 2038, otherwise "". <b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.
OFFDATEEXT	Extended event off date.	12	The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file. For non-hardware alarms,

Field Name	Description	Length	Values
			<p>extended date format (based on the system locale settings).  For example, "dd-mm-yyyy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
OFFMILLI	Millisecond precision for the off time.	6	<p>For high resolution timestamped digital and timestamped analog alarms, it is the milliseconds part of the off time.</p>
OFFTIME	Event off time.	12	<p>This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.  <b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
OFFTIMEDATE	Special SQL formatted date and time.	28	<p>For non-hardware alarms: HH:MI:SS.  <b>Note:</b> The format can be configured in the citect.ini file using the <a href="#">[Alarm]TimeDate</a> parameter.</p>
OFFUTC	Alarm off time in UTC format.	12	<p>Integer that represents a "time" value based on the number of seconds since Jan 1, 1970 (in UTC format not Local).</p>
OLD_DESC	Multi digital old state description.	8	"Invalid", "OFF", "ON", "ON State 2", "ON State

Field Name	Description	Length	Values
			3", "ON State 4", "ON State 5", "ON State 6", "ON State 7".
ONDATE	Event on date (short format).	12	<p>Short date format (based on the system locale settings).  For example, "dd-mm-yy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p> <p><b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.</p>
ONDATEEXT	Event on date (extended format).	12	<p>The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file.  For non-hardware alarms, extended date format (based on the system locale settings).  For example, "dd-mm-yyyy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".</p>
ONMILLI	Millisecond precision for the on time.	6	For high resolution timestamped digital and timestamped analog alarms, it is the milliseconds part of the on time.
ONTIME	Event on time.	12	This can be in 12 hour format (for example, h:mm:ss tt), or 24 hour

Field Name	Description	Length	Values
			format (for example, H:mm:ss) depending on the regional date and time settings on the local computer. <b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".
ONTIMEDATE	Special SQL formatted date and time.	28	For non-hardware alarms: HH:MI:SS. <b>Note:</b> The format can be configured in the citect.ini file using the <a href="#">[Alarm]TimeDate</a> parameter.
ONUTC	Alarm on time in UTC format.	12	Integer that represents a "time" value based on the number of seconds since Jan 1, 1970 (in UTC format not Local).
PAGING	Alarm paged flag.	8	A flag to indicate that the alarm is going to be paged. Values are 1 (TRUE) or 0 (FALSE).
PAGINGGROUP	Paging group for alarm.	80	A freeform text field indicating the sequence of people to notify in the event the alarm occurred.
PRIORITY	Alarm category priority.	4	Numeric value (integer).
PRIV	Alarm privilege.	16	Numeric value (integer) that represents the level of privilege an operator requires to acknowledge or disable the alarm.
RATE	Alarm rate.	12	For analog and time stamped analog alarms, a numeric value (real).
RECEIPTLOCALTIMEDATE	The alarm's occurrence	28	Displays the alarm's

Field Name	Description	Length	Values
	time.		occurrence time (if known), otherwise it displays the receipt time of the alarm.
RECEIPTDATE	The date the master station received the event (short format).  For example, the date the RTU received the event.	12	Short date format (based on the system locale settings).  For example, "dd-mm-yy". This works between UTC time January 1 1970 - January 18 2038, otherwise "".  <b>Note:</b> An extended date format is used by default. The short date format only applies when <a href="#">[Alarm]ExtendedDate</a> is set to 0 in the Citect.ini file.
RECEIPTDATEEXT	The date the master station received the event (extended format).  For example, the date the RTU received the event.	12	The date format used when <a href="#">[Alarm]ExtendedDate</a> is set to 1 in the Citect.ini file.  For non-hardware alarms, extended date format (based on the system locale settings).  For example, "dd-mm-yyyy".  This works between UTC time January 1 1970 - January 18 2038, otherwise "".
RECEIPTMILLISEC	Millisecond precision for the receipt time.	6	For high resolution timestamped digital and timestamped analog alarms, it is the milliseconds part of the receipt time.
RECEIPTTIME	The time the master station received the	16	This can be in 12 hour format (for example,

Field Name	Description	Length	Values
	event. For example, the time the RTU received the event.		h:mm:ss tt) or 24 hour format (for example, H:mm:ss) depending on the regional date and time settings on the local computer.
RECEIPTTIMEINT	Receipt time in UTC format.	12	An integer that represents a "time" value based on the number of seconds since Jan 1, 1970 (in UTC format not Local).
RECEIPTTIMETICKS	Receipt time measured using ticks.	20	Time expressed as the number of ticks elapsed since 12:00:00 midnight, January 1, 1601 in UTC format. Each tick is 100 nanoseconds.
RESPONSE1	The first response defined for the alarm.	254	Up to eight responses can be defined for an alarm to help operators determine the most appropriate course of action.  See <a href="#">Add Cause and Response Information to Alarms</a> .
RESPONSE2	The second response defined for the alarm.	254	
RESPONSE3	The third response defined for the alarm.	254	
RESPONSE4	The fourth response defined for the alarm.	254	
RESPONSE5	The fifth response defined for the alarm.	254	
RESPONSE6	The sixth response defined for the alarm.	254	
RESPONSE7	The seventh response defined for the alarm.	254	
RESPONSE8	The eighth response defined for the alarm.	254	
RESPONSENUM	The number of alarm responses that have been configured for an alarm	8	0 to 8.

Field Name	Description	Length	Values
	tag.		
SETPOINT	The set point value. This is only applicable to analog alarms.	16	An analog variable tag, expression or base value that determines if a deviation alarm is to be triggered.
STATE	Alarm state string.	16	For analog and time stamped analog alarms: "OFF", "DEVIATION", "RATE OF CHANGE", "LOW", "HIGH", "LOW LOW", "HIGH HIGH".  For multi-digital alarms: "OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7".  For double point status alarms: The configured state name.  For all other alarms: "OFF", "ON".
STATE_DESC	Multi-digital state description.	8	"OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7".
STATE_DESC0	Multi-digital state 0.	64	For multi-digital alarms: "OFF".  For others: "INVALID".
STATE_DESC1	Multi-digital state 1.	64	For multi-digital alarms: "ON".  For others: "INVALID".
STATE_DESC2	Multi-digital state 2.	64	For multi-digital alarms: "ON state 2".  For others: "INVALID".

Field Name	Description	Length	Values
STATE_DESC3	Multi-digital state 3.	64	For multi-digital alarms: "ON state 3". For others: "INVALID".
STATE_DESC4	Multi-digital state 4.	64	For multi-digital alarms: "ON state 4". For others: "INVALID".
STATE_DESC5	Multi-digital state 5.	64	For multi-digital alarms: "ON state 5". For others: "INVALID".
STATE_DESC6	Multi-digital state 6.	64	For multi-digital alarms: "ON state 6". For others: "INVALID".
STATE_DESC7	Multi-digital state 7.	64	For multi-digital alarms: "ON state 7". For others: "INVALID".
SUMDESC	Event description text.	80	Analog alarm - event state text.  Otherwise - configured event [or alarm in case event not defined] description.
SUMSTATE	Event state text.	16	Displays the last active state of the alarm. "OFF", "ON", "ON State 2", "ON State 3", "ON State 4", "ON State 5", "ON State 6", "ON State 7", "DEVIATION", "RATE OF CHANGE", "LOW", "HIGH", "LOW LOW", "HIGH HIGH", "Invalid".
SUMTYPE	Event state.	16	For non-hardware alarms: "DISABLED", "UNACKNOWLEDGED", "ACKNOWLEDGED", "CLEARED".
TAG	Tag name.	80	The configured tag name.

Field Name	Description	Length	Values
TAGEX	Extended alarm tag.	32	<cluster>.<tag> when a cluster is defined. Otherwise <tag>.
TAGGENLINK	Indicates a tag was imported from an I/O device.	16	Name of the I/O device from which this tag was generated.
TIME	The time the event occurred.	12	This can be in 12 hour format (hh:mm:ss tt), or 24 hour format (HH:mm:ss) depending on the current date and time settings on the local computer. <b>Note:</b> This works between UTC time January 1 1970 - January 18 2038, otherwise "".
TIMEDATE	Special SQL formatted time.	28	For non-hardware alarms: HH:MI:SS. <b>Note:</b> The format can be configured in the citect.ini file using the [Alarm]TimeDate parameter.
TIMEINT	Event time in UTC format.	12	Integer that represents a time value based on the number of seconds since Jan 1, 1970 (in UTC format).
TIMETICKS	Event time measured using ticks.	20	Time expressed as the number of ticks elapsed since 12:00:00 midnight, January 1, 1601 in UTC format. Each tick is 100 nanoseconds.

Field Name	Description	Length	Values
TSQUALITY	The quality of the alarm tag. This is passed to the alarm server by the I/O server when a value change occurs.	64	Expected time stamp quality strings are: Time Good, Time Uncertain, Clock Not Synchronized, or empty string if no match is found.
TYPE	Alarm type.	16	"DISABLED", "UNACKNOWLEDGED", "ACKNOWLEDGED", "Cleared"
TYPENUM	Alarm type represented as a numeric value.	4	Values are: -1 - Invalid 0 - Digital 1 - Analog 2 - Advanced 3 - Multi-Digital 4 - ArgAna 5 - User event 6 - Time stamped 7 - Hardware 8 - Time stamped digital 9 - Time stamped analog
USERDESC	The text related to a user event.	64	If you specify an asterisk '*' as the first character of the <i>sTag</i> argument in <a href="#">AlarmSumAppend</a> , the summary entry becomes a user event. This value is the text that follows the asterisk.
USERNAME	Event user name.	16	For non-hardware alarms, the configured name of the event user (if it exists). Otherwise "System".
USERLOCATION	The IP of the machine which last raised, modified, or performed	80	<b>Note:</b> If the last action performed is from a system machine, the IP

Field Name	Description	Length	Values
	an action on the alarm.		will not over ride the last user machine IP address.
VALUE (alarm fields)	Formatted alarm value.	16	For analog alarms, a numeric value formatted according to configuration.

## See Also

[Browse Function Field Reference](#)

## EquipBrowseOpen Fields

The fields in the following table can be used with the [EquipBrowseOpen](#) Cicode function.

Field Name	Description	Length	Values
ALIAS	The display name used to describe a piece of equipment.	254	A meaningful name for a piece of equipment, configured using the <b>Display Name</b> property.
AREA	The area number.	16	Numeric value (integer) representing the area to which the equipment belongs.  If the current user does not have access to the area, the equipment will not be returned by the browse session.
CLUSTER	Cluster name.	16	The name of the cluster to which the equipment is assigned.
COMMENT	Comment.	254	A user-configured comment.
CONTENT	Workspace content associated with the equipment.	254	A comma-separated list that names the content associated with the piece of equipment. This can include pages, faceplates and documents.

Field Name	Description	Length	Values
CUSTOM1	Custom field #1.	254	A user-defined string.
CUSTOM2	Custom field #2.	254	A user-defined string.
CUSTOM3	Custom field #3.	254	A user-defined string.
CUSTOM4	Custom field #4.	254	A user-defined string.
CUSTOM5	Custom field #5.	254	A user-defined string.
CUSTOM6	Custom field #6.	254	A user-defined string.
CUSTOM7	Custom field #7.	254	A user-defined string.
CUSTOM8	Custom field #8.	254	A user-defined string.
DEFSTATE	The default state.	64	The default state configured for the equipment.
DEVSCHED	Determines whether or not the report server retrieves the associated schedule from a BACnet device.	6	"TRUE" or "FALSE". Use SCHEDID to identify the associated schedule.
HIDDEN	Indicates if the equipment is visible in the equipment tree at runtime.	6	"TRUE" or "FALSE".
LOCATION	The location of the equipment.	254	A string describing the location of the equipment.
NAME	Equipment name.	254	This can be a name that represents a single piece of equipment, or a name that also reflects the location of the equipment within an equipment hierarchy (where a period (.) is used to indicate levels in the hierarchy).
PAGE	The name of the page on which the equipment appears.	254	Allows you to navigate directly to the page that hosts the piece of equipment. If the graphic object appears on more

Field Name	Description	Length	Values
			than one page, the page specified will represent the primary operational context.
PARENT	The parent equipment.	254	The name of the parent equipment derived from the name of the equipment.
SCHEDID	BACnet device schedule ID.	16	The ID used to identify the associated schedule on a BACnet device. This field is only used if DEVSCHED returns "TRUE".
SCHEDULED	Indicates if the equipment is included in any scheduled actions.	6	"TRUE" or "FALSE".
TYPE	The equipment type.	254	The name of the equipment type associated with the equipment, as defined in the Equipment Types database.

## See Also

[Browse Function Field Reference](#)

## EquipRefBrowseGetField Fields

The fields in the following table can be used with the [EquipRefBrowseGetField](#) Cicode function.

Field Name	Description	Length	Values
ASSOC	The association used by the <a href="#">AssEquipReferences</a> Cicode function.	63	A Super Genie association is created using the name specified in this field.
CLUSTER	Cluster name.	16	The name of the cluster the equipment is associated with.
COMMENT	Comment.	254	A user-defined string.

Field Name	Description	Length	Values
CUSTOM1	Custom field #1.	254	A user-defined string.
CUSTOM2	Custom field #2.	254	A user-defined string.
CUSTOM3	Custom field #3.	254	A user-defined string.
CUSTOM4	Custom field #4.	254	A user-defined string.
CUSTOM5	Custom field #5.	254	A user-defined string.
CUSTOM6	Custom field #6.	254	A user-defined string.
CUSTOM7	Custom field #7.	254	A user-defined string.
CUSTOM8	Custom field #8.	254	A user-defined string.
EQUIP	Equipment name.	254	The name of the main piece of equipment that the referenced equipment is linked to.
ORDER	Ordering index.	6	A unique index per reference category that is used in the ordering of the EquipRefBrowse functions.
REFCAT	Reference category.	63	A category that is used to filter equipment references using the EquipRefBrowse functions.
REFCLUST	Cluster of the referenced piece of equipment.	16	The name of the cluster to which the referenced piece of equipment is assigned.
REFEQUIP	Name of the referenced piece of equipment.	254	The name of the referenced piece of equipment.
REFITEM	Referenced equipment item.	63	The name of an item belonging to the referenced piece of equipment.

## See Also

[Browse Function Field Reference](#)

### EquipRefBrowseOpen Fields

The fields in the following table can be used with the [EquipRefBrowseOpen](#) Cicode function.

Field Name	Description	Length	Values
ASSOC	The association used by the <a href="#">AssEquipReferences</a> Cicode function.	63	A Super Genie association is created using the name specified in this field.
CLUSTER	Cluster name.	16	The name of the cluster the equipment is associated with.
COMMENT	Comment.	254	A user-defined string.
CUSTOM1	Custom field #1.	254	A user-defined string.
CUSTOM2	Custom field #2.	254	A user-defined string.
CUSTOM3	Custom field #3.	254	A user-defined string.
CUSTOM4	Custom field #4.	254	A user-defined string.
CUSTOM5	Custom field #5.	254	A user-defined string.
CUSTOM6	Custom field #6.	254	A user-defined string.
CUSTOM7	Custom field #7.	254	A user-defined string.
CUSTOM8	Custom field #8.	254	A user-defined string.
EQUIP	Equipment name.	254	The name of the main piece of equipment that the referenced equipment is linked to.
ORDER	Ordering index.	6	A unique index per reference category that is used in the ordering of the EquipRefBrowse functions.
REFCAT	Reference category.	63	A category that is used to filter equipment references using the EquipRefBrowse

Field Name	Description	Length	Values
			functions.
REFCLUST	Cluster of the referenced piece of equipment.	16	The name of the cluster to which the referenced piece of equipment is assigned.
REFEQUIP	Name of the referenced piece of equipment.	254	The name of the referenced piece of equipment.
REFITEM	Referenced equipment item.	63	The name of an item belonging to the referenced piece of equipment.

## See Also

[Browse Function Field Reference](#)

## EquipStateBrowseOpen Fields

The fields in the following table can be used with the [EquipStateBrowseOpen](#) Cicode function.

Field Name	Description	Length	Values
CLUSTER	Cluster name.	16	The name of the cluster to which the state belongs.
DELAY	The delay configured for the state's entry action.	12	The amount of time that needs to elapse before the state will initially become active.  If a delay is defined, the Entry Action for the state will not occur until the specified period of time has elapsed.
DESCRIPTION	A description of the state.	48	A user configured string.
DRMODE	Demand and response mode.	16	A numeric value (integer) that represents different level of energy consumption. See <a href="#">Configure a Demand and Response Mode</a>

Field Name	Description	Length	Values
			<a href="#">Response Solution.</a> The default mode is zero (0).
EQUIPMENT	Name of equipment.	254	The equipment associated with this state. This can be a name that represents a single piece of equipment, or a name that also reflects the location of the equipment within an equipment hierarchy (where a period (.) is used to indicate levels in the hierarchy).
NAME	State name.	64	Usually this string will represent an 'action' the associated equipment could be scheduled to do.
PERIOD	The frequency of the repeat action configured for the state.	16	In hh:mm:ss (hours:minutes:seconds).
PRIORITY	State priority.	16	Numeric value (integer) used to determine which schedule entry will run if a schedule conflict occurs between more than one state.

## See Also

[Browse Function Field Reference](#)

### TagBrowseOpen Fields

The fields in the following table can be used with the [TagBrowseOpen](#) Cicode function.

Field Name	Description	Length	Values
ADDR	The register address.	254	The address of the register in the I/O device that the variable is associated with.

Field Name	Description	Length	Values
			The format and prefix of the address will depend on the protocol the I/O device uses.
ARR_SIZE	Array size	11	An integer representing the size of the array configured for a variable tag. 1 means it is not an array.
CLUSTER	Cluster name.	16	The cluster the tag exists on.
COMMENT	Comment.	254	A user-configured comment.
CUSTOM1	Custom field #1.	64	A user-defined string.
CUSTOM2	Custom field #2.	64	A user-defined string.
CUSTOM3	Custom field #3.	64	A user-defined string.
CUSTOM4	Custom field #4.	64	A user-defined string.
CUSTOM5	Custom field #5.	64	A user-defined string.
CUSTOM6	Custom field #6.	64	A user-defined string.
CUSTOM7	Custom field #7.	64	A user-defined string.
CUSTOM8	Custom field #8.	64	A user-defined string.
DEADBAND	The deadband set for a tag.	11	A numeric value (real) that indicates how much the value of a variable tag can fluctuate within a defined threshold without updates being sent through to the system.  The threshold is represented as a percentage of the tag's engineering range. The default value is 0 (zero), which captures every value change.

Field Name	Description	Length	Values
ENG_FULL	The engineering full scale point for a tag.	11	Numeric value (real). This value represents engineering units. It is the upper limit used for trends and bar graphs. It is a scaled value derived from the Raw Full Scale property.
ENG_UNITS	Engineering units.	8	The engineering units that the value represents. The predefined values (see below) are in the Help.dbf in the bin directory, but the user may specify it as a free text: %, Amps, deg, ft, ft/min, ft/s, ft/sec, gal, gal/h, gal/min, gal/s, gal/sec, Hz, kg, kg/h, kg/min, kg/s, kg/sec, km/h, kPa, kW, litres, lt, lt/h, lt/min, lt/s, lt/sec, m/min, m/s, m/sec, metres, Rev, Rev/h, RPM, t, t/h, Tonnes, Volts, Watts.
ENG_ZERO	The engineering zero scale for a tag.	11	Numeric value (real). This value represents engineering units. It is the lower limit used for trends and bar graphs. It is a scaled value derived from the Raw Zero Scale property.
EQUIPMENT	Equipment name.	254	The name of the piece of equipment with which the tag is associated. This can be a name that represents a single piece of equipment, or a name that also reflects the location of the equipment

Field Name	Description	Length	Values
			within the equipment hierarchy (where a period (.) is used to indicate levels in the hierarchy).
FULLNAME	The full name of the variable tag.	254	The full name of the variable tag in the format: <ClusterName>.<TagName>.
HISTORIAN	Determines if the alarm will be included in an automated configuration process within the Historian environment.	6	True or False (blank).
ITEM	Equipment item.	63	The name of an item with which the tag is associated. Items form part of an equipment hierarchy. They can be used to associate tags, alarms and trends with a particular attribute of a physical piece of equipment.
RAW_FULL	The raw full scale for the tag.	11	Numeric value (real). It is a raw value read from the I/O device. It represents the upper limit used by Plant SCADA to calculate scaled engineering values. If not defined, Raw Zero Scale and Raw Full Scale will default to the full range value based on the data type.
RAW_ZERO	The raw zero scale for this value.	11	Numeric value (real). It is a raw value read from the I/O device. It represents the lowest value used by Plant SCADA to calculate scaled

Field Name	Description	Length	Values
			engineering values. If not defined, Raw Zero Scale and Raw Full Scale will default to the full range value based on the data type.
TAG	Tag name.	79	The configured tag name.
TYPE	Tag type string.	16	RDT_DIGITAL 0 RDT_INTEGER 1 RDT_REAL 2 RDT_BCD 3 RDT_LONG 4 RDT_LONG_BCD 5 RDT_STRING 7 RDT_BYTE 8 RDT_UINT32 10 RDT_ULONG 18
WRITE_ROLES	The roles that have write permissions for the variable tag.	254	Indicates which roles have write permissions for a variable tag when using an OPC UA or Industrial Graphics client.  Only users included in the specified role(s) will be able to send values to the variable tag from a client application.

## See Also

[Browse Function Field Reference](#)

### TrnBrowseOpen Fields

The fields in the following table can be used with the [TrnBrowseOpen](#) Cicode function.

Field Name	Description	Length	Values
ACQERROR	Acquisition error.	6	Numeric value (integer).
CLUSTER	Cluster name.	16	The name of the cluster the trend tag is associated with.

Field Name	Description	Length	Values
COMMENT	Comment	254	A user-defined string.
DEADBAND	The deadband set for a trend tag.	16	A numeric value (real) that indicates how much the value of a trend tag can fluctuate within a defined threshold without updates being sent through to the system. The threshold is represented as a percentage of the tag's engineering range. The default value is 0 (zero), which captures every value change.
ENG_UNITS	Engineering units.	8	The engineering units that the value represents. The predefined values (see below) are in the Help.dbf in the bin directory, but the user may specify it as a free text: %, Amps, deg, ft, ft/min, ft/s, ft/sec, gal, gal/h, gal/min, gal/s, gal/sec, Hz, kg, kg/h, kg/min, kg/s, kg/sec, km/h, kPa, kW, litres, lt, lt/h, lt/min, lt/s, lt/sec, m/min, m/s, m/sec, metres, Rev, Rev/h, RPM, t, t/h, Tonnes, Volts, Watts.
EQUIPMENT	The name of the associated piece of equipment.	254	This can be a name that represents a single piece of equipment, or a name that also reflects the location of the equipment within the equipment hierarchy (where a period (.) is used to indicate levels in the hierarchy).

Field Name	Description	Length	Values
EXPRESSION	Logged variable name.	65	The logged variable name (<clustername>.<tagname>).  If the expression contains simple Cicode containing a single tag name, such as "LT131" or "LT131<1000", the tag name is returned (in both these cases LT131).  If the expression contains a function call or more than one tag name, an empty string is returned.
FILENAME	File where the trend data is to be stored.	253	The complete path to the file or a path substitution.
FILES	The number of history files stored on the hard disk (for this tag).	4	The maximum number of files that can be specified per trend tag is 999.
FORMAT	Trend tag format.	11	The format of the variable/expression being logged.  The format is used by the trend scales and trend cursor displays.
HISTORIAN	Determines if the trend tag will be included in an automated configuration process within the Historian environment.	6	True or False (blank).
ITEM	Name of equipment item.	63	The name configured in the trend tag's <b>Item Name</b> field.
LSL	Lower specification limit.	16	This value is used by SPC tags as the lower limit to determine process capability.  When used in conjunction with the USL, it provides a tolerance for your

Field Name	Description	Length	Values
			process.
NAME	Trend tag name.	79	The name assigned to the trend data.
PRIV	Privilege.	16	Numeric value (integer) that represents the level of privilege an operator requires to display trend data.
SAMPLEPER	Sample period of the trend.	16	Sampling periods of greater than one second use the format hh:mm:ss. If a single digit is used without the colon (:), it will be represent seconds. Sampling periods of less than one second are entered as decimals. For example, a period of 200 milliseconds would be 0.2.
SDEVIATION	Standard deviation.	16	For SPC tags, this is the calculation override for process standard deviation (s bar).
SPCFLAG	SPC flag.	4	Indicates that a tag is a Statistical Process Control (SPC) tag.
STORMETHOD	Storage method for the data.	16	"Scaled" or "Floating Point".
SUBGRPSIZE	The size of each SPC subgroup.	8	Indicates the size of the subgroups for SPC tags. Valid values are 1 - 25 inclusive.
TAGGENLINK	Indicates if the tag was imported from an I/O	32	Name of the I/O device from which this tag was

Field Name	Description	Length	Values
	device.		generated.
TIME	File synchronization time.	32	The time of day when the beginning of the history file is synchronized (in hh:mm:ss). The time is specified in Greenwich Mean Time, not the local time zone.
TRIGGER	The variable tag that triggers data logging.	254	The configured tag name (<clustername>.<tagname>). If the trigger contains simple Cicode containing a single tag name, such as "LT131<50", the tag name is returned (in this case LT131). If the expression contains a function call or more than one tag name, an empty string is returned.
TYPE	Trend type.	16	"1" - Periodic "2" - Event "3" - Periodic event.
USL	Upper specification limit.	16	This value is used by SPC tags as the upper limit to determine process capability. When used in conjunction with the LSL, it provides a tolerance for your process.
XDOUBLEBAR	Process mean.	16	For SPC tags, determines the calculation override for process mean (X double bar).

## See Also

[Browse Function Field Reference](#)

## Server Browse Function Fields

Field Name	Description	Length	Values
CLUSTER	Cluster name.	16	The name of the cluster to which the server belongs.
COMMENT	Comment	48	A user-defined string.
MODE	Indicates whether the server is configured as a primary or standby.	16	For alarm, report and trend servers: 1 - primary 0 - standby. For I/O servers, the MODE field is set to "0".
NAME	Server name.	16	The name of the server
NETADDR	The network addresses associated with the server.	70	A comma-separated list of one or more names. The field contains the network address names as configured, and not IP addresses.
PORT	The port the server is listening on.	16	A number between 1 and 65535. If no port number was configured, this field contains the default port number for the server type.
TYPE	The type of server.	16	"Alarm", "I/O", "Report", or "Trend"

### See Also

[Browse Function Field Reference](#)

## Parameters

Parameters determine how each Plant SCADA computer operates in both the Plant SCADA configuration and runtime environments.

## ⚠ WARNING

### UNINTENDED EQUIPMENT OPERATION

- Read and understand the applicable material in this manual before changing or removing any Citect.ini parameters.
- Never change or remove any undocumented Citect.ini parameters.
- Before deleting sections of the Citect.ini file, confirm that no necessary or undocumented parameters will be deleted.
- Do not edit your configuration file while your project is running.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

- See [About Parameters](#) for information that explains how parameters operate in a Plant SCADA system.
- See [Configure Parameters](#) for task-based information that describes how to use parameters in a Plant SCADA system.
- See [Parameters Reference](#) for a technical description of the parameters that Plant SCADA supports.

## See Also

[Parameter Categories](#)

## About Parameters

Plant SCADA supports two types of parameters:

- **Project Database Parameters**

Project database parameters can be used to set nominal defaults for parameters that are local to a specific Plant SCADA project, rather than a particular computer. For example, you could use the project database parameters to specify login requirements for all computers in a Plant SCADA system.

To set (or change) the parameters in a project database, go to the **Setup** activity in Plant SCADA Studio and select **Parameters** from the Navigation Menu. Any changes to your project database parameters are only implemented following a project recompile.

- **Citect.ini File Parameters**

The Citect.ini file is a text file that stores values for a comprehensive set of operating parameter that are used to configure the operational settings for each computer in a Plant SCADA system. These values are read by Plant SCADA on startup to determine how the application should operate.

To view the complete list of built-in operating parameters that you can use in a Citect.ini file, see [Parameter Categories](#).

During installation, a default Citect.ini file is copied to the Plant SCADA Config folder in the Program Data directory. This configuration file contains a series of undocumented parameters used by Plant SCADA as well as a series of default settings.

**Note:** Parameter settings in the Citect.ini file take precedence over project database parameters (see [Parameter Precedence](#)).

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

- Do not under any circumstances change or remove any of the undocumented Citect.ini parameters.
  - Before deleting sections of the Citect.ini file, confirm that no undocumented parameters will be deleted.
- Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

The Citect.ini file has a defined structure made up of a **header**, a series of **sections** (containing parameters and their values), and **comments**.

- **Header**

The header of the Citect.ini file contains comments the user wants to document about the configuration settings contained within the file. Typically it would include a history of edits made to the file and reasons for those edits. For example:

```
# Originally set up by XX on XX for use on XX
# Modified by XX on XX at the recommendation of Technical Support.
```

Comments are located at the top of the file before the first definition of a section. Each line of the header starts with the "hash" or "pound" character (#).

For more information, see [Add Comments to the Citect.ini File](#).

- **Sections**

In a Citect.ini file, parameters are organized into sections according to their purpose.

The syntax used to define a section is as follows:

```
[Section Name]
<parameter name1> = <parameter value1>
<parameter name2> = <parameter value2>
<parameter nameX> = <parameter valueX>
```

For example:

```
[Alarm]
Primary = 1
SavePeriod = 600
SaveSecondary =
ScanTime = 500
```

The following rules apply when the Citect.ini is read at runtime:

- A section continues until a new section is defined or the end of file is reached.
- Each parameter definition finishes with a return.
- Any line starting with a hash or pound (#) character is considered to be a comment.
- Any line starting with an exclamation mark (!) is considered to be disabled and therefore treated as a

comment.

- The maximum length for a parameter is 254 characters.

Sections which relate to server components (Alarms, Trend, Reports, I/O Server) also support hierarchical inheritance to allow parameters to be fine tuned to the cluster or server component level. The syntax used is as follows:

```
[Section Name.ClusterName.ServerName]
<parameter name1> = <parameter value1>
<parameter name2> = <parameter value2>
<parameter nameX> = <parameter valueX>
```

For example:

```
[Alarm.Cluster1.Server1]
SavePeriod = 600
ScanTime = 500
```

- **Comments**

You can add comments to a parameters section or a particular parameter setting in the Citect.ini file.

Comments for a **section** commence with a hash or pound character (#) and occur on the line just above the declaration of the relevant section.

For example:

```
#Alarm Section comment would go here
[Alarm]
```

Comments for a **parameter** commence with a hash or pound character (#) and occur on the line just above the declaration of the relevant parameter.

For example:

```
[Alarm]
#Set to 1 by System Administrator on 30/06/2005 10:13:44 AM.
Primary = 1
```

For more information, see [Add Comments to the Citect.ini File](#).

## Example

An extract from a typical Citect.ini configuration file is shown below.

```
#
#Header Comment on Citect.ini file
#[Alarm]
#Set to 1 by System Administrator on 30/06/2015 10:13:44 AM.
Primary = 1
SavePeriod = 600
SaveSecondary =
ScanTime = 500
Server = 1
```

In most cases, if you set (or change) parameters in the Citect.ini file, you need to restart Plant SCADA before the new parameter settings are used. However, there are a few exceptions to this rule where Citect.ini parameters are read at regular intervals and can be changed during runtime. Where this is the case, the parameter is documented accordingly.

If a parameter applies to a particular process, setting changes are implemented as soon as the associated process is restarted. For example, an Events parameter for an Alarm Server will be used as soon as the specific Alarm Server is restarted.

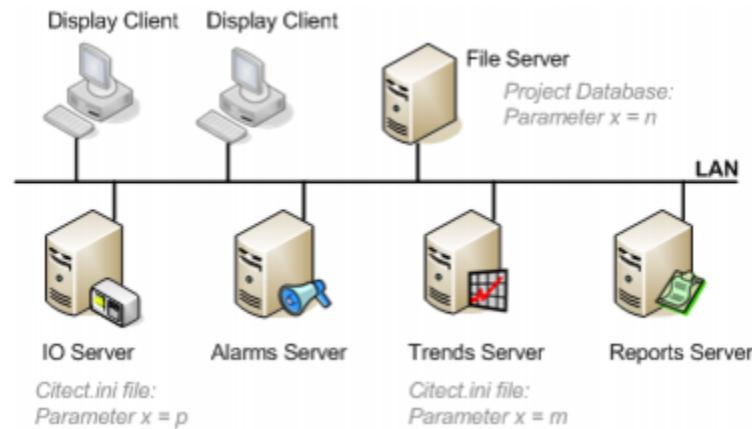
## See Also

- [Configure Project Database Parameters](#)
- [Configure Citect.ini Parameters](#)

## Parameter Precedence

On a machine where a parameter is set in **both** the project database and the Citect.ini file, the value contained in the Citect.ini will be used by that machine.

For example, in the diagram below, the project value for "parameter X" (which is stored in the project database) is **n**. This is the value used for parameter X on every server and client EXCEPT the I/O server and trends server, both of which use the values set in their local Citect.ini files (**p** and **m** respectively).



A parameter which is global to a project and applies to the majority of servers running a project is recommended therefore to be defined in the project database where it can be centrally managed and controlled. Any exceptions to this global value can then be managed by modifying the Citect.ini file on the machine to which the exception applies.

## Included projects

If the Citect.ini has no parameter setting, the value used will be the value specified in the lowest level include project. This means that it will then ignore any value placed in the top level projects at Runtime in favor of the lowest level include project which has the parameter set. Naturally if the parameter is not set anywhere, then the default value for the parameter will be used.

For example, using the parameter[Alarm]SavePeriod. The default value is 600. In this example there is a main project and an included sub project with the parameter being set as shown below. The runtime result column in the table below shows what Plant SCADA considers the value to be, for example if the ParameterGet() Cicode was used.

Citect.ini	Main	Include	Runtime Result
None	None	None	600
None	300	None	300
None	None	400	400
None	300	400	400
500	300	400	500
600	300	400	600

## Duplicate Parameters

If the parameter is defined multiple times either in the same project or different include project, the effective value of the parameter is unpredictable. The compiler will pick up the first entry of the parameter found in the project hierarchy, but does not check for duplicate entries.

This means it is not possible to override parameters defined in the included project by defining the same parameter (with different value) in the including project. It is recommended that when you create a new parameter for your project and give it a default value, you define the default value inline rather than defining it as a project parameter in the included project. For example:

Call ParameterGet("MySection", "MyParameter", MyDefaultValue) whenever needing to read the parameter.

MyDefaultValue can be defined as a label so it is accessible outside Cicode files, or as a global variable if the parameter is only used in Cicode files. By doing this, the value of the parameter is overridden in the including project (by defining the value for the parameter in the system parameter database).

However, if the same parameter is defined in the local Citect.ini file, it will always take precedence over the parameter defined in the project. But , if the parameter is defined multiple times in the Citect.ini file, only the first entry will be picked up.

### See also

[Hierarchical Parameters](#)

## Hierarchical Parameters

As Plant SCADA supports clustering and the ability to run multiple servers of the same type on one machine, there are circumstances when the server component parameters (alarm, report, trend, I/O server) need to be tuned to a finer level than just machine level. For this reason, these parameters are hierarchical parameters that are capable of being implemented at a number of levels:

- **Component type level**

The widest scope, the parameter value will apply to every instance of the server type.

- **Cluster level**

The parameter value will apply to every instance of the server running in the specified cluster.

- **Server level**

The parameter value will apply to the instance of the server running in the specified cluster, on the specified machine.

These parameters support **hierarchical inheritance**, that is, a parameter will:

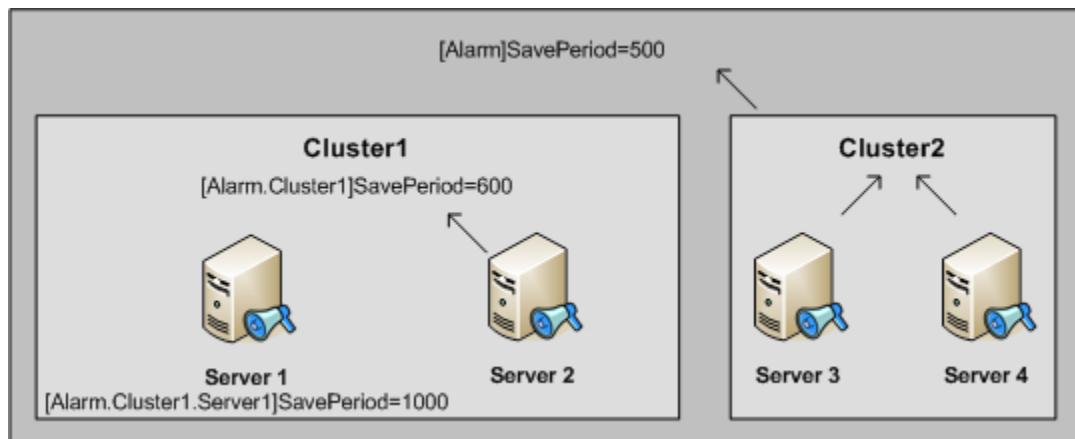
- Apply a value set for it at a server level;
- If that is not specified, apply a value set for it at a cluster level;
- If that is not specified, apply a value set for it at a component level;
- If that is not specified, apply the default value for that parameter.

## Example

The following Citect.ini file is applied to Server1 and Server2 (both in Cluster1), and to Server3 and Server4 (both in Cluster2).

```
[Alarm]SavePeriod = 500
[Alarm.Cluster1]SavePeriod = 600
[Alarm.Cluster1.Server1]SavePeriod = 1000
```

This is illustrated in the diagram below.



The values applied at each server for [Alarm]SavePeriod would be as follows:

Cluster Name	Computer Name	SavePeriod Value
Cluster1	Server1	1000
Cluster1	Server2	600
Cluster2	Server3	500
Cluster2	Server4	500

**Note:** Hierarchical parameters may be set in the parameters database. In this case normal rules of precedence will apply. For more information see [Parameter Precedence](#).

## See Also

[Configure Alarm Parameters for a Specific Cluster or Process](#)

## Computer Setup Editor

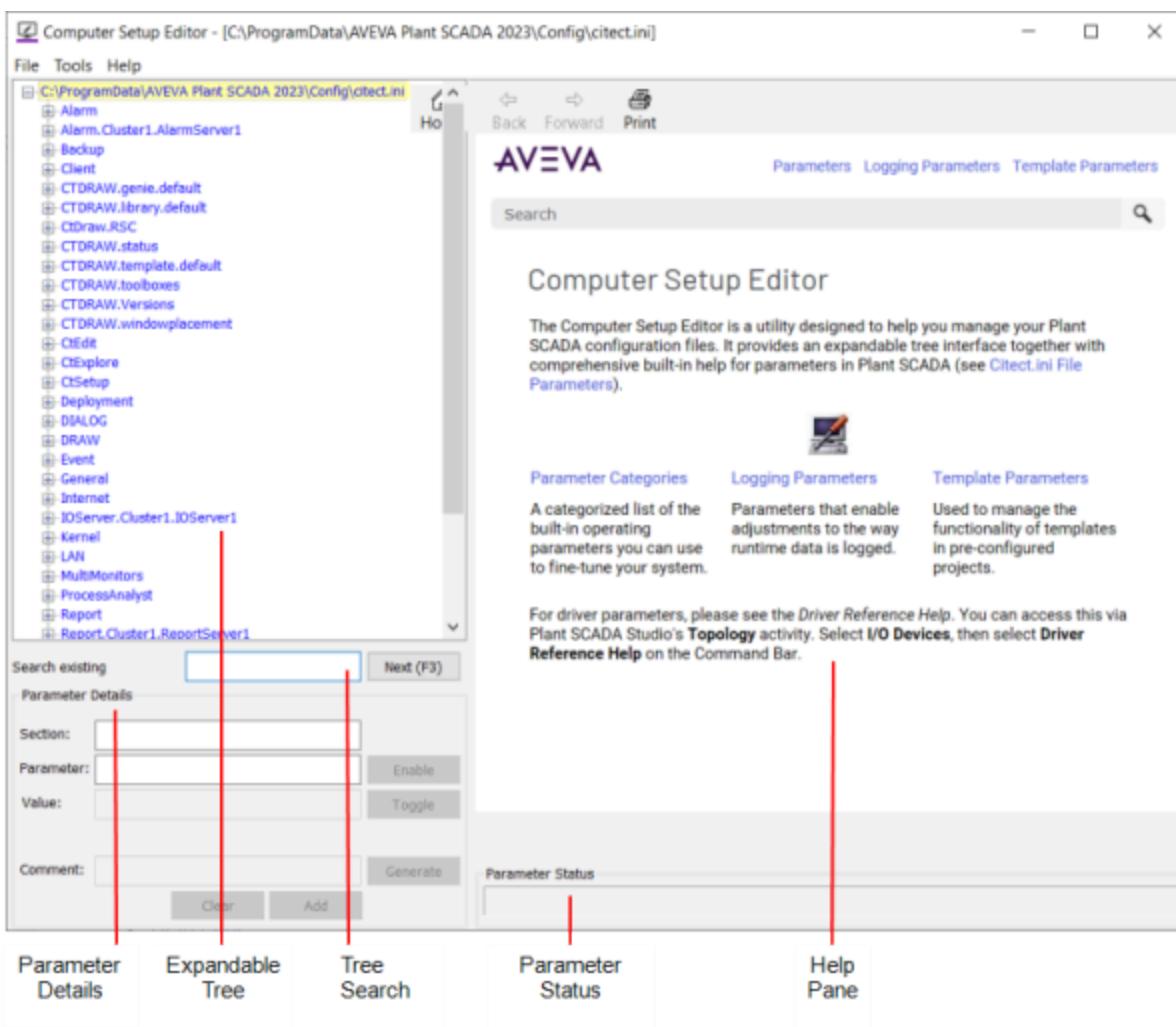
The Computer Setup Editor is a utility designed to help you manage your Citect.ini files. It combines a graphical user interface with an extensive Help system.

The interface provides a convenient way to view, search and modify parameters within a Citect.ini file. The Help system navigates the user through the parameter reference documentation, displaying the topic relevant to the current parameter.

You can use the Computer Setup Editor to perform the following tasks:

- Add or delete parameters
- Change the value of existing parameters (including the addition of optional comments)
- Save changes to disk, cancel all changes or backup to a new file
- Generate a Comparison Report between two separate configuration files, visually highlighting their differences
- Generate an Analysis Report on a configuration file, which provides a useful summary of parameter settings including validity checks on bounds and paths.

The Computer Setup Editor interface consists of the following components, which are described below.



## Expandable Tree

The expandable tree is used to display and navigate the contents of the Citect.ini file.

Each branch on the tree represents a section within the Citect.ini file. Each branch represents the parameters within that section. You can expand and collapse the tree nodes by clicking the "+" and "-" by each element, just as you would in Windows Explorer.

### To expand the tree:

1. Right-click the top element of the tree.
2. Choose Expand all from the shortcut menu.  
The configuration parameter tree expands to show all parameter entries.

### To collapse the tree:

1. Right-click the top element of the tree.
2. Choose Collapse all from the shortcut menu.  
The configuration parameter tree collapses to show only section entries.

## Parameter Details

You can use the Parameter Details pane to view and modify settings and comments for parameters and sections. For details see:

- [Edit Parameter Sections](#)
- [Edit Parameters](#)
- [Add Comments to the Citect.ini File.](#)

When a parameter or section entry is highlighted in the tree pane the corresponding values populate the Parameter Details pane and the corresponding Help files appear in the help pane.

Similarly, the Parameter Details pane populates when a Help topic is selected. This occurs whether or not the parameter exists in the current configuration file.

#### [Tree Search](#)

You can use the tree search to search for a particular element in the Computer Setup Editor's expandable tree.

#### **To search for a specific parameter in the tree:**

1. Type the search string in the **Search existing parameters** field.
  2. The tree expands and highlights the first occurrence of the text string you entered.
- 
- Note:** If the search string is not found a message appears in the Parameter Status pane. In this case, a hyperlink will appear in that pane which can be clicked to perform a search through the help reference topics.
- 
3. To continue to search for the next match within the parameter tree, click **Next** in the tree search pane, or press **F3**. To go to the previous instance, press **Shift + F3**.
  4. If the top or bottom of the Parameter Tree is reached, a message appears in the Parameter Reference pane.

#### [Help Pane](#)

The Help pane displays context-sensitive parameter reference topics.

When a section or parameter entry is selected in the expandable tree pane, the corresponding topic appears in the help pane. If the selected item does not have a help topic, a message to this effect appears in the help pane.

---

**Note:** The converse of the above is also true: when a Help topic is selected in the Help pane, the corresponding entry in the expandable tree pane (provided it exists) is automatically selected.

---

#### [Parameter Status](#)

The parameter status pane becomes active when either:

- A user selects a section or parameter entry in the expandable tree pane.
- A user selects a Help topic in the help pane.

Once active, the parameter status pane displays the name of the section entry being viewed, the name of the parameter entry being viewed (if applicable) and the status of that entry in the current configuration file.

The status shown will depend on whether a help topic exists for this parameter.

If there is no help topic the status pane will tell you so.

If there is a help topic the status pane will state whether:

- The section already exists in this configuration file;
- The parameter already exists in this configuration file;

- The section is not specified in this configuration file; or
- The parameter is not specified in this configuration file.

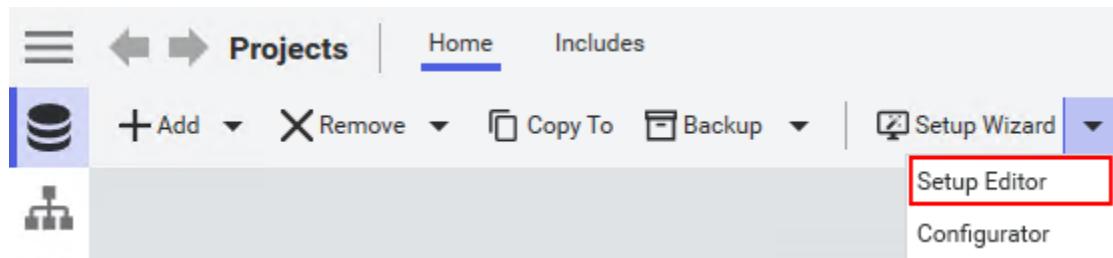
---

**Note:** Section and parameter names are active links that cause the corresponding help topics to be displayed in the help pane.

---

#### To open the Computer Setup Editor:

1. In Plant SCADA Studio, go to the **Projects** activity.
2. On the command bar, open the menu on the **Setup Wizard** button.
3. From the menu, select **Setup Editor**.



---

**Note:** The Computer Setup Editor cannot be used to maintain or set server parameters when being configured at the component or server level using Hierarchical Parameters.

---

#### See Also

[Configure Citect.ini Parameters](#)

## Configure Parameters

Plant SCADA supports both Project Database Parameters and Citect.ini File Parameters. The process you use to configure each type differs greatly.

Select one of the following topics depending on the type of parameter you would like to configure.

- [Configure Project Database Parameters](#)
- [Configure Citect.ini File Parameters](#)

#### See Also

[Parameter Categories](#)

## Configure Project Database Parameters

Project database parameters can be used to set nominal defaults for parameters that are local to a specific Plant SCADA project, rather than a particular computer. For example, you could use the project database parameters to specify login requirements for all computers in a Plant SCADA system.

To create or change parameters in the project database, you use the **Setup** activity.

**To set parameters in the project database:**

1. From the **Setup** activity, select **Parameters**.
2. Enter the **Section Name** of the parameter (16 characters or less).
3. Enter the **Name** of the parameter (16 characters or less).
4. Enter a **Value** for the parameter (254 characters or less).
5. Add a **Comment** if required.
6. Click **Save** to add the record to the database.

If you set (or change) parameters in the project database, you need to re-compile the project before the new parameter settings are used.

---

**Note:** Parameters that are set in the project database take lower precedence than those specified for a particular computer in the Citect.ini file. For more information, see [Parameter Precedence](#).

---

## Configure Citect.ini Parameters

This part of the help includes information that describes how to configure the Citect.ini file using the Computer Setup Editor. It includes the following sections:

- [Open the Citect.ini File](#)
- [Edit Parameter Sections](#)
- [Edit Parameters](#)
- [Add Comments to the Citect.ini File](#)
- [Configure Alarm Parameters for a Specific Cluster or Process](#)
- [Use the Comparison Wizard](#)
- [Saving the Citect.ini File.](#)

### Open the Citect.ini File

**To open the Citect.ini file:**

1. Choose **Open** from the **File** menu.
2. Highlight the required configuration file from the file selection window, then click **Open**.
3. The configuration file will open and display in the expandable tree pane.

**To view load notes:**

1. Choose **View Load Notes** from the **File** menu.
2. A Load Notes window opens and displays any notes taken while loading the configuration file.
3. Click **OK** to close the window.

## See Also

[Configure Citect.ini Parameters](#)

## Edit Parameter Sections

You can edit the sections settings in your Citect.ini file by performing the following tasks:

### To add a section to a Citect.ini file:

1. Right-click on an entry in the expandable tree pane.
2. Choose **Add Section** from the shortcut menu to display the Parameter Details pane.
3. The cursor moves to the **Section** text box in the Parameter Details pane. Type the name of the section you want to add.

Two shortcuts exist to reach this step:

- a. Enter a new section name in the Parameter Details pane. The Computer Setup Editor activates the **Add** button.
  - b. Navigate the Help reference topics within the help pane to locate the topic for the section to be added to the parameter tree, and click the relevant topic. The Computer Setup Editor populates the Parameter Details pane accordingly.
4. Click **Add** to add the new section to the parameter tree.
  5. The Computer Setup Editor checks to see whether the section name is recognized in the parameter reference topics. An alert window displays if the name specified is not recognized. Click **Yes** to proceed.

The new section appears in the parameter tree. It is highlighted as the currently selected entry, and its Help appears in the help pane.

### To delete a section:

1. Right-click the section to be deleted in the expandable tree pane.
2. Choose **Delete Section** from the shortcut menu to display a confirmation window.
3. The Computer Setup Editor opens an alert window requesting confirmation of this action. Click **Yes** to continue.
4. The section is deleted from the parameter tree.

If there are undocumented parameters not currently visible in the tree that exist in the file, an alert appears to advise you that you can view these undocumented parameters, and the undocumented parameters are not deleted. See [View Undocumented Parameters](#) for details.

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

- Do not under any circumstances change or remove any of the undocumented Citect.ini parameters.
- Before deleting sections of the Citect.ini file, confirm that no undocumented parameters will be deleted.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Plant SCADA support personnel regarding undocumented features.

#### To disable a section in a Citect.ini file:

Disabling a section within the parameter tree causes the section and all associated parameters to be treated as a comment in the Citect.ini file when the file is read at startup time.

**Note:** Disabled sections are not deleted from the file and can be enabled later. See [Enable a Section](#) for details.

1. Right-click the section you want to disable in the expandable tree pane.
2. Choose **Disable** from the shortcut menu.
3. The section is disabled and the text in that section within the parameter tree changes to red to indicate its disabled status.

#### To enable a section in a Citect.ini file:

1. Right-click the section you want to enable in the expandable tree pane.
2. Choose **Enable** from the shortcut menu.

The section is enabled and the text in that section within the parameter tree changes to black to indicate its enabled status.

**Note:** Some alarm parameters can be applied to a specified cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

## See Also

[Configure Citect.ini Parameters](#)

## Edit Parameters

You can edit the parameters in a Citect.ini file in the following ways:

## Add a Parameter

#### To add a parameter to a Citect.ini file:

1. Right-click the section in the expandable tree pane you want to add the parameter to.
2. Choose **Add Parameter** from the shortcut menu.
3. The section name pre populates the Parameter Details pane. Type the name of the parameter you want to add in the **Parameter** text box.

Two shortcuts exist to reach this step:

- a. Locate the section for the new parameter in the expandable tree pane, click it, then enter a new parameter name in the **Parameter** text box of the Parameter Details pane. The Computer Setup Editor activates the add button.
- b. Navigate the Help reference topics within the help pane to locate the topic for the parameter to be

added to the parameter tree, and click the relevant topic. The Computer Setup Editor populates the Parameter Details pane. (In the case of Driver parameter reference topics - click on the **Add Parameter Wizard** link to populate the Parameter Details pane.)

4. Enter the value for this parameter in the value field.

The Parameter Details pane changes to reflect the type the parameter:

- If the parameter has a path value a **Browse** button becomes available to help you set this parameter.
- If the parameter has a Boolean value, a **Toggle** button becomes available to help you set this parameter.
- If the parameter has a string value, the button to the left of the field becomes unavailable. Type the value in the **Value** field.

5. Click **Add** to add the new parameter to the parameter tree.

The new parameter appears in the parameter tree. It is highlighted as the currently selected entry, and its Help appears in the help pane.

If the parameter being added to the tree belongs to a section yet to be defined in the tree, both the section and parameter are added to the parameter tree when you click **Add**.

If the parameter being added to the parameter tree is an undocumented parameter, it appears in the parameter tree. However, if you hide undocumented parameters, you'll only be able to see parameters when you view undocumented parameters. See [View Undocumented Parameters](#) for details.

## Edit a Parameter

### To edit a parameter value:

1. Click the parameter in the expandable tree pane.
2. The current value populates the Parameter Details pane.
3. Click in the **Value** text box. The last saved value in the Citect.ini file will be displayed in the area directly below the text box.
4. Change the entry in the **Value** text box to the required value. The changes automatically appear in the parameter tree.

---

**Note:** The changes will not occur to the configuration file until they are saved. See [Saving the Citect.ini File](#) for details.

## Delete a Parameter

### To delete a parameter value from the Citect.ini file:

1. Right-click the parameter in the expandable tree pane.
2. Choose **Delete Parameter** from the shortcut menu.
3. A window opens requesting confirmation to delete the parameter.
4. Click **Yes**. The parameter is deleted from the parameter tree.

---

**Note:** Deleting parameters from the configuration file sets them to the default value (if one exists).

## Disable a Parameter

Disabling a parameter within the parameter tree causes the parameter to be treated as a comment in the Citect.ini file when the file is read at startup time.

Parameter values which are disabled are not deleted from the file and can be enabled at a later time. See Enable a Parameter for details.

### To disable a parameter:

1. Right-click the parameter you want to disable in the expandable tree pane.
2. Choose **Disable** from the shortcut menu.

The parameter is disabled and the text for that parameter within the parameter tree changes to red to indicate it is now disabled.

## Enable a Parameter

### To enable a parameter in the Citect.ini file:

1. Right-click the parameter you wish to enable in the expandable tree pane.
2. Choose **Enable** from the shortcut menu.

The parameter is enabled and the text for that parameter within the parameter tree changes to black to indicate it is now enabled.

## See Also

[Configure Citect.ini Parameters](#)

[View Undocumented Parameters.](#)

## Configure Alarm Parameters for a Specific Cluster or Process

Plant SCADA allows you to configure some alarm parameters for a specific cluster or alarm server process. This is defined in a Citect.ini file using the following syntax.

To define alarm parameters for a specific cluster:

```
[Alarm.<Cluster Name>]  
<Parameter Name>
```

To define alarm parameters for a specific server process:

```
[Alarm.<Cluster Name>.<Server Name>]  
<Parameter Name>
```

For example:

```
[Alarm.Cluster1.AlarmServer1]  
Clusters = Cluster1  
CPU = 0
```

An alarm parameter can also be defined on a client computer or a server computer, depending on the functionality it supports. This means there are five different ways you can use alarm parameters in a Citect.ini

file:

1. For all clusters and server processes in the general [**Alarm**] section.
2. For all alarms in a specific cluster in the [**Alarm.<ClusterName>**] section.
3. For a specific server process in the [**Alarm.<ClusterName>.<ServerName>**] section.
4. On a **client** computer.
5. On a **server** computer.

The following table lists Plant SCADA's alarm parameters and indicates the different ways each can be used.

Alarm Parameter	Supported Usage in Citect.ini				
	1. General	2. Cluster specific	3. Server specific	4. Client computer	5. Server computer
AckHold	x	x	x		x
Active	x	x	x		x
AlarmDisable	x	x	x	x	x
AlarmListRequestTimeout	x			x	x
ArchiveAfter	x	x	x		x
BackgroundColorMode	x			x	
BrowseRowLimit	x			x	x
CacheSize	x	x	x		x
ClientConnectTimeout			x		x
ClientDisconnectTimeout			x		x
ClientRequestTimeout			x		x
Clusters			x		x
CPU			x		x
DBLogDBServer	x	x	x		x
DBLogHistoric	x	x	x		x
DBLogServerCor	x	x	x		x

e					
DefDspFmt	x			x	
DefSOEFmt	x			x	
DefSumFmt	x			x	
DeltaTimeUpdate	x			x	
DisableConnection			x	x	
DisableFilterOptimization	x	x	x		x
DisableSOE	x	x	x		x
DisableSummary	x	x	x		x
DisplayDisable	x	x	x		x
EnableErrorLogging	x			x	x
EnableSmartCustomFilters	x	x	x		x
EnableStateLogging	x			x	x
EventFmt	x	x	x		x
EventQue	x	x	x		x
Events	x	x	x	x	x
ExtendedDate	x			x	
FlushTimeLimit	x	x	x		x
FutureMessages	x	x	x		x
HardHoldTime	x			x	x
HardwareDisable	x			x	x
HeadingFont	x			x	
HeartbeatTimeo	x	x	x		x

ut					
HighResOff	x	x	x		x
Hres24HrDeadBand	x				x
HresTimerExprDelay	x				x
HresType	x				x
HwAlarmFmt	x			x	x
HwAlarmQueMax	x			x	x
HwExclude	x			x	x
IsolationDetectInterval	x	x	x		x
IsolationDetectIP1	x	x	x		x
IsolationDetectIP2	x	x	x		x
IsolationDetectRetryCount	x	x	x		x
KeepOnlineFor	x	x	x		x
LastAlarmCategories	x			x	
LastAlarmDisplayMode	x			x	
LastAlarmFmt	x			x	
LastAlarmPriorities	x			x	
LastAlarmType	x			x	
MaxOptimisedQueries	x	x	x		x
MaxQueryExecuteTime	x				x

MemoryWarningLimit	x	x	x		x
MonitorConnectTimeout	x	x	x		x
MonitorRequestTimeout	x	x	x		x
Period	x	x	x		x
Priority			x	x	
QueryCPUUsage	x	x	x		x
QueryRowLimit	x	x	x		x
QueryTimeout	x	x	x		x
ReloadBackOffTime	x	x	x		x
SavePrimary	x	x	x		x
SaveSecondary	x	x	x		x
ScanTime	x	x	x		x
SetTimeOnAck	x	x	x	x	x
SetTimeOnOff	x	x	x		x
ShutdownCode	x	x	x		x
ShowAllConfigured	x				x
Sort	x			x	
SortMode	x			x	
Sound<n>	x			x	
Sound<n>Interval	x			x	
SoundSilenceTimeoutOnAck	x			x	
StandbyCommandDelay	x	x	x		x

StartTimeout	x	x	x		x
StartupCode			x		x
StreamSize	x	x	x		x
SummaryAutoRefreshMode	x			x	
SummaryBufferSize	x	x	x		x
SummaryMode	x	x	x		x
SummaryShutdownMode	x	x	x		x
SummarySort	x	x	x	x	x
SummarySortMode	x			x	
SummaryTimeout	x	x	x		x
SummaryTimeoutTolerance	x	x	x		x
SumStartupCopy	x	x	x		x
SyncAllHistoricData	x	x	x		x
TimeDate	x			x	
TransferConnectTimeout	x	x	x		x
TransferInterleave	x	x	x		x
TransferInterval	x	x	x		x
TransferRequestTimeout	x	x	x		x
UseConfigLimits	x				x

## See Also

[Configure Citect.ini Parameters](#)

[Alarm Parameters](#)[Alarm Process Parameters](#)

## View Undocumented Parameters

When a configuration file is loaded by the Computer Setup Editor, only those parameters that are documented in the Parameter Help are visible in the parameter tree. There are, however, many other parameters (mostly used by Plant SCADA itself) for which no help topics exist; these are known as undocumented parameters.

### To show undocumented parameter:

1. Right click the top element of the parameter tree.
2. Choose **Show undocumented Sections and Parameters** from the shortcut menu.  
The parameter tree expands to show all undocumented sections and parameters.

### To hide undocumented parameters:

1. Right click the top element of the parameter tree.
2. Choose **Hide undocumented Sections and Parameters** from the shortcut menu.  
The undocumented sections and parameters are hidden.

### Setting the default option for viewing undocumented parameters

1. Choose **Options** from the **Tools** menu.  
The Computer Setup Editor Options window opens.
2. Select the checkbox to display undocumented parameters by default.
3. Click **OK**.

When Computer Setup Editor next opens a configuration file, this setting will be used to determine whether undocumented parameters are displayed in the parameter tree.

## See Also

[Configure Citect.ini Parameters](#)

## Add Comments to the Citect.ini File

You can add comments to your configuration file.

### To add a comment to a section entry:

1. In the expandable tree pane, click the section to which you would like to add a comment.
2. The Parameter Details pane populates with the section information.
3. Type the comment in the **Comment** field.

Or:

Click **Generate** to insert the auto-generated comment "Added by {username} on {date}".

The comment appears next to the section entry in the parameter tree.

**Note:**

- Comments can only be one line long. If the comment needs to be longer than one line, you should include it in the header of the file.
- Using the **Generate** button replaces any existing comment made in relation to that section. To append comments type your text into **Comment** field.

**To add a comment to a parameter entry:**

1. Click the parameter in the expandable tree you want to add a comment to.
2. The Parameter Details pane populates with the parameter information.
3. Type the comment in the **Comment** field.

Or:

Click **Generate** to insert the auto-generated comment "Added by {username} on {date}".

The comment appears next to the parameter entry in the parameter tree.

**Note:** Comments can only be one line long. If the comment needs to be longer than one line, you should include it in the header comment. See Enter Comments in the Header for details.

**To enter comments in the header:**

1. Choose **View Header** from the **File** menu.
2. The Header Information window opens. Comments added in this window are stored in the header of the configuration file.

**Note:** Click **Insert time/date** to enter the current date and time where the cursor is placed in the text window.

3. Click **OK** to store the changes made to the header file.

**See Also**

[Configure Citect.ini Parameters](#)

**Use the Comparison Wizard**

The Comparison Wizard allows you to compare the content of two separate configuration files. For further information see:

- [Run the Comparison Wizard](#)
- [Interpret the Comparison Wizard](#)
- [Copy Values Between Compared Files](#)
- [Save Changes to the Compared File](#)
- [Close Without Saving Changes](#)

## See Also

[Configure Citect.ini Parameters](#)

### Run the Comparison Wizard

#### To run the Comparison Wizard:

1. Open the source configuration file in the Computer Setup Editor.
  2. Choose **Compare INI File** from the **Tools** menu.
  3. A **Choose comparison file** window opens. Navigate to the file you want to compare (**comparison file**) with the file currently loaded in the Computer Setup Editor (**source file**). Click **Open**.
- The Comparison Wizard opens in a new window.

## See Also

[Use the Comparison Wizard](#)

### Interpret the Comparison Wizard

The Comparison Wizard displays the source file and the comparison file side by side.

It includes the following information:

<b>Number of differences between the files</b>	A summary of the number of lines different between the files is shown in the title of the window.
<b>Parameters with the same values</b>	Lines that are not highlighted indicate that the parameters are set in both files and have the same values.
<b>Parameters with different values</b>	Lines that are highlighted in yellow indicate that the parameters are set in both files but have different values.
<b>Parameters not set in both files</b>	Lines that are highlighted in gray indicate that the parameters are set in only one of the files.

## See Also

[Use the Comparison Wizard](#)

### Copy Values Between Compared Files

Where parameter values are different, the Comparison Wizard allows changes to be made to either file to make them the same.

**To copy a value from the source file to the comparison file:**

1. In the source file locate the parameter you want to copy to the comparison file.
2. Right-click the line, it is highlighted in blue and a menu appears with **Copy to RHS**. Select this menu.

The Parameter value copies to the comparison file and the line is no longer highlighted.

---

**Note:** To select multiple lines in the source file use the Ctrl and/or Shift keys in combination with the mouse, and then right-click any line of the selected lines and choose **Copy to RHS**.

---

**To copy a value from the comparison file to the source file:**

1. In the comparison file and locate the parameter you want to copy to the source file.
2. Right-click the line, it is highlighted blue and a menu appear with **Copy to LHS**. Select this menu.

The Parameter value copies to the Source File and the line is no longer highlighted.

---

**Note:** To select multiple lines in the comparison file use the Ctrl and/or Shift keys in combination with the mouse, and then right-click any line of the selected lines and choose **Copy to LHS**.

---

**See Also**

[Use the Comparison Wizard](#)

**Undo and Redo Changes to the Compared Files**

Where parameter values have been copied from one file to another, the **Undo** button becomes active providing you with an option to undo the change.

Undoing the change will return the file to the state before the copy was executed. The undo buffer is limited only by the available memory so all changes made in the file can be undone in the same sequence as they occurred. Once the Undo button has been used the **Redo** button becomes active.

Redoing the change will return the file to the state before the last change which was undone. The redo buffer is limited only by the available memory, so all undo actions can be redone in the same sequence as they occurred.

---

**Note:** As soon as a fresh copy between the files is executed the Redo buffer is reset and all previous undo actions can no longer be redone with the **Redo** button.

---

**See Also**

[Use the Comparison Wizard](#)

**Save Changes to the Compared File**

Changing either the source file or the comparison file activates the **Accept Changes** button. To save changes, click **Accept Changes** located above the relevant file.

Changes accepted to the comparison file are saved directly to the file.

Changes accepted to the Source File are only made to the copy rendered in the expandable tree pane. Use the **Save** command to save the changes to the file. See [Saving the Citect.ini File](#) for details.

## See Also

[Use the Comparison Wizard](#)

### Close Without Saving Changes

To close the Comparison Wizard without saving your changes, click **Cancel** located in the lower right corner of the window.

## See Also

[Use the Comparison Wizard](#)

### Saving the Citect.ini File

#### To save any changes to the Citect.ini file:

1. Choose **Save** from the **File** menu.
2. An alert window opens requesting confirmation that the changes are to be saved. Click **OK**.  
The configuration changes are saved to file.



#### UNINTENDED EQUIPMENT OPERATION

After changes to settings that are not online changes, recompile your project and restart the system to ensure all changed settings are applied.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

#### To save a backup of changes to the Citect.ini file:

1. Choose **Save As** from the **File** menu.
2. A Save As window opens. Enter the name and location of the backup file. Click **Save**.
3. An alert window opens requesting confirmation of the name and location of the file. Click **OK** to proceed.  
A confirmation window opens to confirm the backup has occurred. Click **OK** to continue.

#### To abandon changes to the configuration file:

1. Choose **Reload** from the **File** menu.
2. If there are unsaved changes to the currently loaded file, an alert window opens requesting confirmation that the unsaved changes are to be discarded. Click **Yes** to proceed.  
The configuration file will be reloaded without the unsaved changes.

## See Also

[Configure Citect.ini Parameters](#)

## Parameters Reference

Plant SCADA has a comprehensive set of parameters that you can use to configure the following:

- Operational settings for a project
- Operational settings for each computer in a Plant SCADA system.

For more information, see [About Parameters](#).

This section of the help describes these parameters and the settings that each supports. It includes the following topics:

- [Parameter Categories](#) - a complete list of built-in operating parameters
- [Logging Parameters](#) - information on how parameters can be used to gather Runtime information about your Plant SCADA system
- [Template Parameters](#) - information on parameters you can use to configure templates in pre-configured projects.

### See also

[Configure Parameters](#)

## Parameter Categories

Plant SCADA has built-in operating parameters that you can use to fine-tune your system. The table below lists the parameter categories.

A	B	C
Accumulator Parameters	DisableIO Parameters	OpcDaServer Parameters
Address Forwarding Parameters	DiskDrv Parameters	P
Alarm Parameters	Driver Parameters	Page Parameters
Alarm Process Parameters	E	Path Parameters
AlarmFilterRule Parameters	Event Parameters	Privilege Parameters
Alarm Heading Parameters	F	ProcessAnalyst Parameters
Alarm Log Parameters	Font Parameters	Protocol Parameters
Animator Parameters	G	Proxi Parameters
AnmCursor Parameters	General Parameters	PubSub Parameters
B	I	R
Backup Parameters	Interlock Parameters	RemoteDB Parameters
BrowseTableView Parameters	IOServer Parameters	Report Parameters
		Report Process Parameters

<b>C</b>	<a href="#">IOServer Process Parameters</a> <a href="#">IPC Parameters</a> <a href="#">K</a> <a href="#">Kernel Parameters</a> <a href="#">Keyboard Parameters</a> <a href="#">L</a> <a href="#">LAN Parameters</a> <a href="#">Language Parameters</a> <a href="#">M</a> <a href="#">Memory Parameters</a> <a href="#">MultiMonitors Parameters</a>	<a href="#">Runtime Manager Parameters</a> <a href="#">S</a> <a href="#">[SA_Library.Meter] Parameters</a> <a href="#">Scheduling Parameters</a> <a href="#">Security Parameters</a> <a href="#">Server Parameters</a> <a href="#">Shutdown Parameters</a> <a href="#">SPC Parameters</a> <a href="#">SQL Parameters</a> <a href="#">T</a> <a href="#">Time Parameters</a> <a href="#">Trend Parameters</a> <a href="#">Trend Process Parameters</a> <a href="#">W</a> <a href="#">Win Parameters</a> <a href="#">Workspace Parameters</a>
<b>D</b>	<a href="#">DDE Parameters</a> <a href="#">Debug Parameters</a> <a href="#">Deployment Parameters</a> <a href="#">Device Parameters</a> <a href="#">Dial Parameters</a>	<a href="#">ODBC Parameters</a> <a href="#">OID Parameters</a>

## Computer-specific Parameters

The following parameters need to be set in the main Citect.INI file, and cannot be set via a profile.

- [\[CtEdit\]Data](#)
- [\[CtEdit\]Run](#)
- [\[CtEdit\]Logs](#)
- [\[CtEdit\]Config](#)
- [\[CtEdit\]User](#)
- [\[Client\]PartOfTrustedNetwork](#)

## See Also

[Parameter Categories](#)

## Accumulator Parameters

The Citect.ini file contains the following accumulator parameters:

- [\[Accumulator\]Debug](#) - Enables trace logging messages for accumulators.
- [\[Accumulator\]UpdateTime](#) - The time in seconds between writing the accumulator variables to the I/O device.
- [\[Accumulator\]WatchTime](#) - The time in seconds between each check of the accumulator trigger.

## See Also

[Parameter Categories](#)

### [Accumulator]Debug

Enables trace logging messages.

When set to "1", entries appear in the appropriate SYSLOG.DAT file. Entries are prefixed by "Accumulators:". 0 disables logging.

## Allowable Values:

- 0 = Logging is disabled.
- 1 = Logging is enabled.

## See Also

[Accumulator Parameters](#)

### [Accumulator]UpdateTime

The time in seconds between writing the accumulator variables to the I/O device. You can set this parameter to a small value to get faster update of accumulator values, but you would use more CPU and PLC communication bandwidth. You should set this parameter as high as possible to reduce the load on the PLC.

## Allowable Values:

1 to 36000

## Default Value:

1

## See Also

[Accumulator Parameters](#)

### [Accumulator]WatchTime

The time in seconds between each check of the accumulator trigger. You can set this parameter to a large value to conserve CPU and PLC communication time. If you set the WatchTime to a low value, Plant SCADA must poll the PLC more often for the trigger value, using CPU and PLC communication bandwidth.

**Allowable Values:**

1 to 36000 (seconds)

**Default Value:**

60

**See Also**

[Accumulator Parameters](#)

**Address Forwarding Parameters**

Use address forwarding to override the IP addresses that have been configured inside the project for the server processes. This allows you to quickly transfer responsibility of a server to another computer, or redirect a client to a different server. Address forwarding allows you to override the network address for each Plant SCADA server or client with a different address and port.

Forwarding is achieved by adding a special section to the Citect.ini.

[AddressForwarding]

<ClusterName>.<ServerName>=<IPv4 ipaddress>:<port>

Or:

[AddressForwarding]

<ClusterName>.<ServerName>=<[IPv6] ipaddress>:<port>

Address forwarding is only interpreted and checked at startup of Plant SCADA Runtime. Run the Computer Setup Wizard to confirm your changes. This is useful especially for Time Servers where you can configure a redirected Server as a new time server.

---

**Note:** When using this feature the compiler will not be able to apply consistency checks to your configuration. Therefore, you should not use this feature as the primary mechanism of specifying server/client communications.

---

**See Also**

[Parameter Categories](#)

**Alarm Heading Parameters**

The [AlarmHeading] section contains the following parameters:

- [\[AlarmHeading\]AlarmField](#) - Specifies alternative names for the standard alarm field names for display in the column heading of the new Tab\_Style alarm template.

**See Also**

[Alarm Parameters](#)

[Alarm Process Parameters](#)

## Parameter Categories

### [AlarmHeading]AlarmField

Specifies alternative names to use for the column headings in Tab\_Style and Situational Awareness projects.

#### Allowable Parameters:

Any valid alarm field name.

#### Allowable Values:

Non-blank string.

#### Default Value:

Specified alarm field name.

## Examples

Tab\_Style template:

```
[AlarmHeading]
OnDate = Date
OnTime = Time
SumDesc = Description
Custom1 = Equipment
Type = State
```

Situational Awareness project:

```
[AlarmHeading]
DisableEndDate=Shelve End Date
DisableEndTime=Shelve End Time
DisableComment=Shelve Comment.
```

## See Also

[Alarm Heading Parameters](#)

## Alarm Log Parameters

These parameters affect the Plant SCADA alarm log format. Alarm logs can be accessed through the CtAPI and searched for specific records via the CtAPI function ctFindFirst and using CtAPIAlarmLog for the szTableName parameter.

Log files are named in the format "YYYYMMDD.TXT" and reside in the Plant SCADA DATA directory.

The first line of an alarm log will list the log format used, and each alarm log entry will reside on subsequent separate lines.

Alarms are logged in the order they are received, except for time-stamped alarms, which are stored in the log relevant to the date of the alarm, which may not be the current date. Therefore, when retrieving alarm log records, you should sort the results appropriately.

Alarm logging is disabled by default and enabled by setting "NumFiles" to a value greater than zero.

The Citect.ini file contains the following alarm logging parameters:

- [\[AlarmLog\]DefaultSearchDays](#) - Determines the number of logs to search.
- [\[AlarmLog\]Format](#) - Determines the format of the alarm log files.
- [\[AlarmLog\]NumFiles](#) - Determines the number of log files to be maintained.

## See Also

[Alarm Parameters](#)

[Alarm Process Parameters](#)

[Parameter Categories](#)

### [AlarmLog]DefaultSearchDays

Determines the number of alarm logs to search for.

#### Allowable Values:

0 to 32767

#### Default Value:

The value of [\[AlarmLog\]NumFiles](#) if it has been set, otherwise 100.

## See Also

[Alarm Log Parameters](#)

### [AlarmLog]Format

Determines the format of the alarm log files.

---

**Note:** You should keep the format at its default, as Plant SCADA has been optimized for this format. If the logging format is changed, it is the user's responsibility to archive any existing log files of the previous format before startup.

---

A user specified format must contain the {Tag} and {Category} fields and either the {LocalTimeDate} field or both {Time} and {DateExt} fields. It may also contain any of the following fields: {Desc}, {State}, {LogState}, {Type}, {Name}, {Millisec}, {Help}, {Area}, {Priv} and {Value}.

By default, alarms are logged using the alarm field type of {LocalTimeDate}. This field generates a string in the fixed format "yyyy-mm-dd hh:mm:ss[.ttt]". The time and date are local, and the millisecond field is only appended when the alarm contains millisecond data. This field type is useful for international applications.

**Allowable Values:**

Any combination of Alarm Display Fields.

**Default Value:**

"{LocalTimeDate,23} {Tag,32} {Desc,80} {Category,16} {LogState,16} {State,16} {State\_Desc,16} {Type,16}"

**See Also**

[Alarm Log Parameters](#)

**[AlarmLog]NumFiles**

Alarm logging is disabled by default, and enabled by setting "NumFiles" to a value greater than zero.

Determines the number of log files to be maintained. When this number is exceeded, Plant SCADA deletes the oldest file. However, the user can retain logs by manually marking them as "Read Only".

**Allowable Values:**

- 0 - (Disabled)
- 1 to 32767 - (Enabled)

**Default Value:**

0

**See Also**

[Alarm Log Parameters](#)

**Alarm Parameters**

The Citect.ini file contains the following alarm parameters.

---

**Note:** Some alarm parameters can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

- [Alarm]Ack - Obsolete in version 7.30.
- [Alarm]AckHold - Determines whether alarms that have become inactive (and have been acknowledged) remain in the OFF ACKNOWLEDGED alarm list.
- [Alarm]Active - The period (in milliseconds) at which the clients are advised of a change in alarms.
- [Alarm]AlarmDisable - Enables/disables the processing of alarms.
- [Alarm]AlarmListRequestTimeout - Specifies the length of time (in seconds) that an alarm display will wait to receive data from all clusters.

- [\[Alarm\]ArchiveAfter](#) - Sets the amount of time between each archive of Event Journal data.
- [\[Alarm\]ArgyleTagValueTimeout](#) - Obsolete in version 2015.
- [\[Alarm\]BackgroundColorMode](#) - Sets the background color mode of the alarm list and alarm banner on the Tab\_Style templates.
- [\[Alarm\]BrowseRowLimit](#) - Defines the maximum number of rows per cluster that will be returned in an alarm browse.
- [\[Alarm\]CacheLength](#) - Obsolete in version 7.30.
- [\[Alarm\]CacheSize](#) - Defines the amount of memory (in megabytes) dedicated to the storage of event data.
- [\[Alarm\]DBLogDBServer](#) - Defines logging for the alarm server database.
- [\[Alarm\]DBLogHistoric](#) - Defines logging for Historian operations on the alarm server database.
- [\[Alarm\]DBLogServerCore](#) - Defines logging for the analysis of redundancy and synchronization issues.
- [\[Alarm\]DefaultSOETimeRange](#) - Obsolete in version 2015.
- [\[Alarm\]DefDspFmt](#) - Specifies an Alarm display format to use if the Alarms Display Format field is blank (in Alarm Categories).
- [\[Alarm\]DefSOEFmt](#) - Specifies an SOE display format to use if the SOE Display Format field is blank (in Alarm Categories).
- [\[Alarm\]DefSumFmt](#) - Specifies an Alarm Summary display format to use if the Summary Display Format field is blank (in Alarm Categories).
- [\[Alarm\]DeltaTimeUpdate](#) - Determines if DeltaTime (duration) field is set on non-OFF alarms by calculating volatile duration between current time and the time when the alarm was activated.
- [\[Alarm\]DisableFilterOptimization](#) - Allows you to turn off filter optimization.
- [\[Alarm\]DisableSOE](#) - Used to turn off the storage of events.
- [\[Alarm\]DisableSummary](#) -Used to turn off the storage of summary events.
- [\[Alarm\]DisplayDisable](#) - Changes the meaning of the AlarmDisable() function.
- [\[Alarm\]EnableErrorLogging](#) - Enables or disables the logging of operational errors.
- [\[Alarm\]EnableSmartCustomFilters](#) - Enables/disables the caching of custom alarm queries to facilitate the use of the same filter later.
- [\[Alarm\]EnableStateLogging](#) - Enables or disables the logging of major alarm system state changes.
- [\[Alarm\]EventFmt](#) - Specifies the format of the event queue string.
- [\[Alarm\]EventQue](#) - Enables/disables the Alarm event queue on the Alarms Server.
- [\[Alarm\]ExtendedDate](#) - Specifies if extended date format is used in alarm formatting.
- [\[Alarm\]FilterViewByPrivilege](#) - Obsolete in v7.30.
- [\[Alarm\]FlushTimeLimit](#) - The alarm server flushes Event Journal entries to disk periodically (every 60 seconds). Set this parameter to determine the maximum duration of each alarm server flush.
- [\[Alarm\]FutureMessages](#) - Defines the maximum amount of time in the future that is permitted for future Event Journal messages
- [\[Alarm\]HardHoldTime](#) - The time (in seconds) to hold a hardware alarm in its alarm state before it is reset.
- [\[Alarm\]HardReAlarm](#) - Obsolete in v7.0.
- [\[Alarm\]HardwareDisable](#) - Enables/disables the processing of hardware alarms.
- [\[Alarm\]HeadingFont](#) - Determines the text font used for column headings displayed on the alarm templates.

- [\[Alarm\]HeartbeatTimeout](#) - Defines how long a server will wait before terminating a link that has been used for receiving heartbeat poll requests from its pair server, but is currently idle.
- [\[Alarm\]HighResOff](#) - Sets the precision of time stamped alarms.
- [\[Alarm\]Hres24HrDeadBand](#) - Determines the "previous day" deadband on time stamped alarms.
- [\[Alarm\]HresTimerExprDelay](#) - Allows you to configure and fine tune time delay.
- [\[Alarm\]HresType](#) - Determines the time format used to record time stamped alarms.
- [\[Alarm\]HwAlarmFmt](#) - Defines the format of the data written into the hardware alarm queue and placed in the string field.
- [\[Alarm\]HwAlarmQueMax](#) - Specifies the maximum length of the hardware alarm queue.
- [\[Alarm\]HwExclude](#) - Specifies a list of hardware alarms that the hardware alarm system will not process.
- [\[Alarm\]IsolationDetectInterval](#) - Sets the interval between ICMP packets to detect network isolation on alarm servers.
- [\[Alarm\]IsolationDetectIP1](#) - Determines the status of the disconnected alarm server and keeps the disconnected alarm server in 'non-main' state. Use this parameter to apply setting to all servers on machine.
- [\[Alarm\]IsolationDetectIP2](#) - Determines the status of the disconnected alarm server and keeps the disconnected alarm server in 'non-main' state. Use this parameter to apply setting to all servers on machine.
- [\[Alarm\]IsolationDetectRetryCount](#) - Sets the ICMP retry count to detect network isolation on alarm servers.
- [\[Alarm\]KeepOnlineFor](#) - Sets the amount of time for which the Alarm Server stores event messages on-line.
- [\[Alarm\]LastAlarmCategories](#) - Determines which alarms are displayed on the alarms toolbar, based on category.
- [\[Alarm\]LastAlarmDisplayStyle](#) - Determines whether the last alarm is displayed by category or priority.
- [\[Alarm\]LastAlarmFmt](#) - Specifies the format of the last alarm as displayed on the included templates.
- [\[Alarm\]LastAlarmPriorities](#) - Determines which alarms are displayed on the alarms toolbar, based on priority.
- [\[Alarm\]LastAlarmType](#) - Determines which alarms are displayed in the alarms toolbar, based on type.
- [\[Alarm\]MaxOptimisedQueries](#) - Sets the number of queries that can be cached.
- [\[Alarm\]MaxQueryExecuteTime](#) - Creates a log entry if an internal SQL query takes longer than a specified amount of time.
- [\[Alarm\]MemoryWarningLimit](#) - Value in Mb, of the threshold of the alarm server memory.
- [\[Alarm\]MonitorConnectTimeout](#) - Defines the amount of time, in seconds, that the server will wait for a monitor connection to occur.
- [\[Alarm\]MonitorRequestTimeout](#) - Defines the amount of time, in seconds, that the server will wait for a response from the other server in the pair.
- [\[Alarm\]Period](#) - The period for rate of change alarms.
- [\[Alarm\]Primary](#) - Obsolete in version 7.0.
- [\[Alarm\]Process](#) - Obsolete in version 7.0.
- [\[Alarm\]QueryCPUUsage](#) - Defines the percentage of processor use you want allocate to query searches.
- [\[Alarm\]QueryRowLimit](#) - Defines the maximum number of rows that can be returned in the result set for a single query.
- [\[Alarm\]QueryTimeout](#) - Defines the amount of time (in seconds) that is permitted for query searches.
- [\[Alarm\]ReloadBackOffTime](#) - Back-off time configured to control the pace of the reload on an alarm server.
- [\[Alarm\]SavePeriod](#) - Obsolete in version 7.30.

- [\[Alarm\]SavePrimary](#) - The path to the primary save file.
- [\[Alarm\]SaveSecondary](#) - The path to the secondary save file.
- [\[Alarm\]SaveStyle](#) - Obsolete in version 7.30.
- [\[Alarm\]ScanTime](#) - Determines the rate at which an alarm server can receive updates from an I/O server.
- [\[Alarm\]Server](#) - Obsolete in version 7.0.
- [\[Alarm\]SetTimeOnAck](#) - Determines whether the {Time} field in the Alarm Display and Alarm Log Device contains the time when an alarm is triggered, or the time when the alarm is acknowledged.
- [\[Alarm\]SetTimeOnOff](#) - Determines whether the {Time} field in your Alarm Display and Alarm Log Device contains the time when an alarm is triggered, or the time when the alarm becomes inactive.
- [\[Alarm\]SOERowLimit](#) - Obsolete in version 2015.
- [\[Alarm\]ShowAllConfigured](#) - When set to true, it will show configured alarms on an alarm list using mode 4 (show configured alarms).
- [\[Alarm\]Sort](#) - Determines the order in which alarms are displayed in the alarm list.
- [\[Alarm\]SortMode](#) - Determines the sort order for displaying alarms.
- [\[Alarm\]Sound<n>](#) - Determines the wav file that is played when an alarm of priority <n> occurs.
- [\[Alarm\]Sound<n>Interval](#) - Determines the interval of time before the alarm sound for priority <n> alarms will be repeated.
- [\[Alarm\]SoundSilenceTimeoutOnAck](#) - Determines the timeout period in seconds for silencing alarm after the user acknowledges any alarms using the Tab\_Style templates.
- [\[Alarm\]StandbyCommandDelay](#) - Sets a delay that a standby alarm server will apply when executing alarm commands received from the main alarm server.
- [\[Alarm\]StartTimeout](#) - Sets the timeout period for loading each packet from the primary Alarms Server. This parameter has been reinstated for v2015 only.
- [\[Alarm\]StreamSize](#) - Defines the amount of data that is included in an event data file.
- [\[Alarm\]SummaryAutoRefreshMode](#) - Represents the default value for AlarmSetInfo type 15.
- [\[Alarm\]SummaryBufferSize](#) - Specifies the value to limit the size of the alarm summary buffer.
- [\[Alarm\]SummaryLength](#) - Obsolete in version 2015.
- [\[Alarm\]SummaryMode](#) - Specifies whether alarm summary events are logged to the Alarm Summary log device in the order they went ON, or in the order they went OFF.
- [\[Alarm\]SummarySort](#) - Specifies whether OnTime, OffTime or AckTime will be used to determine the order in which alarms will be listed on the alarm summary page.
- [\[Alarm\]SummarySortMode](#) - Specifies whether alarms listed on the alarm summary page will be displayed in ascending or descending order.
- [\[Alarm\]SummaryTimeout](#) - The length of time after which an alarm summary entry is considered timeout.
- [\[Alarm\]SummaryTimeoutTolerance](#) - The length of time from timeout after which an alarm summary entry is committed to Summary Device regardless the fact that Off Time is not set.
- [\[Alarm\]SummaryTolerance](#) - Obsolete is version 2015.
- [\[Alarm\]SummaryShutdownMode](#) - Specifies whether alarm summary entries that have not been logged are logged at shutdown.
- [\[Alarm\]SumStartupCopy](#) - Determines whether the alarm summary data is copied to the other Alarms Server on startup.

- [Alarm]SumStateFix - Obsolete is version 2015.
- [Alarm]SyncAllHistoricData - On multi-server systems, the Primary server and Standby server synchronize their data so that the Standby server contains an accurate, up to date backup of the Primary server's data.
- [Alarm]TimeDate - Specifies the time/date string format used by Cicode functions that support ONTIMEDATE or OFFTIMEDATE.
- [Alarm]TransferConnectTimeout - Defines the amount of time, in seconds, that the Primary server will wait for a connection to occur.
- [Alarm]TransferInterleave - Controls how often the data synchronization is triggered by the Primary to the Standby Server.
- [Alarm]TransferInterval - Defines the number of seconds between each attempt to update the data on the Standby server.
- [Alarm]TransferRequestTimeout - Defines the amount of time, in seconds, that the Primary server will wait for the Standby server to respond to a data transfer request.
- [Alarm]UseConfigLimits - Forces the Plant SCADA alarm server to use alarm property values from the runtime database.
- [Alarm]WebClientUpdatePollPeriod - Obsolete in version 2015.

## See Also

[Alarm Process Parameters](#)

[Parameter Categories](#)

### [Alarm]AckHold

Determines whether alarms that have become inactive (and have been acknowledged) remain in the OFF ACKNOWLEDGED alarm list. If you enable this parameter, the alarms remain in the list until the operator clears them.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

## Allowable Values:

- 0 - (Disable)
- 1 - (Enable)

## Default Value:

0

## See Also

[Alarm Parameters](#)

**[Alarm]Active**

The period (in milliseconds) at which the clients are advised of a change in alarms. For example, if [Alarm]Active is set to 1000, the Control Clients are advised every 1 second (if a change has occurred in the alarms held on the Alarms Server).

If you are communicating from a Plant SCADA client to the Alarm Server over a slow serial or Wide Area Network (WAN) link, you might want to increase the value of this parameter. This would use less network bandwidth and increase the overall performance of your system.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

**Allowable Values:**

0 to 10000 (milliseconds)

**Default Value:**

1000

**See Also**

[Alarm Parameters](#)

**[Alarm]AlarmDisable**

Enables/disables the processing of all alarms.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

**Allowable Values:**

- 0 - (Enable)
- 1 - (Disable)

**Default Value:**

0

**See Also**

[Alarm Parameters](#)

### [Alarm]AlarmListRequestTimeout

This parameter specifies the length of time (in seconds) that an alarm display will wait to receive data from all clusters. If data is still outstanding at end of this period, the alarm list will display the data that is available and a timeout notification will be generated as a hardware alarm. You can use the Cicode function AlarmGetInfo (type 16) to capture these notifications. Any data that arrives after this timeout period will be displayed as it is received.

When this parameter is set to zero (or a value less than zero), no timeout notification will be generated.

#### Allowable Values:

-1 to 2147483647

#### Default Value:

10 (seconds)

#### See Also

[Alarm Parameters](#)

### [Alarm]ArchiveAfter

This parameter specifies how long (in weeks) alarm summary and SOE data is kept before it is archived. When the specified period of time elapses, the data is archived to another medium (such as a portable hard drive).

If the archiving is enabled, the event messages can be archived on a variety of devices, including:

- Removable hard disk
- Removable flash memory
- DVD-R and DVD-RAM
- Network file share.

The volume where data is to be saved should be labeled. This includes any removable storage devices. If this is not the case, archiving will not succeed and a message will display on the SOE page.

For more information, refer to the topic [Configure the Archiving Parameters](#).

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

#### Allowable Values:

1 – the current value for [\[Alarm\]KeepOnlineFor](#)

If the value specified is out of this range, it will be automatically adjusted to equal the current value for [\[Alarm\]KeepOnlineFor](#). An out-of-range hardware alarm will be raised.

**Default Value:**

4 (weeks)

**Note:** Before upgrading to Plant SCADA 2015 from version 7.20 or 7.30, it is recommended that you extend the setting for this parameter so that it will capture all the data you would like to migrate, as any non-active data that is older than the time range specified may be lost when migration occurs. After upgrade is complete you can then archive your data and return this parameter to its normal setting.

**See Also**

[\[Alarm\]KeepOnlineFor](#)  
[Alarm Parameters](#)

**[Alarm]BackgroundColorMode**

Sets the background color mode of the alarm list and alarm banner on the Tab\_Style templates.

**Note:** This parameter is available only to projects based on the Tab\_Style templates.

**Allowable Values:**

- 0 - Dark blue color
- 1 - White color
- 2 - Transparent color (i.e. the actual color is determined by the background color of the page.)

**Default Value:**

0

**See Also**

[Alarm Parameters](#)

**[Alarm]BrowseRowLimit**

Defines the maximum number of rows per cluster that will be returned in an alarm browse.

Increasing the BrowseRowLimit may affect performance.

**Note:** When changing this parameter you need to restart the client.

**Allowable Values:**

0 to 4294967295.

**Default Value:**

200000

The performance of alarm browsing can be improved by changing the value of the parameter [Alarm]BrowseRowLimit. However, as Historian also uses this parameter to retrieve alarm data, changing the value of this parameter on Plant SCADA machines that Historian is connected to may result in a loss of data.

**See Also**

[Alarm Parameters](#)

**[Alarm]CacheSize**

This parameter defines the amount of memory (in megabytes) dedicated to the storage of event data.

For many systems, the default setting of 25 MB is appropriate. However, if your system is experiencing difficulties with server performance and the Cache Size needs to be adjusted, you can alter the Cache Size settings as required.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

**Allowable Values:**

0 to 400

**Default Value:**

25

**See Also**

[Alarm Parameters](#)

**[Alarm]DBLogDBServer**

Use this parameter to turn on logging for the alarm server database.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

**Allowable Values:**

An integer representing one of the following logging masks, or the sum total of integers representing multiple logging masks:

Logging Mask	Description	Integer
DBLIC	Server license	32
DBDRQ	Driver links	65536
WEBSVR	WebX server and threads	1048576
WEBLNK	WebX server links	2097152
WEBDAT	WebX server request raw data	4194304
WEBREQ	WebX server request content	8388608
WEBCK	WebX logon cookies	268435456

**Default Value:**

0 (logging is not enabled)

**Example**

To enable logging for driver links (set using "65536") and WebX server links (set using "2097152"), you would add the integers for each and set the parameter to the following:

[Alarm]DBLogDBServer=2162688

**See Also**

[\[Alarm\]DBLogServerCore](#)

[\[Alarm\]DBLogHistoric](#)

[Alarm Parameters](#)

**[Alarm]DBLogHistoric**

Use this parameter to define logging for Historian operations on the alarm server database.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

**Allowable Values:**

An integer representing one of the following logging masks, or the sum total of integers representing multiple

logging masks:

Logging Mask	Description	Integer
HISERR	Historian errors	1
HISDAT	Historian file operations	2
HISSBY	Historian standby transfer	4
HISREC	Historian record operations	8
HISWRT	Historian new records	16
HISENUM	Historian enumerate files	32
HISDET	Historian detailed file operations	64

## Default Value:

0 (logging is not enabled)

## Example

To enable logging for Historian errors (set using "1") and Historian file operations (set using "2"), you would add the integers for each and set the parameter to the following:

[Alarm]DBLogHistoric=3

## See Also

[\[Alarm\]DBLogDBServer](#)

[\[Alarm\]DBLogServerCore](#)

[Alarm Parameters](#)

## [Alarm]DBLogServerCore

Use this parameter to enable logging for the analysis of redundancy and synchronization issues.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

## Allowable Values:

An integer representing one of the following logging masks, or the sum total of integers representing multiple logging masks:

Logging Mask	Description	Integer
DBERR	Server errors	1
DBSTD	Server state management	2
DBTRN	Server transaction	4
DBDRV	Server driver control	8
DBADV	Server advise links	16
DBSVR	Server and threads	256
DBSLT	Standby links	512
DBSCX	Links	1024
DBDA	OPC DA links	2048
DBHDA	OPC HDA links	4096
DBAE	OPC AE links	8192
DBMAN	Manager links	16384
DBDIA	Diagnostic links	32768
DBLRW	Link read-write	524288
SBXFR	Standby transfer	16777216
SBDAT	Standby transfer data	33554432
SBPOL	Standby polls	67108864
SBPEX	Standby poll details	134217728
DBSLR	Server link reference counting	536870912

**Default Value:**

0 (logging is not enabled)

**Example**

To enable logging of server errors (set using "1") and server transactions (set using "4"), you would add the integers for each and set the parameter to the following:

[Alarm]DBLogServerCore=5

## See Also

[\[Alarm\]DBLogDBServer](#)  
[\[Alarm\]DBLogHistoric](#)  
[Alarm Parameters](#)

### [Alarm]DefDspFmt

Specifies an Alarm display format to use if the display fields are not defined via the [Format]FormatName parameter, and the Alarms Format field is blank for the Alarm Categories properties.

If all of your alarm display formats are the same, set your alarm display format once using this parameter (instead of repeating the format many times in the Alarm Categories database). You would then save memory and increase the performance of your project.

#### Allowable Values:

Any combination of [Alarm Display Fields](#).

#### Default Value:

{Time,12} {Tag,10} {Name,20} {Desc,32}

## See Also

[Alarm Parameters](#)

### [Alarm]DefSOEFmt

Specifies an SOE display format to use if the display fields are not defined via the [Format]FormatName parameter and the SOE Format field for the Alarm Categories properties is blank.

If the alarm SOE formats are the same, set your alarm SOE format once using this parameter (instead of repeating the format many times in the Alarm Categories database). You would then save memory and increase the performance of your project.

---

**Note:** You can also change the font and format of the system events on the SOE page using the SOE Format field for alarm category 0 (zero).

---

#### Allowable Values:

Any combination of [Alarm SOE Fields](#).

#### Default Value:

{DATE,16} {TIME,16} {TAG,10} {NAME,15} {MESSAGE,64} {STATE,16} {CLASSIFICATION,13} {USERNAME,16} {USERLOCATION,16}

## See Also

[Alarm Parameters](#)

### [Alarm]DefSumFmt

Specifies an Alarm Summary display format to use if the display fields are not defined via the [Format]FormatName parameter and the Summary Format field for the Alarm Categories properties is blank.

If the alarm summary formats are the same, set your alarm summary format once using this parameter (instead of repeating the format many times in the Alarm Categories database). You would then save memory and increase the performance of your project.

#### Allowable Values:

Any combination of [Alarm Summary Fields](#).

#### Default Value:

{Name,20} {OnTime,8} {OffTime,8} {DeltaTime,8} {Comment,22}

## See Also

[Alarm Parameters](#)

### [Alarm]DeltaTimeUpdate

Determines whether the DeltaTime (duration) field is set on non-OFF alarms, by calculating volatile duration between current time and the time when the alarm was activated. Thus providing you with a snapshot of the duration of alarms. This parameter is for display clients and does not affect server side or the alarm database.

---

**Note:** The deltatime field set by this parameter is not refreshed until a new event on an alarm is made, such as a new alarm, state change on an existing alarm or page navigation. For the deltatime to refresh automatically, you will need to implement auto-page refresh using Cicode functions.

---

#### Allowable Values

- 0 - (Disable)
- 1 - (Enable)

#### Default Value:

0

## See Also

[Alarm Parameters](#)

### [Alarm]DisableFilterOptimization

By default, Plant SCADA uses filter optimization for queries of historic alarm data. This turns the filter criteria for configured alarm fields into sets of alarm object IDs before applying them to the final queries.

This is effective when the number of object IDs required by a query is less than a threshold affected by the size of the project, the match rate of the filter and the density of the historic data kept on site. In general, the improvement is more prominent when filtering for a smaller subset of historic data. Its effectiveness drops as the filter becomes more inclusive. In unusual circumstances, the threshold may be crossed which can impact system performance.

This parameter allows you to turn off filter optimization to determine if it is having an impact on system performance.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

#### Allowable Values:

- 0 - (Enable)
- 1 - (Disable)

#### Default Value:

0

#### See Also

[Alarm Parameters](#)

### [Alarm]DisableSummary

Use this parameter to turn off the storage of summary events. If set to 1 (disabled), the following will occur:

- The Alarm Summary page will not work
- Cicode functions that read or modify the alarm summary will not work
- The summary log device will not work
- Alarm summary information may be lost during upgrade.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

#### Allowable Values:

- 0 - (Enable)
- 1 - (Disable)

**Default Value:**

0

**See Also**[Alarm Parameters](#)**[Alarm]DisableSOE**

Use this parameter to turn off the storage of events. If set to 1 (disabled), the following will occur:

- The Sequence of Events (SOE) page will not work
- Alarms will not display in the Process Analyst
- Alarms cannot be accessed through Historian.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

**Allowable Values:**

- 0 - (Enable)
- 1 - (Disable)

**Default Value:**

0

**See Also**[Alarm Parameters](#)**[Alarm]DisplayDisable**

Adjusts the behavior of disabled alarms so they are "suppressed" rather than completely disabled.

By default, when an alarm is disabled it is no longer processed by the system. The alarm is not logged, it is not included on alarm lists (except the disabled list), and does not change state with the I/O device variable.

By setting this parameter to 1, disabled alarms are suppressed. This means they are still processed, logged, and listed on the Sequence Of Events (SOE) and Alarm Summary pages. However, they are not displayed on the Active Alarms page.

While this parameter is set to 1, the User Name and User Location recorded at the time an alarm was disabled will be maintained, despite any subsequent state changes. These properties will not be overwritten by the System User account.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

---

[Configure Alarm Parameters for a Specific Cluster or Process.](#)

---

**Allowable Values:**

- 0 - (Offscan - disable completely)
- 1 - (Suppress - alarm will not display on the Active Alarms page)

**Default Value:**

0

---

**Note:** You need to decide whether you would like to suppress disabled alarms when designing your system. If you do decide to change the parameter to 1, you will have to keep the setting for the lifetime of your system (including after any upgrades to later versions of Plant SCADA). If you change the value of this parameter back to zero (0) after your system has been running, the integrity of the historical data will not be maintained correctly. This will cause issues when historical data is displayed on Process Analyst and the Alarm Summary page.

---

**See Also**

[Alarm Parameters](#)

**[Alarm]EnableErrorLogging**

Enables or disables the logging of operational errors to syslog.dat.

**Allowable Values:**

- 0 - (Disable)
- 1 - (Enable)

**Default Value:**

0

**See Also**

[Alarm Parameters](#)

**[Alarm]EnableSmartCustomFilters**

Enables/disables the caching of custom alarm queries to facilitate later use of the same filter.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

**Allowable Values:**

- 0 - (Disable)
- 1 - (Enable)

**Default Value:**

1

**See Also**[Alarm Parameters](#)**[Alarm]EnableStateLogging**

Enables or disables the logging of major alarm system state changes to syslog.dat.

**Allowable Values:**

- 0 - (Disable)
- 1 - (Enable)

**Default Value:**

0

**See Also**[Alarm Parameters](#)**[Alarm]EventFmt**

Specifies the format of the event queue string.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

**Allowable Values:**

Any combination of [Alarm Display Fields](#).

For example:

[Alarm]  
EventFmt={Date,16}{Time,16}{Name,79}{State,16}{Type,16}

## See Also

[Alarm Parameters](#)

### [Alarm]EventQue

Enables/disables the Alarm event queue on the Alarms Server. Only set this parameter if you are using the Alarm event queue, because it can increase CPU loading and reduce the performance of the Alarms Server.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

#### Allowable Values:

- 0 - (Disable)
- 1 - (Enable)

#### Default Value:

0

## See Also

[Alarm Parameters](#)

### [Alarm]ExtendedDate

Determines whether the short (dd/mm/yy) or extended (dd/mm/yyyy) date format is used when interpreting the {DATE,n} alarm format.

#### Allowable Values:

- 0 - (Use short date format)
- 1 - (Use extended date format)

#### Default Value:

1

## See Also

[Alarm Parameters](#)

**[Alarm]FlushTimeLimit**

The alarm server flushes Event Journal entries to disk periodically (every 60 seconds). This parameter determines the maximum duration (in milliseconds) of each flush.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

**Allowable Values:**

50-3000 ms

**Default Value:**

1000

**See Also**

[Alarm Parameters](#)

**[Alarm]FutureMessages**

Sequence of event records that have a time stamp with a date and time in the future can be stored historically. Typically, records with future dates and times relate forecasted data or data from a device that has different time settings, such as an outstation that has not had its clock set correctly.

You can use this parameter to define the lifetime in weeks for future messages. It defines the maximum amount of time in the future that is permitted for future messages. Any record that has a time stamp that is further in the future than the defined time is not stored historically.

For example, if the current date is October 1st 2021 and the FutureMessages is set to 1 week, any message that has a time stamp between October 2nd and October 7th will be stored historically. A message that has a time stamp beyond October 7th will be lost as the date of the future data is further in the future than the defined Future Message Life.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

**Allowable Values:**

0 - 4294967295

**Default Value:**

4

**See Also**

[Alarm Parameters](#)

**[Alarm]HardHoldTime**

The time (in seconds) to hold a hardware alarm in its alarm state before it is reset.

**Allowable Values:**

1 to 86400

**Default Value:**

30

**See Also**

[Alarm Parameters](#)

**[Alarm]HardwareDisable**

Enables/disables the processing of hardware alarms.

**Allowable Values:**

- 0 - (Enable)
- 1 - (Disable)

**Default Value:**

0

**See Also**

[Alarm Parameters](#)

**[Alarm]HeadingFont**

Determines the text font used for column headings displayed on the alarm templates.

---

**Note:** This parameter is only available for projects based on Tab\_Style templates.

---

**Allowable Values:**

Any font that exists in the project

**Default Value:**

None.

If no heading font is specified, Arial font of point size 10 will be used for displaying the alarm column headings.

**See Also**

[Alarm Parameters](#)

**[Alarm]HeartbeatTimeout**

The HeartbeatTimeout parameter defines how long a server will wait before closing a link that has been used for receiving heartbeat poll requests from its pair server, but is currently idle. By default, the HeartbeatTimeout is set to 300 seconds, which means if there are no heartbeat request polls received via an open link within 300 seconds, the server will close the link.

The other server in the pair normally sends periodic server status polls every 10 seconds. When a server receives a poll request, it replies by sending a response that indicates its status. This allows the servers to check that they can send communications to, and receive communications from, each server in the architecture. During normal conditions, the servers in the Primary – Standby pair will:

- Send heartbeat poll requests to the other server
- Receive heartbeat poll requests from the other server
- Respond to heartbeat poll requests by sending a status response.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

**Allowable Values:**

0 to 4294967295

**Default Value:**

300

**See Also**

[Alarm Parameters](#)

**[Alarm]HighResOff**

Sets the precision of time-stamped alarms. You can specify millisecond accuracy when alarms become active, or when alarms become active and when they become inactive. This parameter is used with time stamped alarms only.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see

---

Configure Alarm Parameters for a Specific Cluster or Process.

---

**Allowable Values:**

- 0 - (Use millisecond (ms) accuracy only when becoming active)
- 1 - (Use millisecond (ms) accuracy for active and inactive transitions)

**Default Value:**

0

**See Also**

[Alarm Parameters](#)

**[Alarm]Hres24HrDeadBand**

Determines how many seconds the PLC time can be ahead of Plant SCADA time before the date field (if used) will show an alarm as occurring on the previous day. This parameter is only relevant if you are using 24 hour time stamped alarms. This is determined by [\[Alarm\]HresType](#). (If [Alarm]HresType is in mode 5, 6, or 7, you are using 24 hour time stamped alarms.) If you are not using 24 hour time stamped alarms, this parameter has no effect.

**Allowable Values:**

0 to 65,535 (seconds)

**Default Value:**

60

**See Also**

[Alarm Parameters](#)

**[Alarm]HresTimerExprDelay**

Allows you to configure the timer delay for HRes timestamped alarms.

**Allowable Values:**

0 to 2147483647 (milliseconds)

**Default Value:**

The same value that is set for the parameter [\[Alarm\]ScanTime](#).

## See Also

[Alarm Parameters](#)

### [Alarm]HresType

Determines the time format used to record time stamped alarms. When a time stamped alarm is triggered, Plant SCADA reads the time of the alarm from the PLC or RTU and stores it in one of the 10 formats listed below.

#### Allowable Values:

Value	Description
0	12 hour BCD timer (stored as hexadecimal)
1	Continuous counter
2	12 hour BCD timer (stored as decimal)
3	12 hour Millisecond timer (stored as decimal)
4	Number of seconds since 1970
5	24 hour BCD timer (stored as hexadecimal)
6	24 hour BCD timer (stored as decimal)
7	24 hour millisecond timer (stored as decimal)
8	Negative BCD Offset (stored as hexadecimal). Set the high byte to represent the total number of hours of the history. The maximum history is 99 hours (four days) from midnight of the next day.
9	Negative BCD Offset (stored as decimal). Set the high byte to represent the total number of hours of the history. The maximum history is 4923 hours (178 days) from midnight of the next day.
10	Negative Millisecond Offset in milliseconds from the next Midnight. The maximum history is 439 days from midnight. This is stored as a positive decimal whole number in a ULONG data type. Therefore, a value of 1 would mean 1ms before midnight today, a value of 4,233,600,000 would equate to 49D, 0H, 0M, 0S, 0ms before midnight today.

**Note:** The BCD types have a resolution of 100th of a second (10ms).

**Default Value:**

0

**See Also**

[Alarm Parameters](#)

**[Alarm]HwAlarmFmt**

Defines the format of the data written into the hardware alarm queue and placed in the string field.

**Allowable Values:**

Any combination of [Alarm Display Fields](#).

**Default Value:**

"Time:{Time,12} Date:{Date,11} Desc:{Desc,40}"

**See Also**

[Alarm Parameters](#)

**[Alarm]HwAlarmQueMax**

Specifies the maximum length of the hardware alarm queue.

**Allowable Values:**

0 - 32767

**Default Value:**

0

**See Also**

[Alarm Parameters](#)

**[Alarm]HwExclude**

Specifies a list of hardware alarms that the hardware alarm system will not process.

**Allowable Values:**

A comma-separated list of hardware error codes

**Default Value:**

NONE

**See Also**

[Alarm Parameters](#)

**[Alarm]IsolationDetectInterval**

This parameter sets the interval between Internet Control Message Protocol (ICMP) packets to detect network isolation on alarm servers. If the ICMP check fails to get a response in a row as configured on [\[Alarm\]IsolationDetectRetryCount](#) parameter, the server will regard itself as isolated and switch itself to an isolated main state.

This parameter is only valid if [\[Alarm\]IsolationDetectIP1](#) and/or [\[Alarm\]IsolationDetectIP2](#) parameters have been configured.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

**Allowable Values:**

Length of time in milliseconds (500 - 120000)

**Default Value:**

5000 (5 seconds)

**See Also**

[Alarm Parameters](#)

**[Alarm]IsolationDetectIP1**

In a system that has redundant pairs of alarm servers, this parameter needs to be set to the IPAddress of a local piece of infrastructure (e.g. a router) on the network that can be pinged by the alarm server. This will be used by the alarm server to detect if it has become completely isolated from the network due to an outage. Setting this parameter will improve outcomes for alarm server redundancy as described below.

When a redundant pair of alarm servers are unable to connect to each other, both take on the primary responsibility for collecting and archiving alarm data. An alarm server taking on this responsibility is called the 'Main' alarm server. When the connection between the servers is reestablished, the alarm servers communicate with each other and decide which server will remain as 'Main', and which server will be 'Standby'.

Consider the situation whereby the alarm servers could not communicate to each other as one of the servers was completely isolated from the network. Being completely isolated from the network, the server would not have obtained any new alarm data during the outage. Whereas the peer alarm server may have still been obtaining data from the field during this time period. To insure all the data that was captured during the outage is maintained by the alarm system, the alarm server that was not isolated needs to stay 'Main' when the connection is recovered. Configuring this parameter correctly as described above will insure that the correct decision is made, as the alarm servers will know if they have been isolated from the network during the outage. Depending on your network topology, each alarm server in a redundant pair may require a different setting for this parameter, as they may have different infrastructure nearby their location in the network.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

## Allowable Values:

- Machine name
- IP address
  - If the Plant SCADA system is using an IPv4 network, use a standard IPv4 address format. For example, 192.1.2.34
  - If the Plant SCADA system is using an IPv6 network, use a standard IPv6 address format (or any acceptable abbreviation). For example, 2001:0db8:0000:0000:8a2e:0123:1234 or 2001:db8::8a2e:123:1234 (abbreviated).

## Default Value:

None

## See Also

- [\[Alarm\]IsolationDetectInterval](#)
- [\[Alarm\]IsolationDetectRetryCount](#)
- [Alarm Parameters](#)

## [Alarm]IsolationDetectIP2

In a system that has redundant pairs of alarm servers, this parameter needs to be set to the IP address of a local piece of infrastructure (e.g. a router) on the network that can be pinged by the alarm server. This will be used by the alarm server to detect if it has become completely isolated from the network due to an outage. Setting this parameter will improve outcomes for alarm server redundancy as described below.

When a redundant pair of alarm servers are unable to connect to each other, both take on the primary responsibility for collecting and archiving alarm data. An alarm server taking on this responsibility is called the 'Main' alarm server. When the connection between the servers is reestablished, the alarm servers communicate with each other and decide which server will remain as 'Main', and which server will be 'Standby'.

Consider the situation whereby the alarm servers could not communicate to each other as one of the servers was completely isolated from the network. Being completely isolated from the network, the server would not have obtained any new alarm data during the outage. Whereas the peer alarm server may have still been

obtaining data from the field during this time period. To insure all the data that was captured during the outage is maintained by the alarm system, the alarm server that was not isolated needs to stay 'Main' when the connection is recovered. Configuring this parameter correctly as described above will insure that the correct decision is made, as the alarm servers will know if they have been isolated from the network during the outage. Depending on your network topology, each alarm server in a redundant pair may require a different setting for this parameter, as they may have different infrastructure nearby their location in the network

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

## Allowable Values:

- Machine name
- IP address
  - If the Plant SCADA system is using an IPv4 network, use a standard IPv4 address format. For example, 192.1.2.34
  - If the Plant SCADA system is using an IPv6 network, use a standard IPv6 address format (or any acceptable abbreviation). For example, 2001:0db8:0000:0000:0000:8a2e:0123:1234 or 2001:db8::8a2e:123:1234 (abbreviated).

## Default Value:

None

## See Also

- [\[Alarm\]IsolationDetectInterval](#)  
[\[Alarm\]IsolationDetectRetryCount](#)  
[Alarm Parameters](#)

## [Alarm]IsolationDetectRetryCount

This parameter sets the Internet Control Message Protocol (ICMP) retry count to detect network isolation on alarm servers. If ICMP check fails more times than this number in a row, the server will regard itself as isolated from the main network and then switch itself to isolated main state. The interval between ICMP packets is configured on [Alarm]IsolationDetectInterval.

This parameter is only valid if [\[Alarm\]IsolationDetectIP1](#) and/or [\[Alarm\]IsolationDetectIP2](#) parameters have been configured.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

## Allowable Values:

0-5

**Default Value:**

2

**See Also**[Alarm Parameters](#)**[Alarm]KeepOnlineFor**

This parameter defines (in weeks) how long event data will remain on the Alarm Summary and SOE pages. When this time expires, the event data is deleted from disk and will no longer appear. This ensures disk space remains available to capture ongoing events.

For example, if this parameter is set to 1 (one week), events that are stored historically on Sunday will remain on the Alarm Summary and/or SOE pages for 7 days. They will then be deleted on the following Sunday. The event messages may be archived within the 7 days using the [\[Alarm\]ArchiveAfter](#) parameter.

During the period of time specified by this parameter, event data can change as new event messages are reported. However, if archiving is enabled, the event data will become read-only once it has been archived.

For more information, refer to the topic [Configure the Archiving Parameters](#) in the Plant SCADA documentation.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

**Allowable Values:**

1 - 5200

If an out-of-range value is specified, the default value (6) will be used and a hardware alarm will be raised.

**Default Value:**

6

**See Also**[Alarm Parameters](#)**[Alarm]LastAlarmCategories**

Determines which alarms are displayed on the alarms toolbar, based on category. For example, if you want the list of alarms to just display category 1 alarms, you would set this parameter to 1.

---

**Note:** This parameter is only available for projects based on the Tab\_Style templates.

**Allowable Values:**

0 - 16375

**Default Value:**

0 - all categories

**See Also**

[Alarm Parameters](#)

**[Alarm]LastAlarmDisplayMode**

Determines whether the last alarm is displayed by category or priority. This parameter only affects the DisplayLastAlarm() Cicode function.

For example, if you set this parameter to Display by Priority, pages based on templates which call DisplayLastAlarm() will be affected. Instead of displaying the current alarm, the latest of those alarms with the highest priority will be displayed.

This parameter also applies to the Tab\_Style templates.

**Allowable Values:**

- 0 - Display by category
- 1 - Display by priority

**Default Value:**

0

**See Also**

[Alarm Parameters](#)

**[Alarm]LastAlarmFmt**

Specifies the format of the last alarm as displayed on the included templates.

---

**Note:** This parameter is available to all Plant SCADA projects, however, its default value is different when implemented as part of a project based on the Tab\_Style templates.

---

**Allowable Values:**

Any combination of Alarm Display Fields

**Default Value:**

- **For any project:**  
{Tag,32} {Name,32} {Desc,32}

- **For Tab\_Style templates:**

```
{Time,12}{Date,12}^t{Name,15}^t{Desc,35}^t{State,9}
```

## See Also

[Alarm Parameters](#)

### [Alarm]LastAlarmPriorities

Determines which alarms are displayed on the alarms toolbar, based on priority. For example, if you want the list of alarms to just display priority 1 alarms, you would set this parameter to 1.

**Note:** This parameter is available only to projects based on Tab\_Style templates.

## Allowable Values:

1 - 255

## Default Value:

0 - all priorities

## See Also

[Alarm Parameters](#)

### [Alarm]LastAlarmType

Determines which alarms are displayed in the alarms toolbar, based on type.

**Note:** This parameter is available only to projects based on Tab\_Style templates.

## Allowable Values:

0 - 14

Non-hardware alarms

- 0 - Active alarms, i.e. Types 1 and 2
- 1 - Unacknowledged alarms, ON and OFF
- 2 - Acknowledged ON alarms
- 3 - Disabled alarms
- 4 - Every configured (non-hardware) alarm, i.e. Types 0 to 3, plus acknowledged OFF alarms.

Hardware alarms

- 5 - Active alarms, i.e. Types 6 and 7

- 6 - Unacknowledged alarms, ON and OFF
- 7 - Acknowledged ON alarms
- 8 - Disabled alarms
- 9 - Every configured alarm, i.e. Types 5 to 8

#### Alarm Summary

- 10 - Summary alarms

#### Alarm General

- 11 - ON alarms
- 12 - OFF alarms
- 13 - ON hardware alarms
- 14 - OFF hardware alarms

### **Default Value:**

0

### **See Also**

[Alarm Parameters](#)

### [Alarm]MaxOptimisedQueries

Used when [Alarm]EnableSmartCustomFilters=1 is set. This is the number of queries that the alarm server caches. When the 101st new query occurs, the oldest query information is discarded. It is unlikely more queries than this will need to be cached.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

### **Allowable Values:**

1 to 32000

### **Default Value:**

100

### **See Also**

[Alarm Parameters](#)

## [Alarm]MaxQueryExecuteTime

This parameter is used for commissioning or debugging purposes. It creates a log entry in the 'tracelog.Alarm.<ClusterName>.<ServerName>.dat' file if an internal SQL query takes longer to process than the specified amount of time (in milliseconds).

To disable this feature, set the parameter to -1.

### Allowable Values:

-1 (disabled), or 0 – 2147483647

### Default Value:

2000

### See Also

[Alarm Parameters](#)

## [Alarm]MemoryWarningLimit

Value in Mb, of the threshold of the alarm server memory.

If the threshold is reached a hardware alarm will be raised which will be visible on all clients connected to that cluster, whether they are connected to the standby or primary alarm servers.

To turn off the hardware alarm change the MemoryWarningLimit to 0.

The hardware alarm message is "Memory above safe limit".

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

### Allowable Values:

0 to 2147483647 (Mb)

### Default Value:

4000 (4Gb)

### See Also

[Alarm Parameters](#)

## [Alarm]MonitorConnectTimeout

Defines the amount of time, in seconds, that the server will wait for a monitor connection to occur. If there is no

monitor connection between the server you are configuring and the other server in the pair after the defined **MonitorConnectTimeout**, the other server will regard itself as isolated and will become Primary.

However, if you need the checks to detect lack of communication more quickly, you can reduce the timeout amount. But be aware that if the timeout is too short, a server may regard itself as isolated before it has had time to process a response to a request (or complete the connection establishment process).

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

## Allowable Values:

0 to 4294967295

## Default Value:

30

## See Also

[Alarm Parameters](#)

## [Alarm]MonitorRequestTimeout

Defines the amount of time, in seconds, that the server will wait for a response from the other server in the pair. If there is no response after the defined **MonitorRequestTimeout**, the other server will regard itself as isolated and will become Primary.

However, if you need the checks to detect faults more quickly, you can reduce the timeout amount. But be aware that if the timeout is too short, a server may regard itself as isolated before it has had time to process a response to a request (or complete the connection establishment process).

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

## Allowable Values:

0 to 4294967295

## Default Value:

30

## See Also

[Alarm Parameters](#)

## [Alarm]Period

The period used to determine when a Rate of Change alarm will be triggered for a variable tag. The Rate specified in Analog Alarm Properties is divided by this period to determine the 'maximum rate' at which the value of the variable tag can change. At each Scan Time (see [\[Alarm\]ScanTime](#)), Plant SCADA checks the value of the tag. If its rate of change is greater than the "maximum rate", a Rate of Change alarm is triggered.

For example, if your goal is to minimize the likelihood a tank from filling too quickly, you might configure a rate of change alarm, using an [\[Alarm\]Period](#) of 60 seconds, an [\[Alarm\]ScanTime](#) of 1 second, and a Rate (see above) of 300 liters. This means that the maximum allowable rate of change for the tank level is 5 l/sec (300 liters / 60 seconds). Plant SCADA calculates the actual rate of change at each ScanTime. i.e. every second, it checks the current level of the tank and compares it to the level recorded a second earlier. If the actual rate of change is, say, 8 l/sec, a Rate of Change Alarm is triggered immediately.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

### Allowable Values:

1 to 3600 (seconds)

### Default Value:

60

### See Also

[Alarm Parameters](#)

## [Alarm]QueryCPUUsage

Defines the percentage of processor use you want to allocate to query searches.

If programs and other processes are experiencing 'freezing' during historic searches, you should reduce the CPU Usage value.

If your searches are slow and are taking longer than expected to return results, you may achieve better results by increasing the CPU Usage value.

However, we recommend that you try to reduce the number of records involved in the search before you increase the CPU usage settings. Increasing the CPU Usage above 90% may have a detrimental effect on drivers and client connections, so we recommend that you should not increase CPU Usage above 90% unless specifically instructed to do so by Technical Support.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

### Allowable Values:

0 to 100

**Default Value:**

80

**See Also**

[Alarm Parameters](#)

**[Alarm]QueryRowLimit**

Defines the maximum number of rows that can be returned in the result set for a single query.

Increasing QueryRowLimit may affect performance.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

**Allowable Values:**

0 to 4294967295

**Default Value:**

1000000

**See Also**

[Alarm Parameters](#)

**[Alarm]QueryTimeout**

Defines the amount of time (in seconds) that is permitted for query searches.

If the search has not completed when the timeout expires, the query will be abandoned.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

**Allowable Values:**

0 to 4294967295

**Default Value:**

110

## See Also

[Alarm Parameters](#)

### [Alarm]ReloadBackOffTime

Adding or deleting alarm records too quickly may put pressure on the server, which could cause delays to the server response.

A back-off time can be configured to control the pace of the reload on an alarm server. The back-off time represents the minimal number of milliseconds for which the system will wait after adding or deleting an alarm record. The system will not attempt to reload another alarm record before this back-off time is out.

This configuration value is read at the beginning of each server reload.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

## Allowable Values:

0 ~ 2147483647 (milliseconds)

## Default Value:

1 (millisecond)

## See Also

[Alarm Parameters](#)

### [Alarm]SavePrimary

This parameter is used to import the alarm history from previous versions of Plant SCADA. This parameter no longer affects how alarm events are recorded on disk. The value for this parameter should not be changed when upgrading.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

## Allowable Values:

Any valid directory

## Default Value:

The directory specified in [\[CtEdit\]Run](#).

## See Also

[Alarm Parameters](#)

### [Alarm]SaveSecondary

This parameter is used to import the alarm history from previous versions of Plant SCADA. This parameter no longer affects how alarm events are recorded on disk. The value for this parameter should not be changed when upgrading.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

## Allowable Values:

Any valid directory

## Default Value:

" " (none)

## See Also

[Alarm Parameters](#)

### [Alarm]ScanTime

Determines the maximum rate at which the alarm server receives updates from the I/O server. A value of 500 (the default value) indicates that the minimum time between alarm transitions for a single alarm tag will be 500ms.

If you increase the scan time, it may reduce the load on the I/O server due to a reduction in the required subscription rate. However, this will depend on whether or not any other servers have subscribed to a device at a higher rate.

This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

## Allowable Values:

0 to 2147483647 (milliseconds)

## Default Value:

500

## See Also

[Alarm Parameters](#)

### [Alarm]SetTimeOnAck

Determines whether the {Time} field in your Alarm Display and Alarm Log Device contains the time when an alarm is triggered, or the time when the alarm is acknowledged.

When this parameter is enabled and the operator acknowledges an alarm, the {Time} field on the alarm page (and associated with the alarm record) is changed to the current time. If you want the {Time} field to remain at the time when the alarm was triggered, set this parameter to 0.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

#### Allowable Values:

- 0 - (Set the alarm time to the time when the alarm is activated)
- 1 - (Set the alarm time to the time when the alarm is acknowledged)

#### Default Value:

1

**Note:** This parameter has no effect on the Alarm Summary Display and Alarm Summary Device, which have separate {OnTime} and {AckTime} fields.

## See Also

[Alarm Parameters](#)

### [Alarm]SetTimeOnOff

Determines whether the {Time} field in your Alarm Display and Alarm Log Device contains the time when an alarm is triggered, or the time when the alarm becomes inactive.

When this parameter is enabled and the alarm becomes inactive, the {Time} field on the alarm page (and in the associated alarm record) is changed to the current time. If you want the {Time} field to remain at the time when the alarm was triggered, set this parameter to 0.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

#### Allowable Values:

- 0: (Set the alarm time to the time when the alarm is activated)
- 1: (Set the alarm time to the time when the alarm becomes inactive)

**Default Value:**

1

---

**Note:** This parameter has no effect on the Alarm Summary Display and Alarm Summary Device, which have a separate {OffTime} field.

---

**See Also**[Alarm Parameters](#)**[Alarm]ShowAllConfigured**

When set to true, it will show configured alarms on an alarm list using mode 4 (show configured alarms). Otherwise, it will only show alarms when the category setting ShowOnActive is true.

**Allowable Values:**

- 0 - Use ShowOnActive setting to determine if an Alarm is displayed when using the show configured alarms mode (mode 4).
- 1 - Show All Configured Alarms when using the show configured alarms mode (mode 4).

**Default Value:**

1

**See Also**[Alarm Parameters](#)**[Alarm]Sort**

Determines the order in which alarms are displayed in the active alarm list.

**Allowable Values:**

- 0 - lists the alarms in order of the most recent change of state
- 1 - lists the alarms in order of their occurrence (On Time).

**Default Value:**

0

If an alarm is acknowledged and OFF, it will no longer display on the active alarms list.

## See Also

[Alarm Parameters](#)

### [Alarm]SortMode

Determines the sort order for displaying alarms. The alarms are listed either in ascending order (the oldest alarms at the top) or descending order (the most recent alarms at the top).

#### Allowable Values:

- 0: (descending)
- 1: (ascending)

#### Default Value:

0

## See Also

[Alarm Parameters](#)

### [Alarm]Sound<n>

Determines the wav file that is used when an alarm sounds, based on the priority <n> of the alarm. For example, [Alarm]Sound1 identifies the .wav file that will be sounded when a priority 1 alarm is triggered.



#### LOSS OF CONTROL

- Do not use Plant SCADA's alarming system as the primary alert mechanism for safety-related alarms or notifications (that is, those classified as critical to process safety, the protection of the plant and equipment, or the protection of human life).
- Do not use Plant SCADA's audible alarm functionality as a replacement for safety-related warning systems critical to process safety, the protection of the plant and equipment or the protection of human life.
- Safety-related alarms and notifications should be independent and separate from the process control system. They should be implemented in the form of a stand-alone physical alarm system driving individual discrete alarm annunciators.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** This parameter is available only to projects based on Tab\_Style templates.

**Allowable Values:**

Any valid wav file name, including a full directory path.

**Note:** You can use Plant SCADA predefined path substitutions to locate a wav file; for example:  
Sound1=[RUN]:Alarm1.wav.

**Default Value:**

(no sound)

**See Also**

[Alarm Parameters](#)

**[Alarm]Sound<n>Interval**

Determines the time to wait before replaying the alarm sound for priority <n> alarms. For example, [Alarm]Sound1Interval identifies the interval of time between audible alerts when a priority 1 alarm is triggered.

**⚠ WARNING****LOSS OF CONTROL**

- Do not use Plant SCADA's alarming system as the primary alert mechanism for safety-related alarms or notifications (that is, those classified as critical to process safety, the protection of the plant and equipment, or the protection of human life).
- Do not use Plant SCADA's audible alarm functionality as a replacement for safety-related warning systems critical to process safety, the protection of the plant and equipment or the protection of human life.
- Safety-related alarms and notifications should be independent and separate from the process control system. They should be implemented in the form of a stand-alone physical alarm system driving individual discrete alarm annunciators.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** This parameter is available only to projects based on Tab\_Style templates.

**Allowable Values:**

250 - 30000 milliseconds

**Default Value:**

2000

## See Also

[Alarm Parameters](#)

### [Alarm]SoundSilenceTimeoutOnAck

Determines the timeout period in seconds for silencing an alarm after the user acknowledges any of the alarms using the Tab\_Style templates. After an alarm is silenced, if any higher priority alarms occur, alarm sound will playback again.

The user can silence alarm explicitly using the silence alarm icon / command on the templates. Alarm silence settings set this way take precedence over the alarm silence effect due to alarm acknowledgment.

**Note:** This parameter is available only to projects based on Tab\_Style templates.

## Allowable Values:

0 to 2147483647 seconds

## Default Value:

0 (silence alarm indefinitely until all alarms are acknowledged)

## See Also

[Alarm Parameters](#)

### [Alarm]StandbyCommandDelay

Sets a delay (in milliseconds) that a standby alarm server will apply when executing alarm commands received from the main alarm server.

You may need to set this parameter if you execute alarm commands within one alarm scan time (for example, you execute acknowledgment as soon as alarm changes state) and you observe the error "Alarm: Property write error: 582" in the syslog.dat file of the alarm server process.

This can occur because redundant alarm servers monitor alarm conditions independently of each other, and it takes up to one alarm scan time for alarm to reach the same state on both servers. If an alarm command is issued within this time period while the alarm states are out of synch, the standby alarm server will not be able to mirror the command.

Setting a command delay on both alarm servers will allow time for the alarm state to reach the same state before mirroring the command. The delay should be typically set to the alarm scan time specified by the alarm server [\[Alarm\]ScanTime](#) INI parameter.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

## Allowable Values:

0 to 2147483647

**Default Value:**

0

**See Also**[Alarm Parameters](#)**[Alarm]StartTimeout**

---

**Note:** This parameter has been reinstated for those doing an online upgrade from 7.20 to v2015 only.

Sets the timeout period for loading each packet from the primary Alarms Server. When a second Alarms Server starts, it tries to get the alarm current states from the primary server. This parameter determines how long to wait for a reply from the primary server. At the end of the timeout period, the Alarms Server either loads the saved data or reads the alarm data (from the I/O devices).

If the Alarms Server is timing out, you will see the message "Timeout from RndAlarm Server" in the Plant SCADA Kernel window. This timeout should only occur if you have a large number of alarms, typically greater than 10,000. If you see this message, increase this parameter until the message no longer displays at startup.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

**Allowable Values:**

10 to 3600 (seconds)

**Default Value:**

120

**See Also**[Alarm Parameters](#)**[Alarm]StreamSize**

This parameter defines the amount of data that is included in an event data file.

By default, the setting of Stream Size is 8 (256 Objects), which means that an event data file will contain an hour's worth of event data for a maximum of 256 database items. If the database contains more than 256 items, it will create extra files for the other database items (with each file containing the event data for a maximum of 256 items).

For many systems, the Stream Size setting of 256 Objects is suitable. However, if you are experiencing reduced performance when displaying and filtering event data, you may need to adjust the Stream Size.

There are several possible causes of slow performance relating to the event data files:

- Your system has a large database (over 100,000 items). The high number of database items can cause an

excessive number of files. In this situation, you may be able to improve performance by choosing a higher Stream Size.

- Your system is logging a high number of events. On especially active systems, the volume of events being logged can cause the individual event data files to become large in size (it is recommended that the size of the largest event data file not exceed 5MB). In this situation, you may be able to improve performance by choosing a lower Stream Size.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

## Allowable Values:

0 (no streams) to 14 (16384 objects)

## Default Value:

8 (256 objects)

## See Also

[Alarm Parameters](#)

### [Alarm]SummaryAutoRefreshMode

This represents the default value for AlarmSetInfo type 15, which is documented as follows:

15 – Auto-refresh mode, where mode is:

- -1 : always auto refresh
- 0 : auto refresh disabled on alarm acknowledge, enable, disable and add comment operations
- 1 : auto refresh page 1, disable on all other pages

## Default Value:

-1

## See Also

[Alarm Parameters](#)

### [Alarm]SummaryBufferSize

Specifies the value to limit the size of the alarm summary buffer.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

**Allowable Values:**

4096 to 2147483647

**Default Value:**

64000

**See Also**

[Alarm Parameters](#)

**[Alarm]SummaryMode**

Specifies how the alarm system commits timeout alarm summary entries into Summary Device.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

**Allowable Values:**

- **0 or 1:** An alarm summary entry is timeout if the Off time is set, and the length between its On time and the current time is greater than the value of the parameter [\[Alarm\]SummaryTimeout](#).

If an alarm summary entry is not timeout, the uncommitted entries of which On time are newer than the On time of this entry will not be committed to log.

In these two modes the alarm system usually logs alarm summary entries into Summary Device in the order they went ON. The order may break if a delayed alarm with an old timestamp is generated, or if an alarm summary entry is appended or split with an old timestamp.

- **2 or 3:** An alarm summary entry is timeout if the Off time is set, and the length between its Off time and the current time is greater than the value of the parameter [\[Alarm\]SummaryTimeout](#).

In these two modes the alarm system does not guarantee the order of On time or Off time of the entries which are committed into Summary Device, but instead attempts to commit timeout entries to Summary Device as soon as possible.

**Note:** Modes 0 and 2 no longer remove alarm summary entries from the summary page.

**Default Value:**

0

**See Also**

[Alarm Parameters](#)

## [Alarm]SummarySort

This parameter allows you to customize the order in which alarms will be displayed on the historical event pages. It is important to use this parameter if you have remote I/O devices in your system and you want to be certain that alarms are listed according to the exact time they occurred as opposed to the time they were reported by the alarms server.

You can choose to organize alarms according to the time each alarm was activated (OnTime), inactivated (OffTime), or the time the alarm was acknowledged by the operator (AckTime). Changing this parameter at runtime will not affect the current way alarms are displayed. It will only take affect at Plant SCADA start up and is usually used in conjunction with the [\[Alarm\]SummarySortMode](#) parameter.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

### Allowable Values:

- 0 None
- 1 OnTime
- 2 OffTime (obsolete in v2015)
- 3 AckTime (obsolete in v2015)

### Default Value:

0

### See Also

[Alarm Parameters](#)

## [Alarm]SummarySortMode

This parameter organizes alarms on the Historical events pages in ascending or descending order. Changing this parameter at runtime will not affect the current way alarms are displayed. It will only take affect at Plant SCADA start up and is usually used in conjunction with the [Alarm]SummarySort parameter.

### Allowable Values:

- 0 Descending order, most recent time at top
- 1 Ascending order, oldest time at top

### Default Value:

0

## See Also

[Alarm Parameters](#)

### [Alarm]SummaryTimeout

The length of time after which an alarm summary entry is considered timeout.

The alarm system searches for timeout entries in the order of On time from oldest to newest. The search may stop or skip entries depending on the value of parameter [\[Alarm\]SummaryMode](#).

The found timeout entries are then committed to the Summary Devices specified in the alarm categories of the entries.

Refer to the Cicode function AlarmSumCommit for committing process.

You can disable this parameter by setting it to -1.

This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

## Allowable Values:

0 to 20000 (minutes)

## Default Value:

60

## See Also

[Alarm Parameters](#)

### [Alarm]SummaryTimeoutTolerance

The length of time from timeout after which an alarm summary entry is committed to Summary Device regardless the fact that Off Time is not set.

The purpose of this parameter is to prevent a pending active alarm with ON condition infinitely blocking other alarm summary entries from being committed when [\[Alarm\]SummaryMode](#) parameter is set to 0 or 1.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

## Allowable Values:

0 to 20000 (minutes)

**Default Value:**

60

**See Also**[Alarm Parameters](#)**[Alarm]SummaryShutdownMode**

Specifies whether alarm summary entries (both complete and incomplete) that have not been logged are logged at shutdown. Alarm summary entries may be duplicated with redundant servers when one server is shutdown and restarted. This parameter should only be changed from the default if you are NOT using redundant alarm servers and/or alarm save files.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

**Allowable Values:**

- 0 - (Do not flush summary entries to devices)
- 1 - (Flush summary entries to devices)

**Default Value:**

0

**See Also**[Alarm Parameters](#)**[Alarm]SumStartupCopy**

Determines whether the alarm summary data is copied to the other Alarms Server on startup. Normally, the second Alarms Server gets all the alarm and summary data from the first server on startup. With this parameter, you can disable the sending of the summary data.

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

**Allowable Values:**

- 0 - (Don't send data)
- 1 - (Send data)

**Default Value:**

1

**See Also**[Alarm Parameters](#)**[Alarm]SyncAllHistoricData**

On multi-server systems, the Primary server and Standby server synchronize their data so that the Standby server contains the most recent date backup of the Primary server's data. During the synchronization process, the Primary server and Standby server synchronize data, including historic data. However, the amount of historic data that is synchronized is dependent on the SyncAllHistoricData parameter in the server configuration.

When SyncAllHistoricData is set to TRUE, the Primary server will update the Standby server with both the writable and non-writable historic data. This provides the Standby server with a copy of the historic data on the Primary server, but takes considerably longer for the synchronization process to complete.

When SyncAllHistoricData is set to FALSE, the Primary server will only update the Standby server with the historic data that is writable, that is the historic data that can still be modified on the system. This is the default setting and is appropriate for many systems as it means the synchronization of historic data is as fast and efficient as possible. There is no need to synchronize the historic data that is non-writable as it does not change and so will be the same on both the Primary and Standby servers.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

**Allowable Values:**

TRUE or FALSE

**Default Value:**

FALSE

**See Also**[Alarm Parameters](#)**[Alarm]TimeDate**

Specifies the time/date string format used by any Cicode functions that support an ONTIMEDATE or OFFTIMEDATE field.

These functions include AlmBrowseOpen, AlmBrowseGetField, AlmTagsOpen, and AlmSummaryOpen.

**Allowable Values:**

The recognizable elements in a time string format are:

- MM
- MONTH
- MON
- DY
- DAY
- HH12
- HH24
- HH
- MI
- SSSS
- SS
- DDD
- DD
- D
- AM
- PM
- A.M.
- P.M.
- Y,YYYY
- YYYY
- YYY
- YY
- Y
- AD
- BC
- A.D.
- B.C.
- TH

A maximum of 28 characters is supported for the formatted alarm field value.

**Default Value:**

HH:MI:SS

## See Also

[Alarm Parameters](#)

### [Alarm]TransferConnectTimeout

Defines the amount of time, in seconds, that the Primary server will wait for a connection to occur. This works in the same way as the **MonitorConnectTimeout** parameter, except that it applies to the server's attempts to establish a connection to the Standby Server for the purpose of transferring data (rather than establishing a connection for monitoring the other server).

Typically, the default setting is appropriate. You may need to increase the **TransferConnectTimeout** if the system incorrectly regards a connection as having been unsuccessful when it is still in the process of being established.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

## Allowable Values:

0 to 4294967295

## Default Value:

30

## See Also

[Alarm Parameters](#)

### [Alarm]TransferInterleave

Controls how often the data synchronization is triggered by the Primary to the Standby Server.

Normally the Primary Server synchronizes the Standby Server on every transfer interval determined by the [\[Alarm\]TransferInterval](#) parameter. The TransferInterleave parameter can be used to make the Primary synchronize the Standby on every *n* intervals.

For example, if the **TransferInterval** = 5 sec and the **TransferInterleave** = 2, then the Standby will be updated every 10 sec.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

## Allowable Values:

0 To 4294967295

**Default Value:**

1

**See Also**

[Alarm Parameters](#)

**[Alarm]TransferInterval**

Defines the number of seconds between each attempt to update the data on the Standby server. For many systems, you will not need to change the default setting of 5 seconds.

If you have a large system, the Primary server may be affected by a high load due to a high number of updates placing a large demand on the server's resources. If your system is experiencing slow performance, you may need to increase the TransferInterval so that the Primary server updates the Standby server less frequently. If the data on the Standby server is being updated too slowly, you may need to decrease the TransferInterval so that updates occur more frequently. Be aware that this may affect the load on the Primary server and so could lead to slow performance.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

**Allowable Values:**

0 to 4294967295

**Default Value:**

5

**See Also**

[Alarm Parameters](#)

**[Alarm]TransferRequestTimeout**

Defines the amount of time, in seconds, that the Primary server will wait for the Standby server to respond to a data transfer request. If there is no response after the defined request timeout, the Standby server is deemed to have become inoperable.

---

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

**Allowable Values:**

0 to 4294967295

**Default Value:**

30

**See Also**[Alarm Parameters](#)**[Alarm]UseConfigLimits**

Use this parameter to force the Plant SCADA alarm server to use alarm property values from the runtime database (RDB), rather than using the values which may be stored in the database file (DBF).

---

**Note:** If you are using [CtEdit]Run and [CtEdit]Copy to manage file server redundancy, you need to set [Alarm]UseConfigLimits=0 so that the alarm properties are not persisted to the RDB files. If not, the RDB files will be overwritten the next time they are accessed by the system.

To permanently update alarm threshold limits using the function AlarmSetThresholdRec(), set the parameter [Alarm]UseConfigLimits to 1.

As part of alarm server reload setup, it is recommended the Citect.ini is read to retrieve the latest value for the [Alarm]UseConfigLimit parameter. This will determine if runtime changes made to an existing alarm will overwrite the alarm property values in the DBF.

**Allowable Values:**

0 or 1.

When set to zero (0):

- The value is set in memory only. Writing back to the RDB and the DBF does not occur.
- On server reload, the alarm property values are retrieved from the stored DBF. Any changes in the RDB will be lost.
- The server will get the alarm property values from the RDB only in the following instances:
  - The database is not populated.
  - If it is a newly configured alarm (hence no values exist).

When set to 1:

- Values are set in memory, the RDB and the DBF.
- On server reload, the alarm property values are retrieved from the disk RDB. The DBF will be updated with the values from RDB.

---

**Note:** Setting this parameter to 1 can result in complicated scenarios if you are writing to alarm properties at runtime, deploying project updates, or running redundant alarm servers. It is recommended that you carefully consider the impact of this parameter in these scenarios within your system setup and test its impact on a runtime system before you implement it within a production facility.

If the value is set to 1, it will affect the property write performance as it writes to RDB files.

## Default Value:

0

**Note:** This parameter can also be applied to a specific cluster or alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

## See Also

[Alarm Parameters](#)

[\[CtEdit\]Copy](#)

[\[CtEdit\]Run](#)

## Alarm Process Parameters

Plant SCADA allows you to specify some alarm parameters for an alarm server process (see [Configure Alarm Parameters for a Specific Cluster or Process](#)).

The following parameters need to be configured for a specific process.

**Note:** The <server name> is the name of the alarm server as specified in the **Topology** activity.

- [\[Alarm.<ClusterName>.<ServerName>\]ClientConnectTimeout](#) - Defines the amount of time in milliseconds, in which the client can attempt to make a connection to the server.
- [\[Alarm.<ClusterName>.<ServerName>\]ClientDisconnectTimeout](#) - Defines the amount of time in milliseconds, in which the client can attempt to terminate a connection to the server.
- [\[Alarm.<ClusterName>.<ServerName>\]ClientRequestTimeout](#) - Defines the amount of time in milliseconds, in which the client can request data from the server.
- [\[Alarm.<ClusterName>.<ServerName>\]ClientUpdatePollPeriod](#) - Obsolete in version 2015.
- [\[Alarm.<ClusterName>.<ServerName>\]Clusters](#) - Sets which clusters this Alarm Server process connects to at startup.
- [\[Alarm.<ClusterName>.<ServerName>\]CPU](#) - Sets the CPU that the Alarm Server process is assigned to.
- [\[Alarm.<ClusterName>.<ServerName>\]Events](#) - The list of events that this Plant SCADA Alarms Server process enables.
- [\[Alarm.<ClusterName>.<ServerName>\]ShutdownCode](#) - Determines the Cicode function to run when Plant SCADA Alarm Server process shuts down.
- [\[Alarm.<ClusterName>.<ServerName>\]StartupCode](#) - Determines the Cicode function to run when Plant SCADA Alarm Server process starts up.

You can also used the following parameters on any computer that connects to a specific server process:

- [\[Alarm.<ClusterName>.<ServerName>\]DisableConnection](#) - used to specify if a client should not connect to a server.
- [\[Alarm.<ClusterName>.<ServerName>\]Priority](#) - used to specify the client priority for the server connection.

## See Also

[Alarm Parameters](#)

### [Alarm.<ClusterName>.<ServerName>]ClientConnectTimeout

Defines the amount of time, in milliseconds, in which the client can attempt to make a connection. If the client is unable to establish a connection to the server within this time, the connection attempt was unsuccessful.

**Note:** This parameter needs to be applied to a specific alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

## Allowable Values:

0 to 4294967295

## Default Value:

30000

## See Also

[Alarm Process Parameters](#)

[Alarm Parameters](#)

### [Alarm.<ClusterName>.<ServerName>]ClientDisconnectTimeout

Defines the amount of time, in milliseconds, in which the client can attempt to terminate a connection to the server. If a disconnection has not occurred by the end of this time, the attempt has been unsuccessful. The Disconnect Timeout is used when the network connection can be detected, but the client cannot disconnect from the system.

**Note:** This parameter needs to be applied to a specific alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

## Allowable Values:

0 to 4294967295

## Default Value:

30000

## See Also

[Alarm Process Parameters](#)

## Alarm Parameters

### [Alarm.<ClusterName>.<ServerName>]ClientRequestTimeout

Defines the amount of time, in milliseconds, in which the client can request data from a server. If the server has not responded by the end of this time, the request has been unsuccessful.

**Note:** This parameter needs to be applied to a specific alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

#### Allowable Values:

0 to 4294967295

#### Default Value:

120000

#### See Also

[Alarm Process Parameters](#)

[Alarm Parameters](#)

### [Alarm.<ClusterName>.<ServerName>]Clusters

Selects which clusters this Alarms Server connects to at startup.

**Note:** This parameter needs to be applied to a specific alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#)

#### Allowable Values:

A comma separated list of configured cluster names.

#### Default Value:

If this parameter is blank, the alarms server will connect to every cluster defined for the project.

#### See Also

[Alarm Process Parameters](#)

[Alarm Parameters](#)

### [Alarm.<ClusterName>.<ServerName>]CPU

This parameter is by Plant SCADA Runtime Manager to determine which CPU(s) the alarm server process is

assigned to. The CPU parameter is configurable for every Plant SCADA process via the **CPU Setup** page in the Computer Setup Wizard (see [CPU Configuration](#)).

CPU numbers are specified in a comma separated list. CPU numbers need to be valid and a subset of the available CPUs on a computer. For example, if there are eight CPUs on a computer, a valid CPU range is 0 to 7.

If this parameter is modified while the Plant SCADA Runtime Manager is running, the changes will not take effect until every instance of Plant SCADA is restarted.

**Note:** This parameter needs to be applied to a specific alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

## Allowable Values:

A comma separated list in the range 0 to 31 (inclusive) where 0 is the first CPU (for example, "0, 1, 2" or "3, 4, 5").

If no value is entered, the process will run on all available CPUs.

**Note:** When a process is set to run on all available CPUs, be aware that performance issues may still occur when your CPU usage is less than 100%. You can detect these issues when your CPU usage for a process is fixed at 100 divided by the number of processors.

---

## Default Value:

No value (the process will run on all CPUs).

**Note:** You cannot manually specify CPU assignments on a computer with more than 32 CPUs. Under these circumstances, the Setup Wizard will automatically set this parameter to no value (all CPUs). Any changes you make to this parameter will be overwritten.

---

### See Also

[Alarm Process Parameters](#)

[\[Client\]CPU](#)

[\[IOServer.<ClusterName>.<ServerName>\]CPU](#)

[\[Report.<ClusterName>.<ServerName>\]CPU](#)

[\[Trend.<ClusterName>.<ServerName>\]CPU](#)

## [Alarm.<ClusterName>.<ServerName>]DisableConnection

This setting is used to specify whether or not the client sub system should connect to the given server. If the Citect.ini on a particular computer has [Type.ClusterName.ServerName]DisableConnection = 1 specified for a particular server, then the client sub system will establish a non-redundant connection to the other server in the redundant pair. If that connection becomes inoperative, the client sub system will not be redirected and will have no connection until the server is restored.

To define connection priority of a particular server use [\[Alarm.<ClusterName>.<ServerName>\]Priority](#).

**Note:** This parameter needs to be applied to a specific alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

**Allowable Values:**

- 0 - Allow the client to connect to the server
- 1 - Stop the client connecting to the server

**Default Value:**

0

**See Also**[Alarm Process Parameters](#)[Alarm Parameters](#)**[Alarm.<ClusterName>.<ServerName>]Events**

The list of events that this Plant SCADA Alarms Server will enable.

This parameter is modified by the Computer Setup Wizard. To set this parameter, we recommend you enable the required events in the "Events Setup" page of the Computer Setup Wizard (instead of entering the events in the Setup Editor).

---

**Note:** This parameter needs to be applied to a specific alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

**Allowable Values:**

List of events (comma separated).

**Default Value:**

NONE

---

**Note:** When Plant SCADA is running in single process mode only the events assigned to [\[Client\]Events](#) will be enabled. Events specified for particular Plant SCADA components are ignored.

---

**See Also**[Alarm Process Parameters](#)[Alarm Parameters](#)**[Alarm.<ClusterName>.<ServerName>]Priority**

This setting is used to specify the client sub system's priority for the connection to this specific server. The client sub system will connect to, or switch over to, any available server that has the highest priority (lowest number). If the priorities for multiple servers are set to the same value, the client sub system will only switch on a loss of the currently active connection.

This is the default operation to provide the same behavior as in versions of the product prior to v7.20.

To disable (not allow) the client to connect to the server use  
[\[Alarm.<ClusterName>.<ServerName>\]DisableConnection](#).

**Note:** This parameter needs to be applied to a specific alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

## Allowable Values:

1 to 32767.

## Default Value:

1

## See Also

[Alarm Process Parameters](#)

[Alarm Parameters](#)

## [Alarm.<ClusterName>.<ServerName>]ShutdownCode

Can be used to specify a Cicode function that runs at shutdown of the Plant SCADA Alarm Server component. Plant SCADA suspends the shutdown process until the function has finished running. The function needs to complete execution within the time specified by the [Code]ShutdownTime parameter, otherwise Plant SCADA will terminate it.

This parameter can be set using the Computer Setup Wizard.

**Note:** This parameter can also be applied to a specific cluster or all alarm server processes. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

## Allowable Values:

Any built-in or user-written Cicode function, 1-31 alpha-numeric characters.

## Default Value:

NONE

**Note:** When Plant SCADA is running in single process mode only the Cicode function assigned to [\[Client\]ShutdownCode](#) will be executed at shutdown. Cicode functions specified for particular Plant SCADA components are ignored.

---

## See Also

[Alarm Process Parameters](#)

[Alarm Parameters](#)

**[Alarm.<ClusterName>.<ServerName>]StartupCode**

Determines the Cicode function to run when the Plant SCADA Alarm Server component starts up.

You can only pass constant data to the startup function call. You cannot pass variables or other functions. For example, MyFunc(1, "str") is valid, but MyFunc(PLCTAG, "str") or MyFunc(YourFunc(), "str") is not supported.

---

**Note:** This parameter needs to be applied to a specific alarm server process. For more information, see [Configure Alarm Parameters for a Specific Cluster or Process](#).

---

This parameter is modified by the Computer Setup Wizard.

**Allowable Values:**

Any built-in or user-written Cicode function, 1-31 alpha-numeric characters.

**Default Value:**

NONE

---

**Note:** When Plant SCADA is running in single process mode only the Cicode function assigned to [\[Client\]StartupCode](#) will be executed at startup. Cicode functions specified for particular Plant SCADA components are ignored.

---

**See Also**

[Alarm Process Parameters](#)

[Alarm Parameters](#)

**AlarmFilterRule Parameters**

Using the AlarmFilterRules parameters you can add new rules or overwrite the pre-defined rules in the alarm filter form. You can define common rules that will appear in all filter forms regardless of alarm display type (for example, Active, SOE, Summary or Disabled), or define rules for a specific alarm display type.

For examples refer to the topic *Using the AlarmFilterRules Parameters* in the Alarm Filter Help (available at runtime).

The Citect.ini file contains the following alarm parameters:

- **[AlarmFilterRuleList.Active]Rule<n>** - Defines the name of the common rules to appear on the Simple Rule drop down list of the alarm active filter form.
- **[AlarmFilterRuleList.Disabled]Rule<n>** - Defines the name of the common rules to appear on the Simple Rule drop down list of the disabled alarm filter form.
- **[AlarmFilterRuleList]Rule<n>** - Defines the name of the common rules to appear on the Simple Rule drop down list of the alarm filter form .
- **[AlarmFilterRuleList.SOE]Rule<n>** - Defines the name of the common rules to appear on the Simple Rule drop down list of the alarm soe filter form.
- **[AlarmFilterRuleList.Summary]Rule<n>** - Defines the name of the common rules to appear on the Simple Rule drop down list of the alarm summary filter form.

- [\[AlarmFilterRules\]<RuleName>](#) - Defines the filter expression represented by the rule name.

## See Also

[Alarm Parameters](#)  
[Alarm Process Parameters](#)  
[Parameter Categories](#)

### [AlarmFilterRuleList.Active]Rule<n>

Defines the name of rules to appear on the Simple Rule drop down list of the active alarm filter form. Rules need to be defined in consecutive order, for example, Rule1, Rule2, Rule3. The content of each rule is defined under parameter [\[AlarmFilterRules\]<RuleName>](#).

For examples refer to the topic *Using the AlarmFilterRules Parameters* in the Alarm Filter Help (available at runtime).

#### Allowable Values:

Any valid rule name

#### Default Value:

Rule1 = Last 24 hours  
Rule2 = Last 7 days

## See Also

[AlarmFilterRule Parameters](#)

### [AlarmFilterRuleList.Disabled]Rule<n>

Defines the name of rules to appear on the Simple Rule drop down list of disabled alarm filter form. Rules need to be defined in consecutive order, for example, Rule1, Rule2, Rule3. The content of each rule is defined under parameter [\[AlarmFilterRules\]<RuleName>](#).

For examples refer to the topic *Using the AlarmFilterRules Parameters* in the Alarm Filter Help (available at runtime).

#### Allowable Values:

Any valid rule name

#### Default Value:

Rule1 = Last 24 hours  
Rule2 = Last 7 days

## See Also

[AlarmFilterRule Parameters](#)

### [AlarmFilterRuleList]Rule<n>

Defines the name of the common rules to appear in the Simple Rule dropdown list of the alarm filter form available at runtime. The common rules defined here are listed at the bottom of the dropdown list, after the rules specific to the filter form.

Rules need to be defined in consecutive order, (for example, Rule1, Rule2, Rule3, ... and so on). The content of each rule is defined under parameter [AlarmFilterRules]<RuleName>.

For examples refer to the topic *Using the AlarmFilterRules Parameters* in the Alarm Filter Help (available at runtime).

## Allowable Values:

Any valid rule name

## Default Value:

None

## See Also

[AlarmFilterRule Parameters](#)

### [AlarmFilterRuleList.SOE]Rule<n>

Defines the name of rules to appear on the Simple Rule drop down list of alarm soe filter form. Rules need to be defined in consecutive order (for example, Rule1, Rule2, Rule3). The content of each rule is defined under parameter [AlarmFilterRules]<RuleName>.

For examples refer to the topic *Using the AlarmFilterRules Parameters* in the Alarm Filter Help (available at runtime).

## Allowable Values:

Any valid rule name

## Default Value:

Rule1 = No system events  
Rule2 = Last 24 hours  
Rule3 = Last 7 days

## See Also

[AlarmFilterRule Parameters](#)

### [AlarmFilterRuleList.Summary]Rule<n>

Defines the name of rules to appear on the Simple Rule drop down list of alarm summary filter form. Rules need to be defined in consecutive order, for example, Rule1, Rule2, Rule3. The content of each rule is defined under parameter [AlarmFilterRules]<RuleName>.

For examples refer to the topic *Using the AlarmFilterRules Parameters* in the Alarm Filter Help (available at runtime).

## Allowable Values:

Any valid rule name

## Default Value:

Rule1 = Last 24 hours - OnTime

Rule2 = Last 7 days - OnTime

## See Also

[AlarmFilterRule Parameters](#)

### [AlarmFilterRules]<RuleName>

Defines the filter expression represented by the rule name. The rule name can be used under parameter group [AlarmFilterRuleList[.<FilterForm>]] so that the rule names are selectable on the respective filter form.

For examples refer to the topic *Using the AlarmFilterRules Parameters* in the Alarm Filter Help (available at runtime).

## Allowable Values:

Any valid filter expression.

Multiple filter expression can be defined, with each separated by semi-colon (;) character. A special keyword, #Now[<offset in seconds>] can be used to represent the current time with an optional offset.

## Default Value:

None

---

**Note:** The following rule names are pre-defined as part of the Library\_Controls project:

No system events = StateNumeric <> 0

Last 24 hours = LocalTimeDate >= #Now[-86400]

Last 7 days = LocalTimeDate >= #Now[-604800]

---

---

Last 24 hours - OnTime = OnTime >= #Now[-86400]

Last 7 days - OnTime = OnTime >= #Now[-604800]

---

## See Also

[AlarmFilterRule Parameters](#)

## Animator Parameters

The Citect.ini file contains the following animator parameters:

- [\[Animator\]BoldWeight](#) - The weight of a text font when bold is specified - the larger the number, the heavier is the font.
- [\[Animator\]ButtonCancelMode](#) - The behavior of push button command cancellation.
- [\[Animator\]CacheSize](#) - The maximum number of symbols that can be held in the cache of a client.
- [\[Animator\]ConfigureMouseCommand](#) - Determines whether you can configure command keys for the mouse buttons (left and right).
- [\[Animator\]EatMouseClick](#) - Determines whether Plant SCADA ignores (eats) the mouse click (left button down or up, or right button down or up), if commands are defined for those keys.
- [\[Animator\]EatMouseFocus](#) - Determines whether Plant SCADA ignores (eats) the left button down message when you change the focus of a window with the mouse.
- [\[Animator\]EnableWebContent](#) - Determines if Web Content objects are enabled at runtime.
- [\[Animator\]FastDisplay](#) - Determines whether Plant SCADA displays fast runtime graphics pages.
- [\[Animator\]FlashTime](#) - The frequency with which colors are flashed.
- [\[Animator\]FormFontWidth](#) - Determines how the text field width for forms with text fields (Check Boxes, Radio Buttons, Prompts, etc) is displayed.
- [\[Animator\]FullScreen](#) - Determines whether pages will be displayed in fullscreen or restored state.
- [\[Animator\]InvalidRegionExpansion](#) - Objects which are less than this distance apart will be re-drawn as a group if they change simultaneously at runtime.
- [\[Animator\]InvalidRegionQueueDepth](#) - The maximum number of non-overlapping objects that will be individually re-drawn when a large number of objects change simultaneously at runtime.
- [\[Animator\]LibraryError](#) - Determines whether a dialog box displays for any symbol that cannot be found.
- [\[Animator\]LibraryLength](#) - The number of objects that will be held in the library cache.
- [\[Animator\]LibraryTime](#) - The period that imported symbols remain in the library cache.
- [\[Animator\]MaxAn](#) - The maximum number of animation points (ANs) on a graphics page.
- [\[Animator\]MaximizedWindow](#) - Defines whether the runtime window is run in maximized mode on start-up.
- [\[Animator\]PrintXScale](#) - The X scaling factor applied when the WinPrint() function is called to print the screen (on the printer).
- [\[Animator\]PrintYScale](#) - The Y scaling factor applied when the WinPrint() function is called to print the screen (on the printer).
- [\[Animator\]TabFactor](#) - Controls the distance between tab stops on the graphics display.
- [\[Animator\]TemplateUpdate](#) - Enables the update of linked graphics pages to reflect template modifications.

- [\[Animator\]TooltipFont](#) - Specifies the font to be used for Plant SCADA's Tooltips.
- [\[Animator\]UseCTGIfNewer](#) - Controls the loading of CTG and CTF files when opening a page.

## See Also

[Parameter Categories](#)

### [Animator]BoldWeight

The weight of a text font when bold is specified - the larger the number, the heavier is the font.

---

**Note:** Although this parameter is still valid, it is only used for V3.xx and V4.xx animations.

---

#### Allowable Values:

0 to 1000

#### Default Value:

600

## See Also

[Animator Parameters](#)

### [Animator]ButtonCancelMode

Specifies the behavior of push button command cancellation.

When set to 0 - On button down, the button down command is executed, and the button up command is executed only if the mouse button is released while the mouse cursor is on top of the button.

When set to 1 - The behavior of command cancellation is determined by the configuration of the button. If you have configured a down command then the up command cannot be cancelled. If you have not configured a down command the up command can be cancelled.

When set to 2 - Command cancellation is disabled. Irrespective of how the button was configured, the up command cannot be cancelled.

---

**Note:** Although this parameter is still valid, it is only used for V3.xx and V4.xx animations.

---

#### Allowable Values:

0, 1, 2

#### Default Value:

1

## See Also

[Animator Parameters](#)

### [Animator]CacheSize

The maximum number of symbols that can be held in the cache of a client. If you increase the cache size, you use Windows resources and could cause problems with other software packages. Note that all symbols are actually cached in memory-this cache is a secondary cache.

**Note:** Although this parameter is still valid, it is only used for V3.xx and V4.xx animations.

## Allowable Values:

10 to 200

## Default Value:

50

## See Also

[Animator Parameters](#)

### [Animator]ConfigureMouseCommand

Determines whether you can configure command keys for the mouse buttons (left and right).

## Allowable Values:

- 0 - (Disable mouse button configuration)
- 1 - (Enable mouse button configuration)

## Default Value:

1

## See Also

[Animator Parameters](#)

### [Animator]EatMouseClicked

Determines whether Plant SCADA ignores (eats) the mouse click (left button down or up, or right button down or up), if commands are defined for those keys.

**Allowable Values:**

- 0 - (Do not ignore mouse)
- 1 - (Ignore mouse)

**Default Value:**

0

**See Also**[Animator Parameters](#)**[Animator]EatMouseFocus**

Determines whether Plant SCADA ignores (eats) the left button down message when you change the focus of a window with the mouse. Normally, if you have two windows on the screen and you click with the mouse from one to the other (to change the focus), any keyboard commands using the left mouse button execute. In addition, if you click on a button, the button command executes.

If you set this option to 1, the left mouse button click is ignored when you change the window focus.

**Allowable Values:**

- 0 - (Do not ignore mouse)
- 1 - (Ignore mouse)

**Default Value:**

0

**See Also**[Animator Parameters](#)**[Animator]EnableWebContent**

Determines if [Web Content](#) objects are enabled at runtime.

You can use this parameter to avoid the creation of multiple Chromium subprocesses, which can cause unnecessary instances of the Plant SCADA runtime process.

If a page containing a Web Content control is displayed while this parameter is set to 0 (disabled), the control will not display and a hardware alarm will be raised.

**Allowable Values:**

- 0 - Web Content objects are disabled
- 1 - Web Content objects are enabled

**Default Value:**

1

**See Also**

[Animator Parameters](#)

**[Animator]FastDisplay**

Determines whether Plant SCADA displays fast runtime graphics pages. If you enable this parameter, the runtime system searches for a fast display image when you display a page. If a fast display image is not found, the runtime system searches for a normal display image.

If you disable this parameter, the runtime system does not search for a fast display image and Graphics Builder will not save a fast display image. This option may also be set using the Options dialog in Graphics Builder (change the "Fast runtime display" option). If you want to save disk space, you can delete all files with the extension CTF in your project when this parameter is disabled.

**Allowable Values:**

- 0 - (Do not display)
- 1 - (Display fast runtime pages)

**Default Value:**

1

**See Also**

[Animator Parameters](#)

**[Animator]FlashTime**

The frequency with which colors are flashed.

**Allowable Values:**

100 to 3000 (milliseconds)

**Default Value:**

500

**See Also**

[Animator Parameters](#)

**[Animator]FormFontWidth**

Determines how the text field width for forms with text fields (Check Boxes, Radio Buttons, Prompts, etc) is displayed. You should only use this parameter if you are using Plant SCADA v2.00 (or earlier) to size the text fields generated with the Cicode form functions.

**Allowable Values:**

- 0 - (String length in characters)
- 1 - (String length in pixels)

**Default Value:**

1

**See Also**

[Animator Parameters](#)

**[Animator]FullScreen**

Determines whether pages will be displayed in fullscreen or restored state.

A page in fullscreen state takes up the entire display area (assuming this does not affect its aspect ratio), and it cannot be resized. Also, a fullscreen page will display without a title bar when [\[Page\]DynamicSizing](#) is enabled or when the page resolution is the same as the screen resolution. Resizing pages can result in lower picture quality. If this is unacceptable, you should re-design the page using the desired resolution.

A page in restored state can be resized by clicking and dragging on the window frame. If you resize a page, subsequent pages will, by default, be resized using the same scale. However, if the [\[Page\]DynamicSizing](#) parameter is set to FALSE, resizing the window will not change the page size.

---

**Note:** This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

**Allowable Values:**

- 0 - (Window is not started in fullscreen mode)
- 1 - (Runtime window will be started fullscreen mode without a title bar)
- 2 – (Runtime window will be started in fullscreen mode with a title bar.)

**Default Value:**

0

**See Also**

[Animator Parameters](#)

**[Animator]InvalidRegionExpansion**

Objects which are less than this distance apart will be re-drawn as a group if they change simultaneously at runtime. This parameter is useful for improving runtime page display times. For example, you could set this parameter to 64 pixels, and draw a group of 5 individual lights (driven by digital tags) with 50 pixels between each. During runtime, if neighboring lights (closer than 64 pixels) changed simultaneously, they would be re-drawn together, rather than individually.

**Allowable Values:**

0 to 32000

**Default Value:**

64

**See Also**

[Animator Parameters](#)

**[Animator]InvalidRegionQueueDepth**

The maximum number of non-overlapping objects (or objects that are further apart than the distance specified by the [\[Animator\]InvalidRegionExpansion](#) parameter) that will be individually re-drawn simultaneously at runtime. For example, if you set this parameter to 30, and you have a graphics page that contains 200 non-overlapping lights(driven by digital tags), and the associated tags change state at once, only 30 of the lights would be re-drawn individually; the remaining lights would be re-drawn (as a group) in one step.

**Allowable Values:**

1 to 32000

**Default Value:**

10

## See Also

[Animator Parameters](#)

### [Animator]LibraryError

Determines whether a dialog box displays for any symbol that cannot be found.

#### Allowable Values:

- 0 - (Do not display a dialog box)
- 1 - (Display a dialog box)

#### Default Value:

0

## See Also

[Animator Parameters](#)

### [Animator]LibraryLength

The number of objects that will be held in the library cache.

#### Allowable Values:

1 to 100000

#### Default Value:

100

## See Also

[Animator Parameters](#)

### [Animator]LibraryTime

The period that imported symbols remain in the library cache.

#### Allowable Values:

- -1 (Objects remain cached until the cache is full) OR

- 0 to 864000 (seconds)

**Default Value:**

600

**See Also**

[Animator Parameters](#)

**[Animator]MaxAn**

The maximum number of animation points (ANs) on a graphics page. You should avoid large values for this parameter. Each AN uses memory and a large number of ANs will reduce the performance of your runtime system.

**Note:** Although this parameter is still valid, it is only used when Graphic Page designers select to use older style v3 and v4 objects.

---

**Allowable Values:**

256 to 32000

**Default Value:**

2048

**See Also**

[Animator Parameters](#)

**[Animator]MaximizedWindow**

Defines whether the runtime window is run in maximized mode on start-up.

The MaximizedWindow parameter only works when DynamicSizing is set to 1 and FullScreen is set to 0. This parameter can also be configured on the "Security Menu Control" dialog in the Computer Setup Wizard.

**Allowable Values:**

- 0 – The runtime window will not be set to maximized state on start-up.
- 1 – The runtime window will be set to maximized state on start-up.

**Default Value:**

0

## See Also

[Animator Parameters](#)

### [Animator]PrintXScale

The X scaling factor applied when the WinPrint() function is called to print the screen (on the printer).

#### Allowable Values:

1 to 10

#### Default Value:

4

## See Also

[Animator Parameters](#)

### [Animator]PrintYScale

The Y scaling factor applied when the WinPrint() function is called to print the screen (on the printer).

#### Allowable Values:

1 to 10

#### Default Value:

4

## See Also

[Animator Parameters](#)

### [Animator]TabFactor

Controls the distance between tab stops on the graphics display. Plant SCADA calculates the width of a tab stop by counting the number of characters, times the average width, times the tab factor. Normally the average width, times the number of characters makes the tab stops too wide. Therefore the tab factor is set to 70% to reduce the width. If you have wide characters, or use a lot of upper case the default tab factor may be too narrow, and overlapping of the display may result. (Increase up to 100% for the widest tab stops.)

**Allowable Values:**

10 to 100

**Default Value:**

70

**See Also**

[Animator Parameters](#)

**[Animator]TemplateUpdate**

Enables the update of linked pages to reflect template modifications.

Setting this parameter allows linked graphics pages to be updated if the templates on which they are based have been modified. The update takes place when you click Update Pages in the Tools menu of Graphics Builder.

With this parameter set, updating linked pages will overwrite page properties with those of the template. This means that if you want to change the properties of a linked page, you must set the template properties. The next time you perform an Update Pages, the page will include the template modifications.

**Allowable Values:**

- 1 - (to enable)
- 0 - (to disable)

**Default Value:**

0

**See Also**

[Animator Parameters](#)

**[Animator]TooltipFont**

Specifies the font to be used for tool tip text. This parameter is checked each time you start Plant SCADA. (You can also use the function DspSetTooltipFont to set the font.)

Unlike other parameters where only one value is supplied, you can enter up to three values to set the tool tip font - the name of the font, the point size, and formatting attributes. Only the name of the font is required. If you don't set values for the other attributes, their default values are used.

**Allowable Values:**

Name, PointSize, BIU

**Default Value:**

12 (PointSize)

" " (BIU)

- Name is the name of the chosen Windows font
- PointSize is the size of the font in points
- B specifies Bold
- I specifies Italic
- U specifies Underline

**See Also**

[Animator Parameters](#)

**[Animator]UseCTGIfNewer**

When a page is opened and Fast Runtime Display is enabled, Plant SCADA searches for both the CTF and the CTG files and uses whichever is newer. Use this parameter to specify that only the CTG file is searched for if the CTF cannot be found.

**Allowable Values:**

- 1 - Always check to see if CTG is newer
- 0 - The CTG is located and loaded only if the CTF cannot be found

**Default Value:**

1

**See Also**

[Animator Parameters](#)

**AnmCursor Parameters**

The cursor is represented by a box with an internal and an external border. Symbols and text have imaginary borders that represent the inside border. When specifying cursor dimensions, you have to specify the size of the external border.

- [\[AnmCursor\]Color](#) - Sets the color of the cursor.
- [\[AnmCursor\]Colour](#) - Obsolete in v7.20 (replaced with [\[AnmCursor\]Color](#)).
- [\[AnmCursor\]Height](#) - The distance (in pixels) from the top and bottom of the inside border to the outside border (of the cursor).
- [\[AnmCursor\]InvertText](#) - Enables/disables the inversion of the color of the text within the cursor box.
- [\[AnmCursor\]MouseSnapToCursor](#) - Specifies the behavior of the Windows cursor in relation to the Plant SCADA cursor
- [\[AnmCursor\]RubberBandColor](#) - Sets the color of the trend selection rubber band.
- [\[AnmCursor\]Thickness](#) - The thickness of the outside cursor border (in pixels).
- [\[AnmCursor\]Width](#) - The distance (in pixels) from the left and right of the inside border to the outside border (of the cursor).

## See Also

[Parameter Categories](#)

### [AnmCursor]Color

Sets the color of the cursor.

### Allowable Values:

0x000000 to 0xFFFFFFFF

Colors defined as RGB values, for example:

- 0x00000000 - (Black)
- 0x00000080 - (Blue)
- 0x00008000 - (Green)
- 0x00008080 - (Cyan)
- 0x00800000 - (Red)
- 0x00800080 - (Magenta)
- 0x00808000 - (Brown)
- 0x00BFBFBF - (Grey)
- 0x007F7F7F - (Dark Grey)
- 0x000000FF - (Light blue)
- 0x0000FF00 - (Light green)
- 0x0000FFFF - (Light cyan)
- 0x00FF0000 - (Light red)
- 0x00FF00FF - (Light Magenta)
- 0x00FFFF00 - (Yellow)
- 0x00FFFFFF - (White)

- 0xFF000000 - (Transparent)

**Default Value:**

0xFFFF (white)

**See Also**

[AnmCursor Parameters](#)

**[AnmCursor]Height**

The distance (in pixels) from the top and bottom of the inside border to the outside border (of the cursor).

**Allowable Values:**

1 to 20

**Default Value:**

1

**See Also**

[AnmCursor Parameters](#)

**[AnmCursor]InvertText**

Enables/disables the inversion of the color of the text within the cursor box.

---

**Note:** Although this parameter is still valid, it is only used for V3.xx and V4.xx animations.

---

**Allowable Values:**

- 0 - (Disable)
- 1 - (Enable)

**Default Value:**

0

**See Also**

[AnmCursor Parameters](#)

## [AnmCursor]MouseSnapToCursor

Specifies the behavior of the Windows cursor in relation to the Plant SCADA cursor.

When set to FALSE the Windows cursor and the Plant SCADA cursor behave independently.

When set to TRUE the Windows cursor snaps to the center of the Plant SCADA cursor. In this case a page update will not result if the cursor position is changing.

### Allowable Values:

- 0 - (False)
- 1 - (True)

### Default Value:

0

### See Also

[AnmCursor Parameters](#)

## [AnmCursor]RubberBandColor

Sets the color of the trend selection rubber band.

### Allowable Values:

0x000000 to 0xFFFF

Colors defined as RGB values, for example:

- 0x000000 - (Black)
- 0x000080 - (Blue)
- 0x008000 - (Green)
- 0x008080 - (Cyan)
- 0x800000 - (Red)
- 0x800080 - (Magenta)
- 0x808000 - (Brown)
- 0xBFBFBF - (Grey)
- 0x7F7F7F - (Dark Grey)
- 0x0000FF - (Light blue)
- 0x00FF00 - (Light green)
- 0x00FFFF - (Light cyan)
- 0xFF0000 - (Light red)

- 0xFF00FF - (Light Magenta)
- 0xFFFF00 - (Yellow)
- 0xFFFFFFFF - (White)

**Default Value:**

0xFFFFFFFF (white)

**See Also**

[AnmCursor Parameters](#)

**[AnmCursor]Thickness**

The thickness of the outside cursor border (in pixels).

**Allowable Values:**

1 to 20

**Default Value:**

1

**See Also**

[AnmCursor Parameters](#)

**[AnmCursor]Width**

The distance (in pixels) from the left and right of the inside border to the outside border (of the cursor).

**Allowable Values:**

1 to 20

**Default Value:**

2

**See Also**

[AnmCursor Parameters](#)

## Backup Parameters

The Citect.ini file contains the following backup parameters:

- [\[Backup\]SaveiniFiles](#) - Determines whether the "Save ini files" checkbox is checked by default during Backup.

## See Also

[Parameter Categories](#)

### [Backup]SaveiniFiles

Determines whether the **Save ini files** checkbox is checked or not when the Backup Project window is opened.

#### Allowable Values:

- 0 - unchecked
- 1 - checked

#### Default Value:

1

## See Also

[Backup Parameters](#)

### BrowseTableView Parameters

The [BrowseTableView] section contains the following parameters:

---

**Note:** These parameters are only available to pages created from the **tab\_style\_1.databrowse** page template.

- [\[BrowseTableView\]<BrowseType>.<ViewName>.ColWidths](#) - Sets the pixel widths of the columns of the current data browse table.
- [\[BrowseTableView\]<BrowseType>.<ViewName>.Fields](#) - Sets the field names of the columns in the current data browse table under a view name configured on the page.

## See Also

[Parameter Categories](#)

### [BrowseTableView]<BrowseType>.<ViewName>.ColWidths

Sets the widths of the columns in pixels for the current data browse table configured on the page. This parameter is only available to pages created from the "tab\_style\_1.databrowse" page template.

The view name is configured by opening the page in Graphics Builder and filling in the "View Name" field in the template form. To open the template form double-click it. Once saved in the INI file, the columns are automatically restored when the user re-enters the page or when the user chooses to restore them via the "Restore saved view" command under the Action menu tab.

The user can erase the saved view by selecting the "Reset view to default" command from the Action menu tab. The table columns is then reset to the original configuration of the project.

---

**Note:** If this parameter is configured as project parameter, it will act as the default configuration and be reset to when the user executes the "Reset view to default" command.

- <BrowseType> field can be any of the following values:

AccumBrowse, AlmBrowse, EquipBrowse, ServerBrowse, TagBrowse, TrnBrowse

This should be the same as the "Browse Type" field in the template parameter form of the Data Browse page.

- <ViewName> – allows the user to specify a name to identify the view that will be saved to or restored from. This should be the same as the "View Name" field in the template parameter form of the Data Browse page.

---

**Note:** You can specify the same view name on multiple Data browse pages The pages will then share the same view.

## Allowable Values:

The width of the columns in pixels, separated by the comma (,) character.

## Default Value:

N/A

## See Also

[BrowseTableView Parameters](#)

### [BrowseTableView]<BrowseType>.<ViewName>.Fields

Sets the field names of the columns in the current data browse table under 'view name' configured on the page. This parameter is only available to pages created from the "tab\_style\_1.databrowse" page template.

The view name is configured by opening the page in Graphics Builder and filling in the "View Name" field in the template form. To open the template form double-click it. Once saved in the INI file, the columns are automatically restored when the user re-enters the page or when the user chooses to restore them via the "Restore saved view" command under the Action menu tab.

The user can erase the saved view by selecting the "Reset view to default" command from the Action menu tab. The table columns is then reset to the original configuration of the project.

---

**Note:** If this parameter is configured as project parameter, it will act as the default configuration and be reset to when the user executes the "Reset view to default" command.

- <BrowseType> field can be any of the following values:

AccumBrowse, AlmBrowse, EquipBrowse, ServerBrowse, TagBrowse, TrnBrowse

This should be the same as the "Browse Type" field in the template parameter form of the Data Browse page.

- <ViewName> – allows the user to specify a name to identify the view that will be saved to or restored from. This should be the same as the "View Name" field in the template parameter form of the Data Browse page.

---

**Note:** You can specify the same view name on multiple Data browse pages. The pages will then share the same view.

---

## Allowable Values:

The name of the fields of the specified data browse type, separated by the comma (,) character.

## Default Value:

N/A

## See Also

[BrowseTableView Parameters](#)

## Client Parameters

The Citect.ini file contains the following client parameters that can be set manually. Other parameters, which are not listed below, can be set by using the Computer Setup Wizard.

- [\[Client\]AdditionalClientsArePartOfTrustedNetwork](#) - Tells an /X client process to authenticate using the stored server password.
- [\[Client\]AllowRPC](#) - Allows a client computer to run a procedure called remotely on a server by the Cicode function MsgRPC.
- [\[Client\]AutoLoginPage](#) - Set to enable auto login. Users can select one of six modes.
- [\[Client\]AutoLoginClearPassword](#) - Set to clear the cached client login credentials.
- [\[Client\]Clusters](#) - Selects which clusters the client is connected to at startup.
- [\[Client\]ComputerRole](#) - Specifies the role of the computer, either server and control client, view-only or control client.
- [\[Client\]CPU](#) - Sets the CPU that the client component is assigned to.
- [\[Client\]DisableDisplay](#) - Specifies that the client process will be run in the background without a visible page or window.
- [\[Client\]Display](#) - Obsolete in version 7.0.
- [\[Client\]Events](#) - Sets the events to be enabled on the client.
- [\[Client\]EvictTimeout](#) - The amount of time a tag reference is cached before it is evicted.
- [\[Client\]ForceClient](#) - Starts only the client process, and connects to configured servers using a network connection.

- [\[Client\]FullLicense](#) - Specifies that a Control Client will use a full license.
- [\[Client\]Manager](#) - Obsolete in version 7.0.
- [\[Client\]PartOfTrustedNetwork](#) - Tells a client process to authenticate using the stored server password.
- [\[Client\]PointCountRequired](#) - Specify what point count a client requires.
- [\[Client\]Primary](#) - Obsolete in version 7.0.
- [\[Client\]Process](#) - Obsolete in version 7.0.
- [\[Client\]ReadOnly](#) - Does not allow writes to any I/O device.
- [\[Client\]ResolveTimeout](#) - Time the clients will wait for a successful server response for a tag exists request.
- [\[Client\]ShutdownCode](#) - Determines the Cicode function to run when Plant SCADA DisplayClient component shuts down.
- [\[Client\]StalenessPeriod](#) - Specifies the total number of seconds that will elapse after the last update before extended quality of the tag element is set to "Stale".
- [\[Client\]StalenessPeriodTolerance](#) - Specifies a percentage value that will be tolerated for the staleness period.
- [\[Client\]Standby](#) - Obsolete in version 7.0.
- [\[Client\]StartupCode](#) - Determines the Cicode function to run when Plant SCADA DisplayClient component starts up.
- [\[Client\]TagReadCachedMode](#) - Specifies which mode TagInfo() and other CiCode functions use for retrieving Tag/Ass properties.
- [\[Client\]TagReadRoundToFormat](#) - Determines if the variable tag values returned by the TagRead and TagReadEx will be rounded to the format of the variable tag.
- [\[Client\]UseLocalLicense](#) - Determines where a Plant SCADA Anywhere client gets its license from.
- [\[Client\]WaitForConnectAtStartup](#) - Specifies that connection to a server will wait until it can establish a connection to the server processes before starting up.
- [\[Client\]WaitForValidDataTimeout](#) - Allows a client to display #COM errors on a project page if a response to a subscription is not received within the specified period of time.

---

**Note:** When Plant SCADA is running in single process mode, the client settings for Cluster, CPU, Events, ShutdownCode and StartupCode are used for the Plant SCADA process.

---

## See Also

[Parameter Categories](#)

### [Client]AdditionalClientsArePartOfTrustedNetwork

This parameter tells an /X Client process to attempt to authenticate using the stored server password, which results in the client node becoming part of a "trusted network".

---

**Note:** To run as part of the trusted network, the Windows user account that is running the process needs to be assigned to the Server Users security role (see the topic [Security Roles](#)).

---

In Plant SCADA, a network with nodes connected via trusted tran channels is called a "trusted network" and the nodes must use the same stored server password to trust each other. Only trusted tran channels can handle remote requests initiated by MsgRPC() and ServerRPC() Cicode functions.

Although all connections between servers are always trusted regardless of this parameter, the tran channel between server and client should be explicitly configured to be trusted by this parameter in order to process remote requests initiated from the client node.

### Allowable Values:

- 0 - /X client is not part of the trusted network
- 1 - /X client is part of the trusted network. Use the Computer Setup Wizard to configure the server password (if not previously done so).

### Default Value:

0

For more information refer to the following topics in the main help:

- [Server Password Configuration](#)
- [Page Table Tran.](#)

### See Also

[\[Client\]PartOfTrustedNetwork](#)

[Client Parameters](#)

### [Client]AllowRPC

Allows a client computer to run a procedure called remotely from a server, or another client, using the Cicode function MsgRPC.

---

**Note:** An MsgRPC call will only succeed if the current user on the remote computer is assigned to a Role that has the **AllowRPC** property set to "TRUE".

---

### Allowable Values:

- 0 – MsgRPC calls will not be successful.
- 1 – MsgRPC calls will be successful.

### Default Value:

0

### See Also

[Client Parameters](#)

## [Client]AutoLoginClearPassword

Set to clear the cached client login credentials for consistency with the server 'AutoLoginClearPassword' ini parameter.

### Allowable Values:

- 0 – Do nothing
- 1 – Clear the stored login credentials (resets to 0 after the login credentials have been cleared)

### Default Value:

0

### See Also

[Client Parameters](#)

## [Client]AutoLoginMode

Set to enable auto login. Users can select one of seven modes.

- **Mode 0** - Auto login is disabled.
- **Mode 1** - System logs in with the current windows user at Startup.
- **Mode 2** - System logs in with the current windows user at Startup and on logout. If the startup login fails the user will be prompted to login at startup and on logout so the system will not switch to view-only mode.

---

**Note:** Modes 1 and 2 act as the same as modes 3 and 4 if the start up login of current windows user fails.

- **Mode 3** - User will be prompted with an empty login form at startup.
- **Mode 4** - User will be prompted with an empty login form at startup and on log out. If the user selects cancel, the process will shut down.

---

**Note:** This mode should not be used on server process (where the ServerLogin is disabled).

- **Mode 5** - Automatically login with saved user credentials at startup. If login fails or there is no saved user credentials the user will be prompted to login.
- **Mode 6** - Attempt to automatically login with saved credential at startup and on logout. If saved credential is not available (or fails to login) it will prompt the user at startup to log in. The re-login of initial user on logout does not validate the user, therefore does not need to prompt even if the password is changed (e.g. in case of windows user).

To remove or change the credentials saved during modes 5 or 6, the user needs to set the mode to 3 or 4 and restart to be prompted for login. When the user successfully logs in the saved credential will be removed and the user can set the mode back to 5 or 6 and restart to be prompted again.

---

**Note:** In modes 1, 3 and 5 when the user logs out, system reverts to view-only mode and no further action is taken.

**Allowable Values:**

- 0 – (Auto login disabled. Control client starts in view-only mode, until valid user logs on)
- 1 – (Login current windows user at system start up)
- 2 - (Login current windows user as system default user at start up)
- 3 - (Prompt user for login at startup)
- 4 - (Prompt user for login at startup and logout)
- 5 - (Try login with saved credential at startup)
- 6 - (Try login with saved credential at startup and on logout)

**Default Value:**

0

**See Also**

[Client Parameters](#)

**[Client]Clusters**

Selects which clusters the client is connected to at startup.

**Allowable Values:**

Comma separated list of the names of any configured clusters.

**Default Value:**

All configured clusters.

---

**Note:** This parameter is used by Plant SCADA to specify the active clusters for all components at startup, when running in single process mode.

---

**See Also**

[Client Parameters](#)

**[Client]ComputerRole**

Specifies the role of the computer. This can be configured on the [Computer Role Setup](#) page of the Setup Wizard.

**Allowable Values:**

- 0 - Server and Control Client

- 1 - Control Client - enables [Client]FullLicense
- 2 - View-only Client
- 3 - HMI (Standalone, no networking)

**Default Value:**

NONE

**See Also**

[Client Parameters](#)

**[Client]CPU**

This parameter is by Plant SCADA Runtime Manager to determine which CPU(s) the client process is assigned to. The CPU parameter is configurable for every Plant SCADA process via the **CPU Setup** page in the Computer Setup Wizard (see [CPU Configuration](#)).

---

**Note:** When Plant SCADA is running in single process mode, this parameter is used to specify the CPU assignment for the Plant SCADA process.

If this parameter is modified while the Plant SCADA Runtime Manager is running, the changes will not take effect until every instance of Plant SCADA is restarted.

CPU numbers are specified in a comma separated list. CPU numbers need to be valid and a subset of the available CPUs on a computer. For example, if there are eight CPUs on a computer, a valid CPU range is 0 to 7.

**Allowable Values:**

A comma-separated list in the range 0 to 31 (inclusive), where 0 is the first CPU.

For example, "0, 1, 2" or "3, 4, 5".

If no value is entered, the process will run on all available CPUs.

---

**Note:** When a process is set to run on all available CPUs, be aware that performance issues may still occur when your CPU usage is less than 100%. You can detect these issues when your CPU usage for a process is fixed at 100 divided by the number of processors.

**Default Value:**

No value (the process will run on all CPUs).

---

**Note:** You cannot manually specify CPU assignments on a computer with more than 32 CPUs. Under these circumstances, the Setup Wizard will automatically set this parameter to no value (all CPUs). Any changes you make to this parameter will be overwritten.

**See Also**

[Client Parameters](#)

[\[Alarm.<ClusterName>.<ServerName>\]CPU](#)

[IOServer.<ClusterName>.<ServerName>]CPU  
[Report.<ClusterName>.<ServerName>]CPU  
[Trend.<ClusterName>.<ServerName>]CPU

### [Client]DisableDisplay

Specifies that the client process will run in the background without a visible page or window.

#### Allowable Values:

- 0 - Enable the display (The client display will be visible).
- 1 - Disable the display (The client will still run in a background process but the client window will not be shown).

#### Default Value:

0

### See Also

[Client Parameters](#)

### [Client]Events

The events to be enabled by the client.

**Note:** All events that have "Global" or nothing in the Name field are enabled automatically in the client.

#### Allowable Values:

List of events (comma separated).

**Note:**

- This parameter is used by Plant SCADA to specify the enabled events for the Plant SCADA process, when running in single process mode.
- To set this parameter, we recommend you enable the required events in the "Events Setup" page of the Computer Setup Wizard (instead of entering the events in the Setup Editor).

### See Also

[Client Parameters](#)

### [Client]EvictTimeout

The amount of time in milliseconds that a tag reference is held in cache without being requested by a client before it is evicted.

**Allowable Values:**

-1 to 2000000

A value of -1 means that [Client]EvictTimeout is turned off. This means that no tag references will be evicted for the life of a session, and this can consume a large amount of memory.

**Default Value:**

300000

**See Also**

[Client Parameters](#)

**[Client]ForceClient**

Starts only the client process and connects to a configured Plant SCADA servers using a network connection.

This is useful where multiple users login to the same server, for instance Windows 2003 Enterprise Edition or similar, and want to run Plant SCADA from their user accounts concurrently to an instance of Plant SCADA that is already running.

In such cases you should start RuntimeManager.exe with the /i option specifying an alternative Citect.ini file where [Client]ForceClient=1 is set.

**Allowable Values:**

- 0 - (Client and servers if they are configured).
- 1 - (Client only).

**Default Value:**

0

**See Also**

[Client Parameters](#)

**[Client]FullLicense**

Specifies that the client will use a full license. This parameter is ignored unless the computer is a client.

**Allowable Values:**

- 0 - do not use a full license
- 1 - use a full license

**Default Value:**

0

**See Also**

[Client Parameters](#)

**[Client]PartOfTrustedNetwork**

This parameter tells a client process to attempt to authenticate using the stored server password, which results in the client node becoming part of a "trusted network". This parameter is automatically set by the Computer Setup Wizard. You do not need to set this parameter on a computer that is running a server process.

**Note:** To run as part of the trusted network, the Windows user account that is running the process needs to be assigned to the Server Users security role (see the topic [Security Roles](#)).

In Plant SCADA, a network with nodes connected via trusted tran channels, is called a "trusted network" and the nodes must use the same stored server password to trust each other. You do not need to set this parameter on computers that are just accepting CTAPI connections. If the CTAPI connection is being used to access information on a remote server via MsgRCP or ServerRCP, then set this parameter manually.

Although all connections between servers with matching passwords are always trusted regardless of this parameter, the tran channel between server and client should be explicitly configured to be trusted by this parameter in order to process remote requests initiated from the client node. Accordingly, this parameter needs to be set to use the client process as a CtAPI server.

**Note:** To include an /X process in a trusted network, use the parameter [\[Client\]AdditionalClientsArePartOfTrustedNetwork](#).

**Allowable Values:**

- 0 - Client is not part of the trusted network
- 1 - Client is part of the trusted network. Use the Computer Setup Wizard to configure the server password (if not previously done so).

**Default Value:**

0

For more information refer to the following topics in the main help:

- [Server Password Configuration](#)
- [Page Table Tran](#).

**See Also**

[Client Parameters](#)

**[Client]PointCountRequired**

A client can specify what point count it requires by this parameter. If the parameter is not specified, the client will get the first available matching point count. If the parameter is specified, but the required point count is not found within the license, no point count (no license) is served to that client.

---

**Note:** This parameter only works with Sentinel Licensing USB keys. It does not apply when using AVEVA™ Enterprise Licensing.

---

**Allowable Values:**

Any integer >= 1 or 0 (unlimited)

**Default Value:**

1 (gets the first available point count)

**See Also**

[Client Parameters](#)

**[Client]ReadOnly**

Commands the Plant SCADA client not to write to any I/O device. This includes disk and physical I/O devices. This is useful when you want to minimize the likelihood that a client will modify the online system (e.g. when you are developing the client). When a write is made to a physical I/O device, and this parameter is set to 1, the following hardware error results: "Write location is protected".

**Allowable Values:**

- 0 - (Enable writes)
- 1 - (Disable writes)

**Default Value:**

0

**See Also**

[Client Parameters](#)

**[Client]ResolveTimeout**

The time period in milliseconds that clients will wait for a successful server response to a tag exists request. This setting may be increased to avoid "Tag resolve timeout" errors (error 532). However such errors are only expected to occur on heavily loaded systems or where communications are slow, and may indicate a

performance issue that requires investigation.

**Allowable Values:**

0 to 4 294 967 296

**Default Value:**

3000

**See Also**

[Client Parameters](#)

**[Client]ShutdownCode**

Can be used to specify a Cicode function that runs at shutdown of the Plant SCADA client component. Plant SCADA suspends the shutdown process until the function has finished running. The function must complete execution within the time specified by the [\[Code\]ShutdownTime](#) parameter, otherwise Plant SCADA will terminate it.

**Allowable Values:**

Any built-in or user-written Cicode function

You do not need to specify all the optional arguments, as the optional arguments will be given runtime defaults (not compiler defaults) as follows:

- Data type INT = 0
- Data type REAL = 0.0
- Data type STRING = ""

**Default Value:**

NONE

**See Also**

[Client Parameters](#)

**[Client]StalenessPeriod**

Specifies the total number of seconds that will elapse after the last update before extended quality of the tag element is set to "Stale". This parameter can be set on the server using the I/O Device configuration dialog.

**Allowable Values:**

- -1 - Use the I/O device setting
- 0 - Disable staleness period processing completely
- >0 - Local definition of staleness period (in seconds) Allowable values are 1 to 2147483647 seconds.

**Default Value:**

-1

**See Also**

[Client Parameters](#)

**[Client]StalenessPeriodTolerance**

Specifies the interval (as a percentage of the staleness period) at which displayed values will be evaluated to determine if they are stale.

**Allowable Values:**

1 to 100 percent of the staleness period

**Default Value:**

10

**See Also**

[Client Parameters](#)

**[Client]StartupCode**

Determines the Cicode function to run when Plant SCADA DisplayClient component starts up.

You can only pass constant data to the startup function call. You cannot pass variables or other functions. For example, MyFunc(1, "str") is valid, but MyFunc(PLCTAG, "str") or MyFunc(YourFunc(), "str") is not supported.

You do not need to specify all the optional arguments, as the optional arguments will be given runtime defaults (not compiler defaults) as follows:

- Data type INT = 0
- Data type REAL = 0.0
- Data type STRING = ""

---

**Note:** This parameter is modified by the Computer Setup Wizard.

---

**Allowable Values:**

Any built-in or user-written Cicode function

**Default Value:**

NONE

**Note:** This parameter is used by Plant SCADA to specify the Cicode function that runs at startup, when running in single process mode.

**See Also**

[Client Parameters](#)

**[Client]TagReadCachedMode**

Specifies which mode TagInfo() and other Cicode functions use for retrieving Tag/Ass properties.

**Allowable Values:**

- 0 - property is retrieved directly from the server
- 1 - property is retrieved from local cache
- 2 - property is retrieved from local configuration
- 3 - property is retrieved from local cache if the cached is loaded; otherwise from local configuration

**Default Value:**

3

**See Also**

[Client Parameters](#)

**[Client]TagReadRoundToFormat**

This setting determines if the variable tag values returned by the Cicode functions TagRead and TagReadEx will be rounded to the format of the variable tag.

Quite often, the value reported for a variable tag will not be precise, resulting in a large number of decimal places. When this parameter is set, the precision of the value returned is rounded to the format of the tag.

For example, if the value from an I/O device for a tag is 3.499999999999, but the format is ##.##, the rounded value will be 3.50.

**Allowable Values:**

- 0 - Values not rounded to format
- 1 - Values rounded to format

**Default Value:**

1

**See Also**[Client Parameters](#)**[Client]UseLocalLicense**

Determines where a Plant SCADA Anywhere client gets its license from.

**Allowable Values**

- 0 – a Plant SCADA Anywhere client will get its license from a remote license server, or directly from an AVEVA Enterprise License Server (if configured).
- 1 – a Plant SCADA Anywhere client will acquire its license from the local "SystemServices" process. This option is only valid if you are running as a service and have a Sentinel license attached to that computer.

**Default Value**

0

**See Also**[Client Parameters](#)**[Client]WaitForConnectAtStartup**

This is a Boolean parameter that specifies whether the client will wait until it can establish a connection to the server processes before starting up.

**Allowable Values:**

0 or 1.

**Default Value:**

For a new installation of Plant SCADA v7, the parameter exists in the Citect.ini file with a default setting of 0. For an existing installation that is being upgraded to v7, and the parameter is absent from this section of the

Citect.ini file, the application behaves as if the parameter existed and set to 1.

If the parameter is set to False (0), there is no delay and the client will attempt to connect in the background as Plant SCADA runtime starts up. If the parameter is set to True (1), the client will wait until it can establish a connection to the server processes before starting up.

## See Also

[Client Parameters](#)

### [Client]WaitForValidDataTimeout

Allows a client to display #COM errors on a project page if a response to a subscription is not received within the specified period of time.

#### Allowable Values:

1 - 60000 (milliseconds)

#### Default Value:

5000 (milliseconds)

## See Also

[Client Parameters](#)

## Code Parameters

The citect.ini file contains the following code parameters:

- [\[Code\]AlarmShutdown](#) - Obsolete in version 7.0.
- [\[Code\]AlarmStartup](#) - Obsolete in version 7.0.
- [\[Code\]AutoReRead](#) - Obsolete in version 7.0.
- [\[Code\]BackwardCompatibleErrHw](#) - Controls whether Plant SCADA Runtime should be backward compatible with versions prior to V5.20.
- [\[Code\]CallbackThreads](#) - Limits the number of tag subscription callback threads that will be executed simultaneously.
- [\[Code\]DebugMessage](#) - Enables/Disables the DebugMsg() logging functionality.
- [\[Code\]DIICallErrorPopup](#) - Creates a popup and logs errors on calls to third-party DLLs.
- [\[Code\]DIICallProtect](#) - Helps protects Plant SCADA Runtime from closing down in the event an external DLL is called and subsequently becomes inoperative.
- [\[Code\]DynamicCallDebug](#) - Allows you to control logging for Cicode that is called dynamically at runtime.
- [\[Code\]EchoError](#) - Controls whether Plant SCADA Runtime echoes errors in Cicode.
- [\[Code\]EnableErrorLogging](#) - Enables or disables the logging of issues with ErrSet() function and related

functions.

- [\[Code\]Export](#) - Controls if your Cicode is to be halted when an error occurs.
- [\[Code\]HaltOnError](#) - Stops your Cicode task if an error occurs in some key Cicode functions and generates a hardware error.
- [\[Code\]HaltOnInvalidTagData](#) - Stops Cicode when any tag read returns invalid data (bad quality).
- [\[Code\]IgnoreCase](#) - Controls whether Plant SCADA Runtime is case insensitive to string data in Cicode.
- [\[Code\]ProfilerEnabled](#) - Set this parameter to 1 to enable the Cicode Profiler when Plant SCADA Runtime starts.
- [\[Code\]IOServerShutdown](#) - Obsolete in version 7.0.
- [\[Code\]IOServerStartup](#) - Obsolete in version 7.0.
- [\[Code\]Process](#) - Obsolete in version 7.0.
- [\[Code\]Queue](#) - The maximum number of elements in queues.
- [\[Code\]ReportShutdown](#) - Obsolete in version 7.0.
- [\[Code\]ReportStartup](#) - Obsolete in version 7.0.
- [\[Code\]ScaleCheck](#) - Controls whether Plant SCADA Runtime checks for scaling over/under flow errors in Cicode.
- [\[Code\]Shutdown](#) - Obsolete in version 7.0.
- [\[Code\]ShutdownTime](#) - Specifies the maximum time allowed for the execution of any Cicode functions triggered by the [Code]Shutdown parameter.
- [\[Code\]Stack](#) - The size of the stack when Plant SCADA Runtime calls a user DLL function through the Cicode function DLLCall().
- [\[Code\]Startup](#) - Obsolete in version 7.0.
- [\[Code\]StrictArgumentCheck](#) - Determines whether the compiler checks for calls to functions missing parameters.
- [\[Code\]Threads](#) - The number of Cicode threads (tasks) that can run concurrently.
- [\[Code\]TimeData](#) - Determines how often (in milliseconds) the I/O server reads I/O device data in order to provide tags with up-to-date Cicode variables.
- [\[Code\]TimeSlice](#) - The period (in milliseconds) that a Cicode thread can run - before it is swapped out.
- [\[Code\]TimeSlicePage](#) - The period (in milliseconds) that a Cicode thread can run - before it is halted.
- [\[Code\]TrendShutdown](#) - Obsolete in version 7.0.
- [\[Code\]TrendStartup](#) - Obsolete in version 7.0.
- [\[Code\]Unsigned](#) - Determines if the capable protocols interpret 16 bit integers as unsigned or not.
- [\[Code\]VBASupport](#) - Enables and disables Plant SCADA VBA Support.
- [\[Code\]WriteLocal](#) - Controls whether Plant SCADA Runtime writes to a local run table in Cicode.

## See Also

[Parameter Categories](#)

## [Code]BackwardCompatibleErrHw

Enables/Disables the backward compatibility of Cicode functions **ErrGetHw** and **ErrSetHw** used after Plant SCADA v5.20

**Note:** Set this flag ONLY if you do not have less than 512 I/O devices in your project, AND you have used the ErrGetHw() or ErrSetHw() functions in that project. This will allow you to mask the "IODevNo" with the value of 512 in the "Device" argument of those functions.

### Allowable Values:

- 0 - (Disable backward compatibility)
- 1 - (Enable backward compatibility)

### Default Value:

0

### See Also

[Code Parameters](#)

## [Code]CallbackThreads

Limits the number of tag subscription callback threads that will executed simultaneously. The upper boundary of the parameter is based on the upper limit of [\[Code\]Threads](#), but in practice it is recommended not to use every available Cicode thread for callbacks .

### Allowable Values:

1 to 512

### Default Value:

5

### See Also

[Code Parameters](#)

## [Code]DebugMessage

Enables or disables the DebugMsg function, which provides inline debug messages to the Kernel and syslog.dat. Also controls the logging functionality of the Assert() function. This parameter can be set and reset at runtime with the DebugMsgSet() function.

**Allowable Values:**

- 0 - (Disable logging)
- 1 - (Enable logging)

**Default Value:**

0

**See Also**[Code Parameters](#)**[Code]DII Call Error Popup**

Creates a popup and logs errors on calls to third-party DLLs via Cicode, but only if the DII Call Protect parameter is set. The popup can be turned off by the global [\[Debug\]SysErrDsp=0](#) setting; however, the log data will be sent to the syslog.

**Allowable Values:**

- 0 - (Disable)
- 1 - (Enable)

**Default Value:**

0

**See Also**[Code Parameters](#)**[Code]DII Call Protect**

If, when calling an external DLL, the file has a software error, this may cause Plant SCADA Runtime to crash. You can help to protect Plant SCADA Runtime by setting the [\[Code\]DII Call Protect=1](#) value in your Citect.ini file.

Setting [\[Code\]DII Call Error Popup=1](#) also logs the Cicode function and arguments being used if it occurs.

Technical Support recommends that these parameters be set during commissioning.

As DLLs can use different languages and versions, it is possible for Plant SCADA Runtime to falsely catch an exception using the DII Call Protect=1 setting. We recommend a run is tried without the DII Call Protect setting, to see if it would happen in the called DLL.

**Allowable Values:**

- 0 - (Disable)
- 1 - (Enable protection)

**Default Value:**

0

**See Also**[Code Parameters](#)**[Code]DynamicCallDebug**

This parameter allows you to control logging for Cicode that is called dynamically at runtime.

If a Cicode function is called from the Cicode window in the Kernel or via the [TaskCall](#) function, this can only be detected at runtime. However, this will typically generate a message in the syslog indicating incorrect usage of Cicode.

If you set this parameter to zero (0), it will suppress the logging of syslog messages for Cicode that is used in this way.

**Allowable Values:**

- 1 - log syslog messages for dynamic Cicode calls  
0 - suppress syslog messages for dynamic Cicode calls

Default Value:

1

**See Also**[Code Parameters](#)**[Code]EchoError**

Controls whether Plant SCADA Runtime echoes errors in Cicode. When a simple user error occurs (for example, if the `PageDisplay()` function is passed a bad page name), Cicode displays an error message at the Error AN, and returns an error code from the function. If you disable this action, Cicode does not display the error message, and the output of the `DspError()` function is stopped. This parameter is global.

**Allowable Values:**

0 or 1

**Default Value:**

1 (Echo)

**See Also**

[Code Parameters](#)

**[Code]EnableErrorLogging**

Enables or disables the logging using the ErrSet Cicode function (and related functions).

When ErrSet(1) is used, this parameter allows watermark messages to the syslog of pairing issues with ErrSet(0) and ErrSet(1).

See the documentation for the [ErrSet](#) Cicode function.

**Allowable Values:**

0 - Disabled

1 - Enabled

**Default Value:**

0

**See Also**

[Code Parameters](#)

**[Code]Export**

Allows built-in Cicode functions to be exported to other DLLs or external applications.

**Allowable Values:**

0 or 1

**Default Value:**

1 (Enable export)

**See Also**

[Code Parameters](#)

**[Code]HaltOnError**

If an error occurs in some critical Cicode functions, your Cicode task will generate a hardware error and will halt. You may stop your Cicode task from being halted by using the ErrSet() function and checking for errors using the IsError() function. You may also stop your Cicode from being halted by setting this parameter to 0, however, the hardware error will still be generated. Setting this parameter is global to all Cicode threads, so you do not have to set ErrSet() for each Cicode tread.

The following functions can halt your current Cicode task when an error occurs: FormNew, DevOpen, DevHistory, DevNext, DevPrev, DevSeek, DevFind, DevFlush, DevRecNo, DevRead, DevReadLn, DevAppend, DevDelete, DevZap, DevControl, DevPrint, DevModify, ErrTrap, FileOpen, FileClose, FileReadBlock, FileWriteBlock, FileSeek, FileDelete, FileReName, FileSize, FileReadLn, FileCopy, SQLConnect, SQLTraceOn, SQLTraceOff, SQLErrMsg, TagReadEx.

**Allowable Values:**

- 1 (Halt Cicode when an error occurs in a function listed above)
- 0 (Do Not halt Cicode when an error occurs in a function listed above).

**Default Value:**

1

**See Also**[Code Parameters](#)**[Code]HaltOnInvalidTagData****WARNING****UNINTENDED EQUIPMENT OPERATION**

Enable the [Code]HaltOnInvalidTagData parameter if your application requires Plant SCADA Runtime to halt when invalid tag data is returned.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

When using compiled tags in Cicode, the LAST VALID VALUE could be returned when the latest tags value is unavailable, and this value may now be incorrect.

The new [Code]HaltOnInvalidTagDataINI parameter has been introduced to allow you to stop Cicode when invalid tag data is returned. The default handling is to NOT stop Cicode in such a case.

**Note:** Existing Cicode error checking FUNCTIONS can still be used instead of this option.

When enabled will cause the Cicode to halt when any tag read returns invalid data (bad quality). This will generate a hardware alarm.

**Note:**

- 
- The foreground code such as page animation and trend/alarm/report Cicode fields are exceptions and will not halt (page shows #COM); however button commands on a page will halt.
  - When enabled in the Citect.ini file you can still change this behavior in code by setting the ErrSet (1) and checking IsError() manually, in this case the code will not halt.
- 

**Allowable Values:**

- 0 - (Disable Halting)
- 1 - (Enable Halting)

**Default Value:**

0

**See Also**

[Code Parameters](#)

**[Code]IgnoreCase**

Controls whether Plant SCADA Runtime is case insensitive to string data in Cicode. Case sensitivity is used when a string comparison operation is performed. For example:

IF sStr1 = sStr2 THEN

Cicode is a case insensitive language and ignores the case of all variables, functions and reserved words. You can also make Plant SCADA Runtime case sensitive to strings within the just current thread, using the CodeSetMode() function.

**Note:** This parameter is global.

---

**Allowable Values:**

0 or 1

**Default Value:**

1 (Insensitive)

**See Also**

[Code Parameters](#)

**[Code]ProfilerEnabled**

Set this parameter to 1 to enable the Cicode Profiler when Plant SCADA Runtime starts. The Cicode Profiler enables you to gather information on how your Cicode is performing.

**Allowable Values:**

0 or 1

**Default Value:**

0 (disabled)

**See Also**

[Code Parameters](#)

**[Code]Queue**

The maximum number of elements in each queue. If you use a large number of queue elements, memory could be exhausted. See the Task functions for information about using queues.

**Allowable Values:**

-1 (Unlimited) or 0 to 1,073,741,824

**Default Value:**

32768

**See Also**

[Code Parameters](#)

**[Code]ScaleCheck**

Controls whether Plant SCADA Runtime checks for scaling over/under flow errors in Cicode. When a variable tag is modified, Cicode checks the new value of the variable against the Scales specified in the Variable Tags database. If the value of the variable is out of scale, Cicode generates a hardware error, and does not write to the I/O device. This parameter is global.

**Allowable Values:**

0 or 1

**Default Value:**

1 (Check)

## See Also

[Code Parameters](#)

### [Code]ShutdownTime

Specifies the maximum time allowed for the execution of any Cicode functions triggered by the [Code]Shutdown parameter. After the specified time passes, Plant SCADA will terminate the function.

#### Allowable Values:

1 to 120 (seconds)

#### Default Value:

30

## See Also

[Code Parameters](#)

### [Code]Stack

The size of the Cicode stack. If a "Cicode stack overflow" hardware error is displayed, increase the size of the Cicode stack. If Plant SCADA Runtime aborts with a "General stack overflow" error, increase the value of the [\[General\]Stack](#) parameter.

#### Allowable Values:

0 to 1024

#### Default Value:

256

## See Also

[Code Parameters](#)

### [Code]StrictArgumentCheck



**UNINTENDED EQUIPMENT OPERATION**

Do not set the [Code]StrictArgumentCheck parameter to a value of 0, except on the advice of Technical Support for this product. Setting this value to 0 can result in Cicode Functions behaving unexpectedly and raising a hardware alarm.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Determines whether the compiler checks for calls to functions with missing parameters. The compiler will by default pass a 1 value or an empty string to function calls that do not have a parameter, and no default defined.

## Allowable Values:

- 0 (do not check for missing parameters)
- 1 (check for missing parameters)

## Default Value:

1

## Example

```
INT
FUNCTION
AddTwoNumbers(INT a, INT b)
RETURN a + b;
END
FUNCTION
CallFunction()
INT sum = AddToNumbers();
This will not compile by default, but will compile when [CODE]StrictArgumentCheck=0.
```

## See Also

[Code Parameters](#)

## [Code]Threads

The number of Cicode threads (tasks) that can run concurrently. Note that each report running simultaneously uses one thread, and each keyboard command running simultaneously uses one thread. If the hardware error "Out of Cicode threads" is displayed, increase the value of this parameter.

## Allowable Values:

5 to 512

## Default Value:

128

## See Also

[Code Parameters](#)

### [Code]TimeData

Determines how often (in milliseconds) the I/O Server reads I/O device data in order to provide tags with up-to-date Cicode variables.

#### Allowable Values:

1 to 60000 (milliseconds)

#### Default Value:

250

## See Also

[Code Parameters](#)

### [Code]TimeSlice

The period (in milliseconds) that a Cicode thread can run - before it is swapped out. After all other threads have run, it is swapped back in. This parameter does not apply to threads that update graphics pages.

#### Allowable Values:

20 to 5000 (milliseconds)

#### Default Value:

100

## See Also

[Code Parameters](#)

### [Code]TimeSlicePage

The period (in milliseconds) that a Cicode thread can run - before it is halted. This parameter applies only to threads that are called from the Graphics databases.

**Allowable Values:**

20 to 5000 (milliseconds)

**Default Value:**

500

**See Also**

[Code Parameters](#)

**[Code]Unsigned**

---

**Note:** This parameter is obsolete. Plant SCADA now supports unsigned integers as a standard data type.

---

This parameter used to determine if 16 bit integers were interpreted as unsigned or signed. But only the MODCELL and COMLI protocols supported this option, and setting this parameter affected both protocols.

Signed integers can have values from -32768 to +32767. Unsigned integers can have values from 0 to 65535.

**Allowable Values:**

- 0 - (Interpret as signed)
- 1 - (Interpret as unsigned)

**Default Value:**

0

**See Also**

[Code Parameters](#)

**[Code]VBASupport**

This parameter allows a user to disable Plant SCADA VBA support.

**Allowable Values:**

- 0 - VBA features disabled
- 1 - VBA features enabled

**Default Value:**

1

## See Also

[Code Parameters](#)

### [Code]WriteLocal

Controls whether Plant SCADA Runtime updates its cache on a tag write rather than waiting for a read from the I/O device to confirm the new value.

With this option on (default) the client initiating the write will immediately show the new value as the operation begins. Other clients connected to the I/O Server will be updated on confirmation of a successful write. If the write does not succeed, the initiating client will revert back to the previous value update for that tag. Turning this option off will make sure that a new value is read from the I/O device before it is shown in any client.

---

**Note:** This parameter is loaded by both clients and I/O servers, so all such machines have to have the same setting in order to avoid potential tag value flicker issues. It is recommended to set this value in the project parameters database to make sure that all clients and servers will use the same value.

---

## Allowable Values:

- 0 - Update the cache from the I/O device
- 1 - Update cache at start of write operation

## Default Value:

1

## See Also

[Code Parameters](#)

### COM Parameters

- [COM]StartTimeout - Obsolete in version 7.10.

## See Also

[Parameter Categories](#)

### CrashHandler Parameters

The Citect.ini file contains the following CrashHandler parameters:

- [\[CrashHandler\]Enable](#) - Enables the Crash Handler, which collects a number of log and data files that may be useful in determining the cause of an unexpected system shut down.
- [\[CrashHandler\]KeepLogFile](#)s - Determines if the log files associated with an unexpected shut down are kept in the default logs directory.

## See Also

[Parameter Categories](#)

### [CrashHandler]Enable

If Plant SCADA suffers an unexpected program shut down, the Crash Handler will create a compressed file containing a number of log and data files that may be useful in determining the cause of the shut down. These files include:

- syslog.dat
- Citect.ini
- tracelog.dat
- kernel.dat
- debug.log

In a multi-process system, these files will use extended names, for example:

`syslog.IOServer.Cluster1.dat`

The compressed file can be emailed to Technical Support for analysis.

You can also save the files to the default Plant SCADA Logs directory. See [\[CrashHandler\]KeepLogFiles](#).

---

**Note:** Technical Support may use the information included in an unexpected shut down email to help resolve problems in future releases, but cannot reply or follow up a problem with users directly. To discuss a particular issue, contact Technical Support.

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Do not change the [CrashHandler]Enable parameter in the Citect.ini file, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Allowable Values:

- 0 - Disabled
- 1 - Enabled

## Default Value:

1

## See Also

[CrashHandler Parameters](#)

## [CrashHandler]KeepLogFiles

This parameter determines if these log files gathered by the Plant SCADA Crash Handler are kept in the default logs directory.

By default, Plant SCADA will not keep files. If you would like to maintain a copy for analysis, you should set this parameter to 1.

If the zip file is saved, its path will be recorded in the syslog.dat. This will be the first entry in the file after an unexpected shut down occurs.

### Allowable Values:

- 0 - disable the keeping of log files
- 1 - enable the keeping of log files

### Default Value:

0

### See Also

[CrashHandler Parameters](#)

### CtAPI Parameters

The Citect.ini file contains the following CtAPI parameters:

- [\[CtAPI\]AcceptThousandsSeparator](#) - Configures the handling of a thousands separator in numeric values passed to ctTagWrite() and ctTagWriteEx().
- [\[CtAPI\]AllowLegacyServices](#) - When set the Plant SCADA Web Service and the Plant SCADA OLEDB Provider can connect to the CtAPI server.
- [\[CtAPI\]CpuLoadCount](#) - Avoids CPU overload when processing ctFindFirst() to make a Trend or Alarm query by controlling how often a request pauses while gathering information. (Used in conjunction with CPULoadSleepMS.)
- [\[CtAPI\]CpuLoadSleepMS](#) - Avoids CPU overload when processing ctFindFirst() to make a Trend or Alarm query by controlling how long a request pauses while gathering information. (Used in conjunction with CpuLoadCount.)
- [\[CtAPI\]Debug](#) - Plant SCADA server side CtAPI debugging information of the communication sessions to the various CtAPI clients.
- [\[CtAPI\]EventLogging](#) - Enables logging of the communications between CtAPI clients and Plant SCADA.
- [\[CtAPI\]MaxConnections](#) - Specifies the maximum number of connections supported by a CTAPI client.
- [\[CtAPI\]Remote](#) - Determines whether remote computers using the CtAPI interface can call in to this computer.
- [\[CtAPI\]RoundToFormat](#) - Indicates whether or not you want the values rounded to format.

## See Also

[Parameter Categories](#)

### [CtAPI]AcceptThousandsSeparator

Configures the handling of a thousands separator in numeric values passed to `ctTagWrite()` and `ctTagWriteEx()`, allowing the value to be written to a numeric variable tag.

**Note:** The only character that is accepted as a thousands separator is "," (comma).

#### Allowable Values:

- 0 - don't accept thousands separator
- 1 - accept thousands separator

#### Default Value:

0

## See Also

[CtAPI Parameters](#)

### [CtAPI]AllowLegacyServices

When set the Plant SCADA Web Service and the Plant SCADA OLEDB Provider can connect to the CtAPI server.

#### Allowable Values:

- 0 - (Disable connection)
- 1 - (Allow connection)

#### Default Value:

0

## See Also

[CtAPI Parameters](#)

### [CtAPI]CpuLoadCount

Depending on the capability of your hardware, it is sometimes possible to overload the server's CPU while making a Trend or Alarm query using `ctFindFirst()`. To avoid this you can use CpuLoadCount in conjunction with

the CpuLoadSleepMS parameter.

CpuLoadCount allows you to control how often the request pauses (sleeps) while it is collecting data. The load count corresponds approximately with the number of rows of query data processed by the server before it pauses. Decreasing this number means the request will pause more often and hence take longer to complete. However the workload will be stretched over a longer time period, decreasing the intensity of demand on the CPU.

---

**Note:** You will need to experiment to see which values are appropriate for optimum performance on your system. When experimenting you should adjust the value of CpuLoadSleepMS before CpuLoadCount.

---

## Allowable Values:

0 to any valid INT

## Default Value:

100

## See Also

[CtAPI Parameters](#)

## [CtAPI]CpuLoadSleepMS

Depending on the capability of your hardware, it is sometimes possible to overload the server's CPU while making a Trend or Alarm query using [ctFindFirst\(\)](#). To avoid this you can use CpuLoadSleepMS in conjunction with the CpuLoadCount parameter.

CpuLoadSleepMS allows you to control how long the request pauses (sleeps) while it is collecting data. Pausing for longer periods will cause the request to take longer but will decrease the intensity of the load on the CPU.

---

**Note:** You will need to experiment to see which values are appropriate for optimum performance on your system. When experimenting you should adjust the value of CpuLoadSleepMS before CpuLoadCount.

---

## Allowable Values:

0 to any valid INT (milliseconds)

## Default Value:

100

## See Also

[CtAPI Parameters](#)

## [CtAPI]Debug

**Note:** This parameter is designed to assist qualified technical support personnel with system diagnosis.

Enables logging of server-side CtAPI debugging information for the communication sessions to the various CtAPI clients. Output is to the Syslog.dat file. This is useful to diagnose CtAPI connection and message exchange issues.

To enable logging of the communications produced by the CTAPI client, see [\[CtAPI\]EventLogging](#).

### Allowable Values:

- 1 = True
- 0 = False

### Default Value:

0

**Note:** You can adjust this parameter during runtime using the SetLogging() function. For more information, see [Adjust Logging During Runtime](#).

### See Also

[CtAPI Parameters](#)

## [CtAPI]EventLogging

**Note:** This parameter is designed to assist qualified technical support personnel with system diagnosis.

Enables logging of the communications produced by the CTAPI client. Debugging traces are provided when a client generates requests for the CTAPI server and when it responds to replies. Logging goes to ipc.log under the "Logs" directory.

For server-side CtAPI debugging information, see [\[CtAPI\]Debug](#).

### Allowable Values:

- 0 - Off
- 1 - Errors
- 2 - Warnings
- 3 - Verbose
- 4 - Verbose Plus

The higher the value, the more logging will occur.

### Default Value:

0

---

**Note:** IPC logging uses the highest of the values set for [CtAPI]EventLogging and [IPC]EventLogging. This means if you try to debug a CtAPI client on a system running multiple Plant SCADA processes, you can not turn off the IPC logging that occurs between these processes.

---

## See Also

[CtAPI Parameters](#)

### [CtAPI]MaxConnections

Specifies the maximum number of connections supported by a CTAPI client. You can increase this value if a CTAPI application requires more than 20 simultaneous connections.

#### Allowable Values:

1 — 50

#### Default Value:

20

## See Also

[CtAPI Parameters](#)

### [CtAPI]Remote

Determines whether remote computers using the CtAPI interface can call in to this computer.

---

**Note:** To use CtAPI on a remote computer without installing Plant SCADA, you will need several CtAPI binaries. These binaries are packaged in the files **CtApi.Win32.Redist.zip** and **CtApi.x64.Redist.zip**, which can be found in the `\Extras\CtAPI` directory of the Plant SCADA installation media.

---

#### Allowable Values:

- 0 - disable remote access
- 1 - allow remote access

#### Default Value:

0

## See Also

[CtAPI Parameters](#)

### [CtAPI]RoundToFormat

This setting configures whether or not variable tag values returned by CTAPI functions [ctTagRead](#), [ctTagReadEx](#) and [ctListData](#) will be rounded to the format of the variable tag. Quite often, the value reported for a variable tag will not be precise, resulting in a large number of decimal places.

When this parameter is set the precision of the value returned is rounded to the format of the tag. For example, if the value coming from the IO device for the tag is 3.4999999999999, but the format is ##.##, the rounded value will be 3.50.

#### Allowable Values:

- 0 - Values not rounded to format
- 1- Values rounded to format

#### Default Value:

1

#### See Also

[CtAPI Parameters](#)

#### CtCicode Parameters

The Citect.ini file contains the following CtCicode parameters:

- [\[CtCicode\]FastFormat](#) - Controls whether fast formatting is used in the Cicode Editor.
- [\[CtCicode\]MLCommentThreshold](#) - Sets the maximum number of characters you can use in a code comment for automatic syntax coloring to be supported.

#### See Also

[Parameter Categories](#)

### [CtCicode]FastFormat

Determines whether or not the Cicode Editor uses fast formatting to format text in Cicode files.

If you are experiencing irregularities in the display of text in Cicode files (particularly the use of bold and italics), it may be related to fast formatting.



#### UNINTENDED EQUIPMENT OPERATION

Do not change the [CtCicode]FastFormat parameter in the Citect.ini file, except on the advice of Technical

Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Allowable Values:

- 0 - Fast formatting is not used
- 1 - Fast formatting is used

## Default Value:

0

## See Also

[CtCicode Parameters](#)

### [CtCicode]MLCommentThreshold

If the number of characters in a code comment is greater than this value, the comment will not be automatically syntax colored by the Cicode Editor.

**Note:** The higher the setting for this parameter, the longer multi-line comments will take to format when their start and end strings are modified.

## Allowable Values:

Any integer greater than zero

## Default Value:

1024

## See Also

[CtCicode Parameters](#)

### CtDraw.RSC Parameters

The Citect.ini file contains the following CtDraw.RSC parameters:

- [\[CtDraw.RSC\]AllowEditSuperGeniePage](#) -When set enables the user to choose whether or not to open and edit a Super Genie page.
- [\[CtDraw.RSC\]BitmapCompression](#) - Controls how much compression is applied to bitmaps and symbols on a page.

- [\[CtDraw.RSC\]LibraryCacheSize](#) - sets the maximum size of a cache used to store library objects.
- [\[CtDraw.RSC\]PurgeGenieLists](#) - This parameter is used if a modified genie is passing a substitution string that was specific to the genie before it was modified, but is now invalid.

## See Also

[Parameter Categories](#)

### [CtDraw.RSC]AllowEditSuperGeniePage

By default a page that is based on a Super Genie library object can not be opened for editing. The user is advised to edit the Super Genie library object directly. This parameter gives the user the choice to edit the page regardless.

#### Allowable Values:

- 0 - User is blocked from opening or editing a page based on a Super Genie library object
- 1 - User is prompted for confirmation before a page based on a Super Genie is opened for editing

#### Default Value:

0

## See Also

[CtDraw.RSC Parameters](#)

### [CtDraw.RSC]BitmapCompression

Controls how much compression is applied to bitmaps and symbols on a page. The higher the value, the smaller the file size. However, the time required for compression increases with a higher value.

#### Allowable Values:

0 - 9

- 0 - no compression used
- 9 - maximum compression used

#### Default Value:

5

## See Also

[CtDraw.RSC Parameters](#)

### [CtDraw.RSC]LibraryCacheSize

This parameter sets the maximum size of a cache used to store library objects (symbols, Genies and templates) that have been previously been opened or used by other graphics since Graphics Builder was last launched.

Increasing the value of this parameter may improve performance if library objects are opened or used again, as the in-memory version can be retrieved instead of a copy that needs to be loaded from disk. This can have an impact on startup performance for projects with a large number of Composite Genies. However, this will also increase Graphics Builder's memory usage.

#### Allowable Values:

128 — 16384

#### Default Value:

4096

## See Also

[CtDraw.RSC Parameters](#)

### [CtDraw.RSC]PurgeGenieLists

This parameter is used if a modified genie is passing a substitution string that was specific to the genie before it was modified, but is now invalid.

After you have restarted the Graphics Builder, perform an Update Pages on the project.

#### Allowable Values:

- 0 Genie Lists will not be purged
- 1 Genie Lists will be purged

#### Default Value:

1

## See Also

[CtDraw.RSC Parameters](#)

## CtEdit Parameters

The citect.ini file contains the following CtEdit parameters:

- [\[CtEdit\]ANSIToOEM](#) - Indicates whether Windows ANSI characters are converted to OEM characters.
- [\[CtEdit\]Backup](#) - The backup directory that is used if a runtime database cannot be located (due to a disk becoming inoperative or a file is unreadable).
- [\[CtEdit\]Bin](#) - The directory where the Plant SCADA binary files are located.
- [CompileEnquiry](#) - Obsolete (now available through options).
- [\[CtEdit\]CompileUnsuccessfulCommand](#) - Indicates to the compiler an optional application, script or batch file to execute after an unsuccessful compile.
- [\[CtEdit\]CompileSuccessfulCommand](#) - Indicates to the compiler an optional application, script or batch file to execute after a successful compile.
- [\[CtEdit\]Config](#) - The directory where the Plant SCADA configuration files, such as Citect.ini, are located.
- [\[CtEdit\]Copy](#) - Sets the COPY directory. Changes made in this directory will be automatically copied to the RUN directory (see below).
- [\[CtEdit\]Data](#) - The directory where the Plant SCADA data files are located.
- [\[CtEdit\]DbFiles](#) - The maximum number of .DBF files that can be open simultaneously.
- [\[CtEdit\]DbfNdxMode](#) - Sets the method used to read DBF files that are indexed for reading records that match a key value.
- [\[CtEdit\]Deploy](#) - The location where a project will be stored when a deployment package is received from the deployment server.
- [\[CtEdit\]DisplayEquipmentItem](#) - Used to control the population of the variable tag list, or equipment item list in Graphics Builder.
- [\[CtEdit\]IncludeProjectEquipmentUpdate](#) - Determines whether an equipment update will also update included projects.
- [\[CtEdit\]IncrementalCompile](#) - Determines whether an incremental compile will occur.
- [\[CtEdit\]IncrementalEquipmentUpdate](#) - Determines whether an incremental equipment update will occur.
- [\[CtEdit\]IncrementalEquipmentUpdateAutoClose](#) - Determines if the Update Equipment confirmation dialog displays when an incremental update is complete.
- [\[CtEdit\]Logs](#) - The directory where the Plant SCADA log files are located.
- [\[CtEdit\]LogLevel](#) - Specifies the level of detail used in the equipment update log file.
- [\[CtEdit\]MaxCicodeFunctions](#) - The maximum number of user functions that can be defined in a project.
- [\[CtEdit\]MaxHelpRec](#) - The maximum number of items that may be contained in a drop-down list box.
- [\[CtEdit\]Run](#) - The directory where the runtime database is located.
- [\[CtEdit\]SaveRetries](#) - Specifies the number of retry attempts Plant SCADA Studio can make to save the content in the data grid if a database file is not accessible.
- [\[CtEdit\]Starter](#) - Specifies the directory where the starter projects are located.
- [\[CtEdit\]SubtAnValue](#) - A string to replace a Genie substitution string of %An%.
- [\[CtEdit\]SubtPageValue](#) - A string to replace a Genie substitution string of %Page%.
- [\[CtEdit\]SuppressCompilerWarning](#) - Allows suppression of specific compiler warning messages.

- [\[CtEdit\]UpdateAnPage](#) - Enables the substitution text of a Genie to be changed when upgrading projects.
- [\[CtEdit\]UpdateAnPageVer5](#) - Enables the substitution text of a Genie to be changed when upgrading projects from v5.10.
- [\[CtEdit\]Upgrade](#) - Allows you to manually upgrade the projects in Plant SCADA.
- [\[CtEdit\]User](#) - The directory where databases are located.

## See Also

[Parameter Categories](#)

### [CtEdit]ANSIToOEM

Indicates whether Windows ANSI characters are converted to OEM characters when Plant SCADA writes to data files (like dBase). This parameter applies to conversion in both directions (OEM To ANSI also).

---

**Note:** The project must be compiled and run with the same setting of this parameter, otherwise string conversions and logins that rely on ANSI to OEM conversions will not work. For example, you cannot compile with the value of 0 and then run with a value of 1.

---

If using an Arabic version of Windows, you should set this parameter to 1.

---

**Note:** This parameter must be set to 0 to support multiple languages.

## Allowable Values

- 0 - (Characters are not converted)
- 1 - (Characters are converted)

## Default Value:

0

## See Also

[CtEdit Parameters](#)

### [CtEdit]Backup

The Backup directory that is searched if a runtime database cannot be located (due to a disk becoming inoperative or a file is not readable). Normally, the runtime database is located in the Run directory. See the [\[CtEdit\]Run](#) parameter.

When both the [CtEdit]Backup and [CtEdit]Run parameters are set, Plant SCADA first searches for the runtime databases in the Run directory, and then may search the directory specified by [CtEdit]Copy. If the files are not found, the Backup directory is searched. If the files are found, the project continues to run from the Backup directory and Plant SCADA issues a file server error message.

To use this parameter, you must ensure that:

- A complete copy of the project exists on both the local computer and the computer specified in the Backup directory
- Both copies have the same filename
- Only UNC paths are used. Using mapped drives may cause included projects to not work correctly

---

**Note:** This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

---

## Allowable Values:

Any valid directory (as a UNC path)

## Default Value:

No default value

## See Also

[CtEdit Parameters](#)

### [CtEdit]Bin

The directory where the Plant SCADA binary files are located.

---

**Note:** This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

---

## Allowable Values:

Any valid directory

## Default Value:

This value is initially set to <Installation Path>\BIN, where <Installation Path> is the directory you chose during product installation. The installer defaults is "%PROGRAMFILES(x86)%\AVEVA Plant SCADA\Bin".

---

**Note:** if running a <filename>.ini, the value for this parameter will always be taken from the local Citect.ini file.

---

## See Also

[CtEdit Parameters](#)

### [CtEdit]CompileSuccessfulCommand

Indicates to the compiler an optional command, script or batch file to execute after a successful compile. To execute an optional application, script or batch file to execute after an unsuccessful compile use the parameter

[\[CtEdit\]CompileUnsuccessfulCommand](#)

**Allowable Values:**

Any valid command line containing replacement strings that represent possible information that the compiler can provide, including such information as project name, project path, INI path, bin path, etc. Enclose the replacement strings in '%' characters, similar to DOS batch file notation. Use the escape character "^" if you want to include the '%' character literally.

If an undefined replacement string is specified in the command line a compiler warning will be generated, and the application will not be called. Compiler warnings will also be generated if the compiler does not execute the command line successfully. However, any errors that occur when the command is executing will not be logged, as they will not be known to the compiler.

**Allowable replacement strings**

Replacement Strings	Description
CtEnvBinPath	Path to the Bin folder of Plant SCADA
CtEnvConfigPath	Path to the config folder
CtEnvDataPath	Path to data folder
CtEnvIncludeProjects	List of include projects for the project being compiled
CtEnvErrors	Number of errors encountered during compile
CtEnvIncludeProjectPaths	List of the directory paths of the include projects for the project being compiled, in the same order as the projects are listed in CtEnvIncludeProjects
CtEnvIniPath	Path to the ini file
CtEnvLogPath	Path to log files
CtEnvProject	Name of the project that is being compiled
CtEnvProjectPath	Path to the project file that is being compiled
CtEnvWarnings	The number of warnings that were encountered during compile

**Example:**

This could be a notification - such as sending an email or network message to a person - or just a log entry to a text file as shown below:

1. Create a text file with the following content:

```
Echo %Date% %Time% '%CtEnvProject%' Compilation successful, [%1] >>
"%CtEnvLogPath%\Log.txt"
```

2. Save the file as a command / batch file, example 'MyError.cmd'
3. Set the INI parameter to run the above file, including the path to where the file was saved. Pass the list of

include projects as the first parameter to the command file.

```
CompileSuccessfulCommand=C:\Temp\MyError.cmd "%CtEnvIncludeProjects%"
```

A Log.txt file will be created and updated each time a project compile is successful.

The replacement strings can be used directly in the INI parameter value and can be accessed as application environment variables. The example demonstrates this method.

## See Also

[CtEdit Parameters](#)

### [CtEdit]CompileUnsuccessfulCommand

Indicates to the compiler an optional command, script or batch file to execute after an unsuccessful compile. Generally this would be used to create a log file to show warnings or errors generated during a compile. To execute an optional application, script or batch file to execute after a successful compile use the parameter [\[CtEdit\]CompileSuccessfulCommand](#).

### Allowable Values:

Any valid command line containing replacement strings that represent possible information that the compiler can provide, including such information as project name, project path, INI path, bin path, etc. Enclose the replacement strings in '%' characters, similar to DOS batch file notation. Use the escape character "^" if you want to include the '%' character literally.

If an undefined replacement string is specified in the command line a compiler warning will be generated, and the application will not be called. Compiler warnings will also be generated if the compiler does not execute the command line successfully. However, any errors that occur when the command is executing will not be logged, as they will not be known to the compiler.

### Allowable replacement strings

Replacement Strings	Description
CtEnvBinPath	Path to the Bin folder of Plant SCADA
CtEnvConfigPath	Path to the config folder
CtEnvDataPath	Path to data folder
CtEnvIncludeProjects	List of include projects for the project being compiled
CtEnvErrors	Number of errors encountered during compile
CtEnvIncludeProjectPaths	List of the directory paths of the include projects for the project being compiled, in the same order as the projects are listed in CtEnvIncludeProjects

Replacement Strings	Description
CtEnvIniPath	Path to the ini file
CtEnvLogPath	Path to log files
CtEnvProject	Name of the project that is being compiled
CtEnvProjectPath	Path to the project file that is being compiled

**Example:**

This could be a notification - such as sending an email or network message to a person - or just a log entry to a text file as shown below:

1. Create a text file with the following content:

```
Echo %Date% %Time% 'CtEnvProject%' Compilation unsuccessful, [%1] >>
"%CtEnvLogPath%\Log.txt"
```

2. Save the file as a command / batch file, example 'MyError.cmd'
3. Set the INI parameter to run the above file, including the path to where the file was saved. Pass the list of include projects as the first parameter to the command file.

```
CompileUnsuccessfulCommand=C:\Temp\MyError.cmd "%CtEnvIncludeProjects%"
```

A Log.txt file will be created and updated each time a project compile is unsuccessful.

The replacement strings can be used directly in the INI parameter value and can be accessed as application environment variables. The example demonstrates this method.

**See Also**

[CtEdit Parameters](#)

**[CtEdit]Config**

The directory where the Plant SCADA configuration files, except Citect.ini, are located. The Citect.ini file needs to be located in the <Config> directory you chose during installation.

Default locations of configuration files are listed in View a Project's Folders.

---

**Note:** if running a <filename>.ini, the value for this parameter will always be taken from the local Citect.ini file.

**Allowable Values:**

Any directory that can be written to by the Windows user account that is used to run Plant SCADA.

---

**Note:** You need to manually configure the access control list for a new folder location so that it matches the permissions that were applied to the original location during installation. See [Configure Directory Security for Modified Folder Locations](#).

## Default Value:

<User>/<Data>/Config, where <User>/<Data> is the directory you chose during installation.

**Note:** If you want the Citect.ini file to be somewhere other than the installed location, move the Citect.ini file to the new location and modify the shortcut to CtExplor.exe to specify an additional argument, e.g. "newpath\ Citect.ini"

## See Also

[CtEdit Parameters](#)

### [CtEdit]Copy

Sets the COPY directory. Changes made to runtime files in this directory will be copied to the RUN directory (set using the [CtEdit]Run parameter). It is strongly recommended that the COPY project is compiled for the changes to appear in the RUN project.

**Note:** Do not use the [CtEdit]Run and [CtEdit]Copy parameters if you intend to use Plant SCADA's Deployment functionality.

This feature is useful when you want to have a local copy of a network project, but don't want to be continually checking that the local copy is the same as the master on the server (i.e. manually copy changed files across to the local project etc.) Plant SCADA takes care of this by copying files whenever it detects that the timestamp of the local file is different to that of the file in the COPY directory. Note that even if the local file has a more recent timestamp, it will still be replaced. It is recommended, therefore, that you only make changes to the project on the server.

**Note:** When using COPY the user folder should be shared. If not shared the project may not update correctly.

For example, you could set up a client so that instead of running the server version of a project (\\\\$ERVER\\PLANT SCADA\\USER\\MYPROJ) across a slow RAS connection, it runs a local copy (C:\\PLANT SCADA\\USER\\MYPROJ). In this instance, you would set your RUN directory to C:\\PLANT SCADA\\USER\\MYPROJ, and your COPY directory to \\\\$ERVER\\PLANT SCADA\\USER\\MYPROJ. You need to specify a UNC path for this parameter, as specifying a mapped drive may have unexpected results.

The Copy parameter can be changed while the system is running, enabling you to switch the SCADA node to copy files from a different location than was utilized when the system started.. This makes it possible to switch to a new runtime configuration that is stored in a different location so that if the changes need to be rolled back, the copy parameter can be restored to its previous value.. The Citect.ini file is read every time an RDB is loaded (at startup, page transition, reload) and the setting is retrieved from the INI file to adjust to your current copy path.

**Note:** A hardware alarm of "Cicode library timestamp differs" will be raised if the Cicode library used by a page has a different timestamp from the one in memory. The timestamps will be different if the project has been fully recompiled, the project has been incrementally recompiled after the page has been modified, or if the project has been incrementally recompiled after any Cicode has been modified.

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Restart the client process if the hardware alarm "Cicode library timestamp differs" is raised after a page is opened.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:**

1. For RUN/COPY (File Server Redundancy) setup, use [\[Alarm\]UseConfigLimits=0](#) so that the alarm properties are not persisted to the DBF/RDB files. If not the RDB files will be over written the next time they are accessed by the system.
2. Set up the directories the same way on the local hard disk as you do on the remote file server. Files are not copied from the COPY directory to the RUN directory until they are accessed - uncompiled files will also be copied once accessed on the local machine when they are used by the Plant SCADA runtime system. Runtime changes in included projects are also copied across. The Citect.ini file is not copied. Any configuration changes implemented in the ini file need to be manually copied across to the local machine.

**Allowable Values:**

Any valid directory (as a UNC path)

**Default Value:**

No default value.

**Note:** For additional information refer to the topic "Client Side Online Changes" in the main help.

**See Also**

[CtEdit Parameters](#)

**[CtEdit]Data**

**⚠ WARNING**

**SYSTEM PERFORMANCE DEGRADATION**

The "on access" scan in anti-virus products can lock files used by Plant SCADA, usually having the effect of slowing Plant SCADA down whilst it waits for the scan of that file to finish.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ CAUTION**

**INOPERABLE SYSTEM OR LOSS OF DATA**

In some extreme cases, anti-virus software may (incorrectly) detect certain patterns within data files as being viruses. Depending on the anti-virus configuration, this may result in files being relocated or deleted, resulting in data being lost or the system being inoperable.

**Failure to follow these instructions can result in injury, or equipment damage.**

It is recommended that the following directories are excluded from scanning by any anti-virus products:

- Program Files installation directory (including files and sub directories)
- Data and Logs directories
- Any alarm server archive paths

The above exclusions are recommended for "on access" or "real time" scans that run continuously and scan each file that is read from or written to. Scheduled scans, which are usually daily or once a week, should also have these exclusions added with the exception of the Program Files exclusion.

The directory where the Plant SCADA data files are located. Default locations of configuration files are listed in New Locations for Configuration and Project Files topic in the main help.

---

**Note:** This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

---

## Allowable Values:

Any valid directory. Be aware that Plant SCADA will not allow data to be stored in any subdirectory of ...\\Plant SCADA <VersionNumber> \\User\\.

---

**Note:** You need to manually configure the access control list for a new folder location so that it matches the permissions that were applied to the original location during installation. See [Configure Directory Security for Modified Folder Locations](#).

---

## Default Value:

<User>/<Data>/Data, where <User>/<Data> is the directory you chose during installation.

---

**Note:** If running a <filename>.ini, the value for this parameter will always be taken from the local Citect.ini file.

---

## See Also

[CtEdit Parameters](#)

### [CtEdit]DbFiles

The maximum number of .DBF files that can be open simultaneously. The Plant SCADA runtime system, Plant SCADA Studio, and the compiler all use this parameter. Increase this parameter if a "No more free handles" error message is displayed.

## Allowable Values:

50 to 32767

## Default Value:

1024

## See Also

[CtEdit Parameters](#)

### [CtEdit]DbfNdxMode

Sets the method used to read DBF files that are indexed for reading records that match a key value.

## Allowable Values

- 0 — Use associated NDX file to perform key based read.
- 1 — Read directly from DBF returning records that match the key value, for Graphics Builder page DBFs.
- 2 — In memory index.

## Default Value:

2

## See Also

[CtEdit Parameters](#)

### [CtEdit]Deploy

Sets the location where a project's runtime files will be stored when a deployment package is received from the deployment server and unpacked.

To run a project from this location, the [\[Deployment\]Enabled](#) parameter must be set to 1 (enabled).

## Allowable Values:

Any valid path on the local computer.

## Default Value:

%PROGRAMDATA%\AVEVA Plant SCADA\Deployment\Client\Projects

## See Also

[CtEdit Parameters](#)

### [CtEdit]DisplayEquipmentItem

Used to control the population of the variable tag list, or equipment item list in Graphics Builder. When set to 1, the equipment item list will be populated; when set to 0, variable tag list will be populated.

---

**Note:** This parameter is modified via the Plant SCADA Studio Options dialog.

---

**Allowable Values:**

0 or 1

**Default Value:**

1

**See Also**

[CtEdit Parameters](#)

**[CtEdit]IncludeProjectEquipmentUpdate**

Determines whether an equipment update will also update included projects.

**Allowable Values:**

- 0 - Equipment update will only update the selected project
- 1 - Equipment update will update both the selected project and its included projects

**Default Value:**

1

**See Also**

[CtEdit Parameters](#)

**[CtEdit]IncrementalCompile**

Determines whether an incremental compile will occur. When enabled, a compile will only include changes that have been made since the last compile occurred.

This parameter sets the **Incremental compile** option on the Plant SCADA Studio Options dialog.

**Allowable Values:**

- 0 - Incremental compile is disabled
- 1 - Incremental compile is enabled

**Default Value:**

1

**See Also**[CtEdit Parameters](#)**[CtEdit]IncrementalEquipmentUpdate**

Determines whether an incremental equipment update will occur. When enabled, an equipment update will only generate tags for equipment and equipment types that have been modified since the last update occurred.

This parameter sets the **Incremental equipment update** option on the Plant SCADA Studio Options dialog.

**Allowable Values:**

- 0 - Incremental equipment update is disabled
- 1 - Incremental equipment update is enabled

**Default Value:**

1

**See Also**[CtEdit Parameters](#)**[CtEdit]IncrementalEquipmentUpdateAutoClose**

Determines if the Update Equipment confirmation dialog displays when an incremental update is complete.

**Allowable Values:**

- 0 - the Update Equipment dialog will display, even if no changes were required during the equipment update. This provides access to the **Show Log** button.
- 1 - the Update Equipment dialog will automatically close if no changes were required during the equipment update. For this setting to work, you need to have incremental equipment updates enabled (see [\[CtEdit\]IncrementalEquipmentUpdate](#)).

**Default Value:**

1

## See Also

[CtEdit Parameters](#)

### [CtEdit]Logs

The directory where the Plant SCADA log files are located.

Default locations of configuration files are listed in New Locations for Configuration and Project Files in the main help.

#### Allowable Values:

Any directory to which is writable by the Windows user account that is used to run Plant SCADA.

---

**Note:** You need to manually configure the access control list for a new folder location so that it matches the permissions that were applied to the original location during installation. See [Configure Directory Security for Modified Folder Locations](#).

---

#### Default Value:

`<User>/<Data>/Logs`, where `<User>/<Data>` is the directory you chose during installation.

---

**Note:** If running a `<filename>.ini`, the value for this parameter will always be taken from the local Citect.ini file.

---

## See Also

[CtEdit Parameters](#)

### [CtEdit]LogLevel

Specifies the level of detail used in the equipment update log file (EquipGen.log).

#### Allowable Values:

- 0 — None
- 3 — Error
- 5 — Summary
- 10 — Details
- 15 — Verbose

#### Default Value:

## See Also

[CtEdit Parameters](#)

### [CtEdit]MaxCicodeFunctions

The maximum number of user functions that can be defined in a project. Increase this parameter if the "Too many Cicode functions" compile error occurs.



#### UNINTENDED EQUIPMENT OPERATION

Do not change the [CtEdit]MaxCicodeFunctions parameter in the citect.ini file, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Allowable Values:

10000 to 32767

## Default Value:

4500

---

**Note:** Increasing this parameter may affect system performance.

---

## See Also

[CtEdit Parameters](#)

### [CtEdit]MaxHelpRec

The maximum number of items that may be contained in a drop-down list box, generally a lower number such as 500 is easier for the user to select. This option is easily changed through the Plant SCADA Options dialog.

Although the upper limit of this parameter is only limited by the size of the INT 32 variable, the size it is set to doesn't affect runtime performance. If a project has 6000 tags, only these 6000 tags will consume memory, even if MaxHelpRec is set much higher.

## Allowable Values:

1 to 2,147,483,647.

## Default Value:

5000

## See Also

[CtEdit Parameters](#)

### [CtEdit]Run

The directory where the runtime project is located. If you run the project from Plant SCADA Studio, Graphics Builder, or Cicode Editor, this parameter will be automatically set to the name of the current project. If you want to run a project independently of these applications, you will need to set this parameter manually. It is recommended that the directory chosen should be a sub directory of the ..\Plant SCADA\User directory.

**Note:** Do not use the [CtEdit]Run and [CtEdit]Copy parameters if you intend to use Plant SCADA's Deployment functionality.



### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Restart the client process if the hardware alarm "Cicode library timestamp differs" is raised after a page is opened.

A hardware alarm of "Cicode library timestamp differs" will be raised if the Cicode library used by a page has a different timestamp from the one in memory. The timestamps will be different if the project has been fully recompiled, the project has been incrementally recompiled after the page has been modified, or if the project has been incrementally recompiled after any Cicode has been modified.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

#### Note:

1. For RUN/COPY (deployment) setup, use [\[Alarm\]UseConfigLimits=0](#) so that the alarm properties are not persisted to the DBF/RDB files. If not the RDB files will be over written the next time they are accessed by the system.
2. If you have set a COPY directory (using the [\[CtEdit\]Copy](#) parameter), any changes made and compiled in that directory will be automatically copied to the RUN directory. Any file with a name that matches one of the files changed in the COPY directory will be overwritten during this process. This copy only occurs when the files are accessed at runtime.

## Allowable Values:

Any valid directory under the ..\Plant SCADA\User directory.

## Default Value:

No default value.

**Note:** If running a <filename>.ini, the value for this parameter will always be taken from the local Citect.Ini file.

For additional information, refer to the topic *Client Side Online Changes* in the main Plant SCADA documentation.

## See Also

[CtEdit Parameters](#)

### [CtEdit]SaveRetries

If any database files are not accessible when you attempt to save the content in the data grid, Plant SCADA Studio can make some retry attempts to save the content.

This parameter represents the number of retry attempts Plant SCADA Studio will make before an error message appears. The duration between two consecutive retry attempts is one second.

Any changes to this parameter will be implemented after you restart Plant SCADA Studio.

## Allowable Values:

0 to 10

## Default Value:

3

## See Also

[CtEdit Parameters](#)

### [CtEdit]Starter

This parameter specifies the directory where the starter projects are located.

## Allowable Values:

Any directory that can be written to by the Windows user account that is used to run Plant SCADA.

---

**Note:** You need to manually configure the access control list for a new folder location so that it matches the permissions that were applied to the original location during installation. See [Configure Directory Security for Modified Folder Locations](#).

---

## Default Value:

<User>/<Data>/Starter, where <User>/<Data> is the directory you chose during installation.

## See Also

[CtEdit Parameters](#)

### [CtEdit]SubtAnValue

A string to replace a Genie substitution string of %An%.

When upgrading Plant SCADA from versions 3.xx or 4.xx to version 5.20 or higher, you need to rename any Genie substitution strings called %An%.

The substitution text will change during the upgrade if you enable [CtEdit]UpdateAnPage. If [CtEdit]UpdateAnPage is enabled, but you do not specify a new Genie substitution string, the string will be renamed to the default value.

#### Allowable Values:

New Genie substitution string (up to 256 characters)

#### Default Value:

AnSubt

#### See Also

[CtEdit Parameters](#)

### [CtEdit]SubtPageValue

A string to replace a Genie substitution string of %Page%.

When upgrading Plant SCADA from versions 3.xx or 4.xx to version 5.20 or higher, you need to rename any Genie substitution strings called %Page%.

The substitution text will change during the upgrade if you enable [CtEdit]UpdateAnPage. If [CtEdit]UpdateAnPage is enabled, but you do not specify a new Genie substitution string, the string will be renamed to the default value.

#### Allowable Values:

New Genie substitution string (up to 256 characters)

#### Default Value:

PageSubt

#### See Also

[CtEdit Parameters](#)

### [CtEdit]SuppressCompilerWarning

Allows suppression of specific compiler warning messages using an associated warning message number.

When a compiler warning message is displayed, it will include a number prefixed by "W". For example:

The specified language is not supported (W1027)

To suppress a particular type of warning message, apply the associated warning number to this parameter. For example, to suppress the unsupported language message demonstrated above, set the parameter to the following value:

[CtEdit]SuppressCompilerWarning=W1027

#### Allowable Values:

One or more warning message numbers. If more than one is used, separate them with commas.

#### Default Value:

No messages suppressed.

#### See Also

[CtEdit Parameters](#)

### [CtEdit]UpdateAnPage

Allows genie substitution strings of %Page% or %An% to be renamed during project upgrades from versions 3.xx or 4.xx to version 5.21 or higher.

Without setting this parameter, you will be unable to specify a value for the substitution strings, as %Page% will automatically be set to the name of the current page, and %An% will be set to the AN of the current page. You will also be unable to change these values.

The substitution strings will be renamed during an upgrade if this parameter is enabled. Use [\[CtEdit\]SubtAnValue](#) and [\[CtEdit\]SubtPageValue](#) to specify the new substitution strings.

#### Allowable Values:

- 0 to disable
- 1 to enable

#### Default Value:

0

#### See Also

[CtEdit Parameters](#)

### [CtEdit]UpdateAnPageVer5

Allows genie substitution strings of %Page% or %An% to be renamed during project upgrades from version 5.10.

Without setting this parameter, you will be unable to specify a value for the substitution strings, as %Page% will automatically be set to the name of the current page, and %An% will be set to the AN of the current page. You will also be unable to change these values.

The substitution strings will be renamed during an upgrade if this parameter is enabled. Use [CtEdit]SubtAnValue and [CtEdit]SubtPageValue to specify the new substitution strings.

This parameter will be reset to 0 (zero) once you have restarted Plant SCADA.

#### Allowable Values:

- 0 to disable
- 1 to enable

#### Default Value:

0

#### See Also

[CtEdit Parameters](#)

### [CtEdit]Upgrade

Allows you to manually upgrade the projects in Plant SCADA when a new version or service pack is installed and the automatic upgrade was not successful, or you chose not to upgrade at that time.

To perform a manual upgrade, set the parameter to 1 then close and restart Plant SCADA.

This parameter will be reset to 0 (zero) once you have restarted Plant SCADA.

#### Allowable Values:

- 0 to disable
- 1 to enable

#### Default Value:

0

#### See Also

[CtEdit Parameters](#)

## [CtEdit]User

The directory where databases are located.

**Note:** This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

### Allowable Values:

Any valid directory.

### Default Value:

<User>/<Data>/User, where <User>/<Data> is the directory you chose during installation.

**Note:** You need to manually configure the access control list for a new folder location so that it matches the permissions that were applied to the original location during installation. See [Configure Directory Security for Modified Folder Locations](#).

### See Also

[CtEdit Parameters](#)

## CtExplore Parameters

The Citect.ini file contains the following default parameters of the Plant SCADA projects page settings:

- [\[CtExplore\]DefaultTemplateStyle](#) - controls the default page setting for *Template style* used in Plant SCADA projects.
- [\[CtExplore\]DefaultTemplateResolution](#) - controls the default page setting for *Template resolution* used in Plant SCADA projects.
- [\[CtExplore\]DefaultTemplateTitleBar](#) - controls the default page setting for *Show template title bar* used in Plant SCADA projects.

### See Also

[Parameter Categories](#)

## [CtExplore]DefaultTemplateStyle

This parameter controls the default page setting for *Template style* used in Plant SCADA projects.

If this INI parameter specifies XP\_Style then the screen resolution will be XGA and the title bar not displayed no matter how the [\[CtExplore\]DefaultTemplateResolution](#) and [\[CtExplore\]DefaultTemplateTitleBar](#) parameters are set.

If this INI parameter specifies a style other than XP\_Style then XP\_Style will not be available from the NewProject | Template Style drop-down list.

**Allowable Values:**

- Standard
- Top
- Bottom
- XP\_Style
- Tab\_Style\_1

**Default Value:**

Tab\_Style\_1

**See Also**

[CtExplore Parameters](#)

**[CtExplore]DefaultTemplateResolution**

This parameter controls the default page setting for *Template resolution* used in Plant SCADA projects.

**Allowable Values:**

- VGA
- SVGA
- XGA
- SXGA
- WUXGA
- HD1080
- User
- Default

**Default Value:**

Default

**See Also**

[CtExplore Parameters](#)

**[CtExplore]DefaultTemplateTitleBar**

The parameter controls the default page setting for *Show template title bar* used in Plant SCADA projects.

**Allowable Values:**

0 or 1

**Default Value:**

0

**See Also**

[CtExplore Parameters](#)

**DBClient Parameters**

The Citect.ini file contains the following DBClient parameters:

- [\[DBClient\]Enabled](#) - enables ODBC logging.
- [\[DBClient\]FileBase](#) - specifies a location for the ODBC log files and a file name prefix.
- [\[DBClient\]MaxFiles](#) - specifies the maximum number of ODBC log files to retain.
- [\[DBClient\]MaxSize](#) - specifies the maximum size for an ODBC log file (in kilobytes).
- [\[DBClient\]OldFiles](#) - specifies the maximum number of log file sets to retain.

**See Also**

[Parameter Categories](#)

**[DBClient]Enabled**

Can be used to enable ODBC logging.

**Allowable Values:**

- 0 = disable ODBC logging
- 1 = enable ODBC logging

**Default Value:**

0

**See Also**

[DBClient Parameters](#)

## [DBClient]FileBase

Specifies the location where ODBC log files are stored, and a prefix for the file names. The following format is used for the parameter value:

{file path}\{file name prefix}

The file name prefix you specify will be appended by "\_<nnn>" when a log file is created, where "nnn" is an incremental numeric value.

For example, you may specify the following parameter value:

%PROGRAMDATA%\MyLogs\DBClient

This would generate the following log file:

%PROGRAMDATA%\MyLogs\DBClient\_001.log

### Allowable Values:

Any valid path and file name prefix in the following format:

{file path}\{file name prefix}

### Default Value:

%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Logs\DBClient\DBClient

### See Also

[DBClient Parameters](#)

## [DBClient]MaxFiles

Specifies the maximum number of ODBC log files to retain.

If the maximum number of files is reached and a new ODBC log file is created (due to an existing file reaching its maximum size), the oldest ODBC log file will be deleted.

### Allowable Values:

0 - 2,147,483,647

### Default Value:

2

### See Also

[DBClient Parameters](#)

**[DBClient]MaxSize**

Sets the maximum size for an ODBC log file (in kilobytes).

By limiting the size of individual log files, you can make the files more manageable. For example, you can limit their size so that they are suitable for viewing in Notepad or sending via e-mail.

**Allowable Values:**

0 - 2,147,483,647

**Default Value:**

2048

**See Also**

[DBClient Parameters](#)

**[DBClient]OldFiles**

Specifies the number of sets of log files that can be stored for each instance of the application.

If the number of log file sets reaches the defined limit, any new sets of log files will replace the oldest existing log files (which are deleted). The existing files are renamed (incremented by one).

**Allowable Values:**

0 - 2,147,483,647

**Default Value:**

1

**See Also**

[DBClient Parameters](#)

**DDE Parameters**

The Citect.ini file contains the following DDE parameters:

- [\[DDE\]AllowCicode](#) - Allows Cicode to be run on the Plant SCADA server via the DDE Execute command.
- [\[DDE\]AllowWrites](#) - Allows tag writes to the Plant SCADA server via the DDE Poke command.
- [\[DDE\]Enable](#) - Allows Plant SCADA runtime to function as a DDE server.
- [\[DDE\]Timeout](#) - The period (in milliseconds) that Plant SCADA waits after a call to an external application -

before it assumes the application does not exist.

## See Also

[Parameter Categories](#)

### [DDE]AllowCicode

Allows Cicode to be run on the Plant SCADA server via the DDE Execute command.

#### Allowable Values:

0 or 1

#### Default Value:

0

## See Also

[DDE Parameters](#)

### [DDE]AllowWrites

Allows tag writes to the Plant SCADA server via the DDE Poke command.

#### Allowable Values:

0 or 1

#### Default Value:

0

## See Also

[DDE Parameters](#)

### [DDE]Enable

Use this parameter to allow Plant SCADA runtime to function as a DDE server. By default, the parameter is set to disable DDE.

**Allowable Values:**

- 0 = disable DDE
- 1 = enable DDE

**Default Value:**

0

**See Also**[DDE Parameters](#)**[DDE]Timeout**

The period (in milliseconds) that Plant SCADA waits after a call to an external application - before it assumes the application does not exist.

**Allowable Values:**

0 to 32767 (milliseconds)

**Default Value:**

30000

**See Also**[DDE Parameters](#)**Debug Parameters**

The Citect.ini file contains the following debug parameters:

- [\[Debug\]ArchiveFiles](#) - Archives log files once maximum size is reached.
- [\[Debug\]CategoryFilter](#) - Enables log message filtering by component category.
- [\[Debug\]CategoryFilterMode](#) - Enables or disables logging of categories declared by the [Debug]CategoryFilter value.
- [\[Debug\]ChromiumDevToolsPort](#) - Enables debugging for a Web Content object located on a Plant SCADA graphics page.
- [\[Debug\]CodeDebug](#) - Enables Cicode debugging for the process containing a specified component.
- [\[Debug\]CrashOnError](#) - Determines whether Plant SCADA generates a protection violation whenever it detects a software error, for example, a memory corruption error such as "Invalid Heap" , "Already Free", 'try to free bad or unknown memory" or "local free no heap".

- [\[Debug\]CrashReserveManagedMB](#) - Specifies how much memory is reserved at startup to allow crash handling to occur for managed code.
- [\[Debug\]CrashReserveNativeMB](#) - Specifies how much memory is reserved at startup to allow crash handling to occur for native code.
- [\[Debug\]DBLogEnable](#) - Controls whether alarm database logs are enabled.
- [\[Debug\]DBLogMaxFiles](#) - Controls the maximum number of alarm database log files to keep for the current run of the server.
- [\[Debug\]DBLogMaxSize](#) - Controls the maximum size of an individual alarm database log file.
- [\[Debug\]DBLogOldFiles](#) - Controls the maximum number of runs to keep for alarm database log files.
- [\[Debug\]DebugAllTrans](#) - Allows TRAN session logging to be turned on or off.
- [\[Debug\]DebugLogHistoryFiles](#) - Determines how many "debug.<process name>.log" files are kept when rolled over.
- [\[Debug\]DebugLogSize](#) - Sets the maximum size of the "debug.<process name>.log" file.
- [\[Debug\]DriverTrace](#) - An extension of the Kernel that can be used to display the commands and information being issued to a driver.
- [\[Debug\]DriverTracePort](#) - Allows a driver trace to be limited to a particular driver port.
- [\[Debug\]DriverTraceMask](#) - Defines a bit mask that can be used to either include or exclude driver commands from a driver trace.
- [\[Debug\]DriverCheck](#) - Defines the maximum number of data buffer elements to be displayed in a driver trace.
- [\[Debug\]DebugAllTrans](#) - Allows TRAN sessions to be logged to the Kernel window for debugging purposes.
- [\[Debug\]DisableWinTop](#) - Allows you to minimize the likelihood of windows or popups from being forced to the top of the screen.
- [\[Debug\]DriverCheck](#) - Monitors the structure used to talk to drivers.
- [\[Debug\]DrWatson](#) - Starts DrWatson when Plant SCADA starts running (if DrWatson is not already running).
- [\[Debug\]EnableHardwareAlarmLogging](#) - Enables or disables logging of stack traces for hardware alarms to syslog.
- [\[Debug\]EnableLogging](#) - Enables or disables the logging mechanism.
- [\[Debug\]EnablePSICounters](#) - Enables or disables the Citect.Platform.PSI.Client and Citect.Platform.PSI.Server performance counters.
- [\[Debug\]EnableTaskFrameworkCounters](#) - Enables or disables Citect.Platform.Tasks.TaskManager, Citect.Platform.Tasks.TaskQueue and Citect.Platform.Tasks.TaskThread performance counters.
- [\[Debug\]EnableTransportCounters](#) - Enables or disables Citect.Platform.Transport.SessionTransport and Citect.Platform.Transport.TransportHost performance counters.
- [\[Debug\]EnableReloadLogging](#) - Enables additional debugging output for server reloads.
- [\[Debug\]FlushPeriod](#) - Sets the period at which Tracelog file entries are to be flushed to the file system.
- [\[Debug\]LogDirect](#) - Indicates whether to log directly to a log file or via the Kernel logging system.
- [\[Debug\]LogsDriveMinimumFreeSpace](#) - Specifies the minimum amount of free space required on the drive that hosts the log directory.
- [\[Debug\]LogShutdown](#) - Only required if ShutdownProtect is set to log the shutdown sequence.
- [\[Debug\]MaximumFileSize](#) - Sets the maximum size of an archive file.

- [\[Debug\]MaxMiniDumps](#) - Sets the maximum number of mini-dump files that are stored in the log directory.
- [\[Debug\]MaxTableOutputLength](#) - When set will limit the length of tables when they are output to a kernel.dat file.
- [\[Debug\]Memory](#) - Obsolete in v7.30.
- [\[Debug\]Menu](#) - Determines whether the Kernel option is displayed on the control menu of the Plant SCADA runtime system.
- [\[Debug\]MiniDumpType](#) - Defines the type of mini-dump that is performed.
- [\[Debug\]Priority](#) - Allows you to filter logging messages by priority.
- [\[Debug\]SeverityFilter](#) - Enables log message filtering by severity.
- [\[Debug\]SeverityFilterMode](#) - Enables or disables logging of severities declared by the [Debug]SeverityFilter value.
- [\[Debug\]Shutdown](#) - Determines whether the Shutdown option is displayed on the control menu of the Plant SCADA runtime system.
- [\[Debug\]ShutDownProtect](#) - Enables shutdown protection.
- [\[Debug\]SnapshotEnable](#) - Controls whether periodic diagnostic snapshots are enabled for the alarm database.
- [\[Debug\]SnapshotInterval](#) - Sets the number of minutes between periodic alarm database snapshots.
- [\[Debug\]SnapshotMaxFiles](#) - Controls the maximum number of alarm database snapshots to keep for the current run of the server.
- [\[Debug\]SnapshotMaxSize](#) - Controls the maximum size of an alarm database snapshot.
- [\[Debug\]SnapshotOldFiles](#) - Controls the maximum number of runs to keep for alarm database snapshot files.
- [\[Debug\]SysErrDsp](#) - Disables many system error logs and popup boxes.
- [\[Debug\]SysLogArchive](#) - Causes the syslog to automatically timestamp files as they are rolled over.
- [\[Debug\]SysLogSize](#) - Sets the size of the SYSLOG.DAT file.
- [\[Debug\]UserTraceMode](#) - Allows you to filter messages logged to the syslog.dat.

## See Also

[Parameter Categories](#)

### [Debug]ArchiveFiles

Enables or disables the archiving of the tracelog.dat log file once the maximum size specified by [\[Debug\]MaximumFileSize](#) is reached.

### Allowable Values:

- 0 - disable archiving
- 1 - enable archiving

## Default Value:

0

The following format is used for the file name:

tracelog.[server][timestamp].dat

where:

- [server] = <type>.<cluster>.<username>
- [timestamp] = \_YYMMDD\_HHMMSS

For example:

tracelog.Alarm.Cluster1.AlarmServer1\_051231\_235959.dat

The files are located in the following directory:

%PROGRANDATA%\AVEVA Plant SCADA <VersionNumber>\Logs

---

**Note:** Consider the amount of available disk space before you start archiving log files, as the disk space required to store traces may be significant. A long term archiving strategy is recommended to avoid running a system close to its limitations. If the system runs out of physical resources such as memory or disk space, it may cease to operate.

## See Also

[Debug Parameters](#)

### [Debug]CategoryFilter

Allows you to filter the logging messages sent to the tracelog.dat file by component category.

The [\[Debug\]CategoryFilterMode](#) parameter indicates whether messages of the specified categories will be included or excluded from the log.

---

**Note:** By default, the setting for [\[Debug\]CategoryFilterMode](#) will exclude the categories specified by [\[Debug\]CategoryFilter](#).

## Allowable Values:

Zero or more of the following categories specified as a comma-separated string:

- AlarmClient - traces events from the display layer of the alarm client
- AlarmClientAdaptor - traces events from the adapter layer of the alarm client that marshals between the old and new alarm system
- AlarmClientComms - traces events from the client side of alarm communications
- AlarmClientModel - traces events from the model layer of the alarm client
- AlarmFilterParse - alarm filter parsing logging
- AlarmServer - traces events from the Alarm Server
- AlarmServerAdaptor - traces events from the adapter layer of the alarm server that marshals between the

old and new alarm system

- AlarmServerComms - traces events from the server side of alarm communications
- AlarmServerModel - traces events from the model layer of the alarm server
- CSAToPSI - traces events from the interface between the Client-Side Adapter (CSA) and the Publisher-Subscriber Infrastructure (PSI)
- DataSource - traces setup and cache changes for data sources on the server side
- Instrumentation - traces information from the instrumentation components
- Logging - traces events from the logging subsystem
- ManagedUtil - traces information from the managed utility module
- OpcDaServerComApi - enables tracing of calls at a COM API level
- OpcDaBrowse205 - enables tracing related to OPC DA v2.05 item browsing
- OpcDaBrowse300 - enables tracing related to OPC DA v3.00 item browsing
- OpcDaServer - enables extensive tracing of OPC DA Server object APIs
- OpcDaGroup - enables extensive tracing of OPC DA Server object APIs
- PlatformHost - traces events from the Plant SCADA platform components that are hosted by runtime
- PSIClient - traces events from the client-side handling of tag requests
- PSIInterfaces - traces events from the PSI interfaces
- PSIServer - traces events from the server-side handling of tags requests
- RuntimeManager - traces events from the Runtime Manager
- TaskFramework - traces events from the framework that handles the scheduling and execution of work tasks at runtime
- TagBrowsing - traces events from the client and server handling of tag browsing requests
- Transport - traces events from the platform networking subsystem
- XmlUtils - traces information from the XML utilities module

## Default Value:

Empty string, that is, no categories excluded or included depending on [\[Debug\]CategoryFilterMode](#).

## Example

The following setting will trace tag requests to the point where the data originates:

```
[DEBUG]
CategoryFilter=CSAToPSI,PSIClient,PSIServer,DataSource
CategoryFilterMode=0
```

When combined with the implementation of [\[Debug\]DriverTrace](#), [\[PubSub\]LogDevice](#) and [\[PubSub\]LogLevel](#), this provides a top to bottom data flow trace.

---

**Note:** You can adjust this parameter during runtime using the `SetLogging()` function. For more information, search the Plant SCADA Help for the topic *Adjusting Logging During Runtime*.

## See Also

[Debug Parameters](#)

### [Debug]CategoryFilterMode

Allows or denies logging of categories declared by the [\[Debug\]CategoryFilter](#).

#### Allowable Values:

- 0 - Log only categories defined by the CategoryFilter value
- 1 - Allow logging of all categories except the ones defined by the CategoryFilter value.

#### Default Value:

1

---

**Note:** You can adjust this parameter during runtime using the SetLogging() function. For more information, search the Plant SCADA Help for the topic *Adjusting Logging During Runtime*.

In addition to setting this parameter you need to also set the [\[Debug\]SeverityFilterMode](#) and the [\[Debug\]SeverityFilter](#) parameters. Other parameters associated with syslog.dat file are; [\[Debug\]ArchiveFiles](#), [\[Debug\]EnableLogging](#), [\[Debug\]MaximumFileSize](#)

## See Also

[Debug Parameters](#)

### [Debug]ChromiumDevToolsPort

This parameter enables debugging for a Web Content object located on a Plant SCADA graphics page.

It lets you specify a port number that Chrome can connect to, allowing you to analyze the Web Content browser using Chrome's developer tools.

For example, if you set this parameter to 1026, you can access information about the Web Content browser by pointing a new instance of Chrome at the following address:

`http://localhost:1026`

---

This URL displays a list of links that represent the contents of any Web Content controls that are currently active. The text used for each link will be the title of the page that is displayed. If any controls are using frames, the list may include additional entries. Select a link to access the developer tools for a particular Web Content control.

---

**Note:** You should confirm that the port you select is not used by a Plant SCADA process or any other application.

#### Allowable Values:

1024 - 65535

**Default Value:**

0 (disabled)

**Note:** It is recommended that you set this parameter back to 0 (disabled) when you have finished debugging a Web Content object.

**See Also**

[Debug Parameters](#)

**[Debug]CodeDebug**

Enables Cicode debugging for the process containing the component specified by this parameter.

The default value is "Client" meaning that the client process has Cicode debugging enabled.

**Allowable Values:**

- Client
- Report or ClusterName.Report
- Alarm or ClusterName.Alarm
- Trend or ClusterName.Trend
- IOServer or ClusterName.IOServer

**Default Value:**

Client

**Note:** You cannot debug a process that is running as a Windows™ service.

**See Also**

[Debug Parameters](#)

**[Debug]CrashOnError**

Determines whether Plant SCADA generates a protection violation whenever it detects a software error, for example, a memory corruption error such as "Invalid Heap", "Already Free", 'try to free bad or unknown memory" or "local free no heap". If you enable this parameter, you must ensure that DrWatson is running. (DrWatson will trap the error and produce a trace.) Note that DrWatson is not supplied with some versions of Windows.

When you have reproduced the error, you must disable this parameter. Otherwise any error found by Plant SCADA will cause a General Protection Fault (and if DrWatson is not running, Plant SCADA will be aborted).

**Allowable Values:**

- 0 - (Disable)
- 1 - (Enable)

**Default Value:**

0

**See Also**[Debug Parameters](#)**[Debug]CrashReserveManagedMB**

This parameter specifies how much memory is reserved at startup to allow crash handling to occur for managed code. In a situation where a process runs out of memory, this reserved memory is released to allow the crash handling package to work.

The default value for this parameter (2 MB) is a minimum value. Values below this default are not accepted.

---

**Note:** It is recommended that you do not change the value for this parameter unless instructed by Technical Support.

---

**Allowable Values:**

2 - 1000 (MB)

**Default Value:**

2 (MB)

**See Also**[\[Debug\]CrashReserveNativeMB](#)[Debug Parameters](#)**[Debug]CrashReserveNativeMB**

This parameter specifies how much memory is reserved at startup to allow crash handling to occur for native code. In a situation where a process runs out of memory, this reserved memory is released to allow the crash handling package to work.

The default value for this parameter (40 MB) is a minimum value. Values below this default are not accepted.

---

**Note:** It is recommended that you do not change the value for this parameter unless instructed by Technical Support.

---

**Allowable Values:**

40 - 1000 (MB)

**Default Value:**

40 (MB)

**See Also**

[\[Debug\]CrashReserveManagedMB](#)

[Debug Parameters](#)

**[Debug]DBLogEnable**

Controls whether alarm database log files are enabled under:

[Logs]\DBLog.Alarm.<cluster>.<server>\DB\_<nnn>.log.

**Allowable Values:**

- 1 = On
- 0 = Off

**Default Value:**

1

**See Also**

[Debug Parameters](#)

**[Debug]DBLogMaxFiles**

Controls the maximum number of alarm database log files to keep for current run of the server.

A rolling log system is used, starting with “DB\_001.log”. When this file reaches the maximum size (as specified by [\[Debug\]DBLogMaxSize](#)), it will be closed and “DB\_002.log” will be opened, and so on.

This parameter determines how many of these files will be retained. When the maximum number is reached, the earliest file will be deleted.

For example, when this parameter is set to 5 and DB\_005.log reaches its maximum size, then DB\_001.log will be deleted and DB\_006.log will be opened.

**Allowable Values**

1-50

## Default Value

2

## See Also

[Debug Parameters](#)

### [Debug]DBLogMaxSize

Controls the maximum size (in kilobytes) of an individual alarm database log file.

A rolling log system is used, starting with “DB\_001.log”. When this file reaches the specified maximum size, it will be closed and “DB\_002.log” will be opened, and so on.

## Allowable Values:

10240 (10 MB) - 204800 (200 MB)

## Default Value:

51200 (50 MB)

## See Also

[Debug Parameters](#)

### [Debug]DBLogOldFiles

Controls the maximum number of runs to keep for alarm database log files. When an alarm server is started, any old files from a previous run will be renamed.

For example, “DBLog\_xxx.log” will be renamed “DBLog\_xxx.log\_2”, “DBLog\_xxx.log\_2” will be renamed “DBLog\_xxx.log\_3”, and so on.

The oldest run will be deleted when the specified maximum is reached. For example, a setting of 1 will keep one set of old files (the current files and “DBLog\_xxx.log\_2” )

## Allowable Values:

0-10

## Default Value:

1

## See Also

[Debug Parameters](#)

**[Debug]DebugAllTrans**



### UNINTENDED EQUIPMENT OPERATION

Do not change [Debug]DebugAllTrans or set the [TranDebug] parameter in the Citect.ini file, except on the advice of Technical Support.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Allows TRAN session logging to be turned on or off. This allows startup messages to be seen during start-up, so that fresh TRAN sessions during runtime can be traced. TRAN is Plant SCADA's PC-to-PC low-level communications protocol. TRAN sessions are logged to the Kernel and the syslog.dat file.

## Allowable Values:

- 0 - Default
- 1 - Enable TRAN logging

## Default Value:

0

---

**Note:** You can adjust [Debug]DebugAllTrans during runtime using the SetLogging() function. For more information, search the Plant SCADA Help for the topic *Adjusting Logging During Runtime*.

---

## Logging a specific TRAN session

You can log a specific TRAN session by entering the following into the Citect.ini file:

[TranDebug]<name>

Where:

- <name> is the name of a TRAN session as viewed in the Kernel.

This will produce the same logs as [Debug]DebugTranAll, though only for the session specified.

## See Also

[Debug Parameters](#)

## [Debug]DebugLogHistoryFiles

Determines how many "debug.<process name>.log" files are kept when rolled over. When a log file is rolled over, it is renamed "debug.<process name>.001". The next is named "debug.<process name>.002", and so on.

If the specified number of files is reached, the oldest file will be deleted when the next rollover occurs. The occurrence of rollovers is determined by the parameter [\[Debug\]DebugLogSize](#).

### Allowable Values:

1 to 10

### Default Value:

1

### See Also

[Debug Parameters](#)

## [Debug]DebugLogSize

Sets the maximum size of the "debug.<process name>.log" file. When the file reaches the specified size, it is rolled over.

When a log file is rolled over, it is renamed "debug.<process name>.001". The next is named "debug.<process name>.002", and so on. The number of log files that are stored is determined by the parameter [\[Debug\]DebugLogHistoryFiles](#), which has a default value of 1.

### Allowable Values:

1 to 10240 (KB)

### Default Value:

4096

### See Also

[Debug Parameters](#)

## [Debug]DisableWinTop

Allows you to disable windows or popups from being forced to the top of the screen. This allows an engineer to perform Kernel debugging or to run Cicode from the Kernel without windows appearing. This INI is intended for commissioning when you need to use the "Kernel".

**Allowable Values:**

- 0 - (Normal)
- 1 - (Disable win on top)

**Default Value:**

0

**See Also**[Debug Parameters](#)**[Debug]DriverCheck**

This parameter is designed for driver developers, to assist users who are experiencing non-recoverable errors due to driver issues.

When this parameter is set, the data structure used for communication between a driver and the I/O server (referred to as the DCB) is monitored for changes. If an error occurs, a Kernel and syslog message is generated.

**Allowable Values:**

- 0 - Disabled
- 1 - Enabled
- 2 - Enabled (with additional confirmation of DCB validity)

**Default Value:**

0

**Example**

With **[debug]drivercheck=1** set, the DCB is checksummed.

On return, the checksum is checked as well as a flag at the end of the DCB buffer; for example:

```
*** WARNING: DriverCheck=1 DCB->buffer overrun ***
```

or

```
*** WARNING: DriverCheck=1 DCB corrupted/changed ***
```

Receipt of this message may not represent a serious error. If a driver cleared the whole DCB buffer of 554 bytes, then expect this error.

If a driver modifies the point information, expect notification via a DCB corruption message.

**[debug]drivercheck=2** will additionally verify that a DCB belongs to a given unit. If this is not the case, an error will be logged.

## See Also

[Debug Parameters](#)

[\[Debug\]DriverTrace](#)



### UNINTENDED EQUIPMENT OPERATION

Do not change the [Debug]DriverTrace parameter in the Citect.ini file, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

The DriverTrace utility is an extension of the Plant SCADA Kernel that can be used to display the commands and information being issued to a driver by Plant SCADA. It can be useful in debugging a driver, for example, it can help determine why a driver is unable to fully initialize.

## Allowable Values:

- OFF - Turns the DriverTrace off.
- CMDS - Turns the DriverTrace on, but does not display the contents of the data buffer associated with the command. Be aware that specific driver commands need to be enabled with the DriverTraceMask INI parameter (or associated Kernel command).
- VER - Turns the DriverTrace on, and includes the contents of the data buffer in the display. Be aware that specific driver commands need to be enabled with the DriverTraceMask INI parameter (or associated Kernel command).
- ERR - Turns the DriverTrace on, but only traces DCBs with errors. (DCBs are the internal data structures used for communication between the I/O server and a driver.)

## Default Value:

OFF

**Note:** You can adjust this parameter during runtime using the SetLogging() function. For more information, search the Plant SCADA Help for the topic Adjusting Logging During Runtime.

## See Also

[Debug Parameters](#)

[\[Debug\]DriverTracePort](#)

[\[Debug\]DriverTraceMask](#)

[\[Debug\]DriverTraceElements](#)

[\[Debug\]DriverTracePort](#)

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not change the [Debug]DriverTracePort parameter in the Citect.ini file, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

This parameter allows a driver trace to be limited to a particular driver port.

**Allowable Values:**

- <port name> = a specified port
- \* = all ports

**Default Value:**

( \* ) all ports

**See Also**

[Debug Parameters](#)

[\[Debug\]DriverTrace](#)

[\[Debug\]DriverTraceMask](#)

[\[Debug\]DriverTraceElements](#)

**[Debug]DriverTraceMask****⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not change the [Debug]DriverTraceMask parameter in the Citect.ini file, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Uses a 4-byte hexadecimal number to define a bit mask that can be used to either include or exclude driver commands from a driver trace.

**Allowable Values:**

The driver commands and their values are as follows:

<b>Command</b>	<b>Bit Position</b>
CTDRV_INIT	00000001
CTDRV_OPEN	00000002
CTDRV_INIT_CHANNEL	00000004
CTDRV_INIT_UNIT	00000008
CTDRV_READ	00000010
CTDRV_WRITE	00000020
CTDRV_CONVERT	00000040
CTDRV_CANCEL	00000080
CTDRV_CPU	00000100
CTDRV_DATABASE	00000200
CTDRV_STOP_UNIT	00000400
CTDRV_STOP_CHANNEL	00000800
CTDRV_CLOSE	00001000
CTDRV_FORMAT	00002000
CTDRV_STATS	00004000
CTDRV_DEBUG	00008000
CTDRV_INFO	00010000
CTDRV_STATUS_UNIT	00020000
CTDRV_INIT_CARD	00040000
CTDRV_UPDATE_INFO	00080000
CTDRV_UI_READ	00100000
CTDRV_UI_WRITE	00200000
CTDRV_EXIT	00400000
CTDRV_UNITACTIVATES	01000000
CTDRV_SUBSCRIPTIONS	02000000
CTDRV_EVENTUPDATES	04000000

Command	Bit Position
CTDRV_STATUS_DISCONNECT	40000000

**Default Value:**

FFFFFFFF

The <mask> used to include only the CTDRV\_OPEN, CTDRV\_INIT\_UNIT and CTDRV\_READ commands would be:  
 [Debug]DriverTraceMask=0000001A

---

**Note:** You may want to exclude the CPU function call, as this happens often. Do this by setting a mask of:  
 [Debug]DriverTraceMask=7ffffeff

---

**Examples**

```
-> 07f96fdc Cmd: 03 CTDRV_INIT_UNIT ,MOCKOPC, Port: PORT1, Unit: IODev1
(UR0,N1)
| 07f96fdc UnitType: 0 (0x0), UnitAddr: 0 (0x0), BitWidth: 1, UnitCount: 16
<-07f96fdc Cmd: 03 CTDRV_INIT_UNIT ErrDriver 23 (0x17) and took 0ms
-> 07ff1584 Cmd: 04 CTDRV_READ ,DISKDRV, Port: DISKDRV, Unit: IODevDisk
(UR1,N2)
| 07ff1584 UnitType: 1 (0x1), UnitAddr: 1 (0x1), BitWidth: 16, UnitCount: 5,
RawType: INTEGER
<+07ff1584 Cmd: 04 CTDRV_READ ErrDriver 0 (0x0) and took 0ms
<+07ff1584 UnitType: 1 (0x1), UnitAddr: 1 (0x1), BitWidth: 16, UnitCount: 5,
RawType: INTEGER
<+07ff1584 000: 1 0 0 0 5
~> 00000000 Cmd: 25 CTDRV_SUBSCRIBE ,MOCKOPC, Port: PORT1, Unit: IODev1
(UR0,N1)
>-00000000 SUBSCRIBE Tag=Tag4 UpdateRate=500ms
~< 000003e9 Cmd: 26 CTDRV_EVENTUPDATES,MOCKOPC, Port: PORT1, Unit: IODev1
(UR0,N1)
<-000003e9 DS Event UPDATE_MODE=ALL Tag=Tag4 RawValue=0x00
Timestamp=<time_not_set> RawQual=0x0000 DSError=0 (0x0)
```

Explanation of arrows used above:

-> Flags information going down to the driver

<- Flags information back from the driver straight away

<+ Flags information back from the driver asynchronously (i.e. after some small time delay)

~> Flags subscription based calls into the driver

>-

~< Flags subscription based updated from the driver

<-

**(URx,Ny) addition to UNIT and Data driver traces**

UR stands for Unit Record number and is 0 based, so a value of 3 would be the 4th device in the IODevice settings. N stands for the (Network) Number in the IODevice settings. The use of this is to distinguish between redundant units which may use the same unit name. Thus, the trace will positively confirm which unit is in use.

## See Also

[Debug Parameters](#)  
[\[Debug\]DriverTrace](#)  
[\[Debug\]DriverTracePort](#)  
[\[Debug\]DriverTraceElements](#)

### [Debug]DriverTraceElements

#### **WARNING**

##### **UNINTENDED EQUIPMENT OPERATION**

Do not change the [Debug]DriverTraceElements parameter in the Citect.ini file, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Defines the maximum number of data buffer elements to be displayed in a driver trace. This parameter can be used to limit the printouts for drivers with large blocking values. For example if 128 are read, and parameter is set to 10 then 10 are displayed. If 5 are being read, then only 5 are displayed.

## Allowable Values:

1 to 256

## Default Value:

256

## See Also

[Debug Parameters](#)  
[\[Debug\]DriverTrace](#)  
[\[Debug\]DriverTraceMask](#)

### [Debug]DrWatson

Starts DrWatson when Plant SCADA starts running (if DrWatson is not already running). Note that DrWatson is not supplied with some versions of Windows.

## Allowable Values:

- 0 - (Disable)
- 1 - (Enable)

**Default Value:**

0

**See Also**[Debug Parameters](#)**[Debug]EnableHardwareAlarmLogging**

Enables or disables logging of stack traces for hardware alarms to syslog. Use this parameter to identify the source of hardware alarms in your Cicode.

**Allowable Values:**

- 0 - Disables logging of stack traces for hardware alarms
- 1 - Enables logging of stack traces for unsuppressed hardware alarms. The Cicode function ErrSet() can be used to suppress hardware alarms.
- 2 - Enables logging of stack traces for all hardware alarms, even if they are suppressed.

**Default Value:**

0

When the value of this parameter is set to 1, the stack trace is displayed as follows:

Date and Time of hardware alarm	Stack frame number	Cicode function with the hardware alarm	Approximate line number in the Cicode file
<i>A (simulated mode 1) hardware alarm looks like this;</i>			
2018-04-26 11:36:02.994	+10:00	Hardware alarm code 274 Page="AssInfo" Desc="TabSuspendUpdate" Code="Navigation_ShowTargetPage>" hPage=6 "Info_Bank_HD1080" hAn=-1	
2018-04-26 11:36:02.994	+10:00	[01] Navigation_ShowTargetPage["","","","",""];	299
2018-04-26 11:36:02.994	+10:00	[02] Workspace_SetContext["","","1"];	468
2018-04-26 11:36:02.995	+10:00	[03] Workspace_SelectEquipment[1];	429
2018-04-26 11:36:02.995	+10:00	[04] _Workspace_ContextChange[""];	1159
2018-04-26 11:36:02.995	+10:00	[05] _Workspace_DisplayAllocatedPages@Workspace@SA_Include(6, 5, "", "");	2107
2018-04-26 11:36:02.995	+10:00	[06] _Workspace_RefreshPanes@Workspace@SA_Include(6, 0, "", "");	2409
2018-04-26 11:36:02.995	+10:00	[07] _Workspace_RefreshPanes@Workspace@SA_Include(6, 0, "", "");	2395
2018-04-26 11:36:02.995	+10:00	[08] _Workspace_ShowContent@Workspace@SA_Include(0, "Info_Bank_HD1080", "Info", "", "", 1);	3267
2018-04-26 11:36:02.995	+10:00	[09] _Workspace_PlaceContent@Workspace@SA_Include(0, "Info_Bank_HD1080", "Info", "", "", 1, 1);	3542
2018-04-26 11:36:02.995	+10:00	[10] AssInfo["TabSuspendUpdate", "0", -1]	
<i>Function call that triggered the hardware alarm indicated by "</i>			

When the value of this parameter is set to 2, the trace includes the word "SUPPRESSED" as shown below:

2018-04-26 11:36:02.566	+10:00	SUPPRESSED hardware alarm code 274 Page="" Desc="" Code="Workspace_InitPanes" hPage=0 "Master_PageMenu1_HD1080" hAn=-1	
2018-04-26 11:36:02.567	+10:00	[01] _Workspace_InitPanes[0, 1];	2817
2018-04-26 11:36:02.567	+10:00	[02] _Workspace_StrToDisplayMode@Workspace@SA_Include("MaintainSize");	4563
2018-04-26 11:36:02.567	+10:00	[03] StrToInt("MaintainSize")	

**See Also**[Debug Parameters](#)

## [Debug]EnableLogging

Enables or disables logging to the tracelog.dat file.

### Allowable Values:

- 0 - Disables logging to tracelog.dat
- 1 - Enables logging to tracelog.dat

### Default Value:

1

---

**Note:** You can adjust this parameter during runtime using the SetLogging() function. For more information, search the Plant SCADA Help for the topic *Adjusting Logging During Runtime*.

---

### See Also

[Debug Parameters](#)

## [Debug]EnablePSICounters

Enables or disables the following performance counters:

- Citect.Platform.PSI.Client
- Citect.Platform.PSI.Server.

When enabled, the results will appear in the Kernel's Page Table (see [Page Table PerformanceCounter](#) in the main Plant SCADA documentation).

---

**Note:** It is recommended that you only enable this parameter when instructed by Technical Support.

---

This parameter can be changed without having to restart the SCADA process. The counter settings are dynamically read every five seconds.

---

**Note:** If you remove this parameter from your Citect.ini file during runtime, it will retain the last setting. For example, if the parameter is set to enabled when you remove it, the performance counters will remain enabled. The setting will not be updated until a server restart occurs.

---

### Allowable Values:

- 0 - Disables the performance counters
- 1 - Enables the performance counters

### Default Value:

0

## See Also

[Debug Parameters](#)

### [Debug]EnableReloadLogging

Enables or disables additional debugging output for server reloads.

When enabled, a server reload will write additional log entries to syslog.dat that identify any added, deleted and modified records. The configuration value is read at the beginning of each server reload.

#### Allowable Values:

- 0 - Disables the reload debugging output
- 1 - Enables the reload debugging output

#### Default Value:

0

## See Also

[Debug Parameters](#)

### [Debug]EnableTaskFrameworkCounters

Enables or disables the following performance counters:

- Citect.Platform.Tasks.TaskManager
- Citect.Platform.Tasks.TaskQueue
- Citect.Platform.Tasks.TaskThread.

When enabled, the results will appear in the Kernel's Page Table (see [Page Table PerformanceCounter](#) in the main Plant SCADA documentation).

---

**Note:** It is recommended that you only enable this parameter when instructed by Technical Support.

This parameter can be changed without having to restart the SCADA process. The counter settings are dynamically read every five seconds.

---

**Note:** If you remove this parameter from your Citect.ini file during runtime, it will retain the last setting. For example, if the parameter is set to enabled when you remove it, the counters will remain enabled. The setting will not be updated until a server restart occurs.

#### Allowable Values:

- 0 - Disables the performance counters
- 1 - Enables the performance counters

**Default Value:**

0

**See Also**[Debug Parameters](#)**[Debug]EnableTransportCounters**

Enables or disables the following performance counters:

- Citect.Platform.Transport.SessionTransport
- Citect.Platform.Transport.TransportHost.

When enabled, the results will appear in the Kernel's Page Table (see [Page Table PerformanceCounter](#) in the main Plant SCADA documentation).

---

**Note:** It is recommended that you only enable this parameter when instructed by Technical Support.

This parameter can be changed without having to restart the SCADA process. The counter settings are dynamically read every five seconds.

---

**Note:** If you remove this parameter from your Citect.ini file during runtime, it will retain the last setting. For example, if the parameter is set to enabled when you remove it, the transport counters will remain enabled. The setting will not be updated until a server restart occurs.

**Allowable Values:**

- 0 - Disables the performance counters
- 1 - Enables the performance counters

**Default Value:**

0

**See Also**[Debug Parameters](#)**[Debug]FlushPeriod**

Sets the period at which Tracelog file entries are to be flushed to the file system.

**Allowable Values:**

Number of seconds (minimum of 1 second)

**Default Value:**

1

**See Also**[Debug Parameters](#)**[Debug]LogDirect**

Indicates whether to log directly to a log file or via the Kernel logging system.

**Allowable Values:**

- 0 - (log via the Kernel logging system)
- 1 - (log directly to the syslog file)

**Default Value:**

0

**See Also**[Debug Parameters](#)**[Debug]LogsDriveMinimumFreeSpace**

Specifies the minimum amount of free space required on the drive that hosts the log directory. If less than this amount of space is available on the drive, then a hardware alarm will be raised, as if a mini-dump does occur, there may be insufficient space to save it.

**Allowable Values:**

0 – 2147483647

- 0 - disable checking of the free space on the drive where the log directory resides.
- $n$  - the amount of available space (in Mb) required on the log directory drive to avoid a hardware alarm being raised.

**Default Value:**

2000 (Mb)

## See Also

[Debug Parameters](#)

### [Debug]LogShutdown

---

**Note:** This parameter is designed to assist qualified technical support personnel with system diagnosis.

---

Logs the shutdown function addresses to syslog.dat.

With default handling, a non-recoverable error can be investigated by determining what was happening at the time logging stopped. If a recoverable error is detected, it will be indicated by a trace marker between shutdown function addresses.

You may be requested to provide this information to Technical Support.

#### Allowable Values:

- 0 - (Disable)
- 1 - (Enable)

#### Default Value:

0

---

**Note:** You can adjust this parameter during runtime using the SetLogging() function. For more information, search the Plant SCADA Help for the topic *Adjusting Logging During Runtime*.

---

## See Also

[Debug Parameters](#)

### [Debug]MaximumFileSize

Sets the maximum size for the tracelog.dat log file, in kilobytes. This works in tandem with [\[Debug\]ArchiveFiles](#) to maintain an ongoing archiving strategy.

#### Allowable Values:

Maximum log file size in kilobytes

#### Default Value:

2048

## See Also

[Debug Parameters](#)

## [Debug]MaxMiniDumps

Sets the maximum number of mini-dump files that are stored in the log directory.

This impacts the following zip files:

- Citect32Exception\_[timestamp].zip (32-bit process)
- CitectException\_[timestamp].zip (64-bit process)

For more information, see *Configure User.dmp* in [The Crash Handler](#).

If an exception occurs while the maximum number of mini-dump files is stored, the oldest file will be deleted to make space.

### Allowable Values:

0 – 32767

- 0 - no mini-dumps are ever deleted
- $n$  - the maximum number of mini dumps that can be stored

### Default Value:

5

### See Also

[\[Debug\]MiniDumpType](#)

[Debug Parameters](#)

## [Debug]MaxTableOutputLength

When set will limit the length of tables when they are output to a kernel.dat file.

### Allowable Values:

- 0 – 32000
- -1 =unlimited

### Default Value:

200

### See Also

[Debug Parameters](#)

## [Debug]Menu

Determines whether the **Kernel** option is displayed on the control menu of the Plant SCADA runtime system. The Kernel option provides access to the Kernel window when Plant SCADA is running.

**Note:** This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

Be aware of the following:

- You should be experienced with Plant SCADA and Cicode before attempting to use the Kernel. These facilities are powerful and, if used incorrectly, can corrupt your system.
- You should only use the Kernel for diagnostics and debugging purposes, and not for normal Plant SCADA operation.
- You need to restrict access to the Kernel. When you are in the Kernel, you can execute any Cicode function with **no** privilege restrictions. You therefore have total control of Plant SCADA (and subsequently your plant and equipment).



### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Do not change the [Debug]Menu parameter in the Citect.ini file, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Allowable Values:

- 0 - (Do not display the Kernel option)
- 1 - (Display Kernel option)

## Default Value:

0

## See Also

[Debug Parameters](#)

## [Debug]MiniDumpType

Defines the type of mini-dump that is performed to capture the state of the process when an unexpected system shutdown occurs.

For more information, see [Configure User.dmp](#) in [The Crash Handler](#).

## Allowable Values:

- 0 - No mini-dump is created when a crash occurs.
  - 1 - A light mini-dump is created (containing local stack memory for unmanaged code).
  - 2 - A medium mini-dump is created (containing local stack memory, global variables and basic thread information for unmanaged code).
- 
- Note:** This type of mini-dump is not supported when running a 64-bit process, such as an alarm server operating in Extended Memory mode. If this setting is applied to a 64-bit process, a light mini-dump will be generated.
- 3 - A heavy mini-dump is created (containing accessible memory from the process, and thread information for unmanaged and managed coding environments).

## Default Value:

3

## See Also

[\[Debug\]MaxMiniDumps](#)

[Debug Parameters](#)

## [Debug]Priority

Allows you to filter messages logged to the tracelog.dat file according to their priority. Only messages with a priority less than or equal to this value will be logged.



### UNINTENDED EQUIPMENT OPERATION

More messages will be logged as the value for this parameter is increased. This may have the potential to seriously degrade performance of the system. This can render a system unresponsive to user input and delay the viewing of data.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Allowable Values:

Value	Description	Performance impact
0	Normal	Will have little impact on system performance (~1 message/sec)
10	Brief	May have some impact on system performance (~10 messages/sec)

Value	Description	Performance impact
20	Information	Allows a system to run, with some impact on performance (~100 messages/sec)
30	Verbose	Is unlikely to allow a system to run efficiently as the logging load may approach 100 percent of CPU usage.
40	Chatty	System will only operate under debug conditions (~100,000 messages/sec). Users may apply intermediate levels for finer control.

**Default Value:**

0

**Note:** You can adjust this parameter during runtime using the SetLogging() function. For more information, search the Plant SCADA Help for the topic *Adjusting Logging During Runtime*.

**See Also**

[Debug Parameters](#)

**[Debug]SeverityFilter**

Allows you to filter messages logged to the tracelog.dat file according to their severity.

The [\[Debug\]SeverityFilterMode](#) parameter indicates whether messages of the specified severities should be included or excluded from the log.

**Allowable Values:**

Zero or more of the following categories specified as a comma-separated string:

- **Critical** - any detected non-recoverable errors
- **Error** - any detected recoverable errors
- **Warning** - any detected minor issues
- **Information** - informative notifications
- **Verbose** - debugging trace

Include all the required values in the comma-separated string (including Critical and Error), as more critical error levels will not be automatically included if only a lower severity error category is specified.

**Default Value:**

Critical,Error

**Note:** You can adjust this parameter during runtime using the SetLogging() function. For more information, search the Plant SCADA Help for the topic *Adjusting Logging During Runtime*.

**See Also**

[Debug Parameters](#)

**[Debug]SeverityFilterMode**

Enables or disables logging of severities declared by the [\[Debug\]SeverityFilter](#).

**Allowable Values:**

- 0 - Log only categories defined by the SeverityFilter value
- 1 - Allow logging of all categories except the ones defined by the SeverityFilter value.

**Default Value:**

0

**Note:** You can adjust this parameter during runtime using the SetLogging() function. For more information, search the Plant SCADA Help for the topic *Adjusting Logging During Runtime*.

**See Also**

[Debug Parameters](#)

**[Debug]Shutdown**

Determines where the **Shutdown** option is displayed in the Plant SCADA runtime system. The Shutdown option allows users to shut down the Plant SCADA system from a graphics page. It can be set to display on the main control menu (the parent graphics page), or only on graphics pages accessed from the control menu (child graphics pages). You can also set the Shutdown option to not display at all.

**Note:**

- When this setting is set to 0, shutdown will also be disabled in the Plant SCADA Runtime Manager.
- This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

**Allowable Values:**

- 0 - (The Shutdown option is not displayed)
- 1 - (The Shutdown option is displayed on the control menu)
- 2 - (The Shutdown option is displayed only on child graphics pages)

**Default Value:**

1

**See Also**[Debug Parameters](#)**[Debug]ShutDownProtect**

Enables protection to veto a non-recoverable error if it occurs during shutdown.

**UNINTENDED EQUIPMENT OPERATION**

Do not change the [Debug]ShutDownProtect parameter in the Citect.ini file, except on the advice of Technical Support.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Allowable Values:**

- 0 - (Disable)
- 1 - (Enable)

**Default Value:**

0

**See Also**[Debug Parameters](#)**[Debug]SnapshotEnable**

Controls whether periodic diagnostic snapshots for the alarms database are created under:

[Logs]\DBLog.Alarm.<cluster>.<server>\DBSnapshot\_<nnn>.log.

**Allowable Values:**

- 1 = On
- 0 = Off

**Default Value:**

0

**See Also**[Debug Parameters](#)**[Debug]SnapshotInterval**

Sets the number of minutes between periodic alarm database snapshots.

**Allowable Values:**

1 - 1440

**Default Value:**

5

**See Also**[Debug Parameters](#)**[Debug]SnapshotMaxFiles**

Controls the maximum number of alarm database snapshot files to keep for current run of the server.

A rolling log system is used, starting with “DBSnapshot\_001.log”. When this file reaches the maximum size (as specified by [\[Debug\]SnapshotMaxSize](#)), it will be closed and “DBSnapshot\_002.log” will be opened, and so on.

This parameter determines how many of these files will be retained. When the maximum number is reached, the earliest file will be deleted.

For example, when this parameter is set to 5 and DBSnapshot\_005.log reaches its maximum size, then DBSnapshot\_001.log will be deleted and DBSnapshot\_006.log will be opened.

**Allowable Values**

1-50

**Default Value**

2

## See Also

[Debug Parameters](#)

### [Debug]SnapshotMaxSize

Controls the maximum size (in kilobytes) of an alarm database snapshot.

A rolling log system is used for snapshots, starting with “DBSnapshot\_001.log”. When this file reaches the specified maximum size, it will be closed and “DBSnapshot\_002.log” will be opened, and so on.

#### Allowable Values:

10240 (10 MB) - 204800 (200 MB)

#### Default Value:

51200 (50 MB)

## See Also

[Debug Parameters](#)

### [Debug]SnapshotOldFiles

Controls the maximum number of runs to keep for alarm database snapshot files.

When an alarm server is started, any old files from a previous run will be renamed. For example:

“DBSnapshot\_xxx.log” will be renamed to “DBSnapshot\_xxx.log\_2”, “DBSnapshot\_xxx.log\_2” will be renamed to “DBSnapshot\_xxx.log\_3”, and so on.

The oldest run will be deleted when the specified maximum is reached.

For example, a setting of 1 will keep one set of old files (the current files and “DBSnapshot\_xxx.log\_2”).

#### Allowable Values:

0-10

#### Default Value:

1

## See Also

[Debug Parameters](#)

## [Debug]SysErrDsp

Enables display of a popup message when a system error occurs. Setting this to 1 will result in the system being paused (stop any communication to clients) whilst the error is displayed. When this is set to 0 and a system error occurs, a hardware alarm will be raised and the error will be logged to the syslog.dat. Users should only enable this on a temporary basis for the purposes of debugging.

### Allowable Values:

- 0 - (No popups are displayed)
- 1 - (Allow popups are displayed)

### Default Value:

0

### See Also

[Debug Parameters](#)

## [Debug]SysLogArchive

Use this parameter to archive the syslog.dat files. The files roll over with timestamp and facilitate the retrieval of logged data from a historical archive.

The time taken for a syslog to fill and roll over depends on the amount of logging being recorded, and the current setting for [\[Debug\]SysLogSize](#).

### Allowable Values:

- 0 (disable syslog.dat archiving)
- 1 (enable syslog.dat archiving)

### Default Value:

0

---

**Note:** Consider the amount of available disk space before you start archiving log files, as the disk space required to store traces may be significant. A long term archiving strategy is recommended to avoid running a system close to its limitations.

The following format is used for the file name:

`syslog.[server][timestamp].dat`

where:

`[server] = <type>.<cluster>.<username>`

`[timestamp] = _YYMMDD_HHMMSS`

For example:

syslog.Alarm.Cluster1.AlarmServer1\_051231\_235959.dat

The files are located in the following directory:

%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Logs

---

**Note:** You can update this parameter within the Citect.ini file while online, as the runtime system will read its value periodically or on demand. For more information, see the topic *Adjusting Logging During Runtime*.

---

## See Also

[Debug Parameters](#)

[\[Debug\]SysLogSize](#)

### [Debug]SysLogSize

Sets the maximum size of the syslog.dat file. When the file reaches this size, it is rolled over.

The syslog.dat file is a low-level logging file used by Plant SCADA. This file is in the Windows directory. Plant SCADA logs debugging information to this file. You should check this file occasionally to see if Plant SCADA has detected any issues.

#### Allowable Values:

0 to 2,000,000 (kilobytes) (i.e. maximum size is 2GB)

#### Default Value:

100000

---

**Note:** You can adjust this parameter during runtime using the SetLogging() function. For more information, search the Plant SCADA Help for the topic *Adjusting Logging During Runtime*.

---

## See Also

[Debug Parameters](#)

[\[Debug\]SysLogArchive](#)

### [Debug]UserTraceMode

Sets the level of tracing for user Cicode. Allows you to filter messages logged to the **syslog.dat** according to their trace level when using the Situational Awareness project. Only messages with a trace level less than or equal to this value will be logged.

#### Allowable Values:

- 0: Nothing
- 1: Error
- 2: Warning

- 3: Information

**Default Value:**

2

**Note:** You can adjust this parameter during runtime using the Debug\_SetTraceLevel() function.

**See Also**[Debug Parameters](#)**Deployment Parameters**

The Citect.ini file contains the following Deployment parameters:

- [\[Deployment\]AskRestartArgs](#) — Passes arguments to the Cicode function called by [\[Deployment\]AskRestartFunc](#).
- [\[Deployment\]AskRestartFunc](#) — Calls a Cicode function instead of displaying a restart notification dialog when a prompted deployment occurs.
- [\[Deployment\]Enabled](#) — Determines if Runtime Manager runs a project that has been deployed from the deployment server, or the Active Project.

**See Also**[Parameter Categories](#)**[Deployment]AskRestartArgs**

This parameter allows you to pass arguments to the Cicode function called by the parameter [\[Deployment\]AskRestartFunc](#).

The arguments need to be defined as a set of comma-separated values, with each argument enclosed in double quote marks. For example:

```
AskRestartArgs = "Hello", "Hey"
```

If any of the arguments include a space, you need to place single quotes around all of the arguments:

```
AskRestartArgs = '"Hello There", "Hey"'  
AskRestartArgs = '"Hello There", "Hey Dude"'
```

**Allowable Values:**

A string defining a set of comma-separated values.

**Default Value:**

No values.

## See Also

[Deployment Parameters](#)

### [Deployment]AskRestartFunc

If "prompt" is specified as the update method for deployment, you can use this parameter to call a Cicode function instead of displaying the inbuilt restart notification dialog.

The Cicode function should return 0 if the restart is approved, or an error code if it is not.

You can use the parameter [\[Deployment\]AskRestartArgs](#) to pass arguments to the specified Cicode function.

## Allowable Values:

Any valid Cicode function.

## Default Value:

Display the inbuilt restart notification dialog.

## See Also

[Deployment Parameters](#)

### [Deployment]Enabled

Determines if Runtime Manager will run a project that has been deployed from the deployment server, or the Active Project (the project that is currently selected in Plant SCADA Studio).

If this parameter is enabled (1), the Runtime Manager will run the project located in the directory specified by the parameter [\[CtEdit\]Deploy](#). This is where unpacked project files are stored after a deployment package has been received from the deployment server.

If disabled (0), the Runtime Manager will run the Active Project.

---

**Note:** This parameter will be set to 1 (enabled) when you use the Configurator to set up a computer as a deployment client.

---

## Allowable Values:

- 0 - Disabled (run the Active Project).
- 1 - Enabled (run the project located in the directory specified by [\[CtEdit\]Deploy](#))

## Default Value:

0

## See Also

[Deployment Parameters](#)

## Device Parameters

The Citect.ini file contains the following device parameters:

- [\[Device\]AlwaysCreateHistory](#) - Determines whether history files will be created even if no alarms have been logged.
- [\[Device>CreateHistoryFiles](#) - Determines whether history files will be created for devices that don't have time and period specified.
- [\[Device\]DiscardSpoolOnError](#) - Discards or retains data in the spooling system.
- [\[Device\]ForceHistoryOpenMode](#) - Determines whether device history files are maintained for data logged by keyboard commands.
- [\[Device\]FormLength](#) - The number of lines that are printed on a page before a new page (form feed) is selected.
- [\[Device\]LptDosANSIToOEM](#) - Indicates whether Windows ANSI characters are converted to OEM characters when printing alarms.
- [\[Device\]MaxSpool](#) - The maximum number of records that Plant SCADA retains in the spool buffer for logging devices (before the log record is discarded).
- [\[Device\]SQLSelect](#) - Performs a SQL Select query when a SQL device is opened.
- [\[Device\]WatchTime](#) - The frequency (in milliseconds) that Plant SCADA checks devices for history files and flushes logging data to disk.

## See Also

[Parameter Categories](#)

### [Device]AlwaysCreateHistory

Before enabling this INI parameter, if you are running multi-process you will need to modify the system include project and add the 'Process' field for all devices. Enter the process type relevant for your usage of that device. Additionally, if a nominated process can run on multiple clusters on the same machine, you will need to add the 'Cluster Name' field for that device, otherwise the process may not startup and you will receive an error in the Runtime Manager.

Determines whether history files will be created even if no alarms have been logged. The normal operation of Plant SCADA is that a history file will not be produced if no alarms were logged for that period. Setting this parameter to 1 creates an empty history file in these cases.

### Allowable Values:

- 0 - (History files not created)
- 1 - (History files created)

**Default Value:**

0

**See Also**[Device Parameters](#)**[Device]CreateHistoryFiles**

Determines whether history files will be created for devices that don't have time and period specified.

**Allowable Values:**

- 0 - (History files not created)
- 1 - (History files created)

**Default Value:**

1

**See Also**[Device Parameters](#)**[Device]DiscardSpoolOnError**

Discards or retains data in the spooling system. Normally, if a device becomes inoperative, the data in the spooling system is discarded. By disabling this option, the data is retained.

**Allowable Values:**

- 0 - (Do not discard data)
- 1 - (Discard data)

**Default Value:**

1

**See Also**[Device Parameters](#)

**[Device]ForceHistoryOpenMode**

Determines whether device history files are maintained for data logged by keyboard commands. By disabling this option, logged data associated with keyboard commands will be appended to the current device file, irrespective of the history setup of your device file.

**Allowable Values:**

- 0 - (Device history files not maintained)
- 1 - (Device history files maintained)

**Default Value:**

1

**See Also**

[Device Parameters](#)

**[Device]FormLength**

The number of lines that are printed on a page before a new page (form feed) is selected. This parameter only applies to a printer device.

**Allowable Values:**

0 to 512 (lines)

**Default Value:**

60

**See Also**

[Device Parameters](#)

**[Device]LptDosANSIToOEM**

Indicates whether Windows ANSI characters are converted to OEM characters when printing alarms and reports to LPTx.DOS. The conversion is required due to differences between Windows and DOS character sets.

**Allowable Values:**

- 0 - (Characters are not converted)
- 1 - (Characters are converted)

**Default Value:**

1

**See Also**

[Device Parameters](#)

**[Device]MaxSpool**

The maximum number of records that Plant SCADA retains in the spool buffer for logging devices (before the log record is discarded).

**Allowable Values:**

0 (Keep unlimited) to 32000

**Default Value:**

4000

**See Also**

[Device Parameters](#)

**[Device]SQLSelect**

Performs an SQL Select query when an SQL device is opened.

**Allowable Values:**

- 0 - (Perform a query to retrieve only table headers when an SQL device is opened)
- 1 - (Perform a query to retrieve table headers and table data when an SQL device is opened)

**Default Value:**

1

**See Also**

[Device Parameters](#)

## [Device]WatchTime

The frequency (in milliseconds) that Plant SCADA checks devices for history files and flushes logging data to disk.

### Allowable Values:

1000 to 3600000 (milliseconds)

### Default Value:

5000

### See Also

[Device Parameters](#)

## Dial Parameters

The Citect.ini file contains the following dial parameters:

- [\[Dial\]CacheRefresh](#) - Enables or disables automatic cache refresh on connection to a remote I/O device.
- [\[Dial\]CallerIDTimeout](#) - The maximum time the I/O Server will wait for a valid caller ID during dial in.
- [\[Dial\]ConnectionRetries](#) - For PSTN connected devices - the number of times the I/O Server will try to dial a dial-up I/O device before declaring that it is OFFLINE. For non-PSTN connected Devices - the number of times the I/O Server will try to bring a device online before declaring that it is offline.
- [\[Dial\]Debug](#) - Enables/disables scheduled I/O device debugging.
- [\[Dial\]DebugLevel](#) - Specifies the level of dial debug traces.
- [\[Dial\]DefaultCallerID](#) - Specifies the caller id to use when taking dial in calls where [Dial]CallerIDTimeout has expired.
- [\[Dial\]MaxConnectionTime](#) - The maximum time a dial-up I/O device will stay connected (not including persistent connections).
- [\[Dial\]MaxQueueTime](#) - The maximum time a dial-up I/O device will remain in the dial queue.
- [\[Dial\]MaxTimeAfterDisconnect](#) - The maximum number of seconds to wait following a disconnect request to allow the dial system time to finish servicing requests.
- [\[Dial\]MissedScheduleTolerance](#) - Specifies how many consecutive scheduled dial attempts can be missed.
- [\[Dial\]ModemRetryDelay](#) - The minimum time to wait (in seconds) between disconnecting from a modem and attempting to use the modem again.
- [\[Dial\]PersistCache](#) - Prevents the cache for remote IO devices being persisted to disk after disconnection and at shutdown.
- [\[Dial\]ReadThroughCache](#) - Determines whether the I/O server will read through the data cache while it is connected to a remote I/O device.
- [\[Dial\]RingCount](#) - The number of rings the I/O Server will wait before it answers a call from a dialing I/O device.

- [\[Dial\]WatchTime](#) - The period at which the I/O Server checks for I/O devices to contact and checks for incoming calls from dialing I/O devices.

## See Also

[Parameter Categories](#)

### [Dial]CacheRefresh

Enables or disables automatic cache refresh on connection to a remote I/O device. By default, the I/O Server will automatically cache data for all variable tags defined for the device. Setting the [Dial]CacheRefresh parameter to 0 will turn off the automatic refresh.

#### Allowable Values:

- 0 to disable
- 1 to enable

#### Default Value:

1

## See Also

[Dial Parameters](#)

### [Dial]CallerIDTimeout

The maximum time the I/O Server will wait for a valid caller ID during dial in.

#### Allowable Values:

1 to 43200 (secs)

#### Default Value:

The value of MaxConnectionTime is converted to seconds and inserted here.

## See Also

[Dial Parameters](#)

### [Dial]ConnectionRetries

For PSTN connected devices - the number of times the I/O Server will try to dial a dial-up I/O device before

declaring that it is OFFLINE.

For non-PSTN connected devices - the number of times the I/O Server will try to bring a device online before declaring that it is OFFLINE.

---

**Note:** If the modem is busy, the attempt will not succeed. If more than one modem is configured on the I/O Server computer, a different one will be selected for the next attempt. If there is only one modem, the I/O Server will try it again. This process will continue until an available modem is found or until the number of retries has been reached.

---

## Allowable Values:

1 to 10

## Default Value:

2

## See Also

[Dial Parameters](#)

## [Dial]Debug

Enables or disables the logging of debug messages to syslog.dat for scheduled I/O devices.

The parameter determines the level of logging used for scheduled communication.

## Allowable Values:

- 0 - (Disable debugging)
- 1 - (Enable debugging)

## Default Value:

0

## See Also

[Dial Parameters](#)

## [Dial]DebugLevel

Allows dynamic control over the amount of dial debug traces sent to syslog.dat.

---

**Note:** [\[Dial\]Debug](#) needs to be set to 1 for this parameter to take effect.

---

**Allowable Values:**

- 0 - No logging performed
- 1 - Minimum amount of logging performed
- 2 - Standard amount of logging performed (default)
- 3 - Maximum logging (everything) useful for debugging

**Default Value:**

2

---

**Note:** You can update this parameter within the Citect.ini file while online, as the runtime system will read its value periodically or on demand. For more information, search the Plant SCADA Help for the topic *Adjusting Logging During Runtime*.

---

**See Also**[Dial Parameters](#)**[Dial]DefaultCallerID**

Specifies the caller id to use when taking dial in calls.

---

**Note:** This is only used if the amount of time specified by [\[Dial\]CallerIDTimeout](#) has been exceeded.

---

**Allowable Values:**

Any string

**Default Value:**

" "

**See Also**[Dial Parameters](#)**[Dial]MaxConnectionTime**

The maximum time a dial-up I/O device will stay connected (not including persistent connections).

**Allowable Values:**

1 to 720 (mins)

**Default Value:**

20 (mins)

**See Also**

[Dial Parameters](#)

**[Dial]MaxQueueTime**

The maximum time a dial-up I/O device will remain in the dial queue, if no modems (or physical ports in the case of non-PSTN connections) are available.

**Allowable Values:**

1 to 720 (mins).

**Default Value:**

The value of [\[Dial\]MaxConnectionTime](#) is inserted here.

**See Also**

[Dial Parameters](#)

**[Dial]MaxTimeAfterDisconnect**

The maximum number of seconds to wait following a disconnect request to allow the dial system time to finish servicing requests. This setting allows some time for the dial-up system to clear any outstanding requests, but will force the connection to close after the specified time even if a problem is preventing the queues being emptied.

**Allowable Values:**

1 to 86400 (seconds)

**Default Value:**

120 (seconds)

**See Also**

[Dial Parameters](#)

**[Dial]MissedScheduleTolerance**

Specifies how many consecutive scheduled dial attempts can be missed before the cache becomes stale.

**Allowable Values:**

1 to 10

**Default Value:**

2

**See Also**

[Dial Parameters](#)

**[Dial]ModemRetryDelay**

The minimum time to wait (in seconds) between disconnecting from a modem and attempting to use the modem again.

**Allowable Values:**

1 to 60

**Default Value:**

10

**See Also**

[Dial Parameters](#)

**[Dial]PersistCache**

Prevents the cache for remote IO devices being persisted to disk after disconnection and at shutdown. Default value is 1 - data is persisted to disk. Setting this to 0 stops persisting data to disk.

If no cache exists on startup, values for remote units will display #WAIT until dialed. If a cache file exists on startup, cached data will be used.

**Allowable Values:**

0 or 1

**Default Value:**

1

**See Also**

[Dial Parameters](#)

**[Dial]ReadThroughCache**

Determines whether the I/O Server will read through the data cache while it is connected to a remote I/O device.

**Allowable Values:**

0 or 1

**Default Value:**

0

**See Also**

[Dial Parameters](#)

**[Dial]RingCount**

The number of rings the I/O Server will wait before it answers a call from a dialing I/O device.

**Allowable Values:**

1 to 10 (rings)

**Default Value:**

3

**See Also**

[Dial Parameters](#)

**[Dial]WatchTime**

The period at which the I/O Server checks for I/O devices to contact and checks for incoming calls from dialing I/O devices. The WatchTime should be less than your modem's inactivity timeout (if you are using dial back),

otherwise the modem will hang up and Plant SCADA will miss the call.

### Allowable Values:

1 to 600 (seconds)

### Default Value:

20

### See Also

[Dial Parameters](#)

### DisableIO Parameters

The Citect.ini file contains the following DisableIO parameters:

- **[DisableIO]<DeviceName>** - Permanently disables the named I/O device for the current session.
- **[DisableIO]<ServerName>** - Permanently disables all I/O devices associated with the named I/O server for the current session.

### See Also

[Parameter Categories](#)

### [DisableIO]<DeviceName>

Permanently disables a named I/O device for the current session so that no hardware errors are generated.

This means that the "PLC Server I/O device off-line" hardware error message will not be seen for the device when the I/O Server is not available.

The effect of this setting is therefore similar to issuing the Cicode function **IODeviceControl** twice with Type set as 0 and 1 respectively, and with the value of the INI setting used as the 'Data' argument (enable or disable).

To enable this IO device, delete the settings from Citect.ini file or assign 0. If you want to comment out the setting, use the exclamation mark character ("!"), not a semi-colon (";"), otherwise Plant SCADA might hang on start-up.

---

**Note:** Parameters of [DisableIO] are not standard ones. On start-up, Plant SCADA tries to load all device names into a symbol table and use ";" to separate them. If ";" is used to comment out [DisableIO] parameters, it sends the initialization of the client system into a dead loop.

---

### Allowable Values:

<I/O device> = 0 or 1

## Default Value:

None

## Example

```
[DisableIO]  
IODevice01=1
```

IODevice01 is an I/O device in your project, and this setting disables IODevice01.

## See Also

[DisableIO Parameters](#)

### [DisableIO]<ServerName>

Permanently disables all I/O devices associated with the named I/O Server for the current session.

This means that the "PLC Server I/O device off-line" hardware error message will not be seen for these devices when the I/O server is not available.

This parameter is useful to bypass the connection to I/O servers on start-up when there are many I/O devices and some of them don't yet exist.

The effect of this setting is therefore similar to issuing the Cicode function **IODevControl** twice with Type set as 0 and 1 respectively, and with the value of the INI setting used as the 'Data' argument (enable or disable).

To enable this I/O server, delete the setting from Citect.ini file or assign 0. If you want to comment out the setting, use the exclamation mark character (!), not a semicolon (;), otherwise Plant SCADA may hang on start-up.

---

**Note:** Parameters of [DisableIO] are not standard ones. On start-up, Plant SCADA tries to load all device names into a symbol table and use ";" to separate them. If ";" is used to comment out [DisableIO] parameters, it will send initialization of the client system into a dead loop.

## Allowable Values:

<I/O server> = 0 or 1

## Default Value:

None

## Example

```
[DisableIO]  
IOServer03=1
```

where IOServer03 is an I/O server in your project. This setting will disable all I/O devices associated with IOServer03.

## See Also

[DisableIO Parameters](#)

### DiskDrv Parameters

The Citect.ini file contains the following DiskDrv parameters:

- [\[DiskDrv\]UpDateTime](#) - The update (write) time for a disk I/O device.

## See Also

[Parameter Categories](#)

### [DiskDrv]UpDateTime

The update (write) time for a Disk I/O device.

#### Allowable Values:

0 to unlimited (secs)

#### Default Value:

5

## See Also

[DiskDrv Parameters](#)

### Driver Parameters

The Citect.ini file contains the following generic driver parameters:

- [\[<DriverName>\]OverrideOSProtection](#) - Determines whether to override the protection mechanism built-in to the I/O Server for drivers that may not be compatible with newer Windows® versions.

## See Also

[Parameter Categories](#)

### [<DriverName>]OverrideOSProtection

Determines whether to override the protection mechanism built-in to the I/O server for drivers that may not be compatible with the current Windows® operating system (OS). You should only set this if you have conducted

your own testing of the driver on the current OS and are satisfied that its operation is correct and reliable.

**Allowable values:**

0, 1

**Default value:**

0

**See Also**

[Driver Parameters](#)

**Event Parameters**

The Citect.ini file contains the following event parameters:

- [Event]Alarm - Obsolete in version 7.0.
- [Event]IOServer - Obsolete in version 7.0.
- [\[Event\]InhibitEvent](#) - Inhibiting events to be enabled.
- [Event]Name - Obsolete in version 7.0.
- [Event]Report - Obsolete in version 7.0.
- [\[Event\]Server](#) - Determines whether this computer is an Events Server.
- [Event]Trend - Obsolete in version 7.0.
- [\[Event\]WatchTime](#) - The time (in seconds) between checks for due events.

**See Also**

[Parameter Categories](#)

**[Event]InhibitEvent**

Inhibits events on startup. For example, you might have an event that is triggered off the rising edge of a bit on startup. The Events Server notices the bit come on, and runs the event. If this parameter is enabled, the Events Server does not run this event until it has read the I/O devices a second time.

**Allowable Values:**

- 0 - (Do not inhibit)
- 1 - (Inhibit)

**Default Value:**

1

**See Also**[Event Parameters](#)**[Event]Server**

Determines whether this computer is an Events Server.

**Note:** This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

**Allowable Values:**

- 0 - (Not an Events Server)
- 1 - (Events Server)

**Default Value:**

0

**See Also**[Event Parameters](#)**[Event]WatchTime**

The time (in seconds) between checks for due events. You can set this parameter to a large value to conserve CPU time; however you should ensure that any Time and Period-based events in the project are a direct multiple of the WatchTime, otherwise they will not be checked.

**Allowable Values:**

1 to 36000 (seconds)

**Default Value:**

1

**See Also**[Event Parameters](#)

## Font Parameters

The Citect.ini file contains the following font parameters:

- **[Font]Echo** - The text font used to echo keyboard commands on the screen (if they are set to echo).
- **[Font]Error** - The text font used to display an error message at an animation point.
- **[Font]General** - The text font used for general text display (other than keyboard commands, error messages, and prompt messages).
- **[Font]Print** - The text font used to print data to the printer device.
- **[Font]Prompt** - The text font used to display a prompt message at an animation point.

## See Also

[Parameter Categories](#)

### [Font]Echo

The text font used to echo keyboard commands on the screen (if they are set to echo).

#### Allowable Values:

Any font that exists in the project.

#### Default Value:

DefaultFont

## See Also

[Font Parameters](#)

### [Font]Error

The text font used to display an error message at an animation point.

#### Allowable Values:

Any font that exists in the project.

#### Default Value:

DefaultFont

## See Also

[Font Parameters](#)

### [Font]General

The text font used for general text display (other than keyboard commands, error messages, and prompt messages).

#### Allowable Values:

Any font that exists in the project.

#### Default Value:

DefaultFont

## See Also

[Font Parameters](#)

### [Font]Print

The text font used to print data to the printer device.

#### Allowable Values:

Any font that exists in the project.

#### Default Value:

PrintFont

## See Also

[Font Parameters](#)

### [Font]Prompt

The text font used to display a prompt message at an animation point.

#### Allowable Values:

Any font that exists in the project.

## Default Value:

DefaultFont

## See Also

[Font Parameters](#)

## Format Parameters

The [Format] section contains the following parameters:

- [\[Format\]FormatName](#) - Define a display format under a specified name.

## See Also

[Parameter Categories](#)

### [Format]FormatName

Defines a display format for tabular data and identifies it with a specified name. This allows you to name a customized display format that you can then apply to a list.

The way the format name is applied depends on the Starter project from which a project was originally created.

## Allowable Values:

Any valid format string.

## Default Values:

- **Situational Awareness projects**

In a project based on the Situational Awareness the display format can be used for alarm, interlock and equipment reference lists.

### Alarm List Display Formats

The display format can be used for alarms lists such as those on alarms pages, the Information Zone on the Operator Dashboard, and the Top Alarms list.

---

**Note:** The syntax of the format string is the same as the standard alarm display format string, with the exception of the width attribute being interpreted as pixels instead of characters.

---

The default values are:

```
[Format]
Top5Alarm = {PriorityAndState,24}{OnDate,90}{OnTime,90}{Name,220}{Desc,300}{State,50}
```

This is the alarm list format for the Top 5 alarm pop-up box in the Header Bar.

```
[Format]
TopActiveAlarms_UHD4K =
```

```
{PriorityAndState,24}{Date,120}{Time,130}{Name,190}{Desc,180}{State,50}
```

This is the alarm list format for the Active Alarms Zone pane in a 4K resolution workspace.

```
[Format]
```

```
InfoAlarm_HD1080 = {PriorityandState,24}{OnTime,90}{Item,160}{Name,180}
```

This is the alarm list format for the Information Zone in 1080 resolution.

```
[Format]
```

```
InfoAlarm_UHD4K = {PriorityandState,24}{OnTime,130}{Item,300}{Name,328}
```

This is the alarm list format for the Information Zone in 4K resolution.

```
[Format]
```

```
Alarm =
```

```
{PriorityAndState,25}{OnDate,80}{OnTime,90}{Name,250}{State,40}{Cluster,70}{Equipment,220}{Item,160}{UserName,100}{Comment,250}{Category,60}
```

This is the alarm list format for the generic alarm page.

```
[Format]
```

```
Summary =
```

```
{PriorityAndState,25}{Name,250}{SumState,40}{OnDate,80}{OnTime,90}{OffDate,80}{OffTime,90}{DeltaTime,90}{Cluster,70}{Equipment,150}{Item,100}{Comment,250}{Category,60}
```

This is the alarm list format for the generic alarm summary page.

```
[Format]
```

```
SOE =
```

```
{PriorityAndState,25}{Date,80}{Time,90}{Message,250}{Name,250}{State,40}{Cluster,70}{Equipment,150}{Item,100}{UserName,100}{UserLocation,100}{Category,60}
```

This is the alarm list format for the generic sequence of events page.

```
[Format]
```

```
Disabled={PriorityAndState,25}{OnDate,80}{OnTime,90}{Name,250}{State,40}{Cluster,70}{Equipment,150}{Item,100}{DisableEndDate,80}{DisableEndTime,90}{DisableComment,250}{UserName,100}{UserLocation,100}{Category,60}
```

This is the alarm list format for the generic disabled alarms page.

```
[Format]
```

```
AlarmPageSOE_HD1080 =
```

```
{PriorityAndState,24}{Date,90}{Time,90}{Message,165}{State,50}{UserName,100}{UserLocation,100}
```

This is the alarm list format for the historical events side panel on the default alarm pages (in HD1080 resolution).

```
[Format]
```

```
AlarmPageSOE_UHD4K =
```

```
{PriorityAndState,24}{Date,120}{Time,130}{Message,325}{State,50}{UserName,100}{UserLocation,100}
```

This is the alarm list format for the historical events side panel on the default alarm pages (in UHD4K resolution).

**Note:** If you use the `AlarmListCreate()` function to create an alarms list and do not specify a format name, the "Alarm", "Summary", "SOE", or "Disabled" format will be used by default, based on the type of alarm list being created.

## Interlocks

The default values are:

```
[Format]Interlock_HD1080 = {Interlock,25}{Bypass,25}{Description,370}
```

This is the Interlock list format for the information zone in 1080HD format.

```
[Format]Interlock_UHD4K = {Interlock,25}{Bypass,25}{Description,720}
```

This is the Interlock list format for the information zone in UHD4K resolution.

### Equipment References

The default values are:

```
[Format]EquipRef_HD1080 = {Checked,24}{Equipment,240}{Category,120}
```

This is the Equipment Reference List format for alarm page in 1080 format.

```
[Format]EquipRef_UHD4K = {Checked,24}{Equipment,240}{Category,120}
```

This is the Equipment Reference List format for alarm page in UHD4K resolution.

- **Tab\_Style and StruxureWare projects**

**Note:** The syntax of the format string is the same as the standard alarm display format string, with the exception of the width attribute being interpreted as pixels instead of characters.

The Tab\_Style alarm templates assume the alarm display format is defined under the following names if no specified format name is specified for an alarm page:

- Alarm - for normal alarm display type except Summary and Hardware
- LastAlarm - for alarm banner
- Summary - for summary alarm display type
- SOE – for Sequence Of Event (SOE) alarm display type

The default values are:

```
[Format]
Alarm = {Date,80}{Time,80}{Tag,120}{Name,120}{Desc,120}
[Format]
LastAlarm = {Date,80}{Time,80}{Tag,120}{Name,120}{Desc,120}
[Format]
Summary = {Date,80}{Time,80}{Tag,120}{Name,120}{Desc,120}
[Format]
SOE = {Date,80}{Time,80}{Tag,120}{Name,120}{Desc,120}
```

This will show the alarm fields Date, Time, Tag, Name and Desc with each field taking up the specified pixel width on the active / disabled alarm pages.

You can assign a format name to an alarm page by opening the page in the Graphics Builder, then double-clicking on the alarm template and entering the name in the pop-up form.

Tab-style templates do not support text-justification operators because the templates already support proper column handling.

## See Also

[Alarm Parameters](#)

## General Parameters

The Citect.ini file contains the following general parameters:

- [\[General\]AdvancedDDEPoll](#) - The rate at which Plant SCADA polls I/O devices for data requested by another application (using DDE).
- [\[General\]AELManagerUrl](#) - This parameter stores the current URL for the AVEVA™ Enterprise License

Manager.

- [\[General\]BadOptimise](#) - Obsolete in version 7.0.
- [\[General\]BufferIO](#) - The size of the compiler's file buffer.
- [\[General\]CheckAddressBoundary](#) - Determines whether the compiler checks for correctly-aligned I/O device variables.
- [\[General\]ClusterReplication](#) - Controls whether tag or tag reference will be replicated in a multi-cluster system.
- [\[General\]CitectRunningCheck](#) - Obsolete in version 7.0.
- [\[General\]Ctl3D](#) - Determines whether dialog boxes in the runtime system display with a three-dimensional effect.
- [\[General\]DeadlockTimeout](#) - Specifies the number of seconds to wait before await on an event is considered timed-out.
- [\[General\]DeadlockWarning](#) - Specifies whether an error message is displayed if a time-out occurs.
- [\[General\]DebugInfo](#) - Determines whether symbolic information is included in the compiled Cicode. This controls whether Cicode can be debugged or not.
- [\[General\]DisablePlotSetupForm](#) - Determines whether the Plot Setup form displays when the TrnPrint() function is used.
- [\[General\]DisableRemoteBlocking](#) - Disables blocking on a remote I/O device.
- [\[General\]EnforceTagGlobalUniqueness](#) - Determines if the compiler generates warning messages for tag names that are not globally unique.
- [\[General\]FormatCheck](#) - Determines whether alarm values are truncated when they are longer than the field in which they are to display.
- [\[General\]InfoDestroy](#) - Determines whether the VARIABLE.DBF file is closed automatically after a call to the DspInfoField() function.
- [\[General\]LicensingFullLogging](#) - Enables licensing information to be logged.
- [\[General\]LicenseReservationTimeout](#) - Specifies the number of seconds to reserve a license for a given IP address in cases where a remote client connection is lost.
- [\[General\]LockDelay](#) - Sets the time delay in millisecond between file operation retries.
- [\[General\]LockRetry](#) - The number of times to retry an operation that does not succeed because a file is locked.
- [\[General\]LongFilename](#) - Enables long filename support for Plant SCADA projects, pages, and filenames.
- [\[General\]Multiprocess](#) - Detemines whether Plant SCADA runs as a multi-process or single-process application.
- [\[General\]OSErrors](#) - Enables/disables Plant SCADA trapping of Operating System errors.
- [\[General\]PasswordExpiry](#) - Determines how long your user and operator passwords will take to expire.
- [\[General\]PointCountHigh](#) - The percentage figure at which the first watermark message will be displayed.
- [\[General\]PointCountHighHigh](#) - The percentage figure at which the second watermark message will be displayed.
- [\[General\]PointLimitMsg](#) - Obsolete in version 7.0.
- [\[General\]PrinterColourMode](#) - Determines the default setting (black & white, or color) for trends printed using the TrnPrint() function.
- [\[General\]RegionalNumbersFormat](#) - Determines how Plant SCADA outputs data containing decimal numbers.

- [\[General\]ServerMonitoringPeriod](#) - Defines how often a Alarm Client will send an additional transaction to its server.
- [\[General\]ShareFiles](#) - Enables/disables the opening of projects by the compiler in Shared Mode.
- [\[General\]ShowDriverError](#) - Determines whether an error dialog displays when an internal driver error occurs.
- [\[General\]Stack](#) - The size of the Plant SCADA general-purpose stack.
- [\[General\]StartDelay](#) - Delays the startup of Plant SCADA for a specified period.
- [\[General\]SymbolicInfo](#) - Enables/disables AN help information.
- [\[General\]TagAssMode](#) - Obsolete in version 7.20. Refer to [\[General\]TagDB](#) instead.
- [\[General\]TagDB](#) - Determines whether the Variable Tags database is loaded when the project is compiled.
- [\[General\]TagDBReloadOnChange](#) - Determines whether the Variable Tags database is checked for changes and reloaded when a new page is displayed.
- [\[General\]TagStartDigit](#) - Allows you to use tags that have a number as the first digit.
- [\[General\]TagWriteEventFmt](#) - Allows the formatting of writes to the TagWriteEventQue.
- [\[General\]TagWriteEventQue](#) - Enables/disables logging of tags from IO Devices to a queue called TagWriteEventQue
- [\[General\]TrnPrinter](#) - The default printer port used in the TrnPrint() function.
- [\[General\]TypeRemapAllowed](#) - When set the Data Type field on the form is not editable when the variable tag is linked .
- [\[General\]Verbose](#) - Enables/disables the display of the startup dialog box, which shows the opening runtime databases and other information.
- [\[General\]VerboseToSysLog](#) - Enables startup messages displayed by the Kernel in "Verbose" mode to be logged in the SYSLOG.DAT file.
- [\[General\]WatermarkedPointCount](#) - A flag indicating if watermarking is to be displayed for proximity to point count limits

## See Also

[Parameter Categories](#)

### **[General]AdvancedDDEPoll**

The rate at which Plant SCADA polls I/O devices for data requested by another application (using DDE).

This parameter is only relevant to data accessed by the Plant SCADA DDE Server (Advanced DDE) using "hot" links. (These links are referred to as "automatic" links by some applications, such as Excel).

### **Allowable Values:**

1000 to 32000 ms

### **Default Value:**

2000

## See Also

[General Parameters](#)

### [General]AELManagerUrl

This parameter stores the current URL for the AVEVA™ Enterprise License Manager.

The URL is specified via the Configure Enterprise License Manager dialog. To access this dialog, go to Plant SCADA Studio's **Licensing** activity and select the **Configure** button on the Enterprise License Manager tile.

## Allowable Values:

A valid URL ending with "/AELicenseManager".

## Default Value:

`http://localhost/AELicenseManager`

## See Also

[General Parameters](#)

### [General]BufferIO

The size of the compiler's file buffer.

## Allowable Values:

0 to 32000

## Default Value:

4096

## See Also

[General Parameters](#)

### [General]CheckAddressBoundary

Determines whether the compiler checks for correctly-aligned I/O device variables. Each analog variable in an I/O device usually occupies a word location (16 bits wide). If the first word is V1, the next is V2, V3, V4, and so on. With some I/O devices, you can access two words as a long or real value (32 bits wide). The first long will be V1 and the next V3, V5, V7, and so on.

For Plant SCADA to read these variables correctly, all double-register variables must be aligned on the same

boundary, either an even or odd boundary. When Plant SCADA compiles your project and finds a double-register variable, it remembers which boundary it is on and checks that all other double register variables are on the same boundary. So, if Plant SCADA finds a double register variable on address V5, Plant SCADA checks that all other double register variables are also on odd boundaries.

The Plant SCADA compiler displays the "Address on bad boundary" message if the address of a long or real variable is not aligned correctly.

### Allowable Values:

- 0 - (Disable checking)
- 1 - (Enable checking and report as errors)
- 2 - (Enable checking and report as alerts)

### Default Value:

1

### See Also

[General Parameters](#)

### [General]ClusterReplication

Determines whether tags and tag references will be replicated in a multi-cluster system.

**Note:** After upgrading Plant SCADA, you will need to change this parameter to 1 if you want cluster replication to be enabled.

### Allowable Values:

- 0 — Disable replication.  
A cluster must be explicitly specified for a tag or tag reference, otherwise a compiler error message will be generated.
- 1 — Enable replication.  
A cluster need not be defined for a tag or tag reference for replication to occur. If the same tag exists in multiple clusters in the variables database, the cluster name will be replicated down to an I/O device via the I/O server to which it is connected.

### Default Value:

0

### See Also

[General Parameters](#)

**[General]Ctl3D**

Determines whether dialog boxes in the runtime system display with a three-dimensional effect.

**Allowable Values:**

- 0 - (No 3-D effect)
- 1 - (Show 3-D effect)

**Default Value:**

1

**See Also**

[General Parameters](#)

**[General]DeadlockTimeout**

Specifies the number of seconds to wait before a wait on an event is considered timed-out. Setting this parameter to 0 would mean that Plant SCADA will wait indefinitely and is not recommended. The value should not be changed from the default unless required for troubleshooting by technical support.

**Default Value:**

300

**See Also**

[General Parameters](#)

**[General]DeadlockWarning**

Specifies whether an error message is displayed if a time-out occurs.

**Allowable Values:**

- 0 - Do not display an error message.
- 1 - Display an error message.

**Default Value:**

1

## See Also

[General Parameters](#)

### [General]DebugInfo

Determines whether symbolic information is included in the compiled Cicode. Symbolic information is used to debug Cicode. The performance gained by turning this option off is only about 1%.

#### Allowable Values:

- 0 - (Do not include symbolic information)
- 1 - (Include symbolic information - allow debugging)

#### Default Value:

1

## See Also

[General Parameters](#)

### [General]DisablePlotSetupForm

The DisablePlotSetupForm parameter works in conjunction with the DisplayForm argument of the **TrnPrint()** function. It determines whether the Plot Setup form displays when TrnPrint() is called, but it will only take effect if TrnPrint() is called, and the DisplayForm argument is set to -1.

This parameter can be set from the Plot Setup form, by clicking in the Do not display this form next time check box.

#### Allowable Values:

- 0 - (Plot Setup form is enabled)
- 1 - (Plot Setup form is disabled)

#### Default Value:

0

## See Also

[General Parameters](#)

### [General]DisableRemoteBlocking

Disables the blocking of remote I/O devices.

To optimize data transfer, multiple requests to an I/O device are blocked into one single request. However, some blocked requests are not satisfied from the I/O Server data cache, resulting in variable tags being displayed as #WAIT.

To disable blocking and send client requests for data individually, set this parameter to 1.

#### Allowable Values:

- 0 - (Remote Blocking is enabled)
- 1 - (Remote Blocking is disabled)

#### Default Value:

0

#### See Also

[General Parameters](#)

### [General]EnforceTagGlobalUniqueness

The Connectivity Server manages tag subscriptions for the OPC UA Server and the Industrial Graphics Server. If a subscription request does not have a cluster explicitly defined, all the tags in the Plant SCADA system need to be unique across all clusters (in respect to both their tag name and equipment.item name) otherwise the subscription will be rejected.

When you compile a Plant SCADA project, the compiler provides a flag to the Connectivity Server indicating whether or not all tags are unique across all clusters. The Connectivity Server can refer to this flag when it encounters a tag subscription request that does not specify a cluster. If the flag indicates that all tags are globally unique, the Connectivity Server can safely broadcast a tag exists request to all clusters. If the flag indicates that this is not the case, any tag subscription requests that do not specify a cluster will be rejected.

When the [\[General\]EnforceTagGlobalUniqueness](#) parameter is enabled (1), the compiler will generate a warning message for each tag name that is not globally unique. These messages can be used to diagnose which tags are not unique across all clusters.

#### Allowable Values:

- 0 — the compiler will not generate warning messages for tags that are not unique across all clusters
- 1 — the compiler will generate warning messages for tags that are not unique across all clusters

#### Default Value:

0

## See Also

[General Parameters](#)

### [General]FormatCheck

Determines whether alarm values are truncated when they are longer than the field they are to display in (as specified in Alarm Categories). For example, if, in your Alarm Format, you allot 12 characters to the Analog Value field (i.e. {Value,12} ), the alarm page will display up to 12 characters of the value. If the value happens to be longer than 12 characters, it will be truncated or replaced with the #OVR ("overflow of format width") error.

#### Allowable Values:

- 0 - (#OVR string used to designate an "overflow of format width")
- 1 - (Numeric value is truncated to fit in field)

#### Default Value:

1

## See Also

[General Parameters](#)

### [General]InfoDestroy

Determines whether the VARIABLE.DBF file is closed automatically after a call to the DspInfoField() function. You can disable this parameter for optimization, but you must call the DspInfoDestroy() function to close the VARIABLE.DBF file. If the VARIABLE.DBF file is not closed, the Plant SCADA compiler will not be able to open VARIABLE.DBF file.

#### Allowable Values:

- 0 - (Disabled (VARIABLE.DBF is not closed))
- 1 - (VARIABLE.DBF is automatically closed)

#### Default Value:

1

## See Also

[General Parameters](#)

**[General]LicenseReservationTimeout**

Specifies the number of seconds to reserve a license for a given IP address in cases where a remote client connection is lost. This does not apply to FULL licenses.

**Default Value:**

10

**See Also**

[General Parameters](#)

**[General]LicensingFullLogging**

Enables licensing information to be logged. This parameter is set to false (0) by default to avoid the same licensing information being repeatedly logged.

**Allowable Values:**

- 0 - (Disable licensing logging)
- 1 - (Enable licensing logging)

**Default Value:**

0

**See Also**

[General Parameters](#)

**[General]LockDelay**

Sets the time delay in millisecond between file operation retries. This parameter is ignored if [\[General\]LockRetry](#) is set to 0. Note that if you set this parameter to a high value, you could reduce performance.

**Allowable Values:**

0 to 100

**Default Value:**

30

## See Also

[General Parameters](#)

### [General]LockRetry

The number of times to retry an operation that does not succeed because a file is locked. Some networks do not unlock files as quickly as you would expect, causing the error message "dBASE record locked by another user" to display frequently. Note that if you set this parameter to a high value, you could reduce performance. The period between retries is determined by the [\[General\]LockDelay](#) parameter.

#### Allowable Values:

0 to 20

#### Default Value:

3

## See Also

[General Parameters](#)

### [General]LongFilename

Enables long filename support. With this parameter set, Plant SCADA project names and page names can be up to 64 characters in length, and filenames can be up to 254 characters, including the path.

#### Allowable Values:

- 1 - (to enable)
- 0 - (to disable)

#### Default Value:

1

---

**Note:** If you disable long filename support, you must ensure that the path of the directory where Plant SCADA is installed uses short filenames and does not include any spaces. Long file names in the path will be invalid, and you will be unable to create or save projects to the directory.

---

## See Also

[General Parameters](#)

## [General]Multiprocess

Determines whether Plant SCADA will run as a multi process or single process application.

**Note:** You can modify this parameter using only the Computer Setup Wizard or the Computer Setup Editor. It cannot be set in the parameters database.

**Note:** During an installation of V7.20 or later, this parameter is set to 1 and included in the INI file. If removed from the INI file, the behavior will return to single-process mode.

### Allowable Values:

- 0 - (single process)
- 1 - (multi process)

### Default Value:

0 (set to 1 by the installer)

### See Also

[General Parameters](#)

## [General]OSErrors

Enables/disables Plant SCADA trapping of operating system errors. Plant SCADA traps disk, network, floating-point, and other system errors. If you disable this parameter, Plant SCADA will not trap these errors and could abort if an error occurs.



### UNINTENDED EQUIPMENT OPERATION

Do not change the [General]OSErrors parameter in the Citect.ini file, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

### Allowable Values:

- 0 - (Disable checking)
- 1 - (Enable checking)

### Default Value:

1

## See Also

[General Parameters](#)

### [General]PasswordExpiry

Determines how long your user and operator passwords will take to expire. Whenever you create a user and assign a password, or edit an existing password, Plant SCADA time and date stamps this change. A user will be permitted to log in while the current time and date is not later than the stamped time and date plus the Password Expiry number of days.

If a user tries to log in with an expired password, the log in will not succeed and a message will appear indicating that the password has expired.

#### Allowable Values:

0 (default: no expiry) to 365 days

#### Default Value:

0

## See Also

[General Parameters](#)

### [General]PointCountHigh

The percentage figure at which the first watermark message will be displayed if the [\[General\]WatermarkedPointCount](#) parameter has been set to true.

#### Allowable Values:

Any value between 0 and 100.

#### Default Value:

95

## See Also

[General Parameters](#)

### [General]PointCountHighHigh

The percentage figure at which the second watermark message will be displayed if the

[General]WatermarkedPointCount parameter has been set to true.

### Allowable Values:

Any value between 0 and 100.

### Default Value:

98.

### See Also

[General Parameters](#)

## [General]PrinterColourMode

The PrinterColourMode parameter works in conjunction with the iModeColour argument of the TrnPrint() function. It determines whether the trends printed using TrnPrint() will be black and white, or color, however, it will only take effect if the iModeColour argument is set to -1.

### Allowable Values:

- 0 - (color)
- 1 - (Black & White)

### Default Value:

1

### See Also

[General Parameters](#)

## [General]RegionalNumbersFormat

Determines how Plant SCADA outputs data containing decimal numbers. When set to 1, Plant SCADA uses the decimal separator from the Windows Regional Numbers setting. When set to 0, the hard coded symbol (.) is used.

### Allowable Values:

0, 1

**Default Value:**

1

**See Also**

[General Parameters](#)

**[General]ServerMonitoringPeriod**

Defines how often a Alarm Client will issue an additional transaction to its server to monitor the health of the active server. If the transaction is not completed successfully and redundancy is available, the client will automatically swap to the alternate server.

**Minimum:**

1 second

**Maximum:**

600 seconds (10 minutes)

**Default Value:**

10 seconds

**See Also**

[General Parameters](#)

**[General]ShareFiles**

Enables/disables the opening of projects by the compiler in Shared Mode. If you enable Shared Mode, you will increase the compilation time, but you are allowed to compile while other users have the project open.

**Allowable Values:**

- 0 - (Exclusive)
- 1 - (Shared)

**Default Value:**

0

## See Also

[General Parameters](#)

### [General]ShowDriverError

Determines whether an error dialog displays when an internal driver error occurs.

#### Allowable Values:

- 0 - No error dialog
- 1 - Display error dialog

#### Default Value:

0

---

**Note:** You can update this parameter within the Citect.ini file while online, as the runtime system will read its value periodically or on demand. For more information, search the Plant SCADA Help for the topic Adjusting Logging During Runtime.

---

## See Also

[General Parameters](#)

### [General]Stack

The size of the Plant SCADA general-purpose stack. Increase this parameter if Plant SCADA aborts with a "General stack overflow" error. This stack is different from the Cicode Stack. If a "Cicode stack overflow" hardware error is displayed, increase the value of the Code Stack parameter.

#### Allowable Values:

5000 to 32000

#### Default Value:

32000

## See Also

[General Parameters](#)

## [General]StartDelay

Delays the startup of Plant SCADA for a specified period (in seconds). If the Plant SCADA runtime application is started, it will not actually run until the delay period ends. This allows you to delay Plant SCADA startup if it must wait for some other process to start/complete.

### Allowable Values:

0 to 3600

### Default Value:

0

### See Also

[General Parameters](#)

## [General]SymbolicInfo

Enables/disables AN help information. See the InfoForm() Cicode function for information about AN help information. Plant SCADA will run faster and use less memory if AN help information is disabled.

### Allowable Values:

- 0 - (Disable)
- 1 - (Enable)

### Default Value:

1

### See Also

[General Parameters](#)

## [General]TagDB

Determines whether the Variable Tags database is loaded at runtime. The Variable Tags database needs to be loaded to allow tags to be referenced with the Equipment.Item syntax, as well as, to obtain the tag list shown by the TagDebug function, for Instant trends, if you are performing CtAPI ctFindFirst requests on the TAG table, for Super Genie associations and if you are using functions such as TagInfo, TagScaleStr and AssScaleStr.

If you do not need to use any of these features, you can save memory by disabling this parameter. Because the information needs to be loaded into memory, this parameter can use a large amount of memory. The amount of memory used by one tag is 30 bytes + the length of the name + 1. If you have 10,000 tags with an average name

length of 16 characters, this parameter will use  $(30 + 16 + 10) \times 10,000 = 470,000$  bytes. This memory usage would only be a problem if you are low on memory or have many variables.

### Allowable Values:

- 0 - Disable variable tag database load
- 1 - Enable variable tag database load

### Default Value:

1

---

**Note:** Plant SCADA can automatically resolve tags without a cluster context being specified if every tag name in the project is unique. However, this feature will only function correctly if this parameter is set to 1 (enabled). Confirm that this feature is not required before you disable this parameter.

---

### See Also

[General Parameters](#)

#### [General]TagDBReloadOnChange

Determines whether the Variable Tags database is checked for changes and reloaded when a new page is displayed. Enabling this parameter allows online changes to the Variable Tags database to be automatically made effective for next page display. The Variable Tags database needs to be loaded to allow tags to be referenced with the Equipment.Item syntax, obtain the tag list shown by the TagDebug function, for Instant trends, if you are performing CtAPI ctFindFirst requests on the TAG table, for Super Genie associations and if you are using functions such as TagInfo, TagScaleStr and AssScaleStr.

This parameter is ignored if [General]TagDB is set to 0. In this case, variable tags database will not be loaded on display client.

### Allowable Values:

- 0 - Disable variable tag database reload on page open
- 1 - Enable variable tag database reload on page open

### Default Value:

1

### See Also

[General Parameters](#)

## [General]TagStartDigit

Allows you to use tags, tag items, or equipment names that have a number as the first digit.

Using a number as the first digit of a tag is not a good engineering practice. Tags starting with numbers are not supported by modern programming languages and they are not allowed in the industrial control systems standard IEC 1131-3. This is because it creates confusion when tags are used in complex expressions. This was not a problem with older simpler systems when you could only get the value of a tag.

This functionality is supported primarily for users converting large existing databases to Plant SCADA.

Only variable tags and Cicode variables can start with numbers. You cannot start labels or label arguments with numbers.

As Plant SCADA supports several types of numbers there can be confusion with tags starting with number clashing with the special types of numbers. For example the following may look like tags:

0X1234	is hex number
0B0011	is binary number
007777	is octal number
0E23	is floating point number

When [General]TagStartDigit is set to 0 (zero), the above tags are treated as valid numbers.

However, if you have [General]TagStartDigit set to 1, and you have these types of numbers in your project, you must observe the following conventions:

- Numbers such as 0E12334 must contain a decimal point to distinguish them from tags, For example, 0E23 is a tag and 0.E23 is a float when the [General]TagStartDigit parameter is enabled.
- Numbers such as 0O01234567 (that's an O as in Orange), 0X01234, and 0B01, must use lowercase for alpha characters, and any potentially conflicting tags must use uppercase. For example, 0x10 will be considered a hex number, and 0X10 a tag. Note that if you use lower case to denote a number, but the number is illegal, it will be considered a tag. For example:

0xG	G is not valid hex digit.
0b2	2 is not valid binary digit.
0o8	8 is not valid octal digit.
0x123456789	more than 8 digits for 4 byte integer

Because of the above considerations you cannot define tags which look like valid numbers. For example 0x123 will be treated as a number, not a tag. You should always use upper case for tags starting with a number and the tag must contain at least one alpha (A..Z) or underscore (\_) character.

Also you cannot define numbers which look like tags. When TagStartDigit=0, then 0X123 will be treated as a hex number. When [General]TagStartDigit=1, 0X123 will be treated as a tag. This can cause problems if you have existing Cicode from older projects which has numbers defined this way. When you switch this option on, you may find you will get the "Tag not found" compiler error. This will occur because Plant SCADA will think that some of your numbers are tags and so try to find them in the Tag database.

---

**Note:** CitectVBA does not recognize Plant SCADA variable tags which are named with an initial digit (0-9).

---

### Allowable Values:

- 0 - (Do not allow tags with number as first digit)
- 1 - (Allow tags with number as first digit)

### Default Value:

0

### See Also

[General Parameters](#)

### [General]TagWriteEventFmt

By default, log messages stored in TagWriteEventQue allocate only 32 characters for the tag name and 12 for the value. Since tag names may be up to 79 characters long and strings tags may be up to 128 characters long, the format may be customized by setting the TagWriteEventFmt parameter. For example:

[General]TagWriteEventFmt = {Name,79} {Value,128}

### See Also

[General Parameters](#)

[\[General\]TagWriteEventQue](#)

### [General]TagWriteEventQue

Enables/disables logging of tag writes to a queue called TagWriteEventQue on the client. When this parameter is enabled, writes will be logged for all tags whose IODevices have their LogWrite parameter enabled.

### Allowable Values:

0 or 1

### Default Value:

0 (disabled)

### See Also

[\[General\]TagWriteEventFmt](#)

[General Parameters](#)

**[General]TrnPrinter**

The default printer port used in the TrnPrint function.

**Allowable Values:**

Any valid printer port

**Default Value:**

LPT1:

**See Also**

[General Parameters](#)

**[General]TypeRemapAllowed**

This setting controls the way the <Variable Tag> FORM displays the "Data Type" field for variable tags that are linked to an external data source (ie:marked as "Linked:Yes").

**Allowable Values:**

- 0 - Data Type field is editable on the form when the tag is linked
- 1 - Data Type field is not editable on the form when the tag is linked

**Default Value:**

0

**See Also**

[General Parameters](#)

**[General]Verbose**

Enables or disables the display of the startup dialog box, which shows the opening runtime databases and other information.

**Allowable Values:**

- 0 - not display
- 1 - display

**Default Value:**

1

---

**Note:** You can adjust this parameter during runtime using the SetLogging() function. For more information, search the Plant SCADA Help for the topic Adjusting Logging During Runtime.

---

**See Also**

[General Parameters](#)

**[General]VerboseToSysLog**

Enables startup messages displayed by the Kernel in "Verbose" mode to be logged in the syslog.dat file.

**Allowable Values:**

- 0 - No logging of "Verbose"
- 1 - Log Kernel messages

**Default Value:**

0

---

**Note:** You can adjust this parameter during runtime using the SetLogging() function. For more information, search the Plant SCADA Help for the topic Adjusting Logging During Runtime.

---

**See Also**

[General Parameters](#)

**[General]WatermarkedPointCount**

Enables two preconfigured "watermark" messages that can display to the user when the dynamic point count reaches 2 defined percentages of their point license limit.

**Allowable Value:**

- True - (Display alert message)
- Yes - (Display alert message)
- False - (Do not display alert message)
- No - (Do not display alert message)

Not case sensitive.

## Default Value:

False

## See Also

[General Parameters](#)

[\[General\]PointCountHigh](#)

[\[General\]PointCountHighHigh](#)

## Interlock Parameters

The Citect.ini file contains the following internet parameters:

- [\[Interlock\]StateItem](#) - Sets the global value of the item if not using the default item name "Running" for all pieces of Equipment.

## See Also

[Parameter Categories](#)

### [Interlock]StateItem

Sets the global default value if using a different item name rather than "Running" for pieces of Equipment with an interlock condition.

---

**Note:** This is a global value and not per instance. To set a value per instance configure using the item name for the target interlocked equipment in the Equipment Runtime Parameter grid. Refer to the topic [Configure Interlocks](#) in the main help for more information.

---

If this parameter and Equipment Runtime Parameters have been defined then the Equipment Runtime Parameters will take precedence.

For example: if there is a piece of equipment named "InterlockedDevice" the precedence would work as follows:

- If there is no Equipment Runtime Parameter and INI parameter set, "InterlockedDevice.Running" will be used to detect interlock status.
- If INI parameter [Interlock]StateItem = PV and no Equipment Runtime Parameter configured, "InterlockedDevice.PV" will be used. This is the same for all other pieces of equipment.
- If INI parameter [Interlock]StateItem = PV is set with an Equipment Runtime Parameter set for "InterlockedDevice" as Interlock\_StateItem = "RunningState", InterlockedDevice.RunningState will be used, whilst all other pieces of equipment will use "PV".

## Allowable Values:

Value for item name e.g. PV

**Default Value:**

"Running"

**See Also**

[Interlock Parameters](#)

**IOServer Parameters**

The Citect.ini file contains the following IOserver parameters:

- [\[IOServer\]AlwaysCallStopChannel](#) - Determines whether Plant SCADA calls the StopChannel() function for a driver.
- [\[IOServer\]AlwaysCallStopUnit](#) - Determines whether Plant SCADA calls the StopUnit function for a driver.
- [\[IOServer\]AsyncConnect](#) - Determines whether Plant SCADA will follow its default behavior and attempt to connect to all configured I/O servers on start-up.
- [\[IOServer\]BlockWrites](#) - Determines whether Plant SCADA will try to block optimize writes to I/O devices.
- [\[IOServer\]CacheDebug](#) - Enables/disables I/O Server cache debugging.
- [\[IOServer\]CacheIgnoreDriver](#) - Ignore the request from the protocol driver not to cache the remote I/O device.
- [\[IOServer\]CacheOptMode](#) - Reduces #COM when using dial-up or scheduled devices.
- [\[IOServer\]CacheReadAhead](#) - Determines whether Plant SCADA will use read ahead caching or not.
- [\[IOServer\]CancelTimeout](#) - Determines how long the I/O server will wait on a reply from a driver before giving up.
- [\[IOServer\]CPU](#) - Obsolete in version 7.0.
- [\[IOServer\]DisableConnectivityPropertiesAndExtensions](#) - Disables properties and tag extensions to improve the performance of the I/O servers.
- [\[IOServer\]DebugLogTagHandshake](#) - Enables detailed tag handshake logging.
- [\[IOServer\]EnableEventQueue](#) - Enables the event queue.
- [\[IOServer\]HeartTime](#) - The period at which the I/O Server sends out heartbeats (to the clients) on the status of each I/O device.
- [\[IOServer\]HWAlarmOnDeviceDisable](#) - Controls whether hardware alarms for disabled IO devices are displayed or suppressed.
- [\[IOServer\]InitMsg](#) - Determines whether communication information is displayed in the Kernel (at startup).
- [\[IOServer\]LogTagHandshake](#) - Enables basic tag handshake logging.
- [\[IOServer\]MaxEventsDrop](#) - Sets the number of events that are dropped when too many are queued.
- [\[IOServer\]MaxEventsQueued](#) - Sets the maximum number of events that can be queued.
- [\[IOServer\]MaxTagHandshakeSize](#) - Sets the maximum number of bytes a tag validation check requests during validation.
- [\[IOServer\]MaxTimeInQueueMs](#) - Sets the maximum time for which an event can be queued.
- [\[IOServer\]Name](#) - Obsolete in version 7.0.

- [\[IOServer\]PeerServerConnectTimeOut](#) - Defines how long an IOServer will attempt to connect to peer IOServers.
- [\[IOServer\]Process](#) - Obsolete in version 7.0.
- [\[IOServer\]RedundancyDebug](#) - Enables/disables I/O device redundancy debugging.
- [\[IOServer\]SaveBackup](#) - Obsolete in version 7.0.
- [\[IOServer\]SaveFile](#) - The indicative name and path of the local I/O device persistence cache files on an I/O Server.
- [\[IOServer\]SaveNetwork](#) - The UNC name of the Persistence Cache file on this I/O Server.
- [\[IOServer\]SavePeriod](#) - Obsolete in version 7.20.
- [\[IOServer\]SaveTime](#) - Obsolete in version 7.20.
- [\[IOServer\]Server](#) - Obsolete in version 7.0.
- [\[IOServer\]StrictTagHandshake](#) - Enforces that the I/O Server contains a matching OID for every OID on the client.
- [\[IOServer\]TagAddressNoCase](#) - Determines if case-sensitivity needs to be considered when validating tag addresses.
- [\[IOServer\]WatchDog](#) - Determines whether Plant SCADA periodically checks the communication connection to your standby I/O devices.
- [\[IOServer\]WatchDogPrimary](#) - Determines whether Plant SCADA periodically checks the communication connection to your primary I/O devices.

## See Also

[Parameter Categories](#)

[IOServer Process Parameters](#)

### [IOServer]AlwaysCallStopChannel

Determines whether Plant SCADA attempts to call the StopChannel() function for a driver.

The StopChannel call releases resources allocated for a PLC connection. Under normal circumstances, this occurs when the PLC is online at shutdown, or disabled. Since the PLC may not get a StopChannel when it is offline at shutdown, its resources may not have been released and Plant SCADA crashes can occur.

By default this parameter is set to 1 which means that a StopChannel is called, and may prevent crashes at shutdown. Usually if setting AlwaysCallStopUnit, you would set AlwaysCallStopChannel.

## Allowable Values:

- 0 - StopChannel is not called.
- 1 - StopChannel is called.

## Default Value:

1

## See Also

[IOServer Parameters](#)

### [IOServer]AlwaysCallStopUnit

Determines whether Plant SCADA attempts to call the StopUnit function for a driver when the unit goes offline so as to release resources allocated for an I/O Device.

The StopUnit call releases resources allocated for a PLC. Under normal circumstances, this occurs when the PLC is online at shutdown, or disabled. Since the unit may not get a StopUnit when it is offline at shutdown, its resources may not have been released and Plant SCADA crashes can occur.

By default this parameter is set to 1 and commands that a StopUnit is called. This may prevent crashes at shutdown. Usually if setting AlwaysCallStopUnit, you would set AlwaysCallStopChannel.

#### Allowable Values:

- 0 - Stop Unit is not called
- 1 - StopUnit is called

#### Default Value:

1

## See Also

[IOServer Parameters](#)

### [IOServer]AsyncConnect

Determines whether Plant SCADA will follow its default behavior and attempt to connect to all configured I/O servers on start-up.

When the AsynConnect parameter is set to 1, Plant SCADA will defer the connection attempt for each I/O Server until that I/O Server is required.

This option may be required in order to speed up the start-up process on sites which have many I/O Servers that do not yet exist.

#### Allowable Values:

- 0 - (I/O Servers will be connected on start-up)
- 1 - (Connection of I/O Servers will be bypassed on start-up)

#### Default Value:

0

## See Also

[IOServer Parameters](#)

### [IOServer]BlockWrites

Determines whether Plant SCADA will try to dynamically optimize writes to I/O devices. This includes combining writes to consecutive memory locations into a single large write, and ignoring unnecessary writes to the same memory location (i.e. when the data is the same).

#### Allowable Values:

- 0 - (Turn dynamic write optimization off)
- 1 - (Turn dynamic write optimization on)

#### Default Value:

1

## See Also

[IOServer Parameters](#)

### [IOServer]CacheDebug

Enables/disables I/O Server cache debugging.

*Allowable Values:*

- 0 - (Disable debugging)
- 1 - (Enable debugging)

*Default Value:*

0

## See Also

[IOServer Parameters](#)

### [IOServer]CacheIgnoreDriver

Ignore the request from the protocol driver not to cache the remote I/O device.

#### Allowable Values:

- 0 - (Observe the request)

- 1 - (Ignore the request)

**Default Value:**

0

**See Also**[IOServer Parameters](#)**[IOServer]CacheOptMode**

This parameter is designed to reduce #COMs when using dialup or scheduled devices. The #COM is often a result of how tags are blocked together in the IOServer.

This is the global INI setting. This setting is allowed at a per device level by:

[<unitname>] CacheOptMode = x

By default, the IOServer only searches for tag data requests to be within or equal to a single cached tag set; for example, cache has tag address points 1 to 10, and then 11 to 20, and a request wants addresses 5 to 6, this is permitted. If, however, the request was for items 9 to 11, the system would not return a match and would request a read from the driver. For a disconnected dialup unit, this would cause a #COM or #WAIT.

CacheOptMode values are:

- 1 - First come first served - Tells the IOServer to search through cached points looking for the data, once it is found return.
- 2 - Scan cache for latest entries - Forces the scan to look through cached items to find data which is the MOST recent.
- 4 - An incomplete match is OK - Tells the system that if some items match and some are not found, to consider this OK.
- +8 - Allow behavior on all devices, by default the settings above only apply to remote units.
- +16 - Allow written data not to remove cache item and update value on return, e.g. 1 to 10 in the cache, write to 8. The default behavior would be to remove the whole cache item so that a fresh read of 1 to 10 would be needed on a write to 8. With +16 added, then 1 to 10 stays in the cache, and item 8 would be updated on the REPLY from the driver.

For dialup devices, CacheOptMode=1 covers the majority of needs.

**Allowable Values:**

- 0 - No login
- 1 to 31 - Combination of above values

**Default Value:**

0

## See Also

[IOServer Parameters](#)

### [IOServer]CacheReadAhead

Determines if read ahead caching is enabled on the I/O Server. When read ahead caching is enabled, the I/O Server will try to read any I/O device data which is coming close to cache 'timeout' time. The I/O Server will only read this data if the communication channel to the I/O device is idle - to give higher priority to other read requests.

#### Allowable Values:

- 0 - (Disable read ahead caching)
- 1 - (Enable read ahead caching)

#### Default Value:

1

## See Also

[IOServer Parameters](#)

### [IOServer]CancelTimeout

Determines how long the I/O server will wait on a reply from a driver before giving up and cancelling the request. This timeout can be used to diagnose driver "not responding issues".

#### **WARNING**

##### **UNINTENDED EQUIPMENT OPERATION**

Do not change the [IOServer]CancelTimeout parameter in the Citect.ini file, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

#### Allowable Values:

1 to 1000000 (in seconds)

#### Default Value:

300

## See Also

[IOServer Parameters](#)

### [IOServer]DisableConnectivityPropertiesAndExtensions

The Connectivity Server manages tag subscriptions for the OPC UA Server and the Industrial Graphics Server, and passes these subscriptions on to the I/O servers. The tag may be in the format of either TagName or Equipment.Item.

If a subscribed tag also specifies a property or extension, extra processing is required by the I/O server. The greater the number of I/O devices, the greater the processing load.

If your project does not require properties or extensions for the OPC UA Server or the Industrial Graphics Server, then support for these can be disabled. This will improve the performance of the I/O servers.

This parameter is only available globally and cannot be set per individual I/O server. For consistency in behavior, it is recommended that the INI is applied uniformly across all computers in your system.

---

**Note:** This parameter only applies to subscriptions from the Connectivity Server. It has no relevance to subscriptions made from alarm, trend or report servers, or from Plant SCADA clients or web clients.

---

## Allowable values:

- 0 — properties and extensions are enabled
- 1 — properties and extensions are disabled

## Default Value:

0

## See Also

[IOServer Parameters](#)

### [IOServer]DebugLogTagHandshake

Enables detailed tag handshake logging during tag validation. This allows you to view detailed individual results for each variable tag as they are validated by system checks. The volume of tags that are processed per check request can be configured via the parameter [\[IOServer\]MaxTagHandshakeSize](#).

## Allowable Values:

- 0 - disables detailed tag handshake logging
- 1 - enable detailed tag handshake logging

**Default Value:**

1

**See Also**

[IOServer Parameters](#)

**[IOServer]EnableEventQueue**

Enables the event queue.

To stop events accumulating in the queue, you should use the Cicode function TagEventQueue to read events, as once an event is read it is removed from the queue.

**Allowable values:**

- 0 - disable event queue
- 1 - enable event queue

**Default Value:**

0

**See Also**

[IOServer Parameters](#)

**[IOServer]HeartTime**

The period at which the I/O Server sends out heartbeats (to the clients) on the status of each I/O device.

**Allowable Values:**

2000 to 60000 (milliseconds)

**Default Value:**

60000

**See Also**

[IOServer Parameters](#)

**[IOServer]HWAlarmOnDeviceDisable**

Determines whether hardware alarms for disabled IO devices will be displayed or suppressed.

**Allowable Values:**

- 0 - (Suppress hardware alarms)
- 1 - (Display hardware alarms)

**Default Value:**

1

**Note:**

- On the Plant SCADA machine that is configured as an IOServer and a client, Plant SCADA will display an initial hardware alarm once to indicate that the device has been disabled and its state is offline. This is true if you only disabled the IO device on the IOServer only by calling IODeviceControl(x, 1,1). Plant SCADA will not display additional hardware alarms for disabled devices if the parameter [IOSERVER]HWAlarmOnDeviceDisable = 0 is set.
- On the Plant SCADA machine that is configured as a Control Client only, Plant SCADA will display an initial hardware alarm once to indicate that the device state is offline. Plant SCADA will not display additional hardware alarms for disabled devices if the parameter [IOSERVER]HWAlarmOnDeviceDisable = 0 is set.
- On the Plant SCADA machine that is configured as an IOServer and a Control Client, if you disabled the IO device on both the IO Server and the Plant SCADA Control Client (by calling both IODeviceControl(x, 1,1) and IODeviceControl(x, 0,1)), Plant SCADA will not display an initial hardware alarm for the disabled IO device. Plant SCADA will not display any additional hardware alarms for disabled devices if the parameter [IOSERVER]HWAlarmOnDeviceDisable = 0 is set.

**See Also**

[IOServer Parameters](#)

**[IOServer]InitMsg**

Determines whether communication information is displayed in the Kernel (at startup).

**Allowable Values:**

- 0 - (Disable)
- 1 - (Displays I/O device status at startup)

**Default Value:**

0

## See Also

[IOServer Parameters](#)

### [IOServer]LogTagHandshake

Enables tag handshake logging during tag validation. See also [\[IOServer\]DebugLogTagHandshake](#).

#### Allowable Values:

- 0 - disables tag handshake logging
- 1 - enables tag handshake logging

#### Default Value:

0

## See Also

[IOServer Parameters](#)

### [IOServer]MaxEventsDrop

Sets the number of events that are dropped when too many events become queued.

#### Allowable values:

An integer representing number of events. Value should be a fraction of the integer set for MaxEventsQueued (for example, 10%).

#### Default Value:

30000

## See Also

[IOServer Parameters](#)

### [IOServer]MaxEventsQueued

Sets the maximum number of events that can accumulate in the events queue. If the maximum number events is reached, events will be removed due to queue overflow.

**Allowable values:**

An integer representing number of events. For example an integer between 1000 -1,000 000.

**Default Value:**

300000

**See Also**

[IOServer Parameters](#)

**[IOServer]MaxTagHandshakeSize**

Allows you to set the maximum size of bytes for a tag validation request when handshaking.

**Default Value:**

4096

**See Also**

[IOServer Parameters](#)

**[IOServer]MaxTimeInQueueMs**

Sets the maximum time (in milliseconds) for which an event can be queued before it is timed out.

**Allowable values:**

An integer representing milliseconds. For example any integer between 1000 ms (1 sec) to 86 400 000 ms (1 day).

**Default Value:**

600000

**See Also**

[IOServer Parameters](#)

**[IOServer]PeerServerConnectTimeOut**

Defines the period the I/O Server will attempt to connect to peer I/O Servers before timeout.

I/O Servers are unable to activate any I/O Devices configured as 'Exclusive' = TRUE until this period expires, or the I/O Server has successfully connected to its peer I/O Servers (thus identifying if any of the Peer I/O Servers have activated any I/O Devices).

**Default Value:**

30000

**See Also**

[IOServer Parameters](#)

**[IOServer]RedundancyDebug**

Enables or disables the logging of I/O device redundancy debug messages to syslog.dat.

*Allowable Values:*

- 0 - (Disable debugging)
- 1 - (Enable debugging)

*Default Value:*

0

---

**Note:** You can adjust this parameter during runtime using the SetLogging() function. For more information, search the Plant SCADA Help for the topic *Adjusting Logging During Runtime*.

**See Also**

[IOServer Parameters](#)

**[IOServer]SaveFile**

The indicative name and path of the local I/O device persistence cache files on an I/O Server.

**Allowable Values:**

Any valid directory.

**Default Value:**

The current project directory.

**See Also**

[IOServer Parameters](#)

**[IOServer]SaveNetwork**

The UNC name of the Persistence Cache file on this I/O Server.

**Allowable Values:**

Any valid UNC filename.

**Default Value:**

None

**See Also**

[IOServer Parameters](#)

**[IOServer]StrictTagHandshake**

Enforces that the I/O Server contains a matching OID for every OID on the client.

**Allowable Values:**

- 0 - Disable
- 1 - Unknown tags will cause a mismatch. Client and Server needs to have identical variable databases (for that unit) to be able to communicate.

**Default Value:**

0

**See Also**

[IOServer Parameters](#)

**[IOServer]TagAddressNoCase**

Determines if case-sensitivity needs to be considered when validating tag addresses for the implementation of Object IDs and the variables database.

**Allowable Values:**

- 0 - the comparison must be case sensitive.
- 1 - the comparison will ignore case sensitivity.

**Default Value:**

0

**See Also**[IOServer Parameters](#)**[IOServer]WatchDog**

Determines whether Plant SCADA periodically checks the communication connection to your standby I/O devices.

If you set this parameter to 1, Plant SCADA will check the Standby devices every watch time (as specified by the WatchTime parameter of the device driver). If you set this parameter to 0, this periodic checking is disabled, but Plant SCADA will still communicate with the I/O device on startup to verify that the connection is valid.

**Note:** If you disable periodic checking, you will not receive a hardware error if the standby I/O device becomes inoperative when Plant SCADA is communicating with the Primary.

**Allowable Values:**

- 0 - disables Watch Dog functionality for standby I/O devices
- 1 - enables Watch Dog functionality for standby I/O devices

**Default Value:**

1

**See Also**[IOServer Parameters](#)**[IOServer]WatchDogPrimary**

Determines whether Plant SCADA periodically checks the communication connection to your primary I/O devices. If you set this parameter to 1, Plant SCADA will check the Primary devices every watch time (as specified by the WatchTime parameter of the device driver). If you set this parameter to 0, you will not be notified of a communication error until a read to the device is attempted.

**Note:** If you disable periodic checking, and no read activity is occurring, you will not receive a hardware error if the primary I/O device becomes inoperative.

**Allowable Values:**

- 0 - disables Watch Dog functionality for primary I/O devices
- 1 - enables Watch Dog functionality for primary I/O devices

**Default Value:**

0

**See Also**[IOServer Parameters](#)**IOServer Process Parameters**

Plant SCADA allows you to specify some parameters for an I/O server process (see [Hierarchical Parameters](#) in the Plant SCADA documentation).

The following parameters need to be configured for a specific process.

**Note:** The <server name> is the name of the I/O server as specified in the **Topology** activity.

- [\[IOServer.<ClusterName>.<ServerName>\]Clusters](#) - Sets the clusters that the IO Server process will connect to at startup.
- [\[IOServer.<ClusterName>.<ServerName>\]CPU](#) - Sets the CPU that the IO Server process is assigned to.
- [\[IOServer.<ClusterName>.<ServerName>\]Events](#) - The list of events that this Plant SCADA IO Server process enables.
- [\[IOServer.<ClusterName>.<ServerName>\]ShutdownCode](#) - Determines the Cicode function to run when Plant SCADA IO Server process shuts down.
- [\[IOServer.<ClusterName>.<ServerName>\]StartupCode](#) - Determines the Cicode function to run when Plant SCADA IO Server process starts up.

**See Also**[Parameter Categories](#)[IOServer Parameters](#)**[IOServer.<ClusterName>.<ServerName>]Clusters**

Sets which clusters this IO Server will connect to at startup.

This parameter needs to be defined at the finest level of hierarchical granularity:  
`IOServer.<ClusterName>.<ServerName>Clusters`.

See [Hierarchical Parameters](#) for more information.

**Allowable Values:**

A comma separated list of configured cluster names.

**Default Value:**

If this parameter is blank, the IO Server will connect to every cluster defined for the project.

## See Also

[IOServer Process Parameters](#)

### [IOServer.<ClusterName>.<ServerName>]CPU

This parameter is by Plant SCADA Runtime Manager to determine which CPU(s) the I/O Server process is assigned to. The CPU parameter is configurable for every Plant SCADA process via the **CPU Setup** page in the Computer Setup Wizard (see [CPU Configuration](#)).

This parameter needs to be defined at the finest level of hierarchical granularity:

`IOServer.<ClusterName>.<ServerName>Clusters`. See [Hierarchical Parameters](#) for more information.

CPU numbers are specified in a comma separated list. CPU numbers need to be valid and a subset of the available CPUs on a computer. For example, if there are eight CPUs on a computer, a valid CPU range is 0 to 7.

---

**Note:** If this parameter is modified while the Plant SCADA Runtime Manager is running, the changes will not take effect until every instance of Plant SCADA is restarted.

## Allowable Values:

A comma separated list in the range 0 to 31 (inclusive) where 0 is the first CPU (for example, "0, 1, 2" or "3, 4, 5").

If no value is entered, the process will run on all available CPUs.

---

**Note:** When a process is set to run on all available CPUs, be aware that performance issues may still occur when your CPU usage is less than 100%. You can detect these issues when your CPU usage for a process is fixed at 100 divided by the number of processors.

## Default Value:

No value (the process will run on all CPUs).

---

**Note:** You cannot manually specify CPU assignments on a computer with more than 32 CPUs. Under these circumstances, the Setup Wizard will automatically set this parameter to no value (all CPUs). Any changes you make to this parameter will be overwritten.

## See Also

[IOServer Process Parameters](#)

[\[Client\]CPU](#)

[\[Alarm.<ClusterName>.<ServerName>\]CPU](#)

[\[Report.<ClusterName>.<ServerName>\]CPU](#)

[\[Trend.<ClusterName>.<ServerName>\]CPU](#)

### [IOServer.<ClusterName>.<ServerName>]Events

The list of events that this Plant SCADA IO Server will enable.

This parameter needs to be defined at the finest level of hierarchical granularity:  
`IOServer.<ClusterName>.<ServerName>Clusters`.

See [Hierarchical Parameters](#) for more information.

---

**Note:** This parameter is modified by the Computer Setup Wizard. To set this parameter, we recommend you to enable the required events in the "Events Setup" page of the Computer Setup Wizard (instead of entering the events in the Setup Editor).

---

## Allowable Values:

List of events (comma separated).

## Default Value:

NONE

---

**Note:** When Plant SCADA is running in single process mode only the events assigned to [Client]Events will be enabled. Events specified for particular Plant SCADA components are ignored.

---

## See Also

[IOServer Process Parameters](#)

### [IOServer.<ClusterName>.<ServerName>]ShutdownCode

Can be used to specify a Cicode function that runs at shutdown of the Plant SCADA IO Server component. Plant SCADA suspends the shutdown process until the function has finished running. The function needs to complete execution within the time specified by the [Code]ShutdownTime parameter, otherwise Plant SCADA will terminate it.

This parameter needs to be defined at the finest level of hierarchical granularity:  
`IOServer.<ClusterName>.<ServerName>Clusters`.

See [Hierarchical Parameters](#) for more information.

---

**Note:** This parameter is modified by Computer Setup Wizard.

---

## Allowable Values:

Any built-in or user-written Cicode function, 1-31 alpha-numeric characters.

## Default Value:

NONE

---

**Note:** When Plant SCADA is running in single process mode only the Cicode function assigned to [Client]ShutdownCode will be executed at shutdown. Cicode functions specified for particular Plant SCADA components are ignored.

---

## See Also

[IOServer Process Parameters](#)

### [IOServer.<ClusterName>.<ServerName>]StartupCode

Determines the Cicode function to run when the Plant SCADA IO Server component starts up.

You can only pass constant data to the startup function call. You cannot pass variables or other functions. For example, MyFunc(1, "str") is valid, but MyFunc(PLCTAG, "str") or MyFunc(YourFunc(), "str") is not supported.

This parameter needs to be defined at the finest level of hierarchical granularity:

IOServer.<ClusterName>.<ServerName>Clusters.

See [Hierarchical Parameters](#) for more information.

---

**Note:** This parameter is modified by the Computer Setup Wizard.

## Allowable Values:

Any built-in or user-written Cicode function, 1-31 alpha-numeric characters.

## Default Value:

NONE

---

**Note:** When Plant SCADA is running in single process mode only the Cicode function assigned to [Client]StartupCode will be executed at startup. Cicode functions specified for particular Plant SCADA components are ignored.

## See Also

[IOServer Process Parameters](#)

## IPC Parameters

IPC stands for Interprocess Communications and is a layer used by Plant SCADA for connections to CtAPI clients and Plant SCADA processes on the same PC. Be aware that not all process to process communication is via this interface.

- [\[IPC\]bPipeToSocket](#) - When set, for local process communications a socket is used instead of a pipe.
- [\[IPC\]EventLogging](#) -Enables logging of the communications between processes.
- [\[IPC\]HeartBeat](#) - Checks if the shared memory connection is still valid and reports an error.
- [\[IPC\]SocketSendTimeout](#) - Time to wait for the status of the last socket write.
- [\[IPC\]SocketShutdownTimeout](#) - Is the time to wait before closing the socket.

## See Also

[Parameter Categories](#)

## [IPC]bPipeToSocket

**Note:** This parameter is designed to assist qualified technical support personnel with system diagnosis.

If this is set TRUE, for local Plant SCADA process communications, a socket is used instead of a pipe. This may be useful for debugging purposes. Only change under direction from Plant SCADA support.

### Allowable Values:

- 1 = True
- 0 = False

### Default Value:

0

### See Also

[IPC Parameters](#)

## [IPC]EventLogging

**Note:** This parameter is designed to assist qualified technical support personnel with system diagnosis.

Enables logging of the communications between processes. Logging goes to ipc.log under the "Logs" directory.

### Allowable Values:

- Off - 0
- Errors - 1
- Warnings -2
- Verbose - 3
- Verbose Plus - 4

The higher the value, the more logging will occur.

### Default Value:

0

### See Also

[IPC Parameters](#)

**[IPC]HeartBeat**

---

**Note:** This parameter is designed to assist qualified technical support personnel with system diagnosis.

Checks if the shared memory connection is still valid and reports the error. Setting this to FALSE means we would find out the connection was lost when we went to use the connection. Only change under direction from technical support.

**Allowable Values:**

- 1 = True
- 0 = False

**Default Value:**

1

**See Also**

[IPC Parameters](#)

**[IPC]SocketSendTimeout**

---

**Note:** This parameter is designed to assist qualified technical support personnel with system diagnosis.

The time in milliseconds to wait for the status of the last socket write.

**Allowable Values:**

- 0 - Wait forever (not recommended)
- 1 to 10000 (milliseconds)

**Default Value:**

500

**See Also**

[IPC Parameters](#)

**[IPC]SocketShutdownTimeout**

---

**Note:** This parameter is designed to assist qualified technical support personnel with system diagnosis.

The time in milliseconds to wait before closing the socket after telling the other end it is to be closed.

**Allowable Values:**

100 to 10000 (milliseconds)

**Default Value:**

1000

**See Also**

[IPC Parameters](#)

**Kernel Parameters****WARNING****UNINTENDED EQUIPMENT OPERATION**

- Do not use the kernel for normal Plant SCADA operation. The kernel is only for diagnostics and debugging purposes.
- Configure your security so that only approved personnel can view or use the kernel.
- Do not view or use the kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

The Citect.ini file contains the following Kernel parameters:

- [\[Kernel\]AlwaysOnTop](#) - Determines if the Kernel window remains on top of all other windows.
- [\[Kernel\]BufPoolProtect](#) - Enables buffer pool debugging.
- [\[Kernel\]DeltaLateCount](#) - Specifies the number of times the DeltaCheckTask can run late before an entry is logged to the syslog.dat file.
- [\[Kernel\]DeltaLateTime](#) - Specifies the time that is used to determine if the DeltaCheckTask is running late.
- [\[Kernel\]DeltaSleepTime](#) - Determines how often the DeltaCheckTask runs.
- [\[Kernel\]ErrorBuffers](#) - The maximum number of error buffers available for logging to the syslog.dat file.
- [\[Kernel\]ExceptionWatchdog](#) - Determines if the Unhandled exception handler has been reset by a third party dll, and set it back to the Plant SCADA Exception Mailer.
- [\[Kernel\]Idle](#) - Determines how many of the Plant SCADA Kernel tasks to run before releasing the CPU to other Windows programs.
- [\[Kernel\]MaxLateTime](#) - The maximum amount of time kernel tasks can run late (in ms).
- [\[Kernel\]MemoryMinimum](#) - Minimum memory limit for warnings (in KB)
- [\[Kernel\]Queue](#) - The number of Kernel queues.
- [\[Kernel\]Retries](#) - Determines the number of Kernel retries before Runtime is terminated.
- [\[Kernel\]Task](#) - The number of Kernel tasks.

- [\[Kernel\]Watchdog](#) - Determines if the Kernel is working.
- [\[Kernel\]WatchdogTime](#) - Determines the number of seconds between watchdog checks.
- [\[Kernel\]WinShutdown](#) - Determines whether Windows can shutdown while Plant SCADA is running.

## See Also

[Parameter Categories](#)

### [Kernel]AlwaysOnTop

Determines if the Kernel window remains on top of all other windows when displayed on a screen.

This parameter is set to 1 when **Always on Top** is selected in the Kernel's **Options** menu.

#### Allowable Values:

- 0 - Kernel window does not display on top of other windows
- 1 - Kernel window displays on top of all other windows

#### Default Value:

0

## See Also

[Kernel Parameters](#)

### [Kernel]BufPoolProtect



#### UNINTENDED EQUIPMENT OPERATION

- Do not use the Kernel for normal Plant SCADA operation. The Kernel is only for diagnostics and debugging purposes.
- Configure your security so that only approved personnel can view or use the Kernel.
- Do not view or use the Kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Enables/disables buffer pool debugging.

#### Allowable Values:

- 1 - (to enable debugging mode)

- 0 - (to disable debugging mode)

**Default Value:**

0

**See Also**

[Kernel Parameters](#)

[\[Kernel\]DeltaLateCount](#)

**WARNING****UNINTENDED EQUIPMENT OPERATION**

- Do not use the kernel for normal Plant SCADA operation. The kernel is only for diagnostics and debugging purposes.
- Configure your security so that only approved personnel can view or use the kernel.
- Do not view or use the kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Plant SCADA uses a DeltaCheckTask to monitor long running tasks. This parameter specifies the number of times the DeltaCheckTask can run late before an entry is logged to the syslog.dat file. The time used to determine if the task is running late is specified by the parameter [\[Kernel\]DeltaLateTime](#).

**Allowable Values:**

Any value, but within the recommended range of 1 to 20.

**Default Value:**

4

**See Also**

[Kernel Parameters](#)

[\[Kernel\]DeltaLateTime](#)

**WARNING****UNINTENDED EQUIPMENT OPERATION**

- Do not use the kernel for normal Plant SCADA operation. The kernel is only for diagnostics and debugging purposes.
- Configure your security so that only approved personnel can view or use the kernel.
- Do not view or use the kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Plant SCADA uses a DeltaCheckTask to monitor long running tasks. This parameter specifies the time (in milliseconds) that is used to determine if the DeltaCheckTask is running late. If this occurs more times than the value specified for [\[Kernel\]DeltaLateCount](#), then an entry will be logged to the syslog.dat file.

If tasks frequently run late, it is likely that the system is overloaded and it will not be acting on values and information at the right time.

## Allowable Values:

Any value.

## Default Value:

250

## See Also

[Kernel Parameters](#)

[\[Kernel\]DeltaSleepTime](#)

## WARNING

### UNINTENDED EQUIPMENT OPERATION

- Do not use the kernel for normal Plant SCADA operation. The kernel is only for diagnostics and debugging purposes.
- Configure your security so that only approved personnel can view or use the kernel.
- Do not view or use the kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Plant SCADA uses a DeltaCheckTask to monitor long running tasks. The delta sleep time determines (in milliseconds) how often the DeltaCheckTask runs.

**Note:** Avoid setting this parameter below 250, otherwise there will be too many false warning messages. Also, if this value is too large, then you will not get any warning messages when the tasks are running significantly late, which could overload the system.

**Allowable Values:**

Any value.

**Default Value:**

250

**See Also**

[Kernel Parameters](#)

[\[Kernel\]ErrorBuffers](#)

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

- Do not use the kernel for normal Plant SCADA operation. The kernel is only for diagnostics and debugging purposes.
- Configure your security so that only approved personnel can view or use the kernel.
- Do not view or use the kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Determines the maximum number of error buffers available for logging to the syslog.dat file.

When the number of pending messages to process exceeds this count, these messages are not logged any more. Messages lost in this manner can be monitored in the General Kernel Window as the item "Lost Errors".

**Allowable Values:**

0 to 100,000. However setting this value to 0 will mean that no buffers will be available for logging.

**Default Value:**

1000

**See Also**

[Kernel Parameters](#)

[\[Kernel\]ExceptionWatchdog](#)

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

- Do not use the kernel for normal Plant SCADA operation. The kernel is only for diagnostics and debugging purposes.
- Configure your security so that only approved personnel can view or use the kernel.
- Do not view or use the kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

When set enables a periodic check that determines if the Unhandled exception handler (the Exception Handler) has been reset by a third party dll, and sets it back to the Exception Mailer.

*Allowable Values:*

- 1 - Enable exception handler watchdog
- 0 - disable exception handler watchdog

*Default Value:*

0

**See Also**

[Kernel Parameters](#)

[\[Kernel\]Idle](#)

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

- Do not use the kernel for normal Plant SCADA operation. The kernel is only for diagnostics and debugging purposes.
- Configure your security so that only approved personnel can view or use the kernel.
- Do not view or use the kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Determines how many of the Plant SCADA Kernel tasks to run before releasing the CPU to other Windows programs or other non Kernel activities within Plant SCADA. Plant SCADA will release the CPU if it runs out of its own tasks. If you set this parameter too high, you will reduce CPU resources for other programs running at the same time, and may delay responses to user input.

**Allowable Values:**

1 to 50. However it is recommended that you use the default value.

**Default Value:**

15

**See Also**

[Kernel Parameters](#)

**[Kernel]MaxLateTime****⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

- Do not use the kernel for normal Plant SCADA operation. The kernel is only for diagnostics and debugging purposes.
- Configure your security so that only approved personnel can view or use the kernel.
- Do not view or use the kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

The maximum amount of time kernel tasks can run late (in ms). The number of consecutive times a task is late.

**Allowable Values:**

Any value.

**Default Value:**

250

**See Also**

[Kernel Parameters](#)

**[Kernel]MemoryMinimum****⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

- Do not use the kernel for normal Plant SCADA operation. The kernel is only for diagnostics and debugging purposes.
- Configure your security so that only approved personnel can view or use the kernel.
- Do not view or use the kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Minimum memory limit for warnings (in KB)

### Allowable Values:

Any value, but do not set a value less than 90000, or too many false warnings about low memory condition will be recorded in the syslog file. If the value is too high, the system will run out of memory before a warning is recorded.

### Default Value:

90000

### See Also

[Kernel Parameters](#)

### [Kernel]Queue

## **WARNING**

### UNINTENDED EQUIPMENT OPERATION

- Do not use the kernel for normal Plant SCADA operation. The kernel is only for diagnostics and debugging purposes.
- Configure your security so that only approved personnel can view or use the kernel.
- Do not view or use the kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

The number of Kernel queues. Increase this parameter if an "Out of Kernel Queues" message displays.

### Allowable Values:

150 to 5000 (recommended)

### Default Value:

5000

## See Also

[Kernel Parameters](#)

[\[Kernel\]Retries](#)



### UNINTENDED EQUIPMENT OPERATION

- Do not use the kernel for normal Plant SCADA operation. The kernel is only for diagnostics and debugging purposes.
- Configure your security so that only approved personnel can view or use the kernel.
- Do not view or use the kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Determines the number of Kernel retries before Runtime is terminated. Used in conjunction with the [\[Kernel\]Watchdog](#) and [\[Kernel\]WatchdogTime](#) parameters.

## Allowable Values:

1 to 10

## Default Value:

3

## See Also

[Kernel Parameters](#)

[\[Kernel\]Task](#)



### UNINTENDED EQUIPMENT OPERATION

- Do not use the kernel for normal Plant SCADA operation. The kernel is only for diagnostics and debugging purposes.
- Configure your security so that only approved personnel can view or use the kernel.
- Do not view or use the kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

The number of Kernel tasks. Increase this parameter if you get an "Out of Kernel Tasks" error message.

**Allowable Values:**

50 to 32767

**Default Value:**

512

**See Also**

[Kernel Parameters](#)

**[Kernel]Watchdog****UNINTENDED EQUIPMENT OPERATION**

- Do not use the kernel for normal Plant SCADA operation. The kernel is only for diagnostics and debugging purposes.
- Configure your security so that only approved personnel can view or use the kernel.
- Do not view or use the kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

When set a task checks if the Plant SCADA Kernel is working (i.e., Plant SCADA is not locked up) and terminates Plant SCADA runtime if the timeout ([Kernel]WatchdogTime) has been exceeded a number of retries ([Kernel]Retries).

This is a backup measure to allow redundancy from other I/O servers to be enabled if the Primary I/O server gets locked (for example, by incorrect user Cicode).

**Allowable Values:**

- 0 - (Disable)
- 1 - (Enable)

**Default Value:**

0

**See Also**

[Kernel Parameters](#)

[\[Kernel\]Retries](#)

[\[Kernel\]WatchdogTime](#)

**[Kernel]WatchdogTime**

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

- Do not use the kernel for normal Plant SCADA operation. The kernel is only for diagnostics and debugging purposes.
- Configure your security so that only approved personnel can view or use the kernel.
- Do not view or use the kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

The number of seconds between Kernel watchdog checks.

### **Allowable Values:**

1 to 60 seconds

### **Default Value:**

5 seconds

### **See Also**

[Kernel Parameters](#)

[\[Kernel\]Watchdog](#)

[\[Kernel\]Retries](#)

**[Kernel]WinShutdown**

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

- Do not use the kernel for normal Plant SCADA operation. The kernel is only for diagnostics and debugging purposes.
- Configure your security so that only approved personnel can view or use the kernel.
- Do not view or use the kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Determines whether Windows can shutdown while Plant SCADA is running. When set to **0**, the user cannot shutdown Windows while Plant SCADA is running. When set to **1**, Windows can be shutdown while Plant SCADA is running.

### Allowable Values:

- 0 - Stop Windows shutting down.
- 1 - Allow Windows to shutdown.

### Default Value:

0

### See Also

[Kernel Parameters](#)

### Keyboard Parameters

The Citect.ini file contains the following keyboard parameters:

- [\[Keyboard\]ButtonOnlyLeftClick](#) - Allows only the left mouse button to "push" buttons in Plant SCADA.
- [\[Keyboard\]ButtonRaw](#) - Puts a string (ASCII character code) into the key command line when a button is pressed.
- [\[Keyboard\]EchoPopUp](#) - Determines whether the keyboard input on a graphical object is echoed in a tips popup.
- [\[Keyboard\]LogExtendedDate](#) - Specifies if the extended date format is used in command log formatting.
- [\[Keyboard\]NonPrint](#) - Enables/disables the display of the non-print character (specified in the [\[Keyboard\]NonPrintChar](#) parameter).
- [\[Keyboard\]NonPrintChar](#) - The ASCII character code to display when a key that does not have a displayable character or Key Name is pressed.
- [\[Keyboard\]Type](#) - The keyboard type.

### See Also

[Parameter Categories](#)

### [Keyboard]ButtonOnlyLeftClick

Determines whether only the left mouse button can press a button object. This parameter has a specific purpose relating to events.

When using events, pressing the right mouse button when over a button object will not call the event handler. To allow a right mouse click to trigger an event when over a button, set this parameter to **1**.

**Allowable Values:**

- 0 - (Allow all mouse buttons)
- 1 - (Allow only the left mouse button)

**Default Value:**

0

**See Also**

[Keyboard Parameters](#)

**[Keyboard]ButtonRaw**

Puts a string (ASCII character code) into the key command line when a button is pressed.

**Allowable Values:**

Any ASCII character (specified in decimal)

**Default Value:**

13

**See Also**

[Keyboard Parameters](#)

**[Keyboard]EchoPopUp**

Determines whether the keyboard input on a graphical object is echoed in a tips popup. This allows keyboard feedback local to the input point. Keyboard commands are echoed to the prompt line regardless of the value of this parameter.

**Allowable Values:**

- 0 - (Do not echo to the popup window)
- 1 - (Echo input to the popup window)

**Default Value:**

1

## See Also

[Keyboard Parameters](#)

### [Keyboard]LogExtendedDate

Determines whether the short (dd/mm/yy) or extended (dd/mm/yyyy) date format is used (in syslog.dat command logs) when interpreting the {DATE,n} log format.

#### Allowable Values:

- 0 - (Use short date format)
- 1 - (Use extended date format)

#### Default Value:

1

## See Also

[Keyboard Parameters](#)

### [Keyboard]NonPrint

Enables/disables the display of the non-print character (specified in the [\[Keyboard\]NonPrintChar](#) parameter). If you enable this parameter, the non-print character will be displayed when the key is pressed. If you disable this parameter, the non-print character will not display.

#### Allowable Values:

- 0 - (Enable)
- 1 - (Disable)

#### Default Value:

0

## See Also

[Keyboard Parameters](#)

### [Keyboard]NonPrintChar

The ASCII character code to display when a key that does not have a displayable character or Key Name is pressed.

**Allowable Values:**

0 to 255

**Default Value:**

63 ("?" character)

**See Also**

[Keyboard Parameters](#)

**[Keyboard]Type**

The keyboard type. Plant SCADA will use keys of this type in the project and keys of (default) type 0 for all commands. If the same keys are defined for type 0 and another keyboard type set by this parameter, then the key defined first in the project will execute. Keys of other types are ignored. Type 0 is used for all keyboards.

**Allowable Values:**

0 to 255

**Default Value:**

0

**See Also**

[Keyboard Parameters](#)

**LAN Parameters**

The Citect.ini file contains the following LAN parameters:

- [\[LAN\]AddressScope](#) - Enables the address scope types for IPv6 addresses.
- [\[LAN\]AddressType](#) - Enables support for IPv4 or IPv6 network types.
- [\[LAN\]AllowLegacyConnections](#) - Obsolete in v7.20.
- [\[LAN\]AllowRemoteReload](#) - Enables remote reloading of servers from a client.
- [\[LAN\]BackOffTime](#) - The time that Plant SCADA will back-off trying to send a message if the message pool is full.
- [\[LAN\] Bridge](#) - Obsolete in v7.0.
- [\[LAN\]CancelOnClose](#) - Obsolete in v7.0.
- [\[LAN\]ClientRetryTime](#) - Sets the amount of time between connection attempts by the client.
- [\[LAN\]ConnectionLogging](#) - Provides for additional traces at connection time for Plant SCADA legacy

connections.

- [\[LAN\]Delay](#) - The period (in milliseconds) to wait to optimize a send message - before sending the message.
- [\[LAN\]Disable](#) - Obsolete in v7.0.
- [\[LAN\]EarliestLegacyVersion](#) - Specify the minimum legacy version from which the current version will accept connections.
- [\[LAN\]GroupName](#) - Obsolete in v7.0.
- [\[LAN\]HeartbeatPeriod](#) - Controls how frequently a tran channel sends a heartbeat packet to the other peer.
- [\[LAN\]HeartbeatTimeout](#) - Controls how much idle time on network is accepted prior to dropping the tran connection.
- [\[LAN\]HighWaterMark](#) - Sets the number of messages waiting to be sent on a particular network connection at which the high water mark event will occur.
- [\[LAN\]KeepAliveInterval](#) - Obsolete in v2015.
- [\[LAN\]KeepAliveTime](#) - Obsolete in v2015.
- [\[LAN\]KillPiggyBackAck](#) - Obsolete in v7.0.
- [\[LAN\]LanA](#) - Obsolete in v7.0.
- [\[LAN\]ListenerRetryTime](#) - The amount of time a server will wait between attempts to listen for a connection.
- [\[LAN\]LocalNet](#) - Improves the performance of cluster switching on a client which is also a server.
- [\[LAN\]LowWaterMark](#) - After the high water mark has been reached on a particular network connection, the low water mark represents the number of messages waiting to be sent at which normal operations will resume
- [\[LAN\]NetBIOS](#) - Obsolete in v7.0.
- [\[LAN\]NetTrace](#) - Obsolete in v7.0.
- [\[LAN\]NetTraceBuf](#) - Obsolete in v7.0.
- [\[LAN\]NetTraceErr](#) - Obsolete in v7.0.
- [\[LAN\]NetTraceLog](#) - Obsolete in v7.0.
- [\[LAN\]Node](#) - The name of this Plant SCADA computer.
- [\[LAN\]Poll](#) - Obsolete in v7.0.
- [\[LAN\]ReadOnlyLegacyConnections](#) - Set to allow legacy v7.10 clients to communicate in read-only mode.
- [\[LAN\]ReadPool](#) - The number of receive message buffers used by Plant SCADA.
- [\[LAN\]RemoteTimeOut](#) - Obsolete in v7.0.
- [\[LAN\]Retry](#) - Obsolete in v7.0.
- [\[LAN\]SendTimeout](#) - Obsolete in v7.0.
- [\[LAN\]ServerLoginEnabled](#) - Obsolete in v7.20.
- [\[LAN\]SesRecBuf](#) - Obsolete in v7.0.
- [\[LAN\]SesSendBuf](#) - Obsolete in v7.0.
- [\[LAN\]Sessions](#) - The maximum number of connections across the LAN, i.e. servers x clients.
- [\[LAN\]SocketNoDelay](#) - Switches off the delay on a socket caused by the Nagle algorithm.
- [\[LAN\]TCPIP](#) - Enables the support of TCP/IP protocol.
- [\[LAN\]TimeOut](#) - Obsolete in v7.0.
- [\[LAN\]WaitBufTime](#) - The time Plant SCADA waits for a write message buffer before aborting the session.

- [\[LAN\]WritePool](#) - The maximum number of send message buffers used by Plant SCADA.

## See Also

[Parameter Categories](#)

### [LAN]AddressScope

When using an IPv6 network, this parameter enables the computer to support address scopes that are Global, Link-Local, or both.

IPv6 support can be enabled via the **Network Model** page of the Setup Wizard.

## Allowable Values:

- 0 - (Only Global)
- 1 - (Only Link-Local)
- 2 - (Both Global and Link-Local)

## Default Value:

0

## See Also

[LAN Parameters](#)

[\[LAN\]AddressType](#)

### [LAN]AddressType

This parameter enables support for IPv4, IPv6, or both for network connections between servers and clients. It helps reduce the number of open connections for the computers.

IPv6 support can be enabled via the **Network Model** page of the Setup Wizard.

## Allowable Values:

- 0 - (Only IPv4)
- 1 - (Only IPv6)
- 2 - (Both IPv4 and IPv6)

## Default Value:

0

## See Also

[LAN Parameters](#)

[\[LAN\]AddressScope](#)

### [LAN]AllowRemoteReload

This parameter provides the ability to remotely reload server processes from a client. A valid user needs to be logged in as the client to remotely reload.

## Allowable Values

- 0 - Reload can only be issued locally
- 1 - Reload can be issued locally and remotely

## Default Value:

0

## See Also

[LAN Parameters](#)

### [LAN]BackOffTime

The time that Plant SCADA will back-off trying to send a message if the message pool is full. This back-off time allows the receiver time to process messages already in the queue.

## Allowable Values:

0 to 1000 (milliseconds)

## Default Value:

1

## See Also

[LAN Parameters](#)

### [LAN]ClientRetryTime

Sets the amount of time between connection attempts by a client.

## Allowable Values

Length of time in milliseconds

### Default Value:

5000

## See Also

[LAN Parameters](#)

### [LAN]ConnectionLogging

This setting provides for additional traces at connection time for Plant SCADA legacy connections, i.e. Server to Server connections, CtAPI to Server. Traces also occur at disconnection.

This setting may be useful in diagnosing connection problems.

---

**Note:** Logging is to the Syslog.dat

---

## Allowable Values

- 0 - No additional traces at connection time
- 1 - Enable additional traces at connection time

### Default Value:

0

## See Also

[LAN Parameters](#)

### [LAN]Delay

The period (in milliseconds) to wait to optimize a send message - before sending the message. Plant SCADA will hold back the message for this delay time, so if another message is to be sent, it can be sent in the same network packet. This parameter adds a small delay, but it can reduce network traffic significantly, and so make the total response time faster.

## Allowable Values:

0 to 2000 (milliseconds)

**Default Value:**

10

**See Also**

[LAN Parameters](#)

**[LAN]EarliestLegacyVersion**

Specify the minimum legacy version from which the current version will accept connections.

Connection attempts from processes which are on or above the specified value will be accepted.

Connection attempts from processes which are below the specified value will be denied. The established connection will only be as secure as the earlier of the two connected versions.

The value for this parameter is created using the following formula:

(Product Major Version x 1000) + (Product Minor Version x 10)

For example, for version 8.20, use (8x1000) + (20x10) = 8200

---

**Note:** When using this parameter as part of an online upgrade, only 8100 (and later) is supported. When the online upgrade is complete, you should clear the setting.

**Allowable Values:**

8100 — 8500

**Default Value:**

8500

**See Also**

[LAN Parameters](#)

**[LAN]HeartbeatPeriod**

Heartbeat parameters are used to detect bad tran connections caused by unstable network or equipment failures. The heartbeat parameters do this by exchanging heartbeat messages periodically. For example, a connection between a Plant SCADA client and server may be idle if there are no requests and no tag value updates sent.

The heartbeat packet is a probe designed to elicit a response from the other process to communicate with. If the remote process is reachable and functioning, the remote process acknowledges the heartbeat transmission by returning an acknowledgment packet. If the remote process has stopped, or is unable to respond, this situation will be detected by the local peer by monitoring packets from the remote peer in order to drop the tran channel.

It does not exchange heartbeat messages when there are other valid messages on the channel to avoid unnecessary network data.

This parameter controls how frequently a tran channel sends a heartbeat packet to the other peer.

Lowering the value for HeartbeatPeriod lets the process have less interval between heartbeat packets so that the network problem detection can speed up by configuring shorter [LAN]HeartbeatTimeout value. However, it also increases the network activity on idle connections. Conversely, increasing the value for this parameter will slow down the health checking but decrease the activity on idle connections.

---

**Note:** In order to keep network failures detected in under 5 seconds, the TCPIP socket keep alive, as a parallel mechanism, has been reinstated. This mechanism is very reliable under load as it is monitored by the TCPIP layer itself. The Keep Alive Period and Keep Alive Timeout for the TCPIP socket has been set to the value of the corresponding [LAN] parameters, divided by 5, for faster detection. This means by default the period will be 1 second and the timeout will be 3 seconds, which matches the default of the keep alive in previous versions.

---

## Allowable Values

1 – 2147483647

Length of time in milliseconds. 0 means no heartbeat message will be sent to the other peer.

## Default Value:

5000 (5 seconds)

## See Also

[LAN Parameters](#)

### [LAN]HeartbeatTimeout

Heartbeat parameters are used to detect bad tran connections caused by unstable network or equipment failures. The heartbeat parameters do this by exchanging heartbeat messages periodically. For example, a connection between a Plant SCADA client and a server could be idle for a while if there are no requests and no tag value updates sent. The heartbeat packet is a probe designed to elicit a response from the other process to communicate with. If the remote process is reachable and functioning, the remote process acknowledges the heartbeat transmission by returning an acknowledgment packet. If the remote process has stopped, or is unable to respond, this situation will be detected by the local peer by monitoring packets from the remote peer in order to drop the tran channel.

To avoid unnecessary network data, heartbeat messages will not be exchanged when there are other valid messages on the channel.

This parameter controls how long the network may be idle prior to dropping the tran connection. In lowering the value for HeartbeatTimeout the process may speed up network problem detection. However, it could also lead to drops in connection due to momentary problems. Conversely, increasing the value for this parameter will slow down the health checking but the process will be able to ignore momentary issues.

---

**Note:** In order to keep network failures detected in under 5 seconds, the TCPIP socket keep alive, as a parallel mechanism, has been reinstated. This mechanism is very reliable under load as it is monitored by the TCPIP layer itself. The Keep Alive Period and Keep Alive Timeout for the TCPIP socket has been set to the value of the corresponding [LAN] parameters, divided by 5, for faster detection. This means by default the period will be 1 second and the timeout will be 3 seconds, which matches the default of the keep alive in previous versions.

---

## Allowable Values

1 – 2147483647

Length of time in milliseconds. 0 means no drop will happen.

## Default Value:

15000 (15 seconds)

## See Also

[LAN Parameters](#)

### [LAN]HighWaterMark

Sets the number of messages waiting to be sent on a particular network connection at which the high water mark event will occur. When the high water mark is reached, a hardware alarm will be raised. If this occurs on a connection made to the IO Server, the IO Server will suspend value updates being sent to subscriptions until the low water mark has been reached.

If this hardware alarm occurs irregularly (e.g.if there is a spike of activity in a large system) it may indicate that the value of this parameter needs to be increased for this system.

If this hardware alarm occurs regularly, it may indicate there is a problem with the network or with the performance of the server from which the connection has been made. It can also indicate that more value changes are occurring then the system can deal with. The deadbands of some of the variable tags may need to be increased to reduce the number of value changes in the system, or the sample period of the trends or alarm scan time may need to be increased to reduce the number of polls made to the IODevices.

---

**Note:** If this parameter is set to a value less than or equal to [\[LAN\]LowWaterMark](#) both of these parameters will be ignored and the system will operate at the default values.

---

## Allowable Values

1 to 2147483647

## Default Value

80000

## See Also

[LAN Parameters](#)

### [LAN]LocalNet

Improves the performance of cluster switching on a client which is also a server.

Using the Cicode function ClusterSetName, a client can switch from displaying information from one Plant SCADA server to display information from another Plant SCADA cluster server. If you want to do this on a client which is also a server, set this parameter to 1.

This enables direct communication between client and server, bypassing the network stack, and improving performance on a local connection. With this parameter set to 0, Plant SCADA establishes client/server communication via the network stack, even when the client and server are on the same computer.

## Allowable Values

- 0 Establish communication via network stack
- 1 Establish communication via Plant SCADA local network

## Default Value:

1

## See Also

[LAN Parameters](#)

### [LAN]LowWaterMark

After the high water mark has been reached on a particular network connection, the low water mark represents the number of messages waiting to be sent at which normal operations will resume.

If this occurred on an I/O Server, on returning to the low water mark, the I/O server will send the latest value for tags whose values have changed since the high water mark was reached and then value updates will resume.

**Note:** If this parameter is set to a value greater than or equal to [LAN]HighWaterMark both of these parameters will be ignored and the system will operate at the default values.

## Allowable Values

1 to 2147483647

## Default Value

20000

## See Also

[LAN Parameters](#)

### [LAN]ListenerRetryTime

Specifies the amount of time a server will wait before retrying to start listening on the configured IP Address and TCP Port number.

In a certain cases the configured Plant SCADAServer IP Address and Port Number may not be available for the server to start listening and accepting client connections. For example if the same IP Address and Port Number is already used by another process or the IP Address is not present on any network interface of the machine or if the network card has been unplugged. In these cases, the Plant SCADAServer will start running and in the background it will asynchronously keep trying to start listening and accepting connections on the configured IP Address and Port Number. The interval of time between two attempts to start listening and accepting connections is determined by this parameter.

As soon as the configured IP Address and Port Number are available, the Server will start listening and accepting client connections. It is recommended to configure a Server IP Address and Port Number which are available and not used by another process on the same machine.

## Allowable Values

Length of time in milliseconds

1 - 2147483647

## Default Value:

5000

## See Also

[LAN Parameters](#)

## [LAN]Node

The name of this Plant SCADA computer. Each computer should have a unique name. This name is used by the MsgOpen() function to identify each computer.

If you leave the computer name blank, Plant SCADA sets the computer name to the same name as the Windows Computer Name (as set up in the Network section of the Windows Control Panel).

**Note:** This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

## Allowable Values:

Any valid computer name

## Default Value:

Unknown

## See Also

[LAN Parameters](#)

## [LAN]ReadOnlyLegacyConnections

Set to allow connections from legacy versions to v7.20 in read-only mode. This parameter is used only if "[LAN]EarlierLegacyVersion" is enabled.

### Allowable Values

- 0 - No read only legacy client connections
- 1 - Read-only legacy client connections are enabled

### Default Value:

0

### See Also

[LAN Parameters](#)

## [LAN]ReadPool

The number of receive message buffers used by Plant SCADA. If you get the error message "Out of buffers" on a server or LAN device, increase this parameter. You might also see the message in the Plant SCADA Kernel or the SYSLOG.DAT file, "Out of buffers lan.read.pool" - if so, increase this parameter.

You should only need to increase this value if you have a large, heavily loaded network system (typically greater than 10 Plant SCADA clients, with greater than 10,000 I/O points). If you have a smaller system and you are getting the above errors, you could have a problem with your LAN hardware or configuration.

### Allowable Values:

16 to 32760

### Default Value:

1024

### See Also

[LAN Parameters](#)

## [LAN]Sessions

The maximum number of connections across the LAN, i.e. servers x clients. If you get an "Out Of Sessions" or "Out of Msg Instance, Increase[LAN]Sessions" error, increase the value of this parameter.

A Plant SCADA client uses 3 sessions for the servers (Alarms, Trends, Reports) + 1 session for each of your I/O Servers, so the default parameter setting is good for Clients.

A Server uses 1 session for each client attached, for each type of server it is. So if a computer is a Alarms, Trends, Reports, and I/O Server and you have 20 clients, the server uses  $4 \times 20 = 80$  sessions. You should also allow 50% extra sessions, because during a redundant changeover you get an extra peak loading. So in the example, allow a total of  $80 + 40 = 120$  sessions.

### Allowable Values:

16 to 4000

### Default Value:

128

### See Also

[LAN Parameters](#)

### [LAN]SocketNoDelay

Switches off the delay on a socket caused by the Nagle algorithm.

A Nagle algorithm is used to buffer packets for up to 200 milliseconds so that they can be combined and sent as a single packet. This results in larger, but less frequent, packet transmission which generally provides more efficient network bandwidth utilization.

### Allowable Values

- 0 - use the Nagle algorithm
- 1 - disable the Nagle algorithm

### Default Value:

1

### See Also

[LAN Parameters](#)

### [LAN]TCPIP

Enables support for TCPIP network connections between Servers and Clients. TCPIP should be enabled for any Station which will connect to any other Station.

### Allowable Values:

- 0 - (Disable TCP/IP protocol)

- 1 - (Enable TCP/IP protocol)

**Default Value:**

0

**See Also**

[LAN Parameters](#)

**[LAN]WaitBufTime**

The time Plant SCADA waits for a write message buffer before aborting the session. Note that this wait is a hard wait - Plant SCADA hangs while it waits for this buffer - so be careful if you use this parameter.

**Allowable Values:**

0 to 30000 (milliseconds)

**Default Value:**

20

**See Also**

[LAN Parameters](#)

**[LAN]WritePool**

The maximum number of send message buffers used by Plant SCADA. If you get the error message "Out of buffers" on a server or LAN device, increase this parameter. You might also see the message in the Plant SCADA Kernel or the SYSLOG.DAT file, "Out of buffers lan.write.pool" - if so, increase this parameter.

You should only need to increase this value if you have a large, heavily loaded network system (typically greater than 10 Plant SCADA clients, with greater than 10,000 I/O points). If you have a smaller system and you are getting the above errors, you could have a problem with your LAN hardware or configuration.

**Allowable Values:**

16 to 32760

**Default Value:**

1024

## See Also

[LAN Parameters](#)

### Language Parameters

Define whether Plant SCADA recognizes differences in case in text marked for automatic language change.

- [\[Language\]CaseSensitive](#) - Determines whether text in the language database is treated as case-sensitive.
- [\[Language\]CharSet](#) - Defines the character set to be used when the LOCAL translation is displayed at runtime.
- [\[Language\]ClientTranslateFile](#) - Determines whether translation of ASCII reports is governed by the client or the Reports Server.
- [\[Language\]CodePage](#) - Determines the codepage to be used when printing database records.
- [\[Language\]DisplayError](#) - Defines whether or not an error message will display when a local translation of a native string cannot be found in the language database.
- [\[Language\]LocalLanguage](#) - Determines the default language used for runtime text items such as alarm descriptions, button text, keyboard/alarm logs, graphic text, Cicode strings and so on.
- [\[Language\]LocalLanguageOnly](#) - Determines whether or not the language drop down list will be displayed on the login form.
- [\[Language\]SuppressWarningUnsupported](#) - Allows suppression of a compiler warning message about unsupported languages.

## See Also

[Parameter Categories](#)

### [Language]CaseSensitive

Determines whether text in the language database is treated as case sensitive. If this value is set to case sensitive, strings that differ only in case will require different local language entries in the language database, and will therefore return a different local translation at runtime.

---

**Note:** This parameter does not apply to Alarm strings. Alarm strings are always case sensitive.

---

#### Allowable Values:

- 0 - (Not case sensitive)
- 1 - (Case sensitive)

#### Default Value:

0

## See Also

[Language Parameters](#)

### [Language]CharSet

Defines the character set to be used when the LOCAL translation is displayed at runtime. If you do not specify this parameter, the runtime text may be garbled (if the local language character set differs from the default character set of the Windows installation).

Definition: Charset: Stands for "character set." A set of characters used in Windows.Charsets refer to the same collections of characters as those defined by Windows code pages.

## Allowable Values:

- 0 - ANSI ASCII
- 1 - System Default Character Set
- 128 - Japanese - Shift JIS
- 129 - Korean - Hangul
- 130 - Korean - Johab
- 134 - Chinese - simplified GB2312
- 136 - Chinese - traditional Big5
- 161 - Greek
- 162 - Turkish
- 163 - Vietnamese
- 177 - Hebrew
- 178 - Arabic
- 186 - Baltic
- 204 - Russian
- 222 - Thai
- 238 - East European

## Default Value:

1

## See Also

[Language Parameters](#)

### [Language]ClientTranslateFile

Specifies whether the Reports Server is used for translating reports containing multi-language text. If the Reports

Server does not translate a report, the client viewing the report must use the LanguageFileTranslate() function to perform translation.

---

**Note:** This parameter is used by the Reports Server. It applies only to reports configured using ASCII devices. Any other logging device will have the report translated into the local language of the Reports Server - regardless of the value of this parameter.

For example, a system has one Reports Server and one client. German is set as the local language on the Reports Server, and French on the client.

If [Language]ClientTranslateFile is set to 0 (zero), when the report is run, the Reports Server will produce the report with German translation.

If [Language]ClientTranslateFile is set to 1 (one), the Reports Server will produce the report untranslated - still containing the @( ) formatting.

To display the report on the client - in French - the LanguageFileTranslate() Cicode function must be used to perform the translation.

## Allowable Values:

- 0 - (Reports Server translates reports)
- 1 - (Reports Server does not translate reports)

## Default Value:

0

## See Also

[Language Parameters](#)

### [Language]CodePage

Defines the codepage to be used when printing database records.

Definition: Code page: An ordered set of characters of a given script in which a numeric index (code-point value) is associated with each character. This term is generally used in the context of code pages defined by Windows and can also be called a "character set" or "charset".

In the newer components of our SCADA software, more particularly the Process Analyst, the displayed texts are managed by using the Unicode standard of specifying characters.

Within the Microsoft OS, conversion facilities are routinely used between coding standards to access Microsoft's own legacy components. Part of the reason is a need for backwards compatibility for older files or programs.

### Background Information

Code pages are a carryover from pre-Unicode days, (Pre 2000) but they are still deeply embedded in many current softwares including our SCADA.

Code Pages 1250 through 1258 are commonly called the ANSI code pages.

The ANSI name is actually a misnomer they are actually Windows code pages. They are the Microsoft implementation of an early ANSI specification. They benefit from providing consistent mapping of identical characters between their languages.

The non proprietary code pages (real ANSI) became ISO 8859-1 through 16.

The frequently mentioned OEM code page usually refers to the Windows OEM Code page CP437 originating from DOS-7 days. It is still used internally in the Windows Operating System, eg..the Win32 Console; a legacy design that primarily supports US English.

The Chinese, Korean and Japanese double byte code pages are also regarded to be Microsoft OEM.

The single byte Vietnamese CP1258 and Thai CP874 code pages are spoken of as being ANSI as well as OEM.

## Allowable Values:

### SBCS (Single Byte Character Set) Codepages

- 1250 Windows Latin II (Central Europe)
- 1251 Windows (Cyrillic)
- 1252 Windows Latin I (ANSI, Western European)
- 1253 Windows (Greek)
- 1254 Windows Latin V (Turkish)
- 1255 Windows (Hebrew)
- 1256 Windows (Arabic)
- 1257 Windows (Baltic)
- 1258 Windows (Vietnam)
- 874 Windows (Thai)

### DBCS (Double Byte Character Set) Codepages

- 932 (Japanese Shift-JIS)
- 936 (Simplified Chinese GBK)
- 949 (Korean)
- 950 (Traditional Chinese Big5)

## Default Value:

The default language CodePage setting is 1252 Latin I.

1252 Latin I (Western European) is also known, correctly as Windows 1252, commonly called ANSI.

It can render US/UK English, French, Danish, German, Italian, Norwegian, Portuguese, Spanish, Swedish, and several others eg. Basque, Galician, Catalan, Malay.

## See Also

[Language Parameters](#)

### [Language]DisplayError

Defines whether or not an error message (#MESS) will display when a local translation of a particular native

string cannot be found in the language database. If you decide not to display the error message, the native string will display at runtime instead.

### Allowable Values:

- 0 - (Display the native text)
- 1 - (Display error message #MESS)

### Default Value:

0

### See Also

[Language Parameters](#)

#### [Language]LocalLanguage

Sets the language database from which the local translations of all native strings in the project will be drawn when the project is run. Native strings are those that are preceded by a @, and enclosed in brackets (e.g. @(Motor Overload)). It determines the language of runtime text items such as alarm descriptions, button text, keyboard/alarm logs, graphic text, Cicode strings etc.

This INI parameter's behaviour has changed from that of 7.20. It is no longer used to switch the language; instead use this parameter to set the default language during startup.

If this parameter is included in the Citect.ini, the language specified needs to be a language defined in the **Languages** view in Plant SCADA Studio's **Setup** activity. If not specified, English will be the default.

Be aware that neutral languages (for example, "French") and region-specific variations (for example, "French (Belgium)") will generate and use their own language database ("French.dbf" and "French(Belgium).dbf"). You cannot refer to a regional variation of a language using just the neutral language name.

Additionally, if a language is not officially supported by Plant SCADA as a native language, but is supported by Windows, before setting it in the [Language]LocalLanguage section or on user login, the language also needs to be added to the Languages view. For instance, if using traditional Chinese, the following string needs to be added: Chinese(Traditional)(Taiwan).

---

**Note:** The **Languages** view should contain every language that will be used at Runtime. If you make any changes to the list of defined languages, you will need to compile your project and restart Runtime.

---

### Allowable Values:

Any string that is defined in the language DBF file which can be recognized by Windows.

### Default Value:

English

---

**Note:** If you change this parameter after or during project configuration, you need to perform an Update Pages

---

---

operation (from the Graphics Builder), followed by a full re-compile of the project.

---

## See Also

[Language Parameters](#)

### [Language]LocalLanguageOnly

Determines whether or not the language drop down list will be displayed on the login form.

#### Allowable Values:

- 0 - language drop-down list displays
- 1 - language drop-down list does not display

#### Default Value:

0

## See Also

[Language Parameters](#)

### [Language]SuppressWarningUnsupported

Allows suppression of the compiler warning message "The specified language is not supported".

#### Allowable Values:

- 0 - The warning message is not suppressed.
- 1 - The warning message is suppressed.

#### Default Value:

0

## See Also

[Language Parameters](#)

### Memory Parameters

The Citect.ini file contains the following memory parameters:

- [\[Memory\]Free](#) - Frees memory for other Windows applications when it is not being used by Plant SCADA.
- [\[Memory\]MinPhyK](#) - Sets the minimum physical memory before Plant SCADA generates the error message "Plant SCADA Runtime low on Physical memory" and the hardware alarm "Low physical memory".
- [\[Memory\]PageLock](#) - Determines whether Plant SCADA locks the memory it uses.
- [\[Memory\]Segment](#) - Controls how Plant SCADA allocates memory from Windows.

## See Also

[Parameter Categories](#)

### [Memory]Free

Frees memory for other Windows applications when it is not being used by Plant SCADA. Plant SCADA reuses memory when it is required, but other Windows applications cannot use memory unless it is free.

#### Allowable Values:

- 0 - (Do not free until shutdown)
- 1 - (Free memory when not in use)

#### Default Value:

1

## See Also

[Memory Parameters](#)

### [Memory]MinPhyK

Sets the minimum physical memory before Plant SCADA generates the error message "Plant SCADA Runtime is low on Physical memory" and the hardware alarm "Low physical memory".

The way Windows calculates the available free memory can be unreliable under some conditions. The error message can display when you do in fact have enough free physical memory, and you might want to disable this error by setting this parameter to 0. However, you should check that you really do have enough free memory to run Plant SCADA before disabling this parameter - running out of memory can cause a severe degradation in system performance.

You should also use an 8 MB persistent swap file to give you some virtual memory. Be aware that if you use temporary memory or a large virtual memory setting, the memory check is unreliable.

#### Allowable Values:

0 (disable) to 300

**Default Value:**

300

**See Also**

[Memory Parameters](#)

**[Memory]PageLock**

Determines whether Plant SCADA locks the memory it uses. Locking memory can cause Windows to abort with a paging error.

**UNINTENDED EQUIPMENT OPERATION**

Do not change the [Memory]PageLock parameter in the Citect.ini file, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Allowable Values:**

- 0 - (Do not lock memory)
- 1 - (Lock memory and do not page to disk)

**Default Value:**

0

**See Also**

[Memory Parameters](#)

**[Memory]Segment**

Controls how Plant SCADA allocates memory from Windows. If you set this parameter to 1, Plant SCADA allocates memory in 64K segments - a faster and more efficient use of memory. If you set this parameter to 0, Plant SCADA allocates many small blocks of memory. Do not disable this parameter unless advised by Technical Support for this product.

**UNINTENDED EQUIPMENT OPERATION**

Do not change the [Memory]Segment parameter in the Citect.ini file, except on the advice of Technical

Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Allowable Values:

- 0 - (Allocate as required)
- 1 - (Allocate memory in 64K segments)

## Default Value:

1

## See Also

[Memory Parameters](#)

## MultiMonitor Parameters

The [MultiMonitors] section contains the following parameters:

## Common Parameters

- [\[MultiMonitors\]DisableAutoStart](#) - Disables the MultiMonitor functionally.
- [\[MultiMonitors\]Monitors](#) - Indicates the number of monitors the project will be displayed on.
- [\[MultiMonitors\]ScreenProfile](#) - Sets the arrangement of monitors on a client workstation.
- [\[MultiMonitors\]StartupPage<n>](#) - Determines the pages that appear when starting up a multi monitor setup.
- [\[MultiMonitors\]StartupPage<Name>](#) - Sets the pages that appear when using a particular screen profile.

## Situational Awareness Parameters

- [\[MultiMonitors\]Context<n>](#) - Allows you to share a single context with one or more screens.
- [\[MultiMonitors\]Workspaces](#) - Indicates the number of workspace pages to be shown at startup.

## See Also

[Parameter Categories](#)

### [MultiMonitors]Context<n>

Allows you to share a single context with one or more screens.

**Note:** This parameter is only valid for Situational Awareness projects.

**Allowable Values:**

Any valid context name.

**Default:**

None

The value assigned to the context is comprised of two parts:

- Node name from the navigation hierarchy
- Name of the screen

For example, Context1 = MyPlant.Area1:Primary. This will display details from Area 1 of MyPlant on the primary screen. If you wish to display the same information on a second monitor, you can add Context2 = MyPlant.Area1:Screen2. Both screens will then display the same content, but are independent of each other. You can also choose different views for each monitor by configuring screens to display different content. For example, Context1=MyPlant.Area1:Screen1, Context2=MyPlant.Area2:Screen2 and Context3=MyPlant.Area3:Screen3. In this case, each screen displays content independently.

When you link contexts, only one object can be selected between the two contexts. Linked contexts can be used to display different content for the same object based on the selection. For example, For example, Context1=MyPlant.Area1:Screen1, Screen2. Here, one screen can be used to display the graphics page with a particular equipment from the specified location while the second one can be used to display the Information Pane. The content in the Information pane will be updated for display as and when the selected equipment's state changes.

**See Also**

[MultiMonitor Parameters](#)

**[MultiMonitors]DisableAutoStart**

Used to stop the MultiMonitor functionality from being loaded automatically at startup.

---

**Note:** Use the MultiMonitorStart() Cicode function to load the MultiMonitor function if required

**Allowable Values:**

- False (0)- Enable startup loading of Multimonitor functionality
- True (1)- Disable startup loading of Multimonitor functionality

**Default Value:**

False (0)

## See Also

[MultiMonitor Parameters](#)

### [MultiMonitors]Monitors

Indicates the number of monitors configured for the project.

**Note:** This parameter is ignored if the [\[MultiMonitors\]ScreenProfile](#) parameter is used.

---

## Allowable Values:

1 to 8.

## Default Value:

1

## See Also

[MultiMonitor Parameters](#)

### [MultiMonitors]ScreenProfile

Sets the arrangement of monitors on a client workstation for a project.

## Allowable Values:

Any valid screen profile name

## Default:

None

## See Also

[MultiMonitor Parameters](#)

### [MultiMonitors]StartupPage<Name>

Determines the page that appears on the monitor that corresponds to the screen named "Screen1" in the screen profile.

On start up, Plant SCADA will compare the physical monitors on the workstation to the screens in the screen profile and attempt to map screens to monitors based on the resolution and relative position to the primary monitor/screen. If a physical monitor cannot be found, for example it has failed or been disconnected or the

configuration is incorrect, that screen will be mapped to the primary monitor. So, you may have multiple pages displayed on the primary monitor.

---

**Note:** If you don't specify a startup page for a monitor in the Setup wizard, any startup page setting in the Citect.ini file will be replaced by ><. This overrides the configuration of a monitor set up with a page in the Parameters database.

---

### Allowable Values:

Any valid page name

### Default Value:

No default. However, if no value has been specified for any of the screens, the page specified by the [\[Page\]Startup](#) parameter will be shown on the primary screen only.

### See Also

[MultiMonitor Parameters](#)

## [MultiMonitors]StartupPage<n>

Determines the pages that appear when starting up a multi monitor setup. For example, StartupPagePrimary determines the page that appears on the first monitor at startup, StartupPage1 (the name you have assigned to the parameter) determines what appears on the second monitor, and so on.

### Allowable Values:

Any valid page name

### Default Value:

If left blank, the page specified in parameter [\[Page\]Startup](#).

### See Also

[MultiMonitor Parameters](#)

## [MultiMonitors]Workspaces

Indicates the number of workspace pages that are shown. This parameter is only relevant when using Situational Awareness workspaces.

---

**Note:** If the number specified is incorrect (value too low), the Workspace functionality may not work as expected, or there may be a delay before workspaces are initialized (value too high).

---

**Allowable values:**

Values between 1 to 20

**Default:**

1

**See Also**

[MultiMonitor Parameters](#)

**ODBC Parameters**

The Citect.ini file contains the following ODBC parameter:

- [\[ODBC\]Server](#) - Enables a Plant SCADA ODBC server on the local computer.

**See Also**

[Parameter Categories](#)

**[ODBC]Server**

Enables a Plant SCADA ODBC server on the local computer. This will allow third-party applications that support ODBC to access data directly from Plant SCADA. The ODBC Server is disabled by default. You need to set this parameter to 1 to enable an ODBC server.

For more information, see the topic *Using Plant SCADA as an ODBC Server* in the main Plant SCADA documentation.

**Allowable Values:**

- 0 — (ODBC server not enabled)
- 1 — (ODBC server is enabled)

**Default Value:**

0

**See Also**

[ODBC Parameters](#)

## OID Parameters

The Citect.ini file contains the following OID parameter:

- [OID]Reset - This parameter is obsolete

## See Also

[Parameter Categories](#)

## OpcDaServer Parameters

The Citect.ini file contains the following OPC DA Server parameter:

- [\[OpcDaServer\]AutoStartByClient](#) - Enables or disables the ability to automatically start the OPC DA Server when a client connection occurs.
- [\[OpcDaServer\]RoundToFormat](#) - Specifies if tag values published to an OPC DA Server are rounded.

## See Also

[Parameter Categories](#)

### [OpcDaServer]AutoStartByClient

Allows you to automatically start the OPC DA Server when a client connection occurs.

When this parameter is enabled, a connection request from an OFS client will start Runtime Manager, then start the client and OPC DA Server processes.

When this parameter is disabled, a connection request from an OFS client will start Runtime Manager, but no processes will be launched.

An appropriate log entry will be written to the Runtime Manager log file.

Be aware that an OPC DA Server needs to be properly configured for this parameter to function correctly.

#### Allowable Values:

- 0 - disable OPC DA Server startup on OPC client connection
- 1 - enable OPC DA Server startup on OPC client connection

#### Default Value:

0

## See Also

[OpcDaServer Parameters](#)

## [OpcDaServer]RoundToFormat

Specifies if tag values published to an OPC DA Server are rounded to the format of the variable tag.

For example, if a tag value coming from an I/O device is 3.4999999999999, but the tag format is ##.##, the rounded value will be 3.50.

### Allowable Values:

- 0 - values are not rounded
- 1 - values are rounded

### Default Value:

1

## See Also

[OpcDaServer Parameters](#)

## Page Parameters

The Citect.ini file contains the following page parameters:

- [\[Page\]AddDefaultMenu](#) - Determine whether to add the default menu items to your tabbed menu bar.
- [\[Page\]Alarm](#) - The AN where the configured alarms message is displayed on each graphics page.
- [\[Page\]AlarmPage](#) - The name of the graphics page to display when calling the Cicode function PageAlarm().
- [\[Page\]AllowHScroll](#) - Defines the default behavior for horizontal scrolling.
- [\[Page\]AllowHScrollBar](#) - Defines the default behavior when displaying horizontal scroll bars.
- [\[Page\]AllowVScroll](#) - Defines the default behavior for vertical scrolling.
- [\[Page\]AllowVScrollBar](#) - Defines the default behavior when displaying vertical scroll bars.
- [\[Page\]AnmDelay](#) - When animating with a symbol set, Plant SCADA switches between frames at the frequency specified in AnmDelay.
- [\[Page\]BackgroundColor](#) - Specifies the color used to fill in the background when a page is smaller than the minimum width of a window.
- [\[Page\]BackgroundColour](#) - Obsolete in v7.20. Replaced by [Page]BackgroundColor.
- [\[Page\]BadDitheringColor](#) - Sets the dithering color for graphics elements which are dithered if the value quality is “bad”.
- [\[Page\]BadDitheringDensity](#) - Sets the dithering density for graphics elements which are dithered if the value quality is “bad”.
- [\[Page\]BadText](#) - Text Objects can be displayed as #COM type errors, or as the text overlaid with a dithered pattern if the ‘display value’ expression has “bad” quality.
- [\[Page\]BadTextBackgroundColor](#) - Sets the background color for numeric or text graphics objects to indicate “bad” quality.

- [\[Page\]ComBreak](#) - Obsolete in version 7.20.
- [\[Page\]ComBreakText](#) - Obsolete in version 7.20.
- [\[Page\]CenterStartupPage](#) - Sets the startup page (not the splash screen) to display in the center of the screen.
- [\[Page\]ControlInhibitDitheringColor](#) - Sets the dithering color for graphics elements which are dithered if their values are in ControlInhibit mode.
- [\[Page\]ControlInhibitDitheringDensity](#) - Sets the dithering density for graphics elements which are dithered if their values are in ControlInhibit mode.
- [\[Page\]ControlInhibitTextColor](#) - Sets the background color for numeric or text graphics objects to indicate that the value presented on the objects is in ControlInhibit mode.
- [\[Page\]Date](#) - The AN where the system date is displayed on each graphics page.
- [\[Page\]DefaultQualityFormat](#) - The format of the string representation of quality when a Cicode QUALITY function is used.
- [\[Page\]DefaultTimestampFormat](#) - The format of the string representation of timestamp when a Cicode TIMESTAMP function is used.
- [\[Page\]Delay](#) - This parameter has been superseded by the [\[Page\]ScanTime](#) parameter.
- [\[Page\]DelayRenderAdvancedAnimation](#) - Delays the re-drawing of graphics objects that have been changed by code in a Cicode Object. This delay will continue until the code in all Cicode Objects has been executed.
- [\[Page\]DelayRenderAll](#) - Delays the re-drawing of graphics objects so that all graphics objects on a page are re-drawn at once.
- [\[Page\]DisabledAlarm](#) - The AN where the disabled alarms message is displayed on each graphics page.
- [\[Page\]DisabledGrayTextColor](#) - Sets the color for 'Grayed' option of the 'Disable style' setting on the 'Access'->Disable Tab of the Text Properties Form.
- [\[Page\]DisabledPage](#) - The name of the graphics page to display when calling the Cicode function PageDisabled().
- [\[Page\]DynamicComBreakColour](#) - Obsolete in version 7.20.
- [\[Page\]DynamicComBreakDensity](#) - Obsolete in version 7.20.
- [\[Page\]DynamicSizing](#) - Determines whether pages can be resized at runtime.
- [\[Page\]EnableQualityToolTip](#) - Controls the quality tooltip.
- [\[Page\]ErrorDitheringColor](#) - Sets the dithering color for graphics elements which are dithered if an internal error occurs.
- [\[Page\]ErrorDitheringDensity](#) - Sets the dithering density for graphics elements which are dithered if an internal error occurs.
- [\[Page\]ErrorTextBackgroundColor](#) - Sets the background color for numeric / text graphics objects to indicate an internal error
- [\[Page\]HardwarePage](#) - The name of the graphics page to display when calling the Cicode function PageHardware().
- [\[Page\]HomePage](#) - Sets the page to display when the user clicks the Home page button on the template or calls the Cicode function PageHome()
- [\[Page\]HwAlarm](#) - The AN where the hardware alarms message is displayed on each graphics page.
- [\[Page\]IgnoreValueQuality](#) - Defines the value quality handling by graphic pages.
- [\[Page\]InheritParentScale](#) - Specifies that each page is automatically made the same size as its parent window.

- [\[Page\]KeyEcho](#) - The AN where keyboard commands are echoed.
- [\[Page\]LastAlarm](#) - The AN where the last alarm is displayed.
- [\[Page\]Logo](#) - This sets a symbol to display at the logo area (lower right corner) of the new templates.
- [\[Page\]MaintainAspectRatio](#) - Causes Plant SCADA windows to maintain their aspect ratio when resized.
- [\[Page\]MaximiseOnCreation](#) - Obsolete in version 7.20, use [\[Page\]RelocateNonResizedOnCreation](#) instead.
- [\[Page\]MaxInt](#) - Obsolete in version 7.20.
- [\[Page\]MaxLast](#) - The maximum number of pages that can be placed on the last page stack.
- [\[Page\]MaxList](#) - The maximum number of pages that can be placed on the page list stack. .
- [\[Page\]MaxRecursion](#) - Allows the specified number of recursions on PageGoto, PageDisplay, PageNext, PagePrev, PageCreate, and PageLast functions.
- [\[Page\]MaxStr](#) - Obsolete in version 7.20.
- [\[Page\]MenuDisable](#) - Determines whether the standard menu page is displayed on startup.
- [\[Page\]MenuReloadOnChange](#) - Enables the menu to be reloaded at runtime if list has been updated.
- [\[Page\]MenuShutdown](#) - Determines whether the shutdown button is displayed on the standard menu page.
- [\[Page\]Name](#) - The AN where the Page Name is displayed.
- [\[Page\]OriginAdjust](#) - Adjusts the display of windows that are either partially or completely off screen.
- [\[Page\]OverrideDitheringColor](#) - Sets the dithering color for graphics elements which are dithered if their values are override ("forced").
- [\[Page\]OverrideDitheringDensity](#) - Sets the dithering density for graphics elements which are dithered if an internal error occurs.
- [\[Page\]OverrideTextBackgroundColor](#) - Sets the background color for numeric / text graphics objects to indicate that the value presented on the objects is override ("forced").
- [\[Page\]PagesReloadOnChange](#) - Enables the list of pages to be reloaded at runtime if list has been updated.
- [\[Page\]PrintPage](#) - Sets a custom Cicode function to print the currently displayed page when the user clicks the Print page button on the template or calls the Cicode function PagePrint().
- [\[Page\]ProcessAnalystPage](#) - Sets the page to display when the user calls the Cicode function PageProcessAnalyst() without specifying the page argument.
- [\[Page\]ProcessAnalystPopupPage](#) - Sets the page to pop up when the user calls the Cicode function ProcessAnalystWin() or ProcessAnalystPopup() without specifying the page argument.
- [\[Page\]Prompt](#) - The AN where prompt and error messages are displayed.
- [\[Page\]RangeCheck](#) - Determines if range error messages are displayed.
- [\[Page\]RelocateNonResizedOnCreation](#) - Determines whether windows created with the Cicode function WinNewAt are automatically maximized.
- [\[Page\]ScaleTextToMax](#) - Causes text on pages to have their text scaled to the maximum.
- [\[Page\]ScanTime](#) - Determines how often (in milliseconds) I/O device data is read in order to update the I/O tags on a page.
- [\[Page\]ShowBadText](#) - Text Objects can be displayed as #BAD text, or as the text overlaid with a dithered pattern if the "display value" expression has "bad" quality.
- [\[Page\]ShowErrorText](#) - Text Objects can be displayed as #COM type errors, or as the text overlaid with a dithered pattern if the 'display value' expression has "uncertain" quality
- [\[Page\]ShowUncertainText](#) - Text Objects can be displayed as #UNC text, or as the text overlaid with a

dithered pattern if the "display value" expression has "uncertain" quality.

- [\[Page\]Splash](#) - Specify the name of the page to use for the Splash Screen.
- [\[Page\]SplashTimeout](#) - Milliseconds for the Splash Screen to remain displayed.
- [\[Page\]SplashWinName](#) - Specify the label of the Splash Window for use with the Cicode function WinNumber().
- [\[Page\]Startup](#) - The Page Name of the graphics page to display when the display system starts up.
- [\[Page\]StartUpCancel](#) - Determines whether the Cancel button displays on the Startup message Box.
- [\[Page\]StartupDelay](#) - Milliseconds between when Splash Screen and Start Screen are displayed.
- [\[Page\]StartupHeight](#) - Height of the Start Page on the main display monitor.
- [\[Page\]StartupMode](#) - Specify the window mode of Startup page on main display monitor.
- [\[Page\]StartupWidth](#) - Width of the Start Page on the main display monitor.
- [\[Page\]StartupWinName](#) - Specify the label of the Start Window for use with the Cicode function WinNumber().
- [\[Page\]StartupX](#) - X coordinate of the Start Page on the main display monitor.
- [\[Page\]StartupY](#) - Y coordinate of the Start Page on the main display monitor.
- [\[Page\]SOEPAGE](#) - The name of the graphics page to display when calling the Cicode function PageSOE().
- [\[Page\]SummaryPage](#) - The name of the graphics page to display when calling the Cicode function PageSummary().
- [\[Page\]Time](#) - The AN where the system time is displayed on each graphics page.
- [\[Page\]Tip](#) - The AN where tool tips are displayed on each graphics page.
- [\[Page\]TipHelp](#) - Determines whether tool tips are displayed.
- [\[Page\]TipTimeOut](#) - The time delay between when the mouse pointer stops within an AN area (i.e. over an object) and a tool tip displays (if the object at the AN has text configured for a tool tip).
- [\[Page\]Title](#) - The AN where the Page Title of the graphics page is displayed.
- [\[Page\]UncertainDitheringDensity](#) - Sets the dithering density for graphics elements which are dithered if the value quality is "uncertain".
- [\[Page\]UncertainDitheringColor](#) - Sets the dithering color for graphics elements which are dithered if the value quality is "uncertain".
- [\[Page\]UncertainText](#) - Text Objects can be displayed as #COM type errors, or as the text overlaid with a dithered pattern if the 'display value' expression has "uncertain" quality.
- [\[Page\]UncertainTextBackgroundColor](#) - Sets the background color for numeric or text graphics objects to indicate "uncertain" quality.
- [\[Page\]WaitForValidData](#) - Specifies whether the animation system will attempt to wait for valid data from all subscriptions required to draw a graphics page before it is animated.
- [\[Page\]ValueAbnormalPattern](#) - Displays a diagonal pattern across a graphics object when the corresponding device goes offline, that is, it loses communication.
- [\[Page\]Windows](#) - The maximum number of windows that can be open simultaneously.
- [\[Page\]WinTitle](#) - The format of the window title banner.

## See Also

[Parameter Categories](#)

### [Page]AddDefaultMenu

Determines whether to add the default menu items to your tabbed menu bar.

**Note:** This parameter is only available to pages based on the Tab\_Style templates

The default menu consists of the following tabs:

- Pages - contains buttons to display non-system pages configured in your project.
- Alarms - contains buttons to display the default alarm pages including: Alarm, Disabled, Summary and Hardware pages.
- Trends - contains buttons to display the default Process Analyst pages including: ProcessAnalyst, !ProcessAnalystPopup

Order of default menu items is 100. The items in the default menu are merged with and ordered against your configured menu items at runtime.

## Allowable Values:

- 0 - Do not add default menu items, menu is populated according to what is defined in the menu configuration database
- 1 - Add default menu if nothing is defined in menu configuration form for the generic pages (i.e. menu configuration record with blank page field).
- 2 - Add default menu even if menus are configured

## Default Value:

1

## See Also

[Page Parameters](#)

### [Page]Alarm

**Note:** This parameter is not used for this version of Plant SCADA. It is included for backward compatibility with versions prior to v3.00.

The AN where the configured alarms message is displayed on each graphics page.

## Allowable Values:

0 (Do not display) or 1 to 8192

**Default Value:**

5

**See Also**

[Page Parameters](#)

**[Page]AlarmPage**

The name of the graphics page to display when you call up an alarm page via the Cicode function PageAlarm().

The alarm page is used to display the last occurrence of any configured alarms.

The alarm page is commonly built on the default alarm page template in the style that matches your project but may be any valid page in the project. If the specific page does not exist, a dialog box will appear to alert the user.

**Allowable Values:**

Valid page name

**Default Value:**

Alarm

**See Also**

[Page Parameters](#)

**[Page]AllowHScroll**

This parameter defines the default behavior for horizontal scrolling. When set to 1, scrolling is enabled. When set to 0, scrolling is disabled.

**Allowable Values:**

1 or 0

**Default Value:**

1

**See Also**

[Page Parameters](#)

**[Page]AllowHScrollBar**

This parameter defines the default behavior when displaying horizontal scroll bars. When set to 1, scroll bars are displayed if the window size dictates they are required. When set to 0, scroll bars are hidden.

**Allowable Values:**

1 or 0

**Default Value:**

1

**See Also**

[Page Parameters](#)

**[Page]AllowVScroll**

This parameter defines the default behavior for vertical scrolling. When set to 1, scrolling is enabled. When set to 0, scrolling is disabled.

**Allowable Values:**

1 or 0

**Default Value:**

1

**See Also**

[Page Parameters](#)

**[Page]AllowVScrollBar**

This parameter defines the default behavior when displaying vertical scroll bars. When set to 1, scroll bars are displayed if the window size dictates they are required. When set to 0, scroll bars are hidden.

**Allowable Values:**

1 or 0

**Default Value:**

1

**See Also**[Page Parameters](#)**[Page]AnmDelay**

When animating symbols, Plant SCADA switches between frames at the frequency specified in AnmDelay. See the DspSymAnm() Cicode function for information about animating symbols.

**Allowable Values:**

100 to 10000 (milliseconds)

**Default Value:**

500

**See Also**[Page Parameters](#)**[Page]BackgroundColor**

This parameter specifies the color used to fill in the background when a page being displayed is smaller than the minimum width of a window.

The minimum width for a window is 132 pixels. The minimum page width required to fill this window therefore becomes **132 pixels - 2 \* <border width in pixels>**. In Plant SCADA, the border width of a Window is determined by the function **WinNewAt**, which specifies the window type used via the **Mode** argument. The following table shows the minimum page width supported by each different display mode:

WinNewAt (Mode)	Border size	Minimum page width
0 = Normal window	4	124
4 = No resize	3	126
8 = No icons	3	126
16 = No caption	1	130

If a page you are trying to display is smaller than the relevant width above, it will display left aligned with the remaining space filled by the color specified by this parameter.

**Note:** This parameter should only be used for existing pages that cannot be redrawn. New pages should be made

---

larger than these minimum widths.

---

## Allowable Values:

0x000000 to 0xFFFFFFF

Colors defined as an RGB value. For example:

- 0x000000 - (Black)
- 0x000080 - (Blue)
- 0x008000 - (Green)
- 0x008080 - (Cyan)
- 0x800000 - (Red)
- 0x800080 - (Magenta)
- 0x808000 - (Brown)
- 0xBFBFBF - (Grey)
- 0x7F7F7F - (Dark Grey)
- 0x0000FF - (Light blue)
- 0x00FF00 - (Light green)
- 0x00FFFF - (Light cyan)
- 0xFF0000 - (Light red)
- 0xFF00FF - (Light Magenta)
- 0xFFFF00 - (Yellow)
- 0xFFFFFFFF - (White)

## Default Value:

0xBFBFBF (Grey)

## See Also

[Page Parameters](#)

### [Page]BadDitheringColor

Sets the dithering color for graphics elements which are dithered if the value quality is "bad" if the display expression contains an "unqualified tagname".

Refer to the topic [Tag Extensions](#) for information regarding unqualified tags.

## Allowable Values:

-1 – indication off

0x000000 to 0xFFFFFFF (The RGB color value in hexadecimal)

**Default Value:**

0x000000 (Black)

**See Also**

[Page Parameters](#)

**[Page]BadDitheringDensity**

Sets the dithering density for graphics elements which are dithered if the value quality is "bad".

**Allowable Values:**

- 1 - (Highest Density)
- 2
- 3
- 4 - (Lowest Density)

**Default Value:**

2

**See Also**

[Page Parameters](#)

**[Page]BadText**

Defines the text which appears after "#" character when the 'display value' expression has "bad" quality.  
This type of indication is active only when [\[Page\]ShowBadText](#) = 1 and [\[Page\]IgnoreValueQuality](#) = 0.

**Allowable Values:**

Any string.

**Default Value:**

BAD

**See Also**

[Page Parameters](#)

## [Page]BadTextBackgroundColor

Sets the background color for numeric or text graphics objects to indicate "bad" quality if the display expression contains an "unqualified tagname".

Refer to the topic [Tag Extensions](#) for information regarding unqualified tags.

### Allowable Values:

-1 – indication off

0x000000 to 0xFFFF (The RGB color value in hexadecimal)

### Default Value:

0x800080 - (Magenta)

**Note:** The parameter needs to be set to 2 for this parameter to be observable.

---

### See Also

[Page Parameters](#)

## [Page]CenterStartupPage

Defines if the startup page (not the splash screen) is placed in the center of the screen, otherwise it will display in the top left of the screen.

### Allowable Values:

- 0 - Display startup page in the top left of the screen
- 1- Display startup page in the center of the screen

### Default Value:

0

### See Also

[Page Parameters](#)

## [Page]ControlInhibitDitheringColor

Sets the dithering color for graphics elements which are dithered if their values are in ControlInhibit mode. This type of indication is active only when [\[Page\]IgnoreValueQuality](#) = 0 or 2.

**Allowable Values:**

-1 – indication off  
0x000000 to 0xFFFF (The RGB color value in hexadecimal)

**Default Value:**

0x000000 (Black)

**See Also**

[Page Parameters](#)

**[Page]ControllInhibitDitheringDensity**

Sets the dithering density for graphics elements which are dithered if their values are in ControllInhibit mode. This type of indication is active only when [\[Page\]IgnoreValueQuality](#) = 0 or 2.

**Allowable Values:**

- 1 - (Highest Density)
- 2
- 3
- 4 - (Lowest Density)

**Default Value:**

2

**See Also**

[Page Parameters](#)

**[Page]ControllInhibitTextColor**

Sets the background color for numeric or text graphics objects to indicate that the value presented on the objects is in ControllInhibit mode. This type of indication is active only when [\[Page\]IgnoreValueQuality](#) = 2.

**Allowable Values:**

-1 – indication off  
0x000000 to 0xFFFF (The RGB color value in hexadecimal)

**Default Value:**

0x008080 - (Cyan)

**See Also**

[Page Parameters](#)

**[Page]Date**

---

**Note:** This parameter is not used for this version of Plant SCADA. It is included for backward compatibility with versions prior to v3.00.

---

The AN where the system date is displayed on each graphics page.

**Allowable Values:**

0 (Do not display) or 1 to 8192

**Default Value:**

10

**See Also**

[Page Parameters](#)

**[Page]DefaultQualityFormat**

This parameter defines the format of the string representation of quality when a Cicode QUALITY function is used to initialize quality fields, or to extract a specific quality field to be displayed on graphical pages.

**Allowable Values:**

Any valid "part" parameter value for the function QualityToStr.

- -2: Short representation in the format <General Quality> [-<Quality Substatus>]
- -1: Full representation in the format <General Quality> [Override] [Control Inhibit] – <Quality Substatus>
- 0: <General Quality>
- 1: <Quality Substatus>
- 2: <Quality Limit>
- 3: <Extended Quality Substatus>
- 4: <Quality Override>
- 5: <Control Inhibit>

**Default Value:**

-1

**See Also**

[Page Parameters](#)

**[Page]DefaultTimestampFormat**

This parameter defines the format of the string representation of timestamp when a Cicode TIMESTAMP function is used to initialize timestamp fields, or to extract a specific timestamp field to be displayed on graphical pages.

**Allowable Values:**

Any valid "format" parameter value for the function TimestampToStr.

- 0 – Short time format, hh:mm
- 1 – Long time format, hh:mm:ss
- 2 – Short date format, dd/MM/yyyy
- 3 – Long date format, dddd, dd MMMM yyyy
- 4 – Short date & short time format, dd/MM/yyyy hh:mm
- 5 – Short date & long time format, dd/MM/yyyy hh:mm:ss
- 6 – Long date & short time format, dddd, dd MMMM yyyy hh:mm
- 7 – Long date & long time format, dddd, dd MMMM yyyy hh:mm:ss
- 8 – Month day format, dd MMMM
- 9 – Year month format, MMMM yyyy
- 10 – RFC1123 format, ddd, dd MMM yy hh:mm:ss GMT
- 11 – Sortable date time format, yyyy-MM-ddThh:mm:ss
- 12 – Short universal format, yyyy-MM-dd hh:mm:ss
- 13 – Long universal format, dddd, dd MMMM yyyy hh:mm:ss
- 14 – Long Time & millisecond, hh:mm:ss.fff

**Default Value:**

4

**See Also**

[Page Parameters](#)

## [Page]Delay

This parameter has been superseded by the [\[Page\]ScanTime](#) parameter.

## See Also

[Page Parameters](#)

### [Page]DelayRenderAdvancedAnimation

---

**Note:** If you have not changed the [\[Page\]DelayRenderAll](#) parameter from its default (TRUE), then you should not need to change this parameter.

Delays the re-drawing of graphics objects that have been changed by code in a Cicode Object. This delay will continue until the code in all Cicode Objects has been executed. The associated graphics objects will then be re-drawn together. This delay will occur every time the page is updated (see the [\[Page\]ScanTime](#) parameter).

## Allowable Values:

- 1 (Delay enabled)
- 0 (Delay disabled)

## Default Value:

1

## See Also

[Page Parameters](#)

### [Page]DelayRenderAll

Delays the re-drawing of graphics objects so that all graphics objects on a page are re-drawn at once (Plant SCADA pauses to wait for all underlying Cicode to execute before re-drawing the graphics objects). This delay will occur every time the page is updated. This parameter is intended to enhance page update times (for pages with numerous or complex objects). Be careful not to mistake this delay for inactivity during runtime.

## Allowable Values:

- 1 (Delay enabled - objects re-drawn simultaneously)
- 0 (Delay disabled - objects re-drawn individually)

## Default Value:

1

## See Also

[Page Parameters](#)

### [Page]DisabledAlarm

---

**Note:** This parameter is not used for this version of Plant SCADA. It is included for backward compatibility with versions prior to v3.00.

---

The AN where the disabled alarms message is displayed on each graphics page.

#### Allowable Values:

0 (Do not display) or 1 to 8192

#### Default Value:

7

## See Also

[Page Parameters](#)

### [Page]DisabledGrayTextColor

Sets the color used when 'Grayed' is selected for the **Disable style** setting on the **Access/Disable** tab of the Text Properties dialog.

#### Allowable Values:

0x000000 to 0xFFFFFFF

Colors defined as an RGB value. For example:

- 0x000000 - (Black)
- 0x000080 - (Blue)
- 0x008000 - (Green)
- 0x008080 - (Cyan)
- 0x800000 - (Red)
- 0x800080 - (Magenta)
- 0x808000 - (Brown)
- 0xBFBFBF - (Grey)
- 0x7F7F7F - (Dark Grey)
- 0x0000FF - (Light blue)
- 0x00FF00 - (Light green)

- 0x00FFFF - (Light cyan)
- 0xFF0000 - (Light red)
- 0xFF00FF - (Light Magenta)
- 0xFFFF00 - (Yellow)
- 0xFFFFFFFF - (White)

**Default Value:**

Standard Windows Disabled Grey Text Color

**See Also**

[Page Parameters](#)

**[Page]DisabledPage**

The name of the graphics page to display when you call up a disabled alarm page via the Cicode function PageDisabled.

The disabled alarm page is used to display alarms in the system that have been disabled.

The disabled alarm page is commonly built on the default disabled alarm page template in the style that matches your project but may be any valid page in the project. If the specific page does not exist, a dialog box will appear to alert the user.

**Allowable Values:**

Valid page name

**Default Value:**

Disabled

**See Also**

[Page Parameters](#)

**[Page]DynamicSizing**

Determines whether pages can be resized at runtime. This option overrides the resolution specified for the page.

If dynamic sizing is enabled, you can resize any page by:

- Clicking and dragging on the window frame;
- Maximizing the window;
- Minimizing the window to an icon; or
- Restoring the window to its original state.

You can use the **[Page]InheritParentScale** parameter to specify that each page automatically uses the same scale as its parent. For example, if you increase the size of Page A by 10%, then call Page B, Page B will automatically become 10% larger than it was when it was configured.

Note the following:

- Even if you disable dynamic page resizing, Plant SCADA windows can still be made smaller - if the window is smaller than the page, scroll bars will display to allow you to view the whole page.
- If you have the **[Animator]FullScreen** parameter set, your page will take up the entire screen, and it will have no title bar. In this mode, you cannot resize the window.
- The maximum size of your pages is restricted to that of a full screen Windows application.

## Allowable Values:

- 1 - (Dynamic Resizing ENABLED)
- 0 - (Dynamic Resizing DISABLED)

## Default Value:

1

## See Also

[Page Parameters](#)

## [Page]EnableQualityToolTip

Set by default this parameter controls the quality tooltip that displays the value, quality and timestamp for the current subscription of the tag element or element item.

This feature applies to text / numeric display objects and symbol set objects. The quality information that is displayed is the result of the evaluation of the main expression for the object e.g. for text objects this is the 'display value' expression and will ignore the scale, fill or movement expressions.

The quality information that is displayed is the result of the evaluation of the main expression for the object e.g. for text objects this is the "Display Value" expression and will ignore the scale, fill or movement expressions. The expression has to directly, and in each page, refresh the cycle of reading at least one external tag, ex. "Tag1", "Tag1 + Tag2" or "QualityToStr(Tag1)", but not "10" or "sin(1.0)". Moreover, a reference to the tag cannot refer to the tag's items, so for example "Tag1.v + Tag2.v" is not going to show a tooltip.

When values displayed by text / numeric display objects are not visible because of error codes, such as #COM, their tooltips will include values as well. Tooltips defined in Graphics Builder have higher priority than quality tooltips.

## Allowable Values:

- 0 – Quality tooltip is disabled.
- 1 – Quality tooltip is enabled.

**Default Value:**

1

**See Also**[Page Parameters](#)**[Page]ErrorDitheringColor**

Sets the dithering color for graphics elements which are dithered if an internal error occurs. This type of indication is active only when IgnoreValueQuality = 0.

**Allowable Values:**

- -1 – indication off
- 0x000000 to 0xFFFFFFF (the RGB color value in hexadecimal)

**Default Value:**

0x000000 (Black)

**See Also**[Page Parameters](#)**[Page]ErrorDitheringDensity**

Sets the dithering density for graphics elements which are dithered if an internal error occurs. This type of indication is active only when IgnoreValueQuality= 0.

**Allowable Values**

- 1 - (Highest Density)
- 2
- 3
- 4 - (Lowest Density)

**Default Value:**

2

## See Also

[Page Parameters](#)

### [Page]ErrorTextBackgroundColor

Sets the background color for numeric / text graphics objects to indicate an internal error. The error can be due to several sources such as the Cicode execution system or IO subsystems. In some cases errors are caused in the same way as tag quality, and they are redundant to tag quality. This type of indication is active only when IgnoreValueQuality= 2.

#### Allowable Values:

-1 – indication off  
0x000000 to 0xFFFF (The RGB color value in hexadecimal)

#### Default Value:

0xFF0000 - (Light Red)

## See Also

[Page Parameters](#)

### [Page]HomePage

This sets the page to display when the user clicks the Home page button on the template or calls the Cicode function PageHome().

---

**Note:** If the user creates a page based on the new Tab\_Style templates, the button that displays the home page will be disabled if the page specified by [Page]HomePage does not exist.

---

#### Allowable Values:

The name of any valid page configured in the project

#### Default Value:

NONE

## See Also

[Page Parameters](#)

## [Page]HardwarePage

The name of the graphics page to display when you call up a hardware alarm page via the Cicode function PageHardware().

The hardware alarm page is used to display alarms that are automatically generated by the system to indicate abnormal operation.

The hardware alarm page is commonly built on the default hardware alarm page template in the style that matches your project but may be any valid page in the project. If the specific page does not exist, a dialog box will appear to alert the user.

### Allowable Values:

Valid page name

### Default Value:

Hardware

### See Also

[Page Parameters](#)

## [Page]HwAlarm

---

**Note:** This parameter is not used for this version of Plant SCADA. It is included for backward compatibility with versions prior to v3.00.

The AN where the hardware alarms message is displayed on each graphics page.

### Allowable Values:

0 (Do not display) or 1 to 8192

### Default Value:

6

### See Also

[Page Parameters](#)

## [Page]IgnoreValueQuality

You can use tag elements (such as "Tag1", "Tag1.Field") and tag element items (such as "Tag1.Field.Q") in Cicode expressions. However, by default, when a tag element or an element item is not accessible, or the element quality is not "Good", graphic objects will indicate that state by displaying some alert messages or dithering.

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

If you change the value of this parameter to 1, no dithering or background color will visually identify that an element is not Good or is not accessible, even if the parameters to display such an identifier have been set elsewhere. In such a case the operator may operate the plant inappropriately by relying on incorrect data.

Use the QualityIsOverride() Cicode function or similar method to identify if a tag element has been overridden.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

However, you have the option to disable the default behavior and to display values even when the element quality is not "Good". This option is controlled by the IgnoreValueQuality Cicode task context attribute, which provides equivalent functionality to [Page] ComBreak, which is now obsolete .

*Allowable Values:*

- 0 - The Control Client supports the existing behaviour (#COM) except dithering (dots) will be not shown on text objects when an error is indicated by #<Error Message> (refer to table of error codes below).
  - 1 - The quality of tag subscriptions will be completely ignored. This is equivalent to having previously used the (now obsolete) [Page] ComBreak = 0. The numeric or text objects will display a "bad" quality value and animation will continue. No dithering will be shown.
  - 2 - The text or numeric objects will display "bad" quality values. The background color of the text / numeric objects will change to indicate the quality status. The background colors for QUAL\_BAD and QUAL\_UNCR are defined by [\[Page\]BadTextBackgroundColor](#) and [\[Page\]UncertainTextBackgroundColor](#) Citect.ini parameters.
- If **[Page]IgnoreValueQuality=0** the following short alert messages may be displayed on graphical pages when the Quality of a tag is 'NOT GOOD'. Refer to Cicode and General Errors for more information.

ERROR	ERROR Number	Parameter Set	
<b>ShowUncertainText=0, ShowBadText=0, ShowErrorText=1</b>	>(Default) <b>ShowUncertainText=1, ShowBadText=1, ShowErrorText=1</b>		
Genie not Associated	323	#ASS	#ASS
Divide by zero	273	#DIV/0	#DIV/0
Cicode stack overflow	295	#STACK	#STACK
Out of memory	272	#MEM	#MEM
Format overflow	344	#OVR	#OVR
The unit the tag is on has not been connected to yet	32	#WAIT	#WAIT
Waiting for initial data. Property not ready	423 432	#PEND	#PEND

ERROR	ERROR Number	Parameter Set	
The tag is not known by the server	53	#COM	#COM
Tag not found	424		
Cluster not specified	402		
Cluster not found	403		
Cluster name and tag mismatch	404		
Cluster not connected	405		
No server of type on cluster	418		
No connector	425		
No server could be found	281		
No Error	0	#OK	#BAD/#UNC
Value is out of range	257	#RANGE	#BAD/#UNC
Tag in Last Usable Value	525	#COM	#BAD/#UNC
General software error	256	#COM	#BAD/#UNC
General software error	256	#ERR	#BAD/#UNC

**Default Value:**

0

You can override the IgnoreValueQuality setting for a particular page by using the **Ignore Quality** field on the Page Properties dialog in Graphics Builder.

---

**Note:** The quality of tags referenced by items, ex. Tag1.v or Tag1.Field.t, is GOOD and its timestamps are 0 (INVALID TIMESTAMP). Therefore they give no visual indication of quality, error or change in handling state such as control inhibit or override mode regardless of the setting used for the [Page]IgnoreValueQuality parameter.

**See Also**

[Page Parameters](#)

**[Page]InheritParentScale**

Specifies that each page automatically uses the same scale as its parent window (i.e. the window it was called from). This applies even if the parent window has already been dynamically resized. For example, if you increase the size of Page A by 10%, then call Page B, Page B will automatically become 10% larger than it was when it was configured.

**Allowable Values:**

- 1 (Pages will use the same scale as the parent)
- 0 (Pages will default to the resolution specified for the page)

**Default Value:**

1

**See Also**[Page Parameters](#)**[Page]KeyEcho**

The AN where keyboard commands are echoed. Set to zero (0) to disable.

**Allowable Values:**

0 to 2000

**Default Value:**

1

**See Also**[Page Parameters](#)**[Page]LastAlarm**

---

**Note:** This parameter is not used for this version of Plant SCADA. It is included for backward compatibility only.

The AN where the last alarm is displayed.

**Allowable Values:**

1 to 8192

**Default Value:**

11

## See Also

[Page Parameters](#)

### [Page]Logo

This sets a symbol to display in the logo area of the templates.

Logo size:

- Situation Awareness Templates: less than or equal to 190 x 48 pixels.
- StruxureWare Templates: less than or equal to 230 x 50 pixels.
- Tab Style Projects: less than or equal to 103 x 20 pixels.

## Allowable Values:

Name of the symbol in the format of library.symbol, for example, global.logo

## Default Value:

<None> (The default product logo will be used)

## See Also

[Page Parameters](#)

### [Page]MaintainAspectRatio

This parameter is a boolean, defaulting to 1 (true). When this parameter is set to 1, Plant SCADA windows will maintain their aspect ratios during resizing. When set to 0, windows ignore aspect ratios during resizing, allowing the window to change to the exact required dimensions.

There is also a new page mode which can be specified when you call the Cicode function WinNewAt to create a new window. This new mode will be ignored if [Page]MaintainAspectRatio has been set to 0, every page will not maintain its aspect ratio on resizing. But when [Page]MaintainAspectRatio is set to 1, this new mode is taken into account. In this situation, if 4096 is added to your mode argument passed into WinNewAt, the new window will NOT maintain the aspect ratio during resizing.

## Allowable Values:

0 or 1

## Default Value:

1

## See Also

[Page Parameters](#)

### [Page]MaxInt

The maximum number of page-based integers that can be stored in an array. Page-based variables are stored in an array, local to each display page. See the PageSetInt() Cicode function.

#### Allowable Values:

0 to 100

#### Default Value:

100

## See Also

[Page Parameters](#)

### [Page]MaxLast

The maximum number of pages that can be placed on the last page stack. See the PageLast() Cicode function for information about displaying pages from the last page stack.

#### Allowable Values:

1 to 100

#### Default Value:

10

## See Also

[Page Parameters](#)

### [Page]MaxList

The maximum number of pages that can be placed on the page list stack. See the PageListDisplay() Cicode function for information about displaying pages from the page list stack. When the number of pages in the list reaches the maximum limit, the last page in the list will be replaced with the new page.

**Allowable Values:**

1 to 500

**Default Value:**

25

**See Also**

[Page Parameters](#)

**[Page]MaxRecursion**

Allows the specified number of recursions on PageGoto, PageDisplay, PageNext, PagePrev, PageCreate, and PageLast functions.

**Allowable Values:**

1 to 30

**Default Value:**

5

**See Also**

[Page Parameters](#)

**[Page]MaxStr**

The maximum number of page-based strings that can be stored in an array. Page-based variables are stored in an array, local to each display page. See the PageSetStr() Cicode function.

**Allowable Values:**

0 to 100

**Default Value:**

2

**See Also**

[Page Parameters](#)

**[Page]MenuDisable**

Determines whether the standard menu page is displayed on startup.

**Allowable Values:**

- 0 - Enable menu page
- 1 - Disable menu page

**Default Value:**

0

**See Also**

[Page Parameters](#)

**[Page]MenuReloadOnChange**

This parameter is set by default. Enables the list of menu items to be reloaded at Plant SCADA runtime process if menu has been updated. If set to 0, the menu will not be reloaded until the runtime has been restarted.

**Allowable Values:**

- 0 - Menu will not be reloaded at runtime.
- 1 - Menu will be reloaded at runtime.

**Default Value:**

1

**See Also**

[Page Parameters](#)

**[Page]MenuShutdown**

Determines whether the shutdown button is displayed on the standard menu page.

**Allowable Values:**

- 0 - (Do not display button).
- 1 - (Display shutdown button).

**Default Value:**

1

**See Also**

[Page Parameters](#)

**[Page]Name**

**Note:** This parameter is not used for this version of Plant SCADA. It is included for backward compatibility only.

The AN where the Page Name is displayed.

**Allowable Values:**

0 (Do not display) or 1 to 8192

**Default Value:**

13

**See Also**

[Page Parameters](#)

**[Page]OriginAdjust**

The [Page]OriginAdjust parameter controls the display of windows that are either partially or completely offscreen.

If this flag is set, offscreen windows display at the top left of the screen when restored or maximized.

**Allowable Values:**

- 0 - (Do not adjust window display)
- 1 - (Adjust window display)

**Default Value:**

1

**See Also**

[Page Parameters](#)

## [Page]OverrideDitheringColor

Sets the dithering color for graphics elements which are dithered if their values are override ("forced"). This type of indication is active only when [Page]IgnoreValueQuality = 0 or 2.

### Allowable Values:

- 1 – indication off
- 0x000000 to 0xFFFF (The RGB color value in hexadecimal)

### Default Value:

0x000000 (Black)

### See Also

[Page Parameters](#)

## [Page]OverrideDitheringDensity

Sets the dithering density for graphics elements which are dithered if an internal error occurs. This type of indication is active only when [Page]IgnoreValueQuality = 0 or 2.

### Allowable Values:

- 1 - (Highest Density)
- 2
- 3
- 4 - (Lowest Density)

### Default Value:

2

### See Also

[Page Parameters](#)

## [Page]OverrideTextBackgroundColor

Sets the background color for numeric or text graphics objects to indicate that the value presented on the objects is override ("forced"). This type of indication is active only when [Page]IgnoreValueQuality = 2.

**Allowable Values:**

- -1 – indication off
- 0x000000 to 0xFFFFFFF (The RGB color value in hexadecimal)

**Default Value:**

0xFFFFFFF - (White)

**See Also**

[Page Parameters](#)

**[Page]PagesReloadOnChange**

This parameter is set by default. Enables the list of pages to be reloaded at Plant SCADA runtime process if list has been updated. If set to 0, the list of pages will not be reloaded until the runtime has been restarted.

**Allowable Values:**

- 0 - Page list not reloaded
- 1 - Page list (if updated) reloaded

**Default Value:**

1

**See Also**

[Page Parameters](#)

**[Page]PrintPage**

This sets a custom Cicode function to print the currently displayed page when the user clicks the Print page button on the template or calls the Cicode function PagePrint().

---

**Note:** It is possible to override this setting on a per page basis by using the page environment variable "PrintPage". Refer to 'Using the Tab\_Style Page Templates > Creating a New Project > Using Environment Variables in Tab\_Style Templates' for more information regarding page environment variables.

---

**Note:** This parameter is available only to projects based on Tab\_Style templates.

---

**Allowable Values:**

"?" followed by a valid Cicode function name and comma separated arguments, for example, ?WinPrint LPT1;,0,0,0

**Default Value:**

None

**See Also**

[Page Parameters](#)

**[Page]ProcessAnalystPage**

Sets the page to display when the user calls the Cicode function PageProcessAnalyst() without specifying the page argument. If the specific page does not exist, a dialog box will appear to alert the user.

**Allowable Values:**

The name of any valid page that contains a process analyst

**Default Value:**

ProcessAnalyst

(This page may need to be created in your project.)

**See Also**

[Page Parameters](#)

**[Page]ProcessAnalystPopupPage**

Sets the page to pop up when the user calls the Cicode function ProcessAnalystWin() or ProcessAnalystPopup() without specifying the page argument. If the specific page does not exist, a dialog box will appear to alert the user.

**Allowable Values:**

The name of any valid page that contains a process analyst

**Default Value:**

!ProcessAnalystPopup

(This page may need to be created in your project)

**See Also**

[Page Parameters](#)

### [Page]Prompt

The AN where prompt and error messages are displayed. Set to zero (0) to disable.

See the Prompt() and DspError() Cicode functions for information about displaying prompts and errors.

### Allowable Values:

0 to 2000

### Default Value:

2

### See Also

[Page Parameters](#)

### [Page]RangeCheck

The [Page]RangeCheck parameter determines if range errors are displayed. If this flag is set, the range errors are displayed as per the [\[Page\]IgnoreValueQuality](#), [\[Page\]ShowErrorText](#), [\[Page\]ShowBadText](#), and [\[Page\]ShowUncertainText](#) parameters.

#RANGE will be visible when IgnoreValueQuality=0, ShowErrorText=1, ShowBadText=0, ShowUncertainText=0

### Allowable Values:

- 0 - (Ignore range errors)
- 1 - (Display range errors)

### Default Value:

0

---

**Note:** If this parameter is disabled, the value of the data displayed may be inaccurate.

---

### See Also

[Page Parameters](#)

### [Page]RelocateNonResizedOnCreation

Determines whether windows created with the Cicode function WinNewAt are automatically maximized. By default this is set to 0 to stop windows positioned completely or partially off screen from automatically maximizing and displaying at the top left corner of the screen.

**Allowable Values:**

- 0 (New windows are not automatically maximized)
- 1 (New windows are automatically maximized)

**Default Value:**

0

**See Also**[Page Parameters](#)**[Page]ScaleTextToMax**

If [Page]ScaleTextToMax is set to 0, pages by default will have their text scaled to the minimum of the x and y page scales. If 8192 is added to the mode argument of the WinNewAt Cicode call that created the page, the page will have its text scaled to the maximum of the x and y page scales.

If [Page]ScaleTextToMax is set to 1, pages will have their text scaled to the maximum of the x and y page scales, regardless of the mode used to create the page.

**Allowable Values:**

0 or 1

**Default Value:**

0

**See Also**[Page Parameters](#)**[Page]ScanTime**

---

**Note:** This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor. Any value you enter into your project parameters will be overridden by that specified in the Wizard.

---

The Page ScanTime determines how often (in milliseconds) the Animator refreshes a graphics page at runtime. A value of 250 indicates that the Animator will refresh the page 4 times per second.

Under some conditions, you might want to slow the update of your pages to reduce the CPU load on the Client computer. For example, you might want faster response on your main operator computers, while slowing the response time on manager computers.

**Allowable Values:**

1 to 60000 (milliseconds)

**Default Value:**

250

Note the following:

- You can dynamically change this value (for this computer) by calling PageSetInt(-2, <scantime>). You can also find out what the current scan time is, by calling PageGetInt(-2).
- You can set this value (for this computer) using the Computer Setup Wizard.
- You can set the Page ScanTime for individual graphics pages, using the Page Properties (from the File menu in the Graphics Builder).

**See Also**

[Page Parameters](#)

**[Page]ShowBadText**

Text objects can be displayed as #BAD text, or as text overlaid with a dithered pattern if the "display value" expression has "bad" quality.

This setting allows switching between these two modes. This type of indication is active only when [\[Page\]IgnoreValueQuality = 0](#).

**Allowable Values:**

- 0 – Display the result of the "display value" expression overlaid with a dithered pattern if the expression is not in an error condition or the [\[Page\]ShowErrorText](#) parameter is set to 0. If the expression is in error condition, it shows text or a value based on the [\[Page\]ShowErrorText](#) parameter.
- 1 – Display "#" followed by text defined by [\[Page\]BadText](#) (the default value is "BAD").

**Default Value:**

1

**See Also****[Page]ShowErrorText**

Text Objects can be displayed as #COM type errors, or as the text overlaid with a dithered pattern if the "display value" expression has "uncertain" quality.

This setting allows switching between these two modes. This type of indication is active only when

[Page]IgnoreValueQuality = 2.

### Allowable Values:

- 0 – Show the result of the "display value" expression overlaid with a dithered pattern.
- 1 – Show error status as #COM type errors.

### Default Value:

1

### See Also

[\[Page\]IgnoreValueQuality](#)

[Page Parameters](#)

### [Page]ShowUncertainText

Text Objects can be displayed as #UNC text, or as the text overlaid with a dithered pattern if the "display value" expression has "uncertain" quality.

This setting allows switching between these two modes. This type of indication is active only when [Page]IgnoreValueQuality = 2.

### Allowable Values:

- 0 – Show the result of the "display value" expression overlaid with a dithered pattern.
- 1 – Show "#" followed by text defined by [Page] UncertainText (default value: "UNC")

### Default Value:

1

### See Also

[\[Page\]IgnoreValueQuality](#)

[Page Parameters](#)

### [Page]SOEPage

The name of the graphics page to display when you call up an alarm sequence of events (SOE) page via the Cicode function PageSOE().

The SOE page is used to display events that have occurred in the system with entries repeated as required if they occurred multiple times.

The SOE page is commonly built on the default SOE alarm page template in the style that matches your project

but may be any valid page in the project. If the specific page does not exist, a dialog box will be displayed to alert the user.

**Allowable Values:**

Valid page name

**Default Value:**

SOE

**See Also**

[Page Parameters](#)

**[Page]Splash**

Specify the name of the page to use for Splash Screen.

This parameter is not supported in multi monitor mode.

---

**Note:** Adding a splash screen will cause your start window to have a window number of 1 instead of 0. Please consider using the windows name functionality to identify the start window number.

---

**Allowable Values:**

Any valid page name

**Default Value:**

None

**See Also**

[Page Parameters](#)

**[Page]SplashTimeout**

Milliseconds for the Splash Screen to display.

**Allowable Values:**

- 0 - Disable
- 1- 32767 (Milliseconds)

**Default Value:**

5000

**Note:** If [Page]SplashTimeout = 0 the window needs to be closed manually using WinFree(). The following cicode can be used to close the splash window from cicode that runs outside the context of the splash page:  
`WinGoto(WinNumber("SplashWin")); // "SplashWin" needs to match the name defined in  
[Page]SplashWinName WinFree();`

**See Also**

[Page Parameters](#)

**[Page]SplashWinName**

Specify the name of the Splash Window for use with the Cicode function WinNumber().

**Allowable Values:**

Any valid name

**Default Value:**

SplashWin

**See Also**

[Page Parameters](#)

**[Page]Startup**

The Page Name of the graphics page to display when Plant SCADA starts up.

**Note:** This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

**Allowable Values:**

Any valid Page Name or Page Number.

**Default Value:**

Startup

**See Also**

[Page Parameters](#)

**[Page]StartUpCancel**

This setting determines whether the Cancel button displays on the Startup message Box. The cancel button allows an operator to cancel the startup of Plant SCADA. (With the cancel button disabled, an operator is unable to cancel Plant SCADA as it starts up.)

When set to 0 this setting will also disable the ability for Plant SCADA Runtime Manager to cancel startup for each instance or the whole system.

If this parameter is modified while the Plant SCADA Runtime Manager is running, the changes will not take effect until all instances of Plant SCADA are restarted.

---

**Note:** This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

---

**Allowable Values:**

- 0 - (Disable display)
- 1 - (Enable display)

**Default Value:**

1

**See Also**

[Page Parameters](#)

**[Page]StartupDelay**

Milliseconds delay before the Startup page is shown. This is to be used in conjunction with the [Page]SplashTimeout parameter to determine the timing of the launching of the splash page and the startup page.

If StartupDelay is negative then the startup page can be called manually by the user using the Cicode function MultiMonitorStart().

If no splash screen is specified then this parameter is ignored.

**Allowable Values:**

-1 - 32767

**Default Value:**

3000

---

**Note:** Setting [Page]StartupDelay to be less than [Page]SplashTimout will cause both the splash page and the startup page to be displayed in two separated windows at the same time momentarily. This will make the window number of the Startup page be 1 instead of 0. If your code depends on the window number of the startup page being 0, you can set StartupDelay to be greater than SplashTimeout so that the splash window and

---

---

the startup window is displayed one after another, and the startup window will end up being allocated window number 0. Alternatively, you can set a window name to your startup page via parameter [\[Page\]StartupWinName](#) (default is StartWin) and retrieve the window number of your startup page via Cicode function WinNumber("<StartupWinName>"). This allows you to find the window number of your startup page regardless of the timing of the startup display sequence.

---

## See Also

[Page Parameters](#)

### [Page]StartupHeight

Height of Startup Page on main display monitor.

#### Allowable Values:

1 - 32767

#### Default Value:

-1

**Note:** If the value is set to be less than 1, for [\[Page\]StartupWidth](#) and [\[Page\]StartupHeight](#) both parameters will be ignored. The window will be displayed in its native dimension or will be reduced in size to fit the dimension of the monitor.

---

## See Also

[Page Parameters](#)

### [Page]StartupMode

Specify the window mode of Startup page on main display monitor. The mode specified in this parameter works in similar fashion as the mode specified in Cicode function WinNewAt. However, some modes used by WinNewAt are not applicable to this parameter.

#### Allowable Values:

Combination of valid window modes. The mode of the window:

- 0 - Normal page
- 4 - No re-size. The window is displayed with thin borders and no maximize/minimize icons. The window cannot be re-sized.
- 8 - No icons. The window is displayed with thin borders and no maximize/minimize or system menu icons. The window cannot be re-sized.
- 16 - No caption. The window is displayed with thin borders, no caption, and no maximize/minimize or system

menu icons. The window cannot be re-sized.

- 64 - Always on top.
- 256 - Display the entire window. This mode commands that no parts of the window will appear off the screen.
- 1024 - Disables dynamic resizing of the new window, overriding the setting of the [Page]DynamicSizing parameter.
- 4096 - Allows the window to be resized without maintaining the current aspect ratio. The aspect ratio defines the relationship between the width and the height of the window, which means this setting allows you to stretch or compress the window to any proportions. This option overrides the setting of the [Page]MaintainAspectRatio parameter.
- 8192 - Text on a page will be resized in proportion with the maximum scale change for a resized window. For example, consider a page that is resized to three times the original width, and half the original height. If this mode is set, the font size of the text on the page will be tripled (in proportion with the maximum scale). This option overrides the setting of the [Page] ScaleTextToMax parameter.
- 16384 - Hide the horizontal scroll bar.
- 32768 - Hide the vertical scroll bar.
- 65536 - Disable horizontal scrolling.
- 131072 - Disable vertical scrolling.

You can select multiple modes by adding modes together (for example, set Mode to 72 to open a window without maximize, minimize, or system menu icons and be always on top).

### Default Value:

0

### See Also

[Page Parameters](#)

### [Page]StartupWidth

Width of Startup Page on main display monitor.

### Allowable Values:

1 - 32767

### Default Value:

-1

**Note:** If the value is set to be less than 1, for [Page]StartupWidth and [Page]StartupHeight both parameters will be ignored. The window will be displayed in its native dimension or will be reduced in size to fit the dimension of the monitor.

## See Also

[Page Parameters](#)

### [Page]StartupWinName

Specify the name of the Startup Window for use with the Cicode function WinNumber().

#### Allowable Values:

Any valid name

#### Default Value:

StartupWin

## See Also

[Page Parameters](#)

### [Page]StartupX

X coordinate of Startup Page on main display monitor.

#### Allowable Values:

-32768 to 32767

#### Default Value:

0

## See Also

[Page Parameters](#)

### [Page]StartupY

Y coordinate of Startup Page on main display monitor.

#### Allowable Values:

-32768 to 32767

**Default Value:**

0

**See Also**

[Page Parameters](#)

**[Page]SummaryPage**

The name of the graphics page to display when you call up an alarm summary page via the Cicode function PageSummary().

The summary alarm page is used to display alarms that have occurred in the system that have yet to be logged with alarms repeated as required if they have occurred multiple times.

The summary alarm page is commonly built on the default summary alarm page template in the style that matches your project but may be any valid page in the project. If the specific page does not exist, a dialog box will appear to alert the user.

**Allowable Values:**

Valid page name

**Default Value:**

Summary

**See Also**

[Page Parameters](#)

**[Page]Time**

---

**Note:** This parameter is not used for this version of Plant SCADA. It is included for backward compatibility only.

The AN where the system time is displayed on each graphics page.

**Allowable Values:**

0 (Do not display) or 1 to 8192

**Default Value:**

9

## See Also

[Page Parameters](#)

### [Page]Tip

**Note:** This parameter is not used for this version of Plant SCADA. It is included for backward compatibility only.

The AN where tool tips are displayed on each graphics page. To control the display of tool tips (on the page), set the [\[Page\]TipHelp](#) parameter.

#### Allowable Values:

0 (Do not display) or 1 to 8192

#### Default Value:

14

## See Also

[Page Parameters](#)

### [Page]TipHelp

Determines whether tool tips are displayed. To control the pause delay of the pop-up windows for tool tips, set the [\[Page\]TipTimeOut](#) parameter.

#### Allowable Values:

- 0 - (Disable)
- 1 - (Enable)

#### Default Value:

1

## See Also

[Page Parameters](#)

### [Page]TipTimeOut

The time delay between when the mouse stops within an AN area and a tool tip displays (if the object at the AN has text configured for a tool tip). If the mouse pointer is within an AN area (over the object) for the Time Out

period, without an event occurring, the tool tip pop-up displays. The operator can then move the mouse from AN to AN, and tool tips display for each AN. If an event other than a mouse movement occurs, or if the mouse is outside an AN area for a period of time, the tool tips boxes do not display.

**Allowable Values:**

200 to 20000 (milliseconds)

**Default Value:**

1000

**See Also**

[Page Parameters](#)

**[Page]Title**

---

**Note:** This parameter is not used for this version of Plant SCADA. It is included for backward compatibility only.

The AN where the Page Title of the graphics page is displayed.

**Allowable Values:**

0 (Do not display) or 1 to 8192

**Default Value:**

12

**See Also**

[Page Parameters](#)

**[Page]UncertainDitheringColor**

Sets the dithering color for graphics elements which are dithered if the value quality is "uncertain" if the display expression contains an "unqualified tagname".

Refer to [Tag Extensions](#) for information regarding unqualified tags.

**Allowable Values:**

-1 – indication off

0x000000 to 0xFFFFFFFF (The RGB color value in hexadecimal)

**Default Value:**

0x000000 (Black)

**See Also**

[Page Parameters](#)

[\[Page\]ValueAbnormalPattern](#)

**[Page]UncertainDitheringDensity**

Sets the dithering density for graphics elements which are dithered if the value quality is "uncertain".

**Allowable Values:**

- 1 - (Highest Density)
- 2
- 3
- 4 - (Lowest Density)

**Default Value:**

2

**See Also**

[Page Parameters](#)

[\[Page\]ValueAbnormalPattern](#)

**[Page]UncertainText**

Defines the text which appears after "#" character when the 'display value' expression has "uncertain" quality.

This type of indication is active only when [\[Page\]ShowUncertainText](#) = 1 and [\[Page\]IgnoreValueQuality](#) = 0.

**Allowable Values:**

Any string.

**Default Value:**

UNC

## See Also

[Page Parameters](#)

### [Page]UncertainTextBackgroundColor

Sets the background color for numeric or text graphics objects to indicate "uncertain" quality if the display expression contains an "unqualified tagname".

#### Allowable Values:

-1 – indication off

0x000000 to 0xFFFF (The RGB color value in hexadecimal)

#### Default Value:

0x800080 - (Magenta)

---

**Note:** The parameter [\[Page\]IgnoreValueQuality](#) needs to be set to 2 for this parameter to be observable.

---

## See Also

[Page Parameters](#)

### [Page]ValueAbnormalPattern

Displays a diagonal line across a graphics object when the corresponding device goes offline, that is, it loses communication. When set to 1, the line appears on the device (for example, equipment) when the communication is lost. The parameter applies to the following bad tag quality conditions as well:

1. Tag Quality in Override mode
2. Tag quality in Override BAD Mode
3. Tag quality in Override Uncertain Mode
4. Tag quality in ControlInhibit Mode

---

**Note:** This parameter can be used in conjunction with existing Page parameters for dithering colors and dithering density.

---

#### Allowable Values:

- 0 - This shows a dithered pattern across the graphic object.
- 1 - Displays a diagonal line across the graphic object.

**Default Value:**

0

**See Also**[Page Parameters](#)[\[Page\]UncertainDitheringDensity](#)[\[Page\]UncertainDitheringColor](#)**[Page]WaitForValidData**

In previous versions of Plant SCADA, the animation system waited for valid data from all subscriptions before drawing the page. If the data took longer than a defined time to arrive, then the page was drawn with "#COM" in place of the missing data items. Plant SCADA now allows you to animate pages immediately, using any available cached values/qualities, and using 0 values (with "bad" quality status and "waiting for initial data" sub-status) for any data items that are not immediately available. As subscription notifications are received, the updated values are used.

**Allowable Values:**

- 0 - The Control Client will animate pages immediately, using the available values.
- 1 - The Control Client will attempt to wait for valid data before animating pages (previous functionality).

**Default Value:**

1

**See Also**[Page Parameters](#)**[Page]Windows**

The maximum number of windows that can be open simultaneously. This parameter sets the maximum number of windows supported by Plant SCADA. You might not be able to open this number of windows - Windows could run out of system resources before you reach this limit.

**Allowable Values:**

1 to 200

**Default Value:**

50

## See Also

[Page Parameters](#)

### [Page]WinTitle

The format of the window title banner. If you use the WinTitle() function, set this parameter to \* (asterisk).

#### Allowable Values:

The following field names: {Name} {Title} {Time} {Date} {Window} or the asterisk ( \* ) character. if you are using the WinTitle() function.

#### Default Value:

{Title,32}

## See Also

[Page Parameters](#)

### Path Parameters

The Citect.ini file contains the following path parameter:

- [\[Path\]<PathName>](#) - Defines a data path.

## See Also

[Parameter Categories](#)

### [Path]<PathName>

Defines a substitution data path in the project's system parameters. You can specify logical data paths (for data storage and retrieval) on your system. You can then use a data path shortcut instead of entering the full path to the data.

For example:

`[path]trenddata=c:\data\mytrends`

In this case, trenddata is defined as the shortcut to the c:\data\mytrends path. This shortcut can be used in Plant SCADA forms. For example, a trend tag's "File Name" field may contain the string:

`"[TRENDATA]:\trend_fast_00001".`

Plant SCADA then expects the trenddata path to be defined in the system parameters or in the INI file. The reserved shortcut names which the system uses can be found in topic Using Path Substitutions.

If one of these reserved path names are used, a "Path not suitable" error is raised on system startup.

---

**Note:** You will need to manually configure the access control list for a folder specified in a path substitution so that it matches the permissions that were applied to the default location during installation. See [Configure Directory Security for Modified Folder Locations](#).

---

### To define (or change a path substitution):

1. Choose **System | Parameters**.
2. In the **Section Name** box, enter **PATH**.
3. In the **Name** property, enter the path substitution string.
4. In the **Value** property, enter the full data path.

## See Also

[Path Parameters](#)

## Privilege Parameters

The Citect.ini file contains the following privilege parameters:

- [\[Privilege\]AckAlarms](#) - Determines the privilege level a user requires to acknowledge alarms.
- [\[Privilege\]DisableAlarms](#) - Determines the privilege level a user requires to disable alarms.
- [\[Privilege>EditUser\]](#) - Determines the privilege level a user requires to edit users using the drop-down menu next to the login user icons on the Tab\_Style template.
- [\[Privilege\]Exclusive](#) - Disables/enables the use of hierarchical privileges.
- [\[Privilege\]Shutdown](#) - Determines the privilege level a user requires to shutdown a project.
- [\[Privilege\]SilenceAlarms](#) - Determines a privilege level a user requires to silence alarms.

## See Also

[Parameter Categories](#)

### [Privilege]AckAlarms

Determines the privilege level a user requires to acknowledge alarms.

---

**Note:** This parameter is available only to projects based on StruxureWare and Tab\_Style templates.

---

### Allowable Values:

0 - 8

### Default Value:

1

## See Also

[Privilege Parameters](#)

### [Privilege]DisableAlarms

Determines the privilege level a user requires to disable alarms.

**Note:** This parameter is available only to projects based on Tab\_Style templates.

---

#### Allowable Values:

0 - 8

#### Default Value:

8

## See Also

[Privilege Parameters](#)

### [Privilege>EditUser

Determines the privilege level a user requires to edit users using the drop-down menu next to the login user icons on the Tab\_Style template.

**Note:** This parameter is available only to projects based on Tab\_Style templates.

---

#### Allowable Values:

0 - 8

#### Default Value:

8

## See Also

[Privilege Parameters](#)

### [Privilege]Exclusive

Disables/enables the use of hierarchical privileges.

If you set this parameter to 1, privileges are exclusive. For example, Privilege 3 means that the operator only has access to Privilege 3 commands.

If you set this parameter to 0, privileges are hierarchical. For example, Privilege 3 means that the operator has access to Privilege 3, 2 and 1 commands.

**Allowable Values:**

- 0 - (Hierarchical)
- 1 - (Not hierarchical)

**Default Value:**

1

**See Also**[Privilege Parameters](#)**[Privilege]Shutdown**

Determines the privilege level a user requires to shutdown a project using the Close window button on the title bar.

**Note:** This parameter is available only to projects based on Tab\_Style templates. It does not stop the user from shutting down Plant SCADA using Cicode functions or using the function OnEvent() to install their own function(s) for handling shutdown confirmation events.

**Allowable Values:**

0 - 8

**Default Value:**

0

**See Also**[Privilege Parameters](#)**[Privilege]SilenceAlarms**

Specifies the privilege level a user is required to have in order to silence the sound of an alarm.

Refer to 'Using the Tab\_Style Page Templates > Creating a New Project > Implementing Audible Alarms' for more information regarding Alarm sounds.

**Note:** This parameter is available only to projects based on Tab\_Style templates.

**Allowable Values:**

0 - 8

**Default Value:**

0

**See Also**[Privilege Parameters](#)**ProcessAnalyst Parameters**

Plant SCADA uses the Cicode functions [Login](#), [UserLogin](#) and [LoginForm](#) to set the preferred language at runtime. To allow the Process Analyst to determine the language it should display, you must map your Plant SCADA language databases to the Process Analyst resource files.

To do this, add a new .ini section called [ProcessAnalyst] to the Citect.ini file on all your Plant SCADA clients and servers, and create a mapping for each language. (Note that this section might already exist in your Citect.ini file.)

The mapping must use this format:

```
[ProcessAnalyst]
```

```
LanguagePath.<dbf>=<ProcessAnalystLanguage>
```

Where <dbf> is the name of a specific Plant SCADA language translation database, and <ProcessAnalystLanguage> is the language code of the resources.dll file containing the equivalent translations.

For example:

```
[ProcessAnalyst]
```

```
LanguagePath.French=fr
```

```
LanguagePath.Chinese=zh-CN
```

```
LanguagePath.German=de
```

Ensure that each of your machines contains the necessary language fonts. Windows provides facilities to add the necessary languages to your machine via the Regional and Language Options dialog box, accessible from the Control Panel. This step is essential if you want to use Asian languages on an English operating system.

**See Also**[Parameter Categories](#)**Protocol Parameters**

The Citect.ini file contains the following protocol parameters:

- [<ProtocolName>]Block - The blocking constant is a trade-off between the time taken to make multiple data requests and the time taken to read more data in a single request.

- [<ProtocolName>]Delay - The period (in milliseconds) to wait between receiving a response and sending the next command.
- [<ProtocolName>]MaxPending - The maximum number of pending commands that the driver holds ready for immediate execution.
- [<ProtocolName>]PollTime - The interrupt or polling service time (in milliseconds). Setting the polling time to 0 puts the driver in interrupt mode.
- [<ProtocolName>]Retry - The number of times to retry a command after a timeout.
- [<ProtocolName>]Timeout - Specifies how many milliseconds to wait for a response before displaying an error message.
- [<ProtocolName>]WatchTime - The frequency (in seconds) that the driver uses to check the communications link to the I/O device.

---

**Note:** The default and allowable values of these parameters are specific to the protocol. To access information about the parameters for a particular driver, see the [The Driver Reference Help](#).

---

## See Also

[Parameter Categories](#)

## Proxi Parameters

- [Proxi]<I/O Server name> - Obsolete in version 7.0.

## See Also

[Parameter Categories](#)

## PubSub Parameters

The Citect.ini file contains the following publish-subscribe parameters:

- [\[PubSub\]CicodeDeviceDelaySwitchInterval](#) - Delays switching to each primary device after an I/O server startup.
- [\[PubSub\]CicodeDeviceDelaySwitchTime](#) - Delays the client from switching tag subscriptions while the primary server is starting.
- [\[PubSub\]LogLevel](#) - Sets the level of logging between the I/O subscription system and the Plant SCADA drivers.
- [\[PubSub\]LogDevice](#) - Selects the device(s) for logging.

## See Also

[Parameter Categories](#)

## [PubSub]CicodeDeviceDelaySwitchInterval

The amount of time (in seconds) to delay the switching for each primary device.

This parameter is set on the target I/O server machine. The I/O server will set its Cicode devices to an 'online' state one at a time using the specified time interval. This will occur after the time specified in [PubSub]CicodeDeviceDelaySwitchTime has expired.

Bringing the devices online one at a time reduces the peak load and allows for performance tuning.

For more information, see Use a Client Tag Subscription Delay for Calculated Variables.

### Allowable values:

0 – 32767

### Default value:

0

### See Also

[PubSub Parameters](#)

## [PubSub]CicodeDeviceDelaySwitchTime

The amount of time (in seconds) to delay a client from switching tag subscriptions while the primary server is starting.

This parameter is set on the target I/O server machine. The I/O server will wait for this delay time before starting and setting its Cicode devices to the 'online' state.

For more information, see Use a Client Tag Subscription Delay for Calculated Variables.

### Allowable values:

0 – 32767

### Default value:

0

When set to the default value (0), the delay is disabled. To enable the delay, set a value of 1 or higher.

### See Also

[\[PubSub\]CicodeDeviceDelaySwitchInterval](#)

[PubSub Parameters](#)

## [PubSub]LogDevice

Determines which devices are logged between the I/O subscription system and the associated Plant SCADA drivers. Messages are logged to the Kernel and syslog.dat.

**Note:** This only occurs on an I/O Server.

### Allowable Values:

Use an asterix "\*" to specify all devices, or use a device name to specify a particular device.(e.g. "PLC\_1234").

### Default Value:

No value, which specifies NO devices.

The parameter [\[PubSub\]LogLevel](#) determines the level of logging used for the specified device(s).

### Example

```
PubSub:(Device IODevDisk ) UNIT ACTIVE change from <none> () Priority -1 to IODevDisk ()  
Priority 1  
PubSub:(Device IODevDisk ) UNIT IODevDisk STATUS change => Offline to Online  
PubSub:(Device IODevDisk ) Update Tag (Force=1): Tag=DiskTag1 RawValue=1  
Timestamp=2010-06-03_16:00.18.225 RawQual=0x00c0 DSError=0 (0x0) Has changed  
PubSub:(Device IODevDisk ) PollRead request : <Block>, Force=0, POINT=A0x00000001  
T0x00000001 RawType 1 Count 5 (DCB=0x07f99a34)
```

**Note:** You can update this parameter within the Citect.ini file while online, as the runtime system will read its value periodically or on demand. For more information, see the topic [Adjust Logging During Runtime](#).

### See Also

[PubSub Parameters](#)

## [PubSub]LogLevel

Sets the degree to which logging is used between the I/O subscription system and the actual Plant SCADA drivers, with respect to I/O requests. Messages are logged to syslog.dat.

Refer to the topic [Log Files](#) in the main help for more information regarding the syslog.dat file.

### Allowable Values:

- 0 - Disable
- 1 - Enable normal logging
- 2 - Enable verbose logging

**Default Value:**

0

This setting impacts the devices specified by the parameter [\[PubSub\]LogDevice](#).

**Note:** You can update this parameter within the Citect.ini file while online, as the runtime system will read its value periodically or on demand. For more information, see the topic [Adjust Logging During Runtime](#).

**See Also**

[PubSub Parameters](#)

**RemoteDB Parameters****[RemoteDB]DefaultComment**

Specifies the default string to be used for the **Comment** field of linked or imported variable tags. If you do not specify a value here, the field will be left blank.

**Allowable Values:**

Any string or blank.

**Default Value:**

No default (the field is left blank).

**See Also**

[RemoteDB Parameters](#)

**[RemoteDB]DefaultEngFull**

Specifies the default value to be used for the Eng Full Scale field of linked or imported variable tags. If you do not specify a default here, the field will be left blank.

**Allowable Values:**

0 to 99999999 (including reals) or blank.

**Default Value:**

No default (the field is left blank).

## See Also

[RemoteDB Parameters](#)

### [RemoteDB]DefaultEngZero

Specifies the default value to be used for the **Eng Zero Scale** field of linked or imported variable tags. If you do not specify a value here, the field will be left blank.

#### Allowable Values:

0 to 99999999 (including reals) or blank.

#### Default Value:

No default (the field is left blank).

## See Also

[RemoteDB Parameters](#)

### [RemoteDB]DefaultEU

Specifies the default value to be used for the **Engineering Units** field of linked or imported variable tags. If you do not specify a value here, the field will be left blank.

#### Allowable Values:

Any string or blank

#### Default Value:

No default (the field is left blank).

## See Also

[RemoteDB Parameters](#)

### [RemoteDB]DefaultRawFull

Specifies the default value to be used for the **Raw Full Scale** field of linked or imported variable tags. If you do not specify a value here, the field will be left blank.

**Allowable Values:**

0 to 9999999999 (including reals) or blank

**Default Value:**

No default (the field is left blank).

**See Also**

[RemoteDB Parameters](#)

**[RemoteDB]DefaultRawZero**

Specifies the default value to be used for the **Raw Zero Scale** field of linked or imported variable tags. If you do not specify a value here, the field will be left blank.

**Allowable Values:**

0 to 9999999999 (including reals) or blank

**Default Value:**

No default (the field is left blank).

**See Also**

[RemoteDB Parameters](#)

**Report Parameters**

The Citect.ini file contains the following report parameters:

- [\[Report\]ComBreak](#) - Determines whether reports that are triggered from the I/O devices are run if a communication break exists.
- [\[Report\]Debug](#) - Enables trace logging messages.
- [\[Report\]Disable](#) - Enables/disables the Reports Server.
- [\[Report\]HeartBeat](#) - Determines whether the Primary and Redundant Reports Servers send out heartbeat signals to each other.
- [\[Report\]HeartBeatTime](#) - Determines the period at which heartbeat signals are sent out by the Primary Reports Server. Also determines the period that the Redundant Reports Server will wait for a response (before assuming control of reports) after sending a heartbeat signal to the Primary Reports Server.
- [\[Report\]InhibitEvent](#) - Inhibits event-type reports on startup.
- [\[Report\]Primary](#) - Obsolete in version 7.0.

- [Report]Process - Obsolete in version 7.0.
- [Report]RtfRollOver - Determines whether an RTF report will save to a new file, or append to an existing file.
- [Report]RunStandby - Enables or disables tandem processing of reports.
- [Report]Server - Obsolete in version 7.0.
- [Report]Startup - The report to run when Plant SCADA starts up.
- [Report]TxtRollOver - Determines whether a text report will save to a new file, or append to an existing file.
- [Report]WatchTime - The period at which Plant SCADA checks for reports to run.

## See Also

[Parameter Categories](#)

[Report Process Parameters](#)

### [Report]ComBreak

Determines whether reports that are triggered from the I/O devices are run if a communication error occurs.  
(This parameter only applies to the trigger of the report - not to the data in the report.)

To control the "Com Break" in the actual report you should use the ERR\_FORMAT\_ON and ERR\_FORMAT\_OFF commands.

This parameter can be defined at the finest level of hierarchical granularity  
([Report.Clustername.ServerName]ComBreak). See [Hierarchical Parameters](#) for more information.

## Allowable Values:

- 0 - (Run reports even if there is a communication error associated with this report trigger)
- 1 - (Do not run reports if there is a communication error associated with this report trigger)

## Default Value:

1

## See Also

[Report Parameters](#)

### [Report]Debug

Enables trace logging messages.

When set to "1", entries appear in the appropriate SYSLOG.DAT for the report server process. Report server entries are prefixed by the string "ReportServer:".

**Allowable Values:**

- 0 = Logging is disabled.
- 1 = Logging is enabled.

**Default Value:**

0

**See Also**

[Report Parameters](#)

**[Report]Disable**

Enables/disables the Reports Server.

**Allowable Values:**

- 0 - (Enable)
- 1 - (Disable)

**Default Value:**

0

**See Also**

[Report Parameters](#)

**[Report]HeartBeat**

Determines whether the Reports Servers send out heartbeats to each other.

When enabled on the Primary Reports Server, heartbeat signals are sent to the Redundant Reports Server, informing it that the Primary Reports Server is controlling reports. If the Redundant Reports Server has been controlling reports, it will cease doing so upon receiving this message. The heartbeat signal is sent out every [Report]HeartBeatTime.

When enabled on the Redundant Reports Server, heartbeat signals are sent to the Primary Reports Server to establish its presence (i.e. a response is required). If the Redundant Reports Server does not receive a response from the Primary Reports Server during the [Report]HeartBeatTime, it will assume control of reports.

This parameter can be defined at the finest level of hierarchical granularity (*[Report.Clustername.ServerName]HeartBeat*). See [Hierarchical Parameters](#) for more information.

**Allowable Values:**

- 0 - (Disable)
- 1 - (Enable)

**Default Value:**

1

**See Also**[Report Parameters](#)**[Report]HeartBeatTime**

**On the Primary Reports Server** - The Primary Reports Server will send out a heartbeat signal once each HeartBeatTime.

**On the Redundant Reports Server** - After sending out a heartbeat message to the Primary Reports Server, the Redundant Reports Server will wait this long for a response. If a response is not received during this time, it will assume the Primary Reports Server is down, and will take control of reports. (If a response is received, control of reports will remain with the Primary Reports Server.)

This parameter can be defined at the finest level of hierarchical granularity (*[/Report.Clustername.ServerName]HeartBeatTime*). See [Hierarchical Parameters](#) for more information.

**Allowable Values:**

500 to 60000 (milliseconds)

**Default Value:**

30000

**See Also**[Report Parameters](#)**[Report]InhibitEvent**

Inhibits event-type reports on startup. For example, you might have a report that is triggered off the rising edge of a bit on startup. The Reports Server notices the bit come on, and runs the report. If this parameter is enabled, the Reports Server does not run this report until it has read the I/O devices a second time.

**Note:** This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

**Allowable Values:**

- 0 - (Do not inhibit)
- 1 - (Inhibit)

**Default Value:**

1

This parameter can be defined at the finest level of hierarchical granularity ([Report.Clustername.ServerName]InhibitEvent). See Hierarchical Parameters for more information.

**See Also**[Report Parameters](#)**[Report]RtfRollOver**

Determines whether an RTF report run using the Report function will save to a new report output file or append to the existing file. If it is saved to a new file, the existing report file is renamed.

**Allowable Values:**

- 0 - (Report information is appended to the report output file)
- 1 - (Report information is saved to a new report output file)

**Default Value:**

0

**See Also**[Report Parameters](#)**[Report]RunStandby**

Enables or disables tandem processing of reports. If this Server is the Standby Reports Server, it can process all reports in tandem with the Primary Server, or it can remain idle until called.

---

**Note:** This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

**Allowable Values:**

- 0 - (Remain idle until called)
- 1 - (Process in tandem with the Primary Server)

**Default Value:**

1

This parameter can be defined at the finest level of hierarchical granularity (*[Report.Clustername.ServerName]RunStandby*). See [Hierarchical Parameters](#) for more information.

**See Also**

[Report Parameters](#)

**[Report]Startup**

The report to run when Plant SCADA starts up. This parameter is only applicable if this Plant SCADA computer is a Reports Server.

**Note:** This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

---

**Allowable Values:**

Any configured report.

**Default Value:**

"Startup".

**See Also**

[Report Parameters](#)

**[Report]TxtRollOver**

Determines whether a text report run using the Report function will save to a new report output file or append to the existing file. If it is saved to a new file, the existing report file is renamed.

**Allowable Values:**

- 0 - (Report information is appended to the report output file)
- 1 - (Report information is saved to a new report output file)

**Default Value:**

1

## See Also

[Report Parameters](#)

### [Report]WatchTime

The period at which Plant SCADA checks for reports to run. The shorter the period, the faster the Reports Server checks if a report is due to run. The faster the Reports Server must check for reports to run, the more I/O requests are made (to the I/O Server) for reports that are triggered from the I/O devices, and an increased CPU loading on the Reports Server results. You can slow the period (this parameter) to reduce CPU and I/O Server loading.

Note that, if a report is triggered off a bit, the bit must remain high for at least twice this period so that the Reports Server notices it. For example, if this parameter is set to 1000, a bit should remain on for 2000 msec so that the Reports Server notices that it goes on and off. The exact period depends on how fast the data can be read from the I/O devices. For example, if it takes 2000 msec to read all the data for the report triggers (this is slow), the reports are checked every  $(1000 + 2000) = 3$  seconds.

This parameter can be defined at the finest level of hierarchical granularity (`[Report.Clustername.ServerName]WatchTime`). See [Hierarchical Parameters](#) for more information.

## Allowable Values:

100 to 60000 (milliseconds)

## Default Value:

1000

## See Also

[Report Parameters](#)

### Report Process Parameters

Plant SCADA allows you to specify some parameters for a report server process (see [Hierarchical Parameters](#) in the Plant SCADA documentation).

The following parameters need to be configured for a specific process.

**Note:** The <server name> is the name of the report server as specified in the **Topology** activity.

- `[Report.<ClusterName>.<ServerName>]Clusters` - Sets the clusters this Reports Server process will connect to at startup.
- `[Report.<ClusterName>.<ServerName>]CPU` - Sets the CPU that this Report Server process is assigned to.
- `[Report.<ClusterName>.<ServerName>]Events` - The list of events that this Plant SCADA Report Server process enables.
- `[Report.<ClusterName>.<ServerName>]ShutdownCode` - Determines the Cicode function to run when this Plant SCADA Reports Server process shuts down.

- **[Report.<ClusterName>.<ServerName>]StartupCode** - Determines the Cicode function to run when this Plant SCADA Reports Server process starts up.

The Citect.ini file on any PC that will be connecting to this <cluster name><server name> report server can contain the following parameters:

- **[Report.<ClusterName>.<ServerName>]DisableConnection** - This setting is used to specify if a client should not connect to a server.
- **[Report.<ClusterName>.<ServerName>]Priority** - This setting is used to specify the client priority for the server connection.

## See Also

[Parameter Categories](#)

[Report Parameters](#)

### **[Report.<ClusterName>.<ServerName>]Clusters**

Sets which clusters this Reports Server will connect to at startup.

#### **Allowable Values:**

A comma separated list of configured cluster names.

#### **Default Value:**

If this parameter is blank, the Reports Server will connect to every cluster defined for the project.

This parameter needs to be defined at the finest level of hierarchical granularity (Report.<ClusterName>.<ServerName>Clusters). See [Hierarchical Parameters](#) for more information.

## See Also

[Report Process Parameters](#)

### **[Report.<ClusterName>.<ServerName>]CPU**

This parameter is by Plant SCADA Runtime Manager to determine which CPU(s) the report server process is assigned to. The CPU parameter is configurable for every Plant SCADA process via the **CPU Setup** page in the Computer Setup Wizard (see [CPU Configuration](#)).

This parameter needs to be defined at the finest level of hierarchical granularity (Report.<ClusterName>.<ServerName>CPU). See [Hierarchical Parameters](#) for more information.

CPU numbers are specified in a comma separated list. CPU numbers need to be valid and a subset of the available CPUs on a computer. For example, if there are eight CPUs on a computer, a valid CPU range is 0 to 7.

**Note:** If this parameter is modified while the Plant SCADA Runtime Manager is running, the changes will not take effect until every instance of Plant SCADA is restarted.

## Allowable Values:

A comma separated list in the range 0 to 31 (inclusive) where 0 is the first CPU (for example, "0, 1, 2" or "3, 4, 5").

If no value is entered, the process will run on all available CPUs.

---

**Note:** When a process is set to run on all available CPUs, be aware that performance issues may still occur when your CPU usage is less than 100%. You can detect these issues when your CPU usage for a process is fixed at 100 divided by the number of processors.

---

## Default Value:

No value (the process will run on all CPUs).

---

**Note:** You cannot manually specify CPU assignments on a computer with more than 32 CPUs. Under these circumstances, the Setup Wizard will automatically set this parameter to no value (all CPUs). Any changes you make to this parameter will be overwritten.

---

## See Also

[Report Process Parameters](#)

[\[Client\]CPU](#)

[\[Alarm.<ClusterName>.<ServerName>\]CPU](#)

[\[IOServer.<ClusterName>.<ServerName>\]CPU](#)

[\[Trend.<ClusterName>.<ServerName>\]CPU](#)

## [\[Report.<ClusterName>.<ServerName>\]DisableConnection](#)

This setting is used to specify whether or not the client sub system should connect to the given server. If the Citect.ini on a particular PC has [Type.ClusterName.ServerName]DisableConnection = 1, specified for a particular server, then the client sub system will establish a non-redundant connection to the other server in the redundant pair. If that connection becomes inoperative, the client sub system will not be redirected and will have no connection until the server is restored.

To define connection priority of a particular server use [\[Report.<ClusterName>.<ServerName>\]Priority](#).

## Allowable Values:

- 0 - Allow the client to connect to the server
- 1 - Stop the client connecting to the server

## Default Value:

0

## See Also

[Report Process Parameters](#)

### [Report.<ClusterName>.<ServerName>]Events

The list of events that this Plant SCADA Report Server will enable.

This parameter needs to be defined at the finest level of hierarchical granularity (Report.<ClusterName>.<ServerName>Events). See [Hierarchical Parameters](#) for more information.

---

**Note:** This parameter is modified by the Computer Setup Wizard.

---

## Allowable Values:

List of events (comma separated).

## Default Value:

NONE

---

**Note:** When Plant SCADA is running in single process mode only the events assigned to [Client]Events will enabled. Events specified for particular Plant SCADA components are ignored.

---

**Note:** To set this parameter, we recommend you to enable the required events in the "Events Setup" page of the Computer Setup Wizard (instead of entering the events in the Setup Editor).

---

## See Also

[Report Process Parameters](#)

### [Report.<ClusterName>.<ServerName>]Priority

This setting is used to specify the client sub system's priority for the connection to this specific server. The client sub system will connect to, or switch over to, any available server that has the highest priority (lowest number). If the priorities for multiple servers are set to the same value, the client sub system will only switch on a loss of the currently active connection. This is the default operation to provide the same behavior as in versions of the product prior to v7.20.

To disable (not allow) the client to connect to the server use  
[\[Report.<ClusterName>.<ServerName>\]DisableConnection](#)

Allowable Values:

1 to 32767

## Default Value:

1

## See Also

[Report Process Parameters](#)

### [Report.<ClusterName>.<ServerName>]ShutdownCode

Can be used to specify a Cicode function that runs at shutdown of the Plant SCADA Report Server component. Plant SCADA suspends the shutdown process until the function has finished running. The function needs to complete execution within the time specified by the [\[Code\]ShutdownTime](#) parameter, otherwise Plant SCADA will terminate it.

This parameter needs to be defined at the finest level of hierarchical granularity (`Report .<ClusterName>.<ServerName>ShutdownCode`). See [Hierarchical Parameters](#) for more information.

#### Allowable Values:

Any built-in or user-written Cicode function, 1-31 alpha-numeric characters.

#### Default Value:

NONE

---

**Note:** When Plant SCADA is running in single process mode only the Cicode function assigned to [\[Client\]ShutdownCode](#) will be executed at shutdown. Cicode functions specified for particular Plant SCADA components are ignored.

---

## See Also

[Report Process Parameters](#)

### [Report.<ClusterName>.<ServerName>]StartupCode

Determines the Cicode function to run when the Plant SCADA Reports Server component starts up.

You can only pass constant data to the startup function call. You cannot pass variables or other functions. For example, `MyFunc(1, "str")` is valid, but `MyFunc(PLCTAG, "str")` or `MyFunc(YourFunc(), "str")` is not supported.

This parameter needs to be defined at the finest level of hierarchical granularity (`Report .<ClusterName>.<ServerName>StartupCode`). See [Hierarchical Parameters](#) for more information.

---

**Note:** This parameter is modified by the Computer Setup Wizard.

---

#### Allowable Values:

Any built-in or user-written Cicode function, 1-31 alpha-numeric characters.

#### Default Value:

NONE

---

**Note:** When Plant SCADA is running in single process mode only the Cicode function assigned to [Client]StartupCode will be executed at startup. Cicode functions specified for particular Plant SCADA components are ignored.

---

## See Also

[Report Process Parameters](#)

### Runtime Manager Parameters

The citect.ini file contains the following Runtime Manager parameters:

- [\[RuntimeManager\]AllowDumpKernel](#) - Enables or disables the Dump Kernel context menu in Runtime Manager.
- [\[RuntimeManager\]AllowReload](#) - Enables or disables the reload option in the Runtime Manager menu.
- [\[RuntimeManager\]AutoRestartTrigger](#) - Specifies how long an instance needs to run before that instance will be automatically restarted if it exits abnormally.
- [\[RuntimeManager\]ExitAfterShutdown](#) - Sets whether the Plant SCADA Runtime Manager exits after all runtime processes are shut down.
- [\[RuntimeManager\]NotRespondingRetries](#) - Specifies the number of retries that the CrashMonitor should wait to detect if an instance of Plant SCADA is not responding.
- [\[RuntimeManager\]Profile](#) - Sets the name of the INI profile to be used. If name does not exist the default citect.ini file will be used.

## See Also

[Parameter Categories](#)

### [RuntimeManager]AllowDumpKernel

Enables or disables the Dump Kernel context menu in Runtime Manager.

#### Allowable Values:

- 0 - (Disables the menu option)
- 1 - (Enables the menu option)

#### Default Value:

1

## See Also

[Runtime Manager Parameters](#)

**[RuntimeManager]AllowReload**

Enables or disables the reload option on the Runtime Manager menu.

**Allowable Values:**

- 0 - (Disables the menu option)
- 1 - (Enables the menu option)

**Default Value:**

1

**See Also**

[Runtime Manager Parameters](#)

**[RuntimeManager]AutoRestartTrigger**

An automatic restart occurs if an instance abnormally exits after the specified trigger time. For example, if set to 3600 seconds (1 hour), and the system exits abnormally after running for 43 minutes, then the automatic restart will not occur. If the system exits abnormally after running for 65 minutes, then the automatic restart will occur.

**Allowable Values:**

- n - the amount of time (in seconds) before an automatic restart occurs following an abnormal exit
- 0 - the instance is restarted automatically following an abnormal exit
- -1 - the instance is not restarted automatically following an abnormal exit

**Default Value:**

3600 seconds (1 hour)

**Note:** If [Runtime]AutoRestartTrigger is less than zero then the minimum value is used, and no auto restart will occur. If  $\geq 0$  then this value is used in seconds. This is the time (from the original Plant SCADA start) before the process will be restarted. If this value is 3600, there is no more than one restart an hour from when the process exited abnormally.

**See Also**

[Runtime Manager Parameters](#)

**[RuntimeManager]ExitAfterShutdown**

Sets whether the Plant SCADA Runtime Manager exits after all runtime processes are shut down.

**Allowable Values:**

- 0 - (Plant SCADA Runtime Manager does not exit)
- 1 - (Plant SCADA Runtime Manager exits)

**Default Value:**

1

**See Also**

[Runtime Manager Parameters](#)

**[RuntimeManager]NotRespondingRetries**

Specifies the number of retries that the CrashMonitor should wait to detect if an instance of Plant SCADA is not responding. One retry is sent by the instance every 30 seconds.

**Allowable Values:**

Minimum: 2

**Default Value:**

3

**See Also**

[Runtime Manager Parameters](#)

**[RuntimeManager]Profile**

Sets the name of the Profile INI file to be used at runtime. If the profile is not found or is blank, the default citect.ini file will be used.

**Allowable Values:**

Name of profile or blank

**Default Value:**

citect.ini file

## See Also

[Runtime Manager Parameters](#)

## Scheduling Parameters

The Citect.ini file contains the following Scheduling parameters:

- [\[Scheduling\]AlwaysExecuteEntryAction](#) - Determines how entry actions are executed by Scheduler.
- [\[Scheduling\]BACnetSchedulePollPeriod](#) - Sets the subscription poll period (rate) for BACnet tags.
- [\[Scheduling\]PersistPath](#) - Sets where the runtime schedule data for the Scheduler is stored.
- [\[Scheduling\]StartDelay](#) - Sets the delay between initialization of the Scheduler's server side components and the start of active schedule processing.

## See Also

[Parameter Categories](#)

### [Scheduling]AlwaysExecuteEntryAction

This parameter determines how entry actions are executed by Scheduler.

When set to true (1), the Scheduler will execute the entry action of all equipment resource states regardless of whether or not their internal states are the same. This ensures backwards compatibility when a piece of equipment remains in the same state but needs to repeatedly perform an action at times configured in the Scheduler.

By default, this parameter is set to false (0), which means only changes that have been made to the selected equipment will take effect. The rest of the equipment states will not be refreshed.

#### Allowable Values:

- 0 – false
- 1 – true.

#### Default Value:

0

## See Also

[Scheduling Parameters](#)

### [Scheduling]BACnetSchedulePollPeriod

Sets the subscription poll period (rate) for BACnet tags. This parameter needs to be set on the report server

machine.

**Allowable Values:**

1 to 3600000 (milliseconds).

**Default Value:**

1000

**See Also**

[Scheduling Parameters](#)

**[Scheduling]PersistPath**

Directs where the configuration data for the Scheduler is stored. Confirm that Windows folder access is set so that path is accessible by the Report Server process.

**Allowable Values:**

- The value is treated as a file path string.

**Default Value:**

The Plant SCADA Data directory

**See Also**

[Scheduling Parameters](#)

**[Scheduling]StartDelay**

This parameter sets the delay from when the Scheduler's server side components are initialized to the point when Scheduler begins processing active schedule entries. This setting is used to allow time to establish the connections to other server processes before the scheduler server executes state actions.

**Note:** A value of [Scheduling]StartDelay=0 indicates no delay resulting in active schedule entries being processed immediately upon Report Server start up.

**Allowable Values:**

- The value is treated as a long number.
- The value units are seconds.

**Default Value:**

10 seconds

**Example**

If the [Scheduling] StartDelay ini parameter is not present in the Plant SCADA ini file, no schedule entries will be processed until 10 seconds after report server start up. If the ini file contains [Scheduling] StartDelay = 5, no schedule entries will be actioned until 5 seconds after the report server start up.

**See Also**

[Scheduling Parameters](#)

**Security Parameters**

The Citect.ini file contains the following Security parameters:

- [\[Security\]AuthenticationLogging](#) -Use this parameter to configure Authentication Event Logging.
- [\[Security\]BlockExec](#) - Can prevent users and groups from running the Exec Cicode function.
- [\[Security>CreateUserByRole](#) - Controls the behavior of the Cicode functions UserCreateForm and UserCreate.
- [\[Security\]DisableDEP](#) - When set will turn off DEP protection for the Plant SCADAruntime.
- [\[Security>DeleteWebContentProfile](#) - Use this parameter to automatically delete the application data associated with users of a Web Content control.

**See Also**

[Parameter Categories](#)

**[Security]AuthenticationLogging**

Use this parameter to configure Authentication Event Logging.

**Allowable Values:**

- 0 - No logging.
- 1 - Log only unsuccessful login attempts.
- 2 - Log every login and logout event.

**Default Value:**

1

**See Also**

[Security Parameters](#)

**[Security]BlockExec**

Can be used to prevent users and groups from running the Exec Cicode function. If the parameter is left blank, it defaults to 1. This parameter is set to 0 for the Example project. This is used in conjunction with the Allow Exec user role permission. A user can run the Exec Cicode function only if this parameter is set to 0 **and** the Allow Exec user role is set to TRUE. For more information about the role, refer to the Plant SCADA online help.

**Allowable Values:**

- 0 - Running Exec is allowed for users or groups with the Allow Exec permission set to TRUE.
- 1 - Running Exec is not allowed.

**Default Value:**

1

**See Also**[Security Parameters](#)**[Security]CreateUserByRole**

This parameter controls the behavior of the Cicode function [UserCreate](#). It determines if a created Plant SCADA user is of a specified Type or assigned to a specified Role.

---

**Note:** This parameter was designed to support the function UserCreate, which is now deprecated. If you use the function [UserCreateByRole](#) instead, you may not need to use this parameter.

---

**Allowable Values:**

- 0 - UserCreate will create a user of a specified type.
- 1 - UserCreate will create a user assigned to a specified role.

**Default Value:**

0

**See Also**[Security Parameters](#)**[Security]DisableDEP**

When set will turn off DEP protection for the Plant SCADA runtime.

**Allowable Values:**

- 0 -DEP is enabled.
- 1 - DEP is disabled.

**Default Value:**

0

**See Also**[Security Parameters](#)**[Security]DeleteWebContentProfile**

Use this parameter to automatically delete the application data associated with users of a Web Content control.

When a Web Content control is displayed, any user-specific data (such as cookies, storage and browsing history) is stored in the profile folder for the current Windows user (for example, C:\Users\<Windows User Name>). The following sub-directory is used:

\AppData\Local\AVEVA Plant SCADA\WebContent\Profiles\<User>\

Where <User> is the name of the user that is currently logged in to Plant SCADA runtime.

If this parameter is set to 1, the <User> folder will be deleted when:

- The user logs out.
- Switching to a different user account occurs.
- If the runtime application is closed.

**Allowable Values:**

- 0 -User's data is not deleted.
- 1 - User's data is deleted.

**Default Value:**

0

**See Also**[Security Parameters](#)

## Server Parameters

- [Server]Name - Obsolete in version 7.0.
- [\[Server\]AutoLoginMode](#) - Determines the auto login method the server will use when establishing a connection to other servers.

## See Also

[Parameter Categories](#)

### [Server]AutoLoginMode

Determines which auto login method the server will use when establishing a connection to other servers.

#### Allowable Values:

- 0 – Auto login is disabled
- 1 –Default server user (full areas and privileges)
- 2 - Specific User (from Computer Setup Wizard)

#### Default Value:

1

## See Also

[Server Parameters](#)

## Shutdown Parameters

The Citect.ini file contains the following shutdown parameters:

- [\[Shutdown\]NetworkIgnore](#) - Determines whether this computer responds to any network shutdown calls.
- [\[Shutdown\]NetworkStart](#) - Enables network shutdown and restart from this computer.
- [\[Shutdown\]Phase](#) - The shutdown phase for this computer.

## See Also

[Parameter Categories](#)

### [Shutdown]NetworkIgnore

Determines whether this computer responds to any network shutdown calls.

This parameter applies to shutdown calls from a remote client and needs to be configured on the server

machine. If using this parameter you need to enable the parameter [Shutdown]NetworkRestart on the remote client.

**Note:** Set this parameter to 0 for all the computers that will send or accept a shutdown command.

---

### Allowable Values:

- 0 - (Allow network shutdown).
- 1 - (Ignore network shutdown).

### Default Value:

1

### See Also

[Shutdown Parameters](#)

### [Shutdown]NetworkStart

Enables network shutdown and restart from this computer.

### Allowable Values:

- 0 - (Do not allow network shutdown and restart)
- 1 - (Allow network shutdown and restart)

### Default Value:

0

### See Also

[Shutdown Parameters](#)

### [Shutdown]Phase

The shutdown phase for this computer. This parameter only comes into effect when [Shutdown]NetworkIgnore is set to 0 and a client receives a shutdown request message from a server. Phase 2 clients receive a shutdown request when the first phase 1 client has reconnected to the server.

### Allowable Values:

- 1 - (Shutdown in first phase)
- 2 - (Shutdown in second phase)

## Default Value:

1

## See Also

[Shutdown Parameters](#)

## SPC Parameters

The Citect.ini file contains the following parameters:

- [\[SPC\]AlarmBufferSize](#) - Specifies the number of subgroups used in SPC Alarm calculations.
- [\[SPC\]XFreak](#) - The number of data points required to set the SPC XFreak type alarm
- [\[SPC\]PartialSubgroup](#) - Enables masking of subgroups from being included in calculations.
- [\[SPC\]RAboveUCL](#) - Sets the number of consecutive data points required to set the RAboveUCL SPC alarm.
- [\[SPC\]RBelowLCL](#) - Sets the number of consecutive data points required to set the RBelowLCL SPC alarm.
- [\[SPC\]ROutsideCL](#) - Sets the number of consecutive data points required to set the ROutsideCL SPC alarm.
- [\[SPC\]XAboveUCL](#) - Sets the number of consecutive data points required to set the XAboveUCL SPC alarm.
- [\[SPC\]XBelowLCL](#) - Sets the number of consecutive data points required to set the XBelowLCL SPC alarm.
- [\[SPC\]XDownTrend](#) - Sets the number of consecutive data points required to set the XDownTrend SPC alarm.
- [\[SPC\]XErratic](#) - Sets the number of consecutive data points required to set the XErratic SPC alarm.
- [\[SPC\]XGradualDown](#) - Sets the number of consecutive data points required to set the XGradualDown SPC alarm.
- [\[SPC\]XGradualUp](#) - Sets the number of consecutive data points required to set the XGradualUp SPC alarm.
- [\[SPC\]XMixture](#) - Sets the number of consecutive data points required to set the XMixture SPC alarm.
- [\[SPC\]XOutsideCL](#) - Sets the number of consecutive data points required to set the XOutsideCL SPC alarm.
- [\[SPC\]XOutsideWL](#) - Sets the number of consecutive data points required to set the XOutsideWL SPC alarm.
- [\[SPC\]XStratification](#) - Sets the number of consecutive data points required to set the XStratification SPC alarm.
- [\[SPC\]XUptrend](#) - Sets the number of consecutive data points required to set the XUptrend SPC alarm.

## See Also

[Parameter Categories](#)

### [SPC]AlarmBufferSize

Specifies the number of subgroups used in SPC Alarm calculations.

**Note:** You may find it useful to make the **AlarmBufferSize** the same as number of subgroups displayed on your SPC charts, so that any SPC alarms directly correlate with those charts.

The following shows the number of subgroups displayed on the standard SPC templates for the given

resolutions:

VGA - 39

SVGA - 52

XGA - 58

SXGA - 76

### Allowable Values:

1 to 200

### Default Value:

39 (standard VGA template)

### See Also

[SPC Parameters](#)

## [SPC]PartialSubgroup

### Allowable Values:

- 0 - (Incomplete subgroups will NOT be included in SPC calculations)
- 1 - (Incomplete subgroups will be included in SPC calculations)

### Default Value:

0

Enables masking of subgroups from being included in calculations if the subgroup data is incomplete. Subgroups that are not complete will have an effect on the accuracy of SPC calculations. The magnitude of this effect is relative to the number of subgroups on screen.

### See Also

[SPC Parameters](#)

## [SPC]RAboveUCL

Sets the number of consecutive data points required to set the RAboveUCL SPC alarm. The alarm is triggered when consecutive subgroup range values are above the upper control limit (UCL).

### Allowable Values:

1 to 20

**Default Value:**

3

**See Also**[SPC Parameters](#)**[SPC]RBelowLCL**

Sets the number of consecutive data points required to set the RBelowLCL SPC alarm. The alarm is triggered when consecutive subgroup range values are below the lower control limit (LCL).

**Allowable Values:**

1 to 20

**Default Value:**

3

**See Also**[SPC Parameters](#)**[SPC]ROutsideCL**

Sets the number of consecutive data points required to set the ROutsideCL SPC alarm. The alarm is triggered when consecutive subgroup range values are outside the control limits (UCL and LCL).

**Allowable Values:**

1 to 20

**Default Value:**

3

**See Also**[SPC Parameters](#)**[SPC]XAboveUCL**

Sets the number of consecutive data points required to set the XAboveUCL SPC alarm. The alarm is triggered

when consecutive subgroup mean values are above the upper control limit (UCL).

**Allowable Values:**

1 to 20

**Default Value:**

3

**See Also**

[SPC Parameters](#)

**[SPC]XBelowLCL**

Sets the number of consecutive data points required to set the XBelowLCL SPC alarm. The alarm is triggered when consecutive subgroup mean values are below the lower control limit (LCL).

**Allowable Values:**

1 to 20

**Default Value:**

3

**See Also**

[SPC Parameters](#)

**[SPC]XDownTrend**

Sets the number of consecutive data points required to set the XDownTrend SPC alarm. The alarm is triggered when consecutive subgroup mean values are gradually descending.

**Allowable Values:**

1 to 20

**Default Value:**

5

## See Also

[SPC Parameters](#)

### [SPC]XErratic

Sets the number of consecutive data points required to set the XErratic SPC alarm. The alarm is triggered when consecutive subgroup mean values are erratic in behaviour.

#### Allowable Values:

1 to 20

#### Default Value:

3

## See Also

[SPC Parameters](#)

### [SPC]XFreak

The number of consecutive data points required to set the SPC XFreak type alarm.

#### Allowable Values:

1 to 10

#### Default Value:

1

## See Also

[SPC Parameters](#)

### [SPC]XGradualDown

Sets the number of consecutive data points required to set the XGradualDown SPC alarm. The alarm is triggered when consecutive subgroup mean values are below the centre line (CL).

#### Allowable Values:

1 to 20

**Default Value:**

8

**See Also**

[SPC Parameters](#)

**[SPC]XGradualUp**

Sets the number of consecutive data points required to set the XGradualUp SPC alarm. The alarm is triggered when consecutive subgroup mean values are above the centre line (CL).

**Allowable Values:**

1 to 20

**Default Value:**

8

**See Also**

[SPC Parameters](#)

**[SPC]XMixture**

Sets the number of consecutive data points required to set the XMixture SPC alarm. The alarm is triggered when consecutive subgroup mean values are both above and below the centre line (CL).

**Allowable Values:**

1 to 20

**Default Value:**

8

**See Also**

[SPC Parameters](#)

**[SPC]XOutsideCL**

Sets the number of consecutive data points required to set the XOutsideCL SPC alarm. The alarm is triggered

when consecutive subgroup mean values are outside the control limits (UCL and LCL).

**Allowable Values:**

1 to 20

**Default Value:**

3

**See Also**

[SPC Parameters](#)

**[SPC]XOutsideWL**

Sets the number of consecutive data points required to set the XOutsideWL SPC alarm. The alarm is triggered when consecutive subgroup mean values are outside the alarm limits.

**Allowable Values:**

1 to 20

**Default Value:**

5

**See Also**

[SPC Parameters](#)

**[SPC]XStratification**

Sets the number of consecutive data points required to set the XStratification SPC alarm. The alarm is triggered when consecutive subgroup mean values are constant.

**Allowable Values:**

1 to 20

**Default Value:**

15

## See Also

[SPC Parameters](#)

### [SPC]XUptrend

Sets the number of consecutive data points required to set the XUptrend SPC alarm. The alarm is triggered when consecutive subgroup mean values are gradually ascending.

#### Allowable Values:

1 to 20

#### Default Value:

5

## See Also

[SPC Parameters](#)

### SQL Parameters

The Citect.ini file contains the following SQL parameters:

- [\[SQL\]DefaultTimestampFormat](#) - Sets the formatting pattern being used by SQLGetField function for converting database date/time values to strings
- [\[SQL\]MaxConnections](#) - Defines the maximum number of DB connection objects.
- [\[SQL\]MaxRecordsets](#) - Defines the maximum number of disconnected recordsets in the system.
- [\[SQL\]QueryTimeout](#) - Sets the timeout period for SQL queries globally.
- [\[SQL\]TextColWidth](#) - Sets the column width for text columns returned from SQL queries.

## See Also

[Parameter Categories](#)

### [SQL]DefaultTimestampFormat

Sets the formatting pattern being used by SQLGetField function for converting database date/time values to strings.

Please note that only main date/time data types are affected by this parameter. For SQL Server it works for DateTime and SmallDateTime, while for MS Access it works for Date/Time type. If it is necessary to use the parameter for converting other parameters, please use conversion macros to main types in your SQL queries. For example in T-SQL, the conversion may use CONVERT macro: SELECT CONVERT(DateTime, MyDateColumn) FROM MyTable.

**Allowable Values:**

Any valid "format" parameter value for the function TimestampToStr.  
0 - Short time format, hh:mm  
1 - Long time format, hh:mm:ss  
2 - Short date format, dd/MM/yyyy  
3 - Long date format, dddd, dd MMMM yyyy.  
4 - Short date & short time format, dd/MM/yyyy hh:mm  
5 - Short date & long time format, dd/MM/yyyy hh:mm:ss  
6 - Long date & short time format, dddd, dd MMMM yyyy hh:mm  
7 - Long date & long time format, dddd, dd MMMM yyyy hh:mm:ss  
8 - Month day format, dd MMMM  
9 - Year month format, MMMM yyyy  
10 - RFC1123 format, ddd, dd MMM yyyy hh:mm:ss GMT  
11 - Sortable date time format, yyyy-MM-ddThh:mm:ss  
12 - Short universal format, yyyy-MM-dd hh:mm:ss  
13 - Long universal format, dddd, dd MMMM yyyy hh:mm:ss

**Default Value:**

11

**See Also**[SQL Parameters](#)**[SQL]MaxConnections**

Defines the maximum number of DB connection objects. Note that the number includes connections which are not opened. If the current number of connections is equal or greater than this parameter then new DB connection objects cannot be created and respective functions return invalid handles. A hardware alarm is generated in this case and an appropriate log is written to TraceLog.

**Allowable Values:**

0 (Unlimited), 0 .. n

**Default Value:**

10

## See Also

[SQL Parameters](#)

### [SQL]MaxRecordsets

Defines the maximum number of disconnected recordsets in the system. If the current number of recordsets is equal or greater than this parameter then new recordsets cannot be created and respective functions return invalid handles. A SCADA hardware alarm is generated in such case and an appropriate log is written to TraceLog

#### Allowable Values:

0 (Unlimited), 0 .. n

#### Default Value:

30

## See Also

### [SQL]QueryTimeout

Sets the timeout period for SQL queries globally. This is the period of time that Plant SCADA will wait for an SQL query to be processed before terminating the query.

A query value of zero (0) seconds has special meaning: Plant SCADA will wait indefinitely for the query to be processed.

#### Allowable Values:

From 0 to 14400 (4 hours)

#### Default Value:

30

## See Also

[SQL Parameters](#)

### [SQL]TextColWidth

Sets the maximum number of text characters retrieved from a database record. This parameter only needs to be set if the SQL query returns text columns longer than 256 characters.

**Allowable Values:**

1 to 255 (characters)

**Default Value:**

255

**See Also****Time Parameters**

- [Time]Address - Obsolete in version 7.20.
- [Time]Deadband - Obsolete in version 7.20.
- [Time]Disable - Obsolete in version 7.20.
- [Time]Name - Obsolete in version 7.20.
- [Time]PollTime - Obsolete in version 7.20.
- [Time]Port - Obsolete in version 7.20.
- [Time]RTsync - Obsolete in version 7.20.
- [Time]Server - Obsolete in version 7.20.

**See Also**

[Parameter Categories](#)

**Trend Parameters**

The Citect.ini file contains the following trend parameters:

- [\[Trend\]AbortStartupOnError](#) - If set and the history file properties of any Trend tag have been modified, restart of the Trend Server will not occur because the history files can not be reloaded.
- [\[Trend\]AcquisitionTimeout](#) - stops a trend server from infinitely acquiring a valid data sample from a disconnected I/O device
- [\[Trend\]AllFiles](#) - Indicates whether trend history files should be created at startup or created when needed.
- [\[Trend\]BlockByIODevice](#) - Obsolete in version 7.0.
- [\[Trend\]BytesReadBeforeSleep](#) - Controls the amount of data read from the trend archive before the trend server has to sleep.
- [\[Trend\]BytesWrittenBeforeSleep](#) - Specifies how many bytes of trend data are written to file by the TrendWriteTask before it sleeps.
- [\[Trend\]CacheSize0](#) - Determines the number of samples that can be stored in the cache of a trend with sample period < 50ms.
- [\[Trend\]CacheSize1](#) - Determines the number of samples that can be stored in the cache of a trend with sample period >= 50ms and < 250ms.

- [\[Trend\]CacheSize2](#) - Determines the number of samples that can be stored in the cache of a trend with sample period >= 250ms and < 1 second.
- [\[Trend\]CacheSize3](#) - Determines the number of samples that can be stored in the cache of a trend with sample period >= 1 second and < 5 seconds.
- [\[Trend\]CacheSize4](#) - Determines the number of samples that can be stored in the cache of a trend with sample period >= 5 seconds and < 10 seconds.
- [\[Trend\]CacheSize5](#) - Determines the number of samples that can be stored in the cache of a trend with sample period >= 10 seconds and < 20 seconds.
- - Determines the number of samples that can be stored in the cache of a trend with sample period >= 20 seconds and < 50 seconds.
- [\[Trend\]CacheSize7](#) - Determines the number of samples that can be stored in the cache of a trend with sample period >= 50 seconds < 100 seconds.
- [\[Trend\]CacheSize8](#) - Determines the number of samples that can be stored in the cache of a trend with sample period >= 100 seconds.
- [\[Trend\]ClientRequestTime](#) - The delay (in milliseconds) after real-time trend data is requested from the Trends Server before the next data is requested.
- [\[Trend\]CopyFilesMessage](#) - Determines whether the redundant Trends Server displays an alert message if it cannot find any trend files.
- [\[Trend\]CPU](#) - Obsolete in version 7.0.
- [\[Trend\]CursorColor](#) - Allows the cursor color to be specified.
- [\[Trend\]CursorColour](#) - Obsolete in version 7.20 (replaced by [\[Trend\]CursorColor](#)).
- [\[Trend\]DeleteIfIncompatible](#) - Deletes incompatible history files automatically.
- [\[Trend\]Disable](#) - Enables/disables the Trends Server.
- [\[Trend\]EnableBackfill](#) - Allows trend backfill to be disabled.
- [\[Trend\]FileCreationWriteSize](#) - Controls the rate at which trend files are initialised (whether created at startup or runtime).
- [\[Trend\]FileNameType](#) - Specifies whether your trend file names will be long or short.
- [\[Trend\]FilterViewByPrivilege](#) - Determines if the results from a trend browse will be filtered according to privilege.
- [\[Trend\]GapFillMode](#) - Determines which method is used to fill gaps in a trend file and graph caused by missing trend samples.
- [\[Trend\]GapFillSamples](#) - Determines whether gaps in a trend graph caused by missing samples are filled (determined by the number of samples missed).
- [\[Trend\]GapFillTime](#) - Determines whether gaps in a trend graph caused by missing samples are filled (determined by the duration of the gap).
- [\[Trend\]InhibitEvent](#) - Inhibits event trends on startup.
- [\[Trend\]InstantTrendIdleTime](#) - Set this parameter to specify the maximum number of seconds that an unused instant trend will be retained.
- [\[Trend\]InstantTrendSamples](#) - Set this parameter to determine the maximum number of samples that will be retained for an individual instant trend.
- [\[Trend\]MaxBackfillsAtOnce](#) - Determines the number of trend to be requested for backfill at a time.
- [\[Trend\]MaxRdnSamples](#) - Determines the maximum samples to be requested for a trend at a time.

- [\[Trend\]MaxRequestLength](#) - Determines the maximum number of trend sample requests allowed per trend.
- [\[Trend\]MaxSetTableQue](#) - Determines the maximum number of SetTrendTable requests on the primary trend server to be transferred to the standby server.
- [\[Trend\]MaxSetTableStnbyPending](#) - Controls the maximum number of pending SetTrendTable requests to the standby trend server.
- [\[Trend\]MissedSamplesAlarmTime](#) - Controls the amount of trend data that can be missed before a hardware alarm is triggered.
- [\[Trend\]Process](#) - Obsolete in version 7.0.
- [\[Trend\]RangeCheck](#) - Checks for invalid trend sample values.
- [\[Trend\]ReadWatchTime](#) - Determines how long the trend server will wait (in milliseconds) between each file read of trend samples.
- [\[Trend\]Redundancy](#) - Obsolete in version 7.0.
- [\[Trend\]ReloadBackOffTime](#) - Back-off time configured to control the pace of the reload on an Trend server.
- [\[Trend\]Server](#) - Obsolete in version 7.0.
- [\[Trend>ShowCacheSizes](#) - Determines whether the currently set values and sample period ranges for parameters [Trend]CacheSize0 . . . [Trend]CacheSize8 are displayed in the Plant SCADA Kernel.
- [\[Trend\]StaggerRequestSubgroups](#) - Obsolete in version 7.0.
- [\[Trend\]TrendDebug](#) - Used to debug the trend system during commissioning.
- [\[Trend\]WatchTime](#) - The delay (in milliseconds) after data is returned from the I/O device before data is requested for the next trend acquisition cycle.
- [\[Trend\]WriteWatchTime](#) - The trend write task writes out trend data to disk, and then sleeps for the time specified in this parameter (before writing the next trend file).

## See Also

[Parameter Categories](#)

[Trend Process Parameters](#)

### [Trend]AbortStartupOnError

When set the trend server will abort the startup if it finds a history file that does not match the required format.

---

**Note:** The configuration can be returned to its previous state so that the history file will match the required format. Alternatively, the invalid files may be backed up and deleted so that new files can be created on the next restart.

---

### Allowable Values:

- 0 - Start the trend server but with trends that have miss-matched history file configuration disabled.
- 1 - Abort the trend server startup if any trends have miss-matched history file configuration.

### Default Value:

0

## See Also

[Trend Parameters](#)

### [Trend]AcquisitionTimeout

Sets a timeout to stop a trend server waiting indefinitely for a valid data sample from an I/O device while the connection to the I/O server is lost or the device is offline.

On each sample collection, the gap between current time and the last sample request time is checked. If this gap is greater or equal to the value set for this parameter, and the sample data is invalid, a gap fill action is requested (i.e. synchronization will be performed). This implementation is designed to avoid large gaps in trend history when the connection to an I/O device is lost.

#### Allowable Values:

0 to 65536 (seconds)

#### Default Value:

300 (seconds)

## See Also

[Trend Parameters](#)

### [Trend]AllFiles

Indicates whether all trend history files should be created at startup or created when needed. By default, Plant SCADA creates all trend history files at startup, so that the required disk space is allocated at the same time, without fragmentation. (Because files are created at startup, you know the total disk requirements.) Creating history files at startup happens only once - the first time Plant SCADA starts up.

On large systems with many trends, the first startup could take considerable time. If you change the default of this parameter, history files are created whenever needed, reducing the first time startup delay. However, if a disk space shortage occurs while Plant SCADA is running, data loss will occur because new history files cannot be created.

#### Allowable Values:

- 0 - (Create a file when needed)
- 1 - (Create all files at startup)

#### Default Value:

1

## See Also

[Trend Parameters](#)

### [Trend]BlockByIODevice

**Note:** This parameter is obsolete. From version 7.0, trend data from a device is no longer blocked by the Trend Server, but instead is blocked by the I/O device. Trend tags are automatically updated with trend data.

Setting this parameter to 1 commands that if I/O problems (e.g. with hardware, the I/O device, or the communications link) resulted in gaps for one trend tag, other trend tags were unaffected.

Leaving this parameter set to 0 meant that one trend tag with gaps could cause gaps for **all** trend tags.

## Allowable Values:

- 0 - Gaps caused for one trend tag will cause gaps for all trend tags
- 1 - Gaps caused for one trend tag will leave other trend tags unaffected

## Default Value:

1

## See Also

### [Trend]BytesReadBeforeSleep

Increasing the value of this parameter allows more data to be read from the trend archive before the trend server has to sleep. This may improve the read performance of trend requests on the trend server.

**Note:** Increasing this value may also increase the CPU usage on the trend server.

## Allowable Values:

1 to 2147483647

## Default Value:

65536

## See Also

[Trend Parameters](#)

### [Trend]BytesWrittenBeforeSleep

This parameter specifies how many bytes of trend data are written to file by the TrendWriteTask, before it sleeps

for the amount of time specified by the WriteWatchTime parameter.

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

Do not change the [Trend]BytesWrittenBeforeSleep parameter in the Citect.ini file, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

### **Allowable Values:**

1 to 1000000 (bytes)

### **Default Value:**

65536

### **See Also**

[Trend Parameters](#)

### **[Trend]CacheSize0**

Determines the number of samples that can be stored in the cache of trends with sample period < 50ms.

### **Allowable Values:**

Any integer between 1 and 65536 inclusive (trend samples)

### **Default Value:**

8188

If your sample period is  $\geq$  50ms, Plant SCADA uses a different parameter to set the number of samples the cache can store. See the following table:

[Trend]Parameter	Sample Period	Default Value (No. of samples)
[Trend]CacheSize0	< 50 ms	8188
[Trend]CacheSize1	$\geq$ 50 ms and < 250 ms	2044
[Trend]CacheSize2	$\geq$ 250 ms and < 1 second	1020
[Trend]CacheSize3	$\geq$ 1 second and < 5 seconds	764
[Trend]CacheSize4	$\geq$ 5 seconds and < 10 seconds	508

[Trend]Parameter	Sample Period	Default Value (No. of samples)
[Trend]CacheSize5	>= 10 seconds and < 20 seconds	380
[Trend]CacheSize6	>= 20 seconds and < 50 seconds	252
[Trend]CacheSize7	>= 50 seconds and < 100 seconds	192
[Trend]CacheSize8	>= 100 seconds	128

## See Also

[Trend Parameters](#)

### [Trend]CacheSize1

Determines the number of samples that can be stored in the cache of trends with sample period >= 50ms and < 250ms.

#### Allowable Values:

Any integer between 1 and 65536 inclusive (trend samples)

#### Default Value:

2044

If your sample period is < 50 ms or >= 250ms, Plant SCADA uses a different parameter to set the number of samples the cache can store. See the following table:

[Trend]Parameter	Sample Period	Default Value (No. of samples)
[Trend]CacheSize0	< 50 ms	8188
[Trend]CacheSize1	>= 50 ms and < 250 ms	2044
[Trend]CacheSize2	>= 250 ms and < 1 second	1020
[Trend]CacheSize3	>= 1 second and < 5 seconds	764
[Trend]CacheSize4	>= 5 seconds and < 10 seconds	508
[Trend]CacheSize5	>= 10 seconds and < 20 seconds	380
[Trend]CacheSize6	>= 20 seconds and < 50 seconds	252
[Trend]CacheSize7	>= 50 seconds and < 100 seconds	192
[Trend]CacheSize8	>= 100 seconds	128

## See Also

[Trend Parameters](#)

### [Trend]CacheSize2

Determines the number of samples that can be stored in the cache of trends with sample period  $\geq 250\text{ms}$  and  $< 1\text{ second}$ .

#### Allowable Values:

Any integer between 1 and 65536 inclusive (trend samples)

#### Default Value:

1020

If your sample period is  $<250\text{ ms}$  or  $\geq 1\text{ second}$ , Plant SCADA uses a different parameter to set the number of samples the cache can store. See the following table:

[Trend]Parameter	Sample Period	Default Value (No. of samples)
[Trend]CacheSize0	< 50 ms	8188
[Trend]CacheSize1	$\geq 50\text{ ms}$ and $< 250\text{ ms}$	2044
[Trend]CacheSize2	$\geq 250\text{ ms}$ and $< 1\text{ second}$	1020
[Trend]CacheSize3	$\geq 1\text{ second}$ and $< 5\text{ seconds}$	764
[Trend]CacheSize4	$\geq 5\text{ seconds}$ and $< 10\text{ seconds}$	508
[Trend]CacheSize5	$\geq 10\text{ seconds}$ and $< 20\text{ seconds}$	380
[Trend]CacheSize6	$\geq 20\text{ seconds}$ and $< 50\text{ seconds}$	252
[Trend]CacheSize7	$\geq 50\text{ seconds}$ and $< 100\text{ seconds}$	192
[Trend]CacheSize8	$\geq 100\text{ seconds}$	128

## See Also

[Trend Parameters](#)

### [Trend]CacheSize3

Determines the number of samples that can be stored in the cache of trends with sample period  $\geq 1\text{ second}$  and  $< 5\text{ seconds}$ .

**Allowable Values:**

Any integer between 1 and 65536 inclusive (trend samples)

**Default Value:**

764

If your sample period is <1 second or  $\geq$  5 seconds, Plant SCADA uses a different parameter to set the number of samples the cache can store. See the following table:

[Trend]Parameter	Sample Period	(No. of samples)
[Trend]CacheSize0	< 50 ms	8188
[Trend]CacheSize1	$\geq$ 50 ms and < 250 ms	2044
[Trend]CacheSize2	$\geq$ 250 ms and < 1 second	1020
[Trend]CacheSize3	$\geq$ 1 second and < 5 seconds	764
[Trend]CacheSize4	$\geq$ 5 seconds and < 10 seconds	508
[Trend]CacheSize5	$\geq$ 10 seconds and < 20 seconds	380
[Trend]CacheSize6	$\geq$ 20 seconds and < 50 seconds	252
[Trend]CacheSize7	$\geq$ 50 seconds and < 100 seconds	192
[Trend]CacheSize8	$\geq$ 100 seconds	128

**See Also**

[Trend Parameters](#)

**[Trend]CacheSize4**

Determines the number of samples that can be stored in the cache of trends with sample period  $\geq$  5 seconds and < 10 seconds.

**Allowable Values:**

Any integer between 1 and 65536 inclusive (trend samples)

**Default Value:**

508

If your sample period is <5 seconds or  $\geq$  10 seconds, Plant SCADA uses a different parameter to set the number of samples the cache can store. See the following table:

[Trend]Parameter	Sample Period	Default Value (No. of samples)
[Trend]CacheSize0	< 50 ms	8188
[Trend]CacheSize1	>= 50 ms and < 250 ms	2044
[Trend]CacheSize2	>= 250 ms and < 1 second	1020
[Trend]CacheSize3	>= 1 second and < 5 seconds	764
[Trend]CacheSize4	>= 5 seconds and < 10 seconds	508
[Trend]CacheSize5	>= 10 seconds and < 20 seconds	380
[Trend]CacheSize6	>= 20 seconds and < 50 seconds	252
[Trend]CacheSize7	>= 50 seconds and < 100 seconds	192
[Trend]CacheSize8	>= 100 seconds	128

## See Also

[Trend Parameters](#)

### [Trend]CacheSize5

Determines the number of samples that can be stored in the cache of trends with sample period >= 10 seconds and < 20 seconds.

#### Allowable Values:

Any integer between 1 and 65536 inclusive (trend samples)

#### Default Value:

380

If your sample period is < 10 seconds or >= 20 seconds, Plant SCADA uses a different parameter to set the number of samples the cache can store. See the following table:

[Trend]Parameter	Sample Period	Default Value (No. of samples)
[Trend]CacheSize0	< 50 ms	8188
[Trend]CacheSize1	>= 50 ms and < 250 ms	2044
[Trend]CacheSize2	>= 250 ms and < 1 second	1020
[Trend]CacheSize3	>= 1 second and < 5 seconds	764

[Trend]Parameter	Sample Period	Default Value (No. of samples)
[Trend]CacheSize4	>= 5 seconds and < 10 seconds	508
[Trend]CacheSize5	>= 10 seconds and < 20 seconds	380
[Trend]CacheSize6	>= 20 seconds and < 50 seconds	252
[Trend]CacheSize7	>= 50 seconds and < 100 seconds	192
[Trend]CacheSize8	>= 100 seconds	128

## See Also

[Trend Parameters](#)

### [Trend]CacheSize6

Determines the number of samples that can be stored in the cache of trends with sample period >= 20 seconds and < 50 seconds

#### Allowable Values:

Any integer between 1 and 65536 inclusive (trend samples)

#### Default Value:

252

If your sample period is < 20 seconds or >= 50 seconds, Plant SCADA uses a different parameter to set the number of samples the cache can store. See the following table:

[Trend]Parameter	Sample Period	Default Value (No. of samples)
[Trend]CacheSize0	< 50 ms	8188
[Trend]CacheSize1	>= 50 ms and < 250 ms	2044
[Trend]CacheSize2	>= 250 ms and < 1 second	1020
[Trend]CacheSize3	>= 1 second and < 5 seconds	764
[Trend]CacheSize4	>= 5 seconds and < 10 seconds	508
[Trend]CacheSize5	>= 10 seconds and < 20 seconds	380
[Trend]CacheSize6	>= 20 seconds and < 50 seconds	252
[Trend]CacheSize7	>= 50 seconds and < 100 seconds	192

[Trend]Parameter	Sample Period	Default Value (No. of samples)
[Trend]CacheSize8	>= 100 seconds	128

## See Also

[Trend Parameters](#)

### [Trend]CacheSize7

Determines the number of samples that can be stored in the cache of trends with sample period >= 50 seconds and < 100 seconds.

#### Allowable Values:

Any integer between 1 and 65536 inclusive (trend samples)

#### Default Value:

192

If your sample period is <50 seconds or >= 100 seconds, Plant SCADA uses a different parameter to set the number of samples the cache can store. See the following table:

[Trend]Parameter	Sample Period	(No. of samples)
[Trend]CacheSize0	< 50 ms	8188
[Trend]CacheSize1	>= 50 ms and < 250 ms	2044
[Trend]CacheSize2	>= 250 ms and < 1 second	1020
[Trend]CacheSize3	>= 1 second and < 5 seconds	764
[Trend]CacheSize4	>= 5 seconds and < 10 seconds	508
[Trend]CacheSize5	>= 10 seconds and < 20 seconds	380
[Trend]CacheSize6	>= 20 seconds and < 50 seconds	252
[Trend]CacheSize7	>= 50 seconds and < 100 seconds	192
[Trend]CacheSize8	>= 100 seconds	128

## See Also

[Trend Parameters](#)

**[Trend]CacheSize8**

Determines the number of samples that can be stored in the cache of trends with sample period >= 100 seconds.

**Allowable Values:**

Any integer between 1 and 65536 inclusive (trend samples)

**Default Value:**

128

If your sample period is < 100 seconds, Plant SCADA uses a different parameter to set the number of samples the cache can store. See the following table:

[Trend]Parameter	Sample Period	Default Value (No. of samples)
[Trend]CacheSize0	< 50 ms	8188
[Trend]CacheSize1	>= 50 ms and < 250 ms	2044
[Trend]CacheSize2	>= 250 ms and < 1 second	1020
[Trend]CacheSize3	>= 1 second and < 5 seconds	764
[Trend]CacheSize4	>= 5 seconds and < 10 seconds	508
[Trend]CacheSize5	>= 10 seconds and < 20 seconds	380
[Trend]CacheSize6	>= 20 seconds and < 50 seconds	252
[Trend]CacheSize7	>= 50 seconds and < 100 seconds	192
[Trend]CacheSize8	>= 100 seconds	128

**See Also**

[Trend Parameters](#)

**[Trend]ClientRequestTime**

The interval between client requests for trend data from the Trends Server. For instance, if you set this value to 500ms, any trends with a display period of 300ms will only be updated with new data every 500ms. You can set this parameter to any value, but the lower the value, the more frequent the requests to the Trends Server, and this will cause an increased CPU load.

**Allowable Values:**

1 to 60000 (milliseconds)

**Default Value:**

250

**See Also**[Trend Parameters](#)**[Trend]CopyFilesMessage**

Determines whether the redundant Trends Server displays an alert message if it cannot find any trend files. Such an error might be reported when the redundant Trends Server starts up for the first time, or if files have been deleted.

If the projects being run by both Trends Servers are not identical, the redundant copy may not succeed, displaying an alert message. For example, if trend tags are missing or in a different order.

**Allowable Values:**

- 0 - (Do not display alert message)
- 1 - (Display alert message)

**Default Value:**

1

**See Also****[Trend]CursorColor**

Specifies the trend cursor color.

**Allowable Values:**

0x000000 to 0xFFFFF

Color defined as an RGB value, for example:

- 0x000000 - (Black)
- 0x000080 - (Blue)
- 0x008000 - (Green)
- 0x008080 - (Cyan)
- 0x800000 - (Red)
- 0x800080 - (Magenta)
- 0x808000 - (Brown)

- 0xBFBFBF - (Grey)
- 0x7F7F7F - (Dark Grey)
- 0x0000FF - (Light blue)
- 0x00FF00 - (Light green)
- 0x00FFFF - (Light cyan)
- 0xFF0000 - (Light red)
- 0xFF00FF - (Light Magenta)
- 0xFFFF00 - (Yellow)
- 0xFFFFFFFF - (White)

**Default Value:**

0x000000 (Black)

**See Also**

[Trend Parameters](#)

**[Trend]DeleteIfIncompatible**

Causes automatic deletion of incompatible history files at start-up.

**Allowable Values:**

- 0 - (No file deletion. Plant SCADA shuts down.)
- 1 - (Incompatible history files are automatically deleted. Plant SCADA logs a message to the SYSLOG.DAT file and continues.)

**Default Value:**

0

**See Also**

[Trend Parameters](#)

**[Trend]Disable**

Enables/disables the Trends Server.

**Allowable Values:**

- 0 - (Enable)

- 1 - (Disable)

**Default Value:**

0

**See Also**

[Trend Parameters](#)

**[Trend]EnableBackfill**

Allows users to disable trend backfill at startup and server re-connection.

**Allowable Values:**

- 0 - (Disable)
- 1 - (Enable)

**Default Value:**

1

**See Also**

[Trend Parameters](#)

**[Trend]FileCreationWriteSize**

Controls the rate at which trend files are initialised (whether created at startup or runtime). Optimal performance is achieved with values of 327680 bytes or lower. File writes larger than this may be batched in chunks of 65536 bytes, reducing the performance benefit. Environmental factors may also alter this threshold.

**Allowable Values:**

1 to 2147483647

**Default Value:**

16384

**See Also**

[Trend Parameters](#)

## [Trend]FileNameType

Specifies whether your trend file names will be long or short. There are three options: Compatible file names, Long file names, and Short file names.

- Compatible File Names - All new trend files will have long file names if supported by the operating system. Existing short file names will be unaffected. The trend system will check for short file names, so Plant SCADA startup may take longer.
- Long File Names - All new and existing trend files will be assigned long file names. Existing short file names will not be deleted, but the trend system will be unable to use them.
- Short File Names - All new and existing trend files will be assigned short file names. Existing long file names will not be deleted, but the trend system will be unable to use them.

### Allowable Values:

- 0 - (Compatible file names)
- 1 - (Long file names)
- 2 - (Short file names)

### Default Value:

0

### See Also

[Trend Parameters](#)

## [Trend]FilterViewByPrivilege

The setting applied to this parameter effects any any trend browse or data request performed by the cicode functions TrnBrowseOpen, TrnGetTable, TrnEventGetTable or TrnEventGetTableMS, or through CTAPI using CtFindFirst for the tables TrnQuery and CTAPITrend.

When this parameter is set to 0 on the trend server, the set of records returned from the browse will not be filtered by privilege. When this parameter is set to 1, the set of records returned from the browse will be filtered by privilege. The privilege and area of the current logged in user who initiated the browse will be checked against that configured in the privilege and area fields of each trend in the trend forms in Plant SCADA Studio, and the trend will only be included if the user met that level of access.

### Allowable Values:

- 0 - browse sessions not filtered by privilege
- 1 - browse sessions filtered by privilege

**Default Value:**

0

**See Also**

[Trend Parameters](#)

**[Trend]GapFillMode**

Determines the method used to fill gaps in a trend file and graph caused by missing trend samples. You can choose either the Step or Ratio method.

If you choose Step, Plant SCADA will fill the gap with a sample of the same value as the last real sample.

If you choose Ratio, Plant SCADA will fill the gap with a sample which is calculated using the ratio of sample times and values immediately before and after the gap. The gap in the trend graph will then be filled with a straight line.

**Note:** For gaps to be filled, you must also set the parameter [Trend]GapFillTime or [Trend]GapFillSamples.

If you want to fill gaps left by missing trend samples on the trend graph only (and not in the trend's file), you must use TrnSetDisplayMode() instead of this parameter.

**Allowable Values:**

- 0 - Step
- 1 - Ratio

**Default Value:**

0

**See Also**

[Trend Parameters](#)

**[Trend]GapFillSamples**

Determines how Plant SCADA processes missing trend samples (i.e. gaps in the trend file and graph). If the number of missing samples is less than the value you enter here, the gap will be filled. Gaps larger than or equal to this value will not be filled.

If you enter 0, no gaps will be filled - the gaps will actually appear on your trend graphs.

**Note:** To use time (instead of samples) to determine which gaps will be filled, set this parameter to -1.

[Trend]GapFillTime will then be used. If you set it to any value greater than or equal to 0, [Trend]GapFillTime will be ignored.

The method used to fill the gap is specified by the [Trend]GapFillMode parameter.

To fill gaps left by missing trend samples on the trend graph only (and not in the trend's file), you must use

TrnSetDisplayMode() instead of this parameter.

### Allowable Values:

-1 to 1000000

### Default Value:

-1

### See Also

[Trend Parameters](#)

## [Trend]GapFillTime

Determines how Plant SCADA processes missing trend samples (i.e. gaps in the trend file and graph). Any gaps equal to or shorter than the time you enter here will be filled with values calculated by Plant SCADA. If you enter 0, no gaps will be filled - the gaps will actually appear on your trend graphs.

---

**Note:** To use samples (instead of time) to determine which gaps will be filled, set [Trend]GapFillSamples to a value greater than or equal to 0, [Trend]GapFillTime will then be ignored.

The method used to fill the gap is specified by the [Trend]GapFillMode parameter.

To fill gaps left by missing trend samples on the trend graph only (and not in the trend's file), use TrnSetDisplayMode() instead of this parameter.

### Allowable Values:

0 to 60000 (milliseconds)

### Default Value:

10000

### See Also

[Trend Parameters](#)

## [Trend]InhibitEvent

Inhibits event trends on startup. For example, you might have a trend that is triggered off the rising edge of a bit on startup. The Trends Server notices the bit come on, and displays the trend. If this parameter is enabled, the Trends Server does not display this trend until it has read the I/O devices a second time.

---

**Note:** This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

**Allowable Values:**

- 0 - (Do not disable)
- 1 - (Disable)

**Default Value:**

1

**See Also**[Trend Parameters](#)**[Trend]InstantTrendIdleTime**

Set this parameter to specify the number of seconds that data will be held in the cache for an instant trend without client display requests. If no requests for the data occur for this period of time, the cached data will be cleared and the instant trend will need to restart the cache for the next client display request.

---

**Note:** Increasing this value may increase the memory usage of the display client, which may impact performance.

---

**Allowable Values:**

0 to 1000000 (seconds).

**Default Value:**

600

**See Also**[Trend Parameters](#)**[Trend]InstantTrendSamples**

Set this parameter to determine the maximum number of samples that will be retained for an individual instant trend. When the limit is reached, the oldest samples are discarded.

---

**Note:** Increasing this value may increase the memory usage of the display client, which may impact performance.

---

**Allowable Values:**

0 to 1000000.

**Default Value:**

6000

**See Also**

[Trend Parameters](#)

**[Trend]MaxBackfillsAtOnce**

Limits the number of trends that can be requested for backfill at a time. Useful for restricting the CPU and network traffic for trend backfilling upon startup or network recapture.

**UNINTENDED EQUIPMENT OPERATION**

Do not change the [Trend]MaxBackfillsAtOnce parameter in the Citect.ini file, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Allowable Values:**

1 to 32000

**Default Value:**

5

**See Also**

[Trend Parameters](#)

**[Trend]MaxRdnSamples**

Determines the maximum number of samples in a single redundancy backfilling request for a particular trend. Increasing the value of this parameter will cause trend redundancy requests to be larger which may speed up trend redundancy backfilling, but at the potential cost of making the trend server less responsive to display client requests during redundancy synchronisation. Decreasing the value of this parameter will cause trend redundancy requests to be smaller, potentially allowing display client trend requests to be responded to faster during redundancy synchronisation, but this may slow down trend redundancy synchronisation.

**Allowable Values:**

1 to 32767

**Default Value:**

500

**See Also**

[Trend Parameters](#)

**[Trend]MaxRequestLength**

The maximum size of a trend request measured in the number of samples that a trend server will accept from clients. This parameter protects the trend server from being overloaded by requests that are too large.

Decreasing this parameter will cause smaller trend requests to be rejected, increasing its value will let the trend server process large requests which could reduce the responsiveness of the trend server to smaller requests.

**Note:** Trend redundancy backfilling requests from the peer trend server are never rejected, regardless of what setting is applied to this parameter. In order to control the size of trend redundancy requests, use the parameter [\[Trend\]MaxRdnSamples](#).

**Allowable Values:**

1 to 100000000

**Default Value:**

4000

**See Also**

[Trend Parameters](#)

**[Trend]MaxSetTableQue**

The maximum number of SetTrendTable requests on the primary trend server allowed when these have to be transferred to the standby server.

**Note:** When [Trend]TrendDebug is on you will get a syslog message: "SET TABLE Que size too big. Abandoning request for trend: xxxx" when the queue hits the high water mark on the primary trend server.

**Allowable Values:**

1000 to 32000

**Default Value:**

32000

## See Also

[Trend Parameters](#)

### [Trend]MaxSetTableStnbyPending

Controls the maximum number of pending SetTrendTable requests to the standby trend server. It is the primary trend server which uses this INI parameter.

This has been done so that the standby trend server does not receive too many requests.

---

**Note:** When [trend]TrendDebug logging is on you will get a syslog message: "SET TABLE Standby server update has reached 100 max pending high water mark" when the queue hits the high water mark on the primary trend server.

---

## Allowable Values:

1 to 32000

## Default Value:

100

## See Also

[Trend Parameters](#)

### [Trend]MissedSamplesAlarmTime

Controls the amount of trend data that can be missed before a hardware alarm is triggered. A value of 0 disables the alarm. If the value is > 0, an alarm will be raised if a gap occurs which is larger than this value.

## Allowable Values:

0 to 60000 (milliseconds)

## Default Value:

5000 (milliseconds)

## See Also

[Trend Parameters](#)

### [Trend]RangeCheck

Plant SCADA checks for trend values that have fallen outside their full scale and zero scale range. The RangeCheck parameter allows you to specify the action that Plant SCADA should take when such values are detected.

#### Allowable Values:

- 0 - (Clamp values to limits)
- 1 - (Invalidate values - <NA> is displayed)
- 2 - (Clamp values to limits for Scaled [2-byte] storage method.  
Leave values outside the range for Floating Point [8-byte] storage method)
- 3 - (Invalidate values for Scaled [2-byte] storage method. Leave values outside the range for Floating Point [8-byte] storage method)

#### Default Value:

1

#### See Also

### [Trend]ReadWatchTime

Determines how long the trend server will wait (in milliseconds) between each read of trend samples.

Setting this parameter to 0 (zero) causes the read thread to give up the rest of its current CPU time slice.

If this value is -1, there will be no delay between reads. You should avoid setting this parameter to -1 if you have many clients. In extreme cases of many clients doing many trend read requests, it may be necessary to increase the value of ReadWatchTime so that the main Plant SCADA thread still gets all its work done.

#### **WARNING**

##### **UNINTENDED EQUIPMENT OPERATION**

Do not change the [Trend]ReadWatchTime parameter in the Citect.ini file, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

#### Allowable Values:

-1 to 60 000 (milliseconds)

#### Default Value:

1

## See Also

[Trend Parameters](#)

### [Trend]ReloadBackOffTime

Adding or updating trend records too quickly could potentially put pressure to the server disk, causing delays to server responsiveness or trend sample collection.

A back-off time can be configured to control the pace of the reload on a trend server. The back-off time represents the minimal number of milliseconds for which the system will wait after adding or updating a trend record. The system will not attempt to reload another trend record before this back-off time is out.

This configuration value is read at the beginning of each server reload.

#### Allowable Values:

- 0 ~ 2147483647 (milliseconds)

#### Default Value:

1 (millisecond)

## See Also

[Trend Parameters](#)

### [Trend]ShowCacheSizes

Determines whether the currently set values and sample period ranges for [Trend] parameters CacheSize0..CacheSize8 are displayed in the Plant SCADA Kernel.

#### Allowable Values:

- 0 - (Information is not displayed)
- 1 - (Information is displayed)

Default Value:

0

## See Also

[Trend Parameters](#)

## [Trend]TrendDebug

Used to diagnose the trend system during commissioning. Options values are:

- 1 - Logs the client, server, and redundancy message types and also the samples being written in the trend server from normal acquisition, to syslog.dat.
- 2 - Logs detailed information about the currently active backfill process, to syslog.dat. This will include the redundant samples written to the archive.
- 4 - Logs detailed information for the TrendSetTable functions, to syslog.dat.
- 8 - Logs the summary information only of the currently active backfill process, to syslog.dat.
- 16 - Logs the client trend data, to syslog.dat.

These settings can be added together to have combinations of logging levels. For example:

```
[Trend]
TrendDebug=6 ! 4 + 2
```

This will log the detailed backfill process and TrendSetTable functions.

### Allowable Values:

0 to 31

### Default Value:

0

**Note:** You can update this parameter within the Citect.ini file while online, as the runtime system will read its value periodically or on demand. For more information, search the Plant SCADA Help for the topic *Adjusting Logging During Runtime*.

### See Also

[Trend Parameters](#)

## [Trend]WatchTime

The interval between Trends Server requests for data from the I/O devices. This parameter is ignored if you have a trend in your system that requests data at a higher rate - the Trends Server will request data as often as the trends in your system demand. For instance, if you set this value to 500ms, and a trend in your system has a sampling period of 300ms, the Trends Server will cater for this sampling period.

**Note:** The parameter value is not altered when the interval is automatically set for fast trends.

You can set this parameter to any value, but the lower the value, the more frequent the requests to the I/O devices, and this will cause an increased CPU load.

### Allowable Values:

100 to 60000 (milliseconds)

**Default Value:**

500

**See Also**

[Trend Parameters](#)

**[Trend]WriteWatchTime**

**WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not change the [Trend]WriteWatchTime parameter in the Citect.ini file, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

There are two uses for this parameter. The trend write task writes out trend data to disk, and then sleeps for the time specified in this parameter (before writing the next trend file). This small delay stops the trend write task from using too much CPU time when it is writing out a large number of files. If this parameter is set to 0, no delay occurs and the trend data is written to disk as fast as possible.

In addition, this parameter is used to define how long a trend will sleep for after the parameter, BytesWrittenBeforeSleep, has determined how long data will be written to disk.

**Allowable Values:**

0 to 60000 (milliseconds)

**Default Value:**

40

**See Also**

[Trend Parameters](#)

**Trend Process Parameters**

Plant SCADA allows you to specify some parameters for a trend server process (see [Hierarchical Parameters](#) in the Plant SCADA documentation).

The following parameters need to be configured for a specific process.

**Note:** The <server name> is the name of the trend server as specified in the **Topology** activity.

- [\[Trend.<ClusterName>.<ServerName>\]Clusters](#) - Sets the clusters this Trends Server process will connect to

at startup.

- [\[Trend.<ClusterName>.<ServerName>\]CPU](#) - Sets the CPU that the Trend Server process is assigned to.
- [\[Trend.<ClusterName>.<ServerName>\]Events](#) - The list of events that this Plant SCADA Trend Server process enables.
- [\[Trend.<ClusterName>.<ServerName>\]ShutdownCode](#) - Determines the Cicode function to run when Plant SCADA Trend Server process shuts down.
- [\[Trend.<ClusterName>.<ServerName>\]StartupCode](#) - Determines the Cicode function to run when Plant SCADA Trend Server process starts up.

The Citect.ini file on any PC that will be connecting to this <cluster name><server name> trend server can contain the following parameters:

- [\[Trend.<ClusterName>.<ServerName>\]DisableConnection](#) - This setting is used to specify if a client should not connect to a server.
- [\[Trend.<ClusterName>.<ServerName>\]Priority](#) - This setting is used to specify the client priority for the server connection.

## See Also

[Parameter Categories](#)

[Trend Parameters](#)

### [Trend.<ClusterName>.<ServerName>]Clusters

Sets which clusters this Trends Server will connect to at startup.

#### Allowable Values:

A comma separated list of configured cluster names.

#### Default Value:

If this parameter is blank, the Trend Server will connect to every cluster defined for the project.

This parameter needs to be defined at the finest level of hierarchical granularity (Trend.<ClusterName>.<ServerName>Clusters). See [Hierarchical Parameters](#) for more information.

## See Also

[Trend Process Parameters](#)

### [Trend.<ClusterName>.<ServerName>]CPU

This parameter is by Plant SCADA Runtime Manager to determine which CPU(s) the trend server process is assigned to. The CPU parameter is configurable for every Plant SCADA process via the **CPU Setup** page in the Computer Setup Wizard (see [CPU Configuration](#)).

This parameter needs to be defined at the finest level of hierarchical granularity (Trend.<ClusterName>.<ServerName>CPU). See [Hierarchical Parameters](#) for more information.

CPU numbers are specified in a comma separated list. CPU numbers need to be valid and a subset of the available CPUs on a computer. For example, if there are eight CPUs on a computer, a valid CPU range is 0 to 7.

**Note:** If this parameter is modified while the Plant SCADA Runtime Manager is running, the changes will not take effect until every instance of Plant SCADA is restarted.

## Allowable Values:

A comma separated list in the range 0 to 31 (inclusive) where 0 is the first CPU (for example, "0, 1, 2" or "3, 4, 5").

If no value is entered, the process will run on all available CPUs.

**Note:** When a process is set to run on all available CPUs, be aware that performance issues may still occur when your CPU usage is less than 100%. You can detect these issues when your CPU usage for a process is fixed at 100 divided by the number of processors.

## Default Value:

No value (the process will run on all CPUs).

**Note:** You cannot manually specify CPU assignments on a computer with more than 32 CPUs. Under these circumstances, the Setup Wizard will automatically set this parameter to no value (all CPUs). Any changes you make to this parameter will be overwritten.

## See Also

[Trend Process Parameters](#)

[\[Client\]CPU](#)

[\[Alarm.<ClusterName>.<ServerName>\]CPU](#)

[\[IOServer.<ClusterName>.<ServerName>\]CPU](#)

[\[Report.<ClusterName>.<ServerName>\]CPU](#)

## [Trend.<ClusterName>.<ServerName>]DisableConnection

This setting is used to specify whether or not the client sub system should connect to the given server. If the Citect.ini on a particular pc has [Type.ClusterName.ServerName]DisableConnection = 1, specified for a particular server, then the client sub system will establish a non-redundant connection to the other server in the redundant pair. If that connection becomes inoperative, the client sub system will not be redirected and will have no connection until the server is restored.

To define connection priority of a particular server use [\[Trend.<ClusterName>.<ServerName>\]Priority](#)

## Allowable Values:

- 0 - Allow the client to connect to the server
- 1 - Stop the client connecting to the server

**Default Value:**

0

**See Also**[Trend Process Parameters](#)**[Trend.<ClusterName>.<ServerName>]Events**

The list of events that this Plant SCADA Trend Server will enable.

This parameter must be defined at the finest level of hierarchical granularity (Trend.<ClusterName>.<Server>Events). See [Hierarchical Parameters](#) for more information.

---

**Note:** This parameter is modified by the Computer Setup Wizard.

---

**Allowable Values:**

List of events (comma separated).

**Default Value:**

NONE

---

**Note:** When Plant SCADA is running in single process mode only the events assigned to [\[Client\]Events](#) will be enabled. Events specified for particular Plant SCADA components are ignored.

---

**See Also**[Trend Process Parameters](#)**[Trend.<ClusterName>.<ServerName>]Priority**

This setting is used to specify the client sub system's priority for the connection to this specific server. The client sub system will connect to, or switch over to, any available server that has the highest priority (lowest number). If the priorities for multiple servers are set to the same value, the client sub system will only switch on a loss of the currently active connection. This is the default operation to provide the same behavior as in versions of the product prior to v7.20.

To disable (not allow) the client to connect to the server use

[\[Trend.<ClusterName>.<ServerName>\]DisableConnection](#)

**Allowable Values:**

1 to 32767

**Default Value:**

1

**See Also**

[Trend Process Parameters](#)

**[Trend.<ClusterName>.<ServerName>]ShutdownCode**

Can be used to specify a Cicode function that runs at shutdown of the Plant SCADA Trend Server component. Plant SCADA suspends the shutdown process until the function has finished running. The function needs to complete execution within the time specified by the [\[Code\]ShutdownTime](#) parameter, otherwise Plant SCADA will terminate it.

This parameter needs to be defined at the finest level of hierarchical granularity (Trend.<ClusterName>.<ServerName>ShutdownCode). See [Hierarchical Parameters](#) for more information.

---

**Note:** This parameter is modified by the Computer Setup Wizard.

**Allowable Values:**

Any built-in or user-written Cicode function, 1-31 alpha-numeric characters.

**Default Value:**

NONE

---

**Note:** When Plant SCADA is running in single process mode only the Cicode function assigned to [\[Client\]ShutdownCode](#) will be executed at shutdown. Cicode functions specified for particular Plant SCADA components are ignored.

**See Also**

[Trend Process Parameters](#)

**[Trend.<ClusterName>.<ServerName>]StartupCode**

Determines the Cicode function to run when the Plant SCADA Trend Server component starts up.

You can only pass constant data to the startup function call. You cannot pass variables or other functions. For example, MyFunc(1, "str") is valid, but MyFunc(PLCTAG, "str") or MyFunc(YourFunc(), "str") is not supported.

This parameter needs to be defined at the finest level of hierarchical granularity (Trend.<ClusterName>.<ServerName>StartupCode). See [Hierarchical Parameters](#) for more information.

---

**Note:** This parameter is modified by the Computer Setup Wizard.

**Allowable Values:**

Any built-in or user-written Cicode function, 1-31 alpha-numeric characters.

**Default Value:**

NONE

---

**Note:** When Plant SCADA is running in single process mode only the Cicode function assigned to [Client]StartupCode will be executed at startup. Cicode functions specified for particular Plant SCADA components are ignored.

---

**See Also**

[Trend Process Parameters](#)

**Win Parameters**

The Citect.ini file contains the following Win parameters:

- [\[Win\]AltEsc](#) - Determines whether the Windows key command [Alt]+[Esc] can be used in the runtime system (to cycle through open applications).
- [\[Win\]AltSpace](#) - Determines whether the Windows key command [Alt]+[Space] can be used in the runtime system (to display the control menu).
- [\[Win\]AltTab](#) - Determines whether the Windows key command [Alt]+[Tab] can be used in the runtime system (to cycle through open applications).
- [\[Win\]CtrlEsc](#) - Obsolete in version 7.10.
- [\[Win\]Configure](#) - Determines whether the Project Editor and Graphics Builder options are displayed on the control menu of the Plant SCADA runtime system.
- [\[Win\]ScreenSaver](#) - Determines whether the Windows screen saver is active when Plant SCADA is running.
- [\[Win>ShowAbout](#) - Determines if the Parameter to control whether the about box is added to the ALT-SPACE menu of the Plant SCADA runtime system.

**See Also**

[Parameter Categories](#)

**[Win]AltEsc**

Determines whether the Windows key command [Alt]+[Esc] can be used in the runtime system (to cycle through open applications).

---

**Note:** This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

---

**Allowable Values:**

- 0 - (Ignore the [Alt] [Esc] key)
- 1 - (Cycle through open applications)

**Default Value:**

1

**See Also**[Win Parameters](#)**[Win]AltSpace**

Determines whether the Windows key command [Alt]+[Space] can be used in the runtime system (to display the control menu).

---

**Note:** This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

---

**Allowable Values:**

- 0 - (Ignore the [Alt] [Space] key)
- 1 - (Display the control menu)

**Default Value:**

1

**See Also**[Win Parameters](#)**[Win]AltTab**

Determines whether the Windows key command [Alt]+[Tab] can be used in the runtime system (to cycle through open applications).

---

**Note:** This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

---

**Allowable Values:**

- 0 - (Ignore the [Alt] [Tab] key)
- 1 - (Cycle through open applications)

**Default Value:**

1

**See Also**[Win Parameters](#)**[Win]Configure**

Determines whether the **Graphics Builder** options are displayed on the control menu of the Plant SCADA runtime system.

---

**Note:** This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

---

**Allowable Values:**

- 0 - (Do not display control menu options)
- 1 - (Display options)

**Default Value:**

1

**See Also**[Win Parameters](#)**[Win]ScreenSaver**

Determines whether the Windows screen saver is active when Plant SCADA is running. The screen saver changes the image on the screen to a random image (if it is not changed within a certain period).

---

**Note:** This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

---

**Allowable Values:**

- 0 - (Screen saver disabled)
- 1 - (Screen saver enabled)

**Default Value:**

1

## See Also

[Win Parameters](#)

### [Win]ShowAbout

Determines if the Parameter to control whether the About box is added to the ALT-SPACE menu of the Plant SCADA runtime system.

**Note:** This parameter is modified by the Computer Setup Wizard as well as the Computer Setup Editor.

## Allowable Values:

- 0 - (About is not listed in the ALT-SPACE menu.)
- 1 - (About is listed in the ALT-SPACE menu.)

## Default Value:

0

## See Also

[Win Parameters](#)

## Workspace Parameters

The [Workspace] section contains the following parameters:

- [\[Workspace\]AlarmSortOrder](#) - Defines a customized sort order for the default alarms pages in a Situational Awareness project.
- [\[Workspace\]EquipmentNavigationModelFilter](#) - Filters the equipment model loaded by the workspace.
- [\[Workspace\]InfoZoneAlarms.IncludeReferences](#) - Determines if alarms from referenced equipment are included in the Alarms page on the Information Zone.
- [\[Workspace\]InfoZoneTrends](#) - Determines the pens that will be displayed in the Information Zone of the workspace based on the equipment items configured.
- [\[Workspace\]InfoZoneTrends.Type](#) - Determines the pens that will be displayed in the Information Zone of the workspace based on the equipment types configured in the Equipment Editor.
- [\[Workspace\]NumberOfTopPriorities](#) - Determines how many alarm priorities are included in the alarm counts presented on an alarm page tree view and the Navigation Zone.

## See Also

[Situational Awareness Parameters](#)

[Parameter Categories](#)

## [Workspace]AlarmSortOrder

Defines a customized sort order for the following alarm lists in a Situational Awareness project:

- Active Alarms page
- Alarms panel in the Information Zone
- Top 5 alarms pop-up for the HD1080 workspace
- Alarm Summary panel for the UHD4K workspace
- Shelved Alarms page.

To specify a sort order, use the following syntax:

{*KeyName,SortDirection*} {*KeyName,SortDirection*}

*KeyName* specifies the name of the pre-defined order-by key to be used. The valid options are a subset of the alarm display fields:

- Tag
- Name
- Category
- Priority
- Area
- Equipment
- Priv
- Time
- State.

*SortDirection* (optional) indicates whether the sort will be ascending or descending. Valid options are:

- 0 = Descending (default)
- 1 = Ascending

### Allowable Values:

Any valid sort order string.

### Default Value:

Empty string.

If the above parameter is not defined, the alarm list will be sorted according to the current setting for the alarm server's **Highest Priority Order** field ("Priority, Alarm State" or "Alarm State, Priority").

### Examples

Sort by **Tag** ascending, and **Time** descending:

```
[Workspace]AlarmSortOrder={Tag,1}{Time}
```

Sort by **Time** ascending and **Tag** descending:

```
[Workspace]AlarmSortOrder={Time,1}{Tag,0}
```

Sort by **Equipment** ascending and **Tag** descending:

```
[Workspace]AlarmSortOrder={Equipment,1}{Tag,0}
```

## See Also

[Workspace Parameters](#)

### [Workspace]EquipmentNavigationModelFilter

Use this parameter to filter the equipment model loaded by the workspace. For example the equipment tree on the trend page, and alarm pages. It is also used as part of the workspace auto-fill functionality.

#### Allowable Values:

Any valid filter.

An empty string indicates that all records will be returned. Where a fieldname is not specified in the filter, it is assumed to be equipment name. For example, the filter "AAA" is equivalent to "name=AAA".

The following regular expressions are supported: \*expr, expr\*, and \*expr\*. To specify an exclusion filtering condition, use the NOT keyword after the = operator.

See the topic **EquipBrowseOpen** in the Cicode Reference for list of Fields.

#### Default Value:

Empty string.

#### Examples

Filter by Drive01:

```
[Workspace]EquipmentNavigationModelFilter=PlantA.Pump1.Drive01
```

Filter by Pump1:

```
[Workspace]EquipmentNavigationModelFilter=PlantA.Pump1
```

Filter by Pump1 and subequipment:

```
[Workspace]EquipmentNavigationModelFilter=PlantA.Pump1.*
```

Filter by Cluster1 and all sub equipment in PlantA:

```
[Workspace]EquipmentNavigationModelFilter = Cluster=Cluster1; Name=PlantA.*
```

Filter by Cluster1 and Pump1 and Pump2:

```
[Workspace]EquipmentNavigationModelFilter = Cluster=Cluster1;  
Name=PlantA.Pump1,PlantA.Pump2
```

Filter by Cluster2 and all subquipment in PlantC:

```
[Workspace]EquipmentNavigationModelFilter = Cluster=Cluster2; Name=PlantC.*
```

## See Also

[Workspace Parameters](#)

### [Workspace]InfoZoneAlarms.IncludeReferences

Determines if alarms from referenced equipment are included in the Information Zone **Alarms** tab on the Situational Awareness Operator Dashboard. This is a client-side setting and will affect all equipment.

#### Allowable Values:

- 0 = Alarms from referenced equipment are not included
- 1 = Alarms from referenced equipment are included

#### Default Value:

1

## See Also

[Workspace Parameters](#)

### [Workspace]InfoZoneTrends

Determines the pens that will be displayed in the Information Zone of the workspace based on the equipment items configured. The value of this parameter is a comma-separated list of equipment item names. Up to five equipment items can be configured. Pens will be displayed in the order in which the equipment items are defined by this parameter.

---

**Note:** A warning is logged to syslog.dat if there are any invalid or duplicate items, that is, a value does not correspond to an item with a trend tag.

---

#### Allowable Values:

Any valid equipment items with a trend tag configured.

#### Default Value:

PV,OP, FB,SP

## See Also

[Workspace Parameters](#)

### [Workspace]InfoZoneTrends.Type

Determines the pens that will be displayed in the Information Zone of the workspace based on the equipment types configured in the Equipment Editor. The value of this parameter is a comma-separated list of equipment item names. Up to five equipment items can be configured. Pens will be displayed in the order in which the equipment items are defined by this parameter.

**Note:** A warning is logged to syslog.dat if there are any invalid or duplicate items, that is, a value does not correspond to an item with a trend tag.

#### Allowable Values:

Any valid equipment items with a trend tag configured.

#### Default Values:

Equipment Type	Default Value
InfoZoneTrends.FeederGate	PV,OP,FB,Height
InfoZoneTrends.TargetMeter	PV,PVTarget,OP,FB,SP
InfoZoneTrends.WindCompass	WindPV,WindPVAvg,WindSpeed,WindSpeedAvg

#### Example

InfoZoneTrends.Drive=PV,SP,FB

#### See Also

[Workspace Parameters](#)

### [Workspace]NumberOfTopPriorities

Determines how many alarm priorities are included in the alarm counts presented in the following locations:

- The Tree View on alarm and trend pages
- The tabs and buttons in the Navigation Zone.

Specifically, you can use this parameter to include the top four alarm priorities rather than just the top three. This setting only applies to projects based on the Situational Awareness Starter project.

#### Allowable Values:

3 or 4

**Default Value:**

3

For more information on how to implement a fourth alarm priority, see the topic *Configure Display Properties for a Fourth Alarm Priority* in the main Plant SCADA help.

**See Also**[Workspace Parameters](#)

## Logging Parameters

These parameters can be used to gather runtime information about your Plant SCADA system. Before making adjustments to these parameters, you need to understand the concepts and processes described in the Monitoring Runtime chapter in the Plant SCADA help.

Similarly, the section on I/O Devices includes information that explains how to log drivers (see [Troubleshooting Device Communications](#)).

Plant SCADA includes the following logging parameters.

Parameter	Description
[Alarm]EnableErrorLogging	Enables or disables the logging of operational errors.
[Alarm]EnableStateLogging	Enables or disables the logging of major alarm system state changes.
[Code]DebugMessage	Enables the DebugMsg Cicode function, and controls the logging functionality of the Assert function.
[Code]EnableErrorLogging	Enables or disables the logging of issues with the ErrSet() Cicode function and related functions.
[Debug]ArchiveFiles	Enables or disables the archiving of the tracelog.dat log file once the maximum size specified by [Debug]MaximumFileSize is reached.
[Debug]CategoryFilter	Allows you to filter logging messages sent to tracelog.dat by component category. <sup>1</sup>
[Debug]CategoryFilterMode	Enables logging of categories declared by [Debug]CategoryFilter. <sup>1</sup>
[Debug]DriverTrace	Enables the Kernel's Driver Trace utility. <sup>1</sup>
[Debug]EnableLogging	Enables or disables logging to the tracelog.dat file. <sup>1</sup>
[Debug]LogDirect	Indicates whether to log directly to a log file or via the Kernel logging system.

Parameter	Description
[Debug]MaximumFileSize	Sets the maximum size for tracelog.dat log file, in kilobytes.
[Debug]Priority	Allows you to filter messages logged to the tracelog.dat file according to their priority. <sup>1</sup>
[Debug]SeverityFilter	Allows you to filter messages logged to the tracelog.dat file according to their severity. <sup>1</sup>
[Debug]SeverityFilterMode	Enables logging of severities declared by the [Debug]SeverityFilter value. <sup>1</sup>
[Debug]SysLogArchive	Enable the creation of syslog.dat archive files. <sup>1,2</sup>
[Debug]SysLogSize	Sets the maximum size of the syslog.dat file. <sup>1</sup>
[Debug]SysErrDsp	Allows you to disable many system error logs and popup boxes.
[Dial]Debug	Enables or disables scheduled I/O device debugging.
[Dial]DebugLevel	Allows you to set the trace level for scheduled I/O device debugging. <sup>2</sup>
[General]Verbose	Enables a startup dialog box which shows the opening runtime databases and other information. <sup>1</sup>
[General]VerboseToSysLog	Sends additional startup information generated by [General]Verbose to syslog.dat. <sup>1</sup>
[Kernel]ErrorBuffers	Determines the maximum number of error buffers available for logging to the syslog.dat file.
[Trend]TrendDebug	Used to determine problems with the trend system during commissioning. <sup>2</sup>

1. You can adjust these parameters during runtime using the SetLogging() Cicode function.
2. You can update this parameter within the Citect.ini file while online, as the runtime system will read its value periodically or on demand.

For more information on 1 and 2, search the Plant SCADA Help for the topic *Adjusting Logging During Runtime*.

## See Also

[Advanced Logging Parameters](#)

## Advanced Logging Parameters

The following parameters are recommended only for advanced users.

Parameter	Description
[CtAPI]EventLogging	Enables logging of the communications between CtAPI clients and Plant SCADA.
[Debug]DebugAllTrans	Allows TRAN session logging to be turned on or off. <sup>1</sup>
[Debug]EnableReloadLogging	Enables additional debugging output for server reloads. <sup>2</sup>
[Debug]LogShutdown	Logs the shutdown function addresses to syslog.dat. This INI is only needed if [Debug]ShutDownProtect is active. <sup>1</sup>
[General]ShowDriverError	Determines whether an error dialog displays when an internal driver error occurs. <sup>2</sup>
[IOServer]RedundancyDebug	Enables I/O device redundancy debugging. <sup>1</sup>
[IPC]EventLogging	Enables logging of the communications between processes.
[Keyboard]LogExtendedDate	Determines the date format is used in command logs.
[LAN]ConnectionLogging	Provides legacy component connection logging to the syslog.dat.
[PubSub]LogLevel	Sets the logging level between the I/O subscription system and the Plant SCADA drivers. <sup>2</sup>
[PubSub]LogDevice	Selects the device(s) logged between the I/O subscription system and the Plant SCADA drivers. <sup>2</sup>

1. You can adjust these parameters during runtime using the SetLogging() function.
2. You can update this parameter within the Citect.ini file while online, as the runtime system will read its value periodically or on demand.

For more information on 1 and 2, search the Plant SCADA Help for the topic *Adjusting Logging During Runtime*.

## See Also

[Logging Parameters](#)

## Template Parameters

These parameters can be used to manage the configuration of the templates in pre-configured projects.

The following Plant SCADA template projects include parameters that you can modify:

- [Tab\\_Style Template Parameters](#)
- [Situational Awareness Parameters](#)

### Tab\_Style Template Parameters

Below are the sections that describe the Tab\_Style Template parameters recognized by Plant SCADA.

Section
<a href="#">Alarm Parameters</a>
<a href="#">Alarm Heading Parameters</a>
<a href="#">Format Parameters</a>
<a href="#">Page Parameters</a>
<a href="#">Privilege Parameters</a>
<a href="#">[TabAlarm.Custom] Parameters</a>
<a href="#">[Tabmenu.Custom] Parameters</a>

#### Alarm Parameters

The Tab\_Style Templates will recognize the following alarm parameters:

- [\[Alarm\]BackgroundColorMode](#) - Sets the background color mode of the alarm list and alarm banner on the Tab\_Style templates.
- [\[Alarm\]LastAlarmCategories](#) - Determines which alarms are displayed on the alarms toolbar, based on category.
- [\[Alarm\]LastAlarmDisplayMode](#) - Determines whether the last alarm is displayed by category or priority.
- [\[Alarm\]LastAlarmFmt](#) - Specifies the format of the last alarm as displayed on the included templates.
- [\[Alarm\]LastAlarmPriorities](#) - Determines which alarms are displayed on the alarms toolbar, based on priority.
- [\[Alarm\]LastAlarmType](#) - Determines which alarms are displayed in the alarms toolbar, based on type.
- [\[Alarm\]Sound<n>](#) - Determines the .wav file that is used when an alarm sounds, based on the priority of the alarm <n>.
- [\[Alarm\]Sound<n>Interval](#) - Determines the interval of time between an alarm sound, based on the priority of the alarm <n>.
- [\[Alarm\]SoundSilenceTimeoutOnAck](#) - Determines the timeout period in seconds for silencing alarm after the user acknowledges any alarms using the Tab\_Style templates.

- [\[Alarm\]HeadingFont](#) - Determines the text font used for column headings displayed on the alarm templates.

## Alarm Heading Parameters

The Tab\_Style Templates will recognize the following [AlarmHeading] parameters:

- [\[AlarmHeading\]AlarmField](#) - Specifies alternative names for the standard alarm field names for display in the column heading of the new Tab\_Style alarm template.

## Format Parameters

The Tab\_Style Templates will recognize the following [Format] parameters:

- [\[Format\]FormatName](#) - Define a display format under a specified name.

## Page Parameters

The Tab\_Style Templates will recognize the following page parameters:

- [\[Page\]AddDefaultMenu](#) - Determine whether to add the default menu items to your tabbed menu bar.
- [\[Page\]AlarmPage](#) - The name of the graphics page to display when calling the Cicode function PageAlarm().
- [\[Page\]DisabledPage](#) - The name of the graphics page to display when calling the Cicode function PageDisabled().
- [\[Page\]HardwarePage](#) - The name of the graphics page to display when calling the Cicode function PageHardware().
- [\[Page\]HomePage](#) - Sets the page to display when the user clicks the Home page button on the template or calls the Cicode function PageHome()
- [\[Page\]Logo](#) - This sets a symbol to display at the logo area (lower right corner) of the new templates.
- [\[Page\]PrintPage](#) - Sets a custom Cicode function to print the currently displayed page when the user clicks the Print page button on the template or calls the Cicode function PagePrint().
- [\[Page\]ProcessAnalystPage](#) - Sets the page to display when the user calls the Cicode function PageProcessAnalyst() without specifying the page argument.
- [\[Page\]ProcessAnalystPopupPage](#) - Sets the page to pop up when the user calls the Cicode function ProcessAnalystWin() or ProcessAnalystPopup() without specifying the page argument.
- [\[Page\]SummaryPage](#) - The name of the graphics page to display when calling the Cicode function PageSummary().

## Privilege Parameters

The Tab\_Style Templates will recognize the following [Privilege] parameters:

- [\[Privilege\]AckAlarms](#) - Determines the privilege level a user requires to acknowledge alarms.
- [\[Privilege\]DisableAlarms](#) - Determines the privilege level a user requires to disable alarms.
- [\[Privilege>EditUser](#) - Determines the privilege level a user requires to edit users using the drop-down menu next to the login user icons on the Tab\_Style template.

- [\[Privilege\]Shutdown](#) - Determines the privilege level a user requires to shutdown a project.
- [\[Privilege\]SilenceAlarms](#) - Determines a privilege level a user requires to silence alarms.

### [TabAlarm.Custom] Parameters

The [TabAlarm.Custom] section contains the following parameters:

- [\[TabAlarm.Custom\]Function.AlarmGetDsp](#) - Define display format under a specified name.
- [\[TabAlarm.Custom\]Function.Row.ShowContextMenu](#) - Define display format under a specified name.
- [\[TabAlarm.Custom\]Function.Row.ShowHWContextMenu](#) - Define display format under a specified name.

## [TabAlarm.Custom]Function.AlarmGetDsp

Sets your own function to return the value of the specified alarm field on the tab style alarm templates. If specified, your function, instead of the default one, will be called to return the value for the requested alarm field.

**Note:** This parameter is available only to projects based on the Tab\_Style templates.

### Allowable Values:

Name of user defined function which conforms to the following specification:

### Syntax

<function>(INT *listID*, INT *recordAN*, STRING *field*)

- *listID*: ID of the alarm list
- *recordAN*: Animation Number (AN) of the alarm record
- *field*: Name of the alarm field being requested

### Default Value:

None

### Return Value:

Value of the alarm field

### Example

```
[TabAlarm.Custom]
Function.AlarmGetDsp = MyAlarmValue
STRING FUNCTION MyAlarmValue(INT listID, INT recordAN, STRING field)
    // return a textual description for priority
```

```
IF (field = "Priority") THEN
    SELECT CASE AlarmGetDsp(recordAN, field)
    CASE "1"
        RETURN "Critical";
    CASE "2"
        RETURN "Major";
    CASE ELSE
        RETURN "Minor";
    END SELECT
ELSE
    RETURN AlarmGetDsp(recordAN, field);
END
END
```

## See Also

[\[TabAlarm.Custom\] Parameters](#)

# [TabAlarm.Custom]Function.Row.ShowContextMenu

Sets your own function to show the context menu to response to the right-clicking of an alarm on the tab style alarm templates.

**Note:** This parameter is available only to projects based on the Tab\_Style templates.

## Allowable Values:

Name of user defined function which conforms to the following specification:

## Syntax

```
<function>(INT listID, INT rowID)  
listID: ID of the alarm list  
rowID: Row number (starting from 0) of the clicked alarm
```

## Default Value:

None

## Return Value:

0 if run successfully or error code otherwise

### Example

```
[TabAlarm.Custom]
Function.Row.ShowContextMenu =
.
INT FUNCTION MyAlarmContextMenu(INT listID, INT rowID)
```

```
// Get AN of the alarm list
INT listAN = TabAlarm_GetAN(listID);
INT selection;
// Show menu
DspPopupMenu(0, "Show Details");
DspPopupMenu(0, "Acknowledge");
selection = DspPopupMenu();
// Run selected command
SELECT CASE selection
CASE 1
    RETURN TabAlarm_Row_ShowInfo(listID, rowID);
CASE 2
    IF TabAlarm_GetAckPriv() THEN
        RETURN AlarmAckRec(AlarmGetDsp(listAN + rowID, "RecNo"));
    ELSE
        Message("Error", ErrMsg(276), 16);
        RETURN 276; // no privilege for operation
    END
END SELECT
RETURN 0;
END
```

## See Also

[\[TabAlarm.Custom\] Parameters](#)

# [TabAlarm.Custom]Function.Row.ShowHWContextMen

Sets your own function to show the context menu to response to the right-clicking of an alarm on the tab style hardware alarm template.

---

**Note:** This parameter is available only to projects based on the Tab\_Style templates.

---

## Allowable Values:

Name of user defined function which conforms to the following specification:

## Syntax

<function>(INT *recordAN*)  
*recordAN*: Animation Number (AN) of the alarm record

## Default Value:

None

**Return Value:**

0 if run successfully or error code otherwise

**Example**

```
[TabAlarm.Custom]
Function.Row.ShowHWContextMenu = MyHardwareContextMenu

INT FUNCTION MyHardwareContextMenu(INT recordAN)
    INT selection;

    // Show menu
    DspPopupMenu(0, "Acknowledge");
    selection = DspPopupMenu();

    // Run selected command
    SELECT CASE selection
    CASE 1
        IF TabAlarm_GetAckPriv() THEN
            KeySetCursor(recordAN);
            RETURN AlarmAck(0, 0);
        ELSE
            Message("Error", ErrMsg(276), 16);
            RETURN 276; // no privilege for operation
        END
    END SELECT

    RETURN 0;
END
```

**See Also**

[\[TabAlarm.Custom\] Parameters](#)

**[Tabmenu.Custom] Parameters**

The [Tabmenu.Custom] section contains the following parameters:

- [\[Tabmenu.Custom\]Font.Disabled](#) - Define display format under a specified name.
- [\[Tabmenu.Custom\]Font.Normal](#) - Define display format under a specified name.
- [\[Tabmenu.Custom\]Function.CreateDefaultMenu](#) - Define display format under a specified name.

## [Tabmenu.Custom]Font.Disabled

Sets a user defined font for the text displayed on the tab menu when an item is disabled.

**Note:** This parameter is available only to projects based on the Tab\_Style templates.

**Allowable Values:**

Name of user defined font

**Default Value:**

None (When not specified, the Arial, 10pt font in grey-blue color is used)

**See Also**

[\[Tabmenu.Custom\] Parameters](#)

## [Tabmenu.Custom]Font.Normal

Sets a user defined font for the text displayed on the tab menu when an item is in normal state.

**Note:** This parameter is available only to projects based on the Tab\_Style templates.

**Allowable Values:**

Name of user defined font

**Default Value:**

None (When not specified, the Arial, 10pt font in dark-blue color is used)

**See Also**

[\[Tabmenu.Custom\] Parameters](#)

## [Tabmenu.Custom]Function.CreateDefaultMenu

Sets your own function to create the default menu for the tab style templates.

**Note:** This parameter is available only to projects based on the Tab\_Style templates.

This parameter is used to replace the built-in default menu and is effective only if the Menu Configuration form is blank. If there are items defined in the Menu Configuration form, the menu will be displayed according to the contents of the form.

Note that the INI parameter [Page]AddDefaultMenu determines items to be displayed on the tabbed menu bar. The default value of this parameter is 1 where the menu is populated as described earlier. However, if [Page]AddDefaultMenu=0 and a [Tabmenu.Custom]Function.CreateDefaultMenu is specified, the menu is populated based on the contents of the Menu Configuration settings, not the Cicode defined in [Tabmenu.Custom]Function.CreateDefaultMenu.

Setting [Page]AddDefaultMenu=2 and specifying a Cicode function in

[Tabmenu.Custom]Function.CreateDefaultMenu, will result in merging items defined in Menu Configuration form to items populated via Cicode function.

## Allowable Values:

Name of user defined function which conforms to the following specification:

## Syntax

<function>(INT *winNo*)

*winNo*: The window number of the window where the menu is displayed

## Default Value:

None

## Return Value:

The menu node handle for the current window

### Example

```
[Tabmenu.Custom]
Function.CreateDefaultMenu = MyDefaultMenu

INT FUNCTION MyDefaultMenu(INT winNo)
    INT winNode = MenuGetWindowNode(winNo);
    INT tabNode, childNode;

    // Add Pages tab
    tabNode = Tabmenu_AddChild(winNode, "Pages");
    IF (tabNode >= 0) THEN
        // Populate predefined pages
        Tabmenu_AddChild(tabNode, "Page 1", "PageDisplay", "^^Page1^^");
        Tabmenu_AddChild(tabNode, "Page 2", "PageDisplay", "^^Page2^^");
        Tabmenu_AddChild(tabNode, "Page 3", "PageDisplay", "^^Page3^^");
    END

    // Add Alarms tab
    tabNode = Tabmenu_AddChild(winNode, "Alarms");
    IF (tabNode >= 0) THEN
        // Populate alarm pages
        Tabmenu_AddChild(tabNode, "Active Alarms", "PageAlarm", "");
        Tabmenu_AddChild(tabNode, "Alarm Summary", "PageSummary", "");
        Tabmenu_AddChild(tabNode, "Disabled Alarms", "PageDisabled", "");
        Tabmenu_AddChild(tabNode, "Hardware Alarms", "PageHardware", "");
    END

    // Add Trends tab
    tabNode = Tabmenu_AddChild(winNode, "Trends");
    IF (tabNode >= 0) THEN
        // Populate trend pages
        Tabmenu_AddChild(tabNode, "Trends", "PageProcessAnalyst", "^^^,^^");
    END
```

```
    Tabmenu_AddChild(tabNode, "Trend Pop-up", "ProcessAnalystPopup", "^\"^");
END

// Add Tools tab
tabNode = Tabmenu_AddChild(winNode, "Tools");
IF (tabNode >= 0) THEN
    // Populate administrative pages / commands
    Tabmenu_AddChild(tabNode, "Log", "PageFile", "^"[DATA]:OperLog.txt^");
    childNode = Tabmenu_AddChild(tabNode, "Exit", "ShutdownForm", "");
    MenuNodeSetProperty(childNode, 2, 8); // set privilege to 8
END

RETURN winNode;
END
```

## See Also

- [\[Tabmenu.Custom\] Parameters](#)
- [\[Page\]AddDefaultMenu](#)

## Situational Awareness Parameters

Below are the sections that contain the Situational Awareness Include parameters recognized by Plant SCADA.

- [\[Format\] Parameters](#)
- [\[MultiMonitors\] Parameters](#)
- [\[SA\\_Library.Meter\] Parameters](#)
- [\[Workspace\] Parameters](#)

## See Also

- [Template Parameters](#)

### Format Parameters

The Tab\_Style Templates will recognize the following [Format] parameters:

- [\[Format\]FormatName](#) - Define a display format under a specified name.

### [MultiMonitors] Parameters

- [\[MultiMonitors\]Context<n>](#) - Allows you to share a single context with one or more screens.
- [\[MultiMonitors\]Workspaces](#) - Indicates the number of workspace pages to be shown at startup.

## See Also

- [Template Parameters](#)

## [SA\_Library.Meter] Parameters

The [SA\_Library.Meter] parameters allow you to define and display a range of PLC alarm limits. Where standard analog alarm limits are limited to four states, there is no limit on the number of PLC alarm limits that can be defined and displayed.

The [SA\_Library.Meter] section comprises the following parameters:

- [\[SA\\_Library.Meter\]UseDefaultPLCLimits](#)
- [\[SA\\_Library.Meter\]PLCLimitNames](#)
- [\[SA\\_Library.Meter\]PLCLimitLabels](#)
- [\[SA\\_Library.Meter\]DefaultPLCLimits](#)

## See Also

[Situational Awareness Parameters](#)

## [SA\_Library.Meter]UseDefaultPLCLimits

Allows you to use PLC alarm limits for faceplates.

### Allowable Values:

TRUE, FALSE.

### Default:

None

When set to TRUE, the system will display the defined PLC alarm limits. Otherwise, it will use the standard analog alarm limits for display on faceplates.

## See Also

[\[SA\\_Library.Meter\]PLCLimitNames](#)

[\[SA\\_Library.Meter\]PLCLimitLabels](#)

[\[SA\\_Library.Meter\]DefaultPLCLimits](#)

## [SA\_Library.Meter]PLCLimitNames

Names for PLC alarm limits. This is a comma-separated list of values. For example, Limit1, Limit2, Limit3 and so on. You should specify all possible limits that may be used by equipment in the project. There is no limit on the number of PLC alarm limits that you can define.

**Note:** Each limit should correspond to an Item Name, and may also be associated with a piece of equipment

---

through a tag. For example, if the equipment is Meter1 and the PLC alarm limit is Limit1, you could have a tag Meter1.Limit1.

---

PLC alarm limits are applicable to the entire project.

**Allowable Values:**

Any valid limit names.

**Default:**

None

**See Also**

[\[SA\\_Library.Meter\]UseDefaultPLCLimits](#)

[\[SA\\_Library.Meter\]PLCLimitLabels](#)

[\[SA\\_Library.Meter\]DefaultPLCLimits](#)

## [SA\_Library.Meter]PLCLimitLabels

Display labels that correspond to each PLC alarm limit defined. This is a comma-separated list of values. For example, L1, L2, L3 and so on. You should define one label for each limit defined. It is recommended that the label not be more than 2 to 3 characters long so that the entire label is visible on the faceplates.

---

**Note:** The order of the PLC limit labels needs to match the order of the PLC limit names.

---

**Allowable Values:**

Any valid names.

**Default:**

None

**See Also**

[\[SA\\_Library.Meter\]UseDefaultPLCLimits](#)

[\[SA\\_Library.Meter\]PLCLimitNames](#)

[\[SA\\_Library.Meter\]DefaultPLCLimits](#)

## [SA\_Library.Meter]DefaultPLCLimits

A number that determines the order in which to use the defined PLC alarm limits. 0 corresponds to the lowest

PLC alarm limit defined, while the largest number corresponds to the highest limit defined. Note that if you have defined limits as Limit1, Limit2 and Limit3:

- 0 will correspond to Limit1
- 1 will correspond to Limit2
- 2 will correspond to Limit3

In the example above, you cannot specify "3" or "4" or any other number as one of the values for the DefaultPLCLimit. The values can only be 0,1 and/or 2.

This is a comma-separated list of values.

For example, if your list contains the values 0,1,4,5,3 and the limits you have defined are Limit1, Limit2, Limit3, Limit4, Limit5 and Limit6, the order will be:

DefaultPLCLimit	PLCLimitNames
0	Limit1
1	Limit2
4	Limit5
5	Limit6
3	Limit4

It is possible to override the limits set up at a project level, and apply these limits to an individual piece of equipment. This can be done in the **System Model** activity | **Runtime Parameters** view.

To associate PLC alarm limits with a piece of equipment:

- Define the **[SA\_Library.Meter]PLCLimitNames** to use.
- In Plant SCADA Studio, navigate to **System Model** activity | **Equipment**. Select **Runtime Parameters** from the dropdown menu.
- For each equipment, in the **Name** property, type PLCLimits.
- In the **Value** property, specify your comma-separated list of default PLC limits, that is the order in which the defined PLC alarm limits need to be used.

---

**Note:** Setting the **Value** property to \* reverses the setting of the UseDefaultPLCLimits parameter. Therefore, if the UseDefaultPLCLimits parameter is set to TRUE, it will be reversed and applied as if it were set to FALSE. In this example, analog alarm limits will apply. If UseDefaultPLCLimits is FALSE, setting the value of the runtime parameter to \* will cause the associated equipment to use the default PLC limits defined at the project level.

---

- Set the **IsTag** property to FALSE so that the PLC limit labels are displayed on the meter faceplates.
- Click **Save**.

## Allowable Values:

Any whole number.

**Default:**

None

**See Also**

[\[SA\\_Library.Meter\]UseDefaultPLCLimits](#)  
[\[SA\\_Library.Meter\]PLCLimitNames](#)  
[\[SA\\_Library.Meter\]PLCLimitLabels](#)

**[Workspace] Parameters**

The [Workspace] section contains the following parameters:

- [\[Workspace\]AlarmSortOrder](#) - Defines a customized sort order for the default alarms pages in a Situational Awareness project.
- [\[Workspace\]EquipmentNavigationModelFilter](#) - Filters the equipment model loaded by the workspace.
- [\[Workspace\]InfoZoneAlarms.IncludeReferences](#) - Determines if alarms from referenced equipment are included in the Alarms page on the Information Zone.
- [\[Workspace\]InfoZoneTrends](#) - Determines the pens that will be displayed in the Information Zone of the workspace based on the equipment items configured.
- [\[Workspace\]InfoZoneTrends.Type](#) - Determines the pens that will be displayed in the Information Zone of the workspace based on the equipment types configured in the Equipment Editor.
- [\[Workspace\]NumberOfTopPriorities](#) - Determines how many alarm priorities are included in the alarm counts presented on an alarm page tree view and the Navigation Zone.

**See Also**

[Situational Awareness Parameters](#)

## VBA Programming Reference

VBA is a Visual Basic for Applications (VBA) and VBScript-compatible Basic scripting language. Plant SCADA has embedded support for VBA.

---

**Note:** VBA is not supported on a 64-bit process, such as an alarm server operating in Extended Memory mode. If a call to VBA code occurs from a 64-bit process, an error code will be returned, a hardware alarm will be raised and the Cicode thread will stop.

---

VBA has the following features:

- VBA code is multithreaded and fully scheduled within the Plant SCADA Kernel.
- VBA uses the same well proven engine that Cicode uses and can be used wherever Cicode is used.
- VBA has a small footprint of under 400K.
- VBA code is directly callable from Plant SCADA Command and Expression fields.

- VBA code is callable from Cicode and vice versa.
- VBA code provides native support for ActiveX objects, Plant SCADA Variable Tags and Alarm Tags.
- VBA makes ActiveX object manipulating easier. It allows direct interaction with the object models from 3rd party applications such as Word, Excel, etc.

---

**Note:** You may notice slight differences between VBA and VBA in other applications; this is normal as each application has a different object model.

---

The Cicode Editor has been upgraded to fully support VBA. Supported features include:

- Integrated Cicode and VBA compiler
- Integrated Cicode and VBA source code editor
- Integrated Cicode and VBA debugger

This section contains information on the VBA programming language and describes the following:

- [Integrating VBA into a Project](#)
- [Understanding VBA Language Basics](#)
- [VBA Function Reference](#)
- [ASCII/ANSI Character Code Listings](#)

---

**Note:** For information regarding methods you can use to extend Plant SCADA that do not require VBA, see the topic [Extensibility](#).

---

## Integrating VBA into a Project

You can integrate VBA into your Plant SCADA project in two ways:

- [Using VBA in Command or Expression Fields](#)
- Store user-defined VBA script in a separate [VBA Files](#).

In either case, all procedures within a VBA script can access (read and write) any Plant SCADA variable tag in the same way Cicode can access Plant SCADA tags.

### Access Cicode Tags with VBA

VBA can use your Plant SCADA project variable tag and alarm tag variables in the same way as could Cicode (except for syntax differences). Both programming languages refer to a project's variable tags by using the name of the tags as defined in the project.

For instance, in the following example, two variable tags in your Plant SCADA project may be named **B1\_PUMP\_101\_SP** and **B1\_PUMP\_101\_PV** respectively, representing the Set Point and Process Variable values of Pump 101. These variable tag names can be used within a VBA statement (just as you would use any other variable in VBA). Both values can be read-from and written-to directly using VBA:

```
' set pump speed to 500 rpm
B1_PUMP_101_SP = 500
' calculate pump speed error
Dim varPumpSpeedError
```

```
varPumpSpeedError = B1_PUMP_101_PV - B1_PUMP_101_SP
```

You should note that VBA does not recognize Plant SCADA variable tags that are named with an initial digit (0-9).

To access such tags using VBA, you must precede the tag name with a case-insensitive string containing the letters 'V', 'B,' and the underscore character (VB\_) as in the following example:

```
Plant SCADA Tag Name: "123Pump"  
CitectVBA reference "VB_123Pump"
```

For details of using tags that have a number as their first digit in your project, consider using the [\[General\]TagStartDigit](#) Citect.ini parameter.

## Access Plant SCADA Array Values with VBA

To access Plant SCADA Array Values in VBA use the following format:

```
<TagName>.<Index>(<indexNumber>)
```

Where:

- <TagName> = The name of the array
- <Index> = The function that retrieves the values located at the specified <indexNumber>
- <indexNumber> = Position of the array to read/write the value.

### Example

In this example, an array is declared with name "sampleArray" in Plant SCADA Studio with 10 elements.

- To read the value at index 2 from sampleArray use the following syntax:

```
sampleArray.Index(2)
```

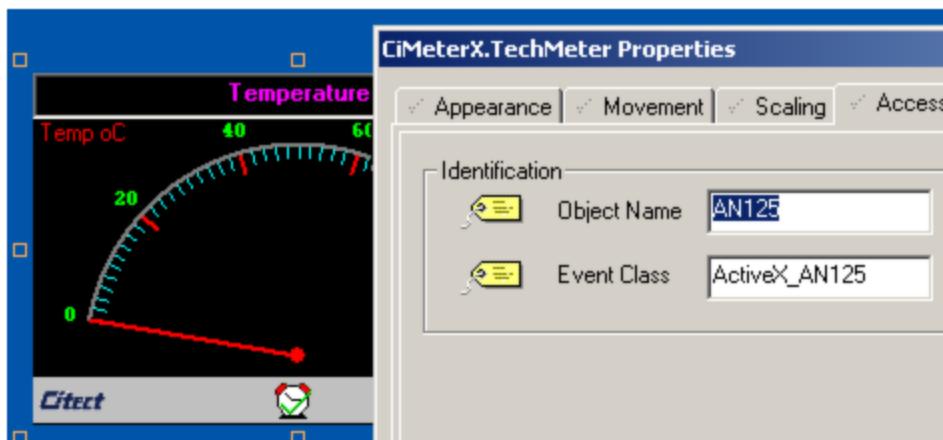
- To write to the value at index 3 in sampleArray use the following syntax:

```
sampleArray.Index(3) = 4
```

## Access ActiveX Objects with VBA

ActiveX objects which have been added to a graphics page in your Plant SCADA project can be referred to in VBA by constructing a unique reference name using the page name, the underscore character, the letters 'AN', and the animation number of the object.

This reference name is called the Event Class name in Plant SCADA. To view the reference name, double-click the ActiveX object, select the **Access** tab, then click the **Identification** tab.



### Example

In this example, the reference name for the Temperature meter object would be referred to in VBA as ActiveX\_AN125.

All object properties can be accessed and manipulated using VBA in the same way that object properties can be manipulated using Cicode.

## Using VBA in Command or Expression Fields

Plant SCADA expects that all code contained within a Plant SCADA Command or Expression field to be Cicode by default. When using VBA code script directly in a Plant SCADA Command or Expression field within Plant SCADA, you must precede the VBA script with the keyword **CiVBA**, as shown here:

```
CiVBA
TestTag_1 = TestTag_1 + 1
```

This is known as the language override command. When the Plant SCADA compiler reads the keyword CiVBA, it knows to handle that code (within the same Plant SCADA Command or Expression field) as VBA script, and compiles it as such. No such override command is required to use Cicode.

The CiVBA language override statement must be placed first in the Plant SCADA Command or Expression field if you want to use VBA script code instead of Cicode in that Plant SCADA Command or Expression field.

---

**Note:** You must use either Cicode or VBA in a Plant SCADA Command or Expression field. You cannot change or swap between the two programming languages (within the same Plant SCADA Command or Expression field) once you've started using one or the other.

---

You can, however, call a single Cicode function from within VBA script if you wrap the Cicode call within special VBA functions **CicodeCallOpen()** and **CicodeCallReturn()**.

## Calling Cicode from VBA

Calling a Cicode function from VBA is accomplished by two VBA functions: **CicodeCallOpen()** and **CicodeCallReturn()**.

To call a given Cicode function, use the **CicodeCallOpen** function which will create and execute a Cicode thread that runs the function. For multitasking purposes, a separate function **CicodeCallReturn** is used to obtain the return-value of the completed Cicode function most recently called by the **CicodeCallOpen** function.

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not nest the CicodeCallOpen and CicodeCallReturn functions. Nesting these functions can lead to unintended equipment operation when your program is run.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

The return value is initialized when control is returned to the Plant SCADA kernel. This occurs only after completion of the line of VBA code containing CicodeCallOpen. For details on multithreading in VBA, see [Multithread Considerations with VBA](#).

To call a given Cicode function or subroutine from VBA, use the CicodeCallOpen function. Upon return from CicodeCallOpen, you can call the CicodeCallReturn function to obtain the return value of the Cicode function called.

The CicodeCallOpen function is a VBA function used to call a Cicode function from VBA. It is used to initiate and execute a call to the Cicode function and returns an integer value representing the success or the type of error encountered by the CicodeCallOpen function.

```
<ReturnValue> = CicodeCallOpen(<FunctName>, <ArgList>)
```

Where:

- <ReturnValue> represents the return value of:
  - 0 if CicodeCallOpen function was successful
  - 1 for CicodeCallOpen function general error
  - 2 for specified Cicode function not found
  - 3 for incorrect number of arguments for specified Cicode function passed in <ArgList>.
- <FunctName> is a string representing the name of the Cicode function being called. The function name should be enclosed in double quotes.
- <ArgList> represents a variable length comma separated argument list of all the arguments to be passed to the Cicode function being opened (dependant upon which Cicode function is being called and the arguments that Cicode function requires). The argument list should not be enclosed within brackets, although when using variable names as arguments, those variable arguments within the list need to be individually enclosed within brackets to force the passing of the variable to Cicode by value.

The CicodeCallReturn function is a VBA function used to obtain the return value of the most recently completed Cicode function opened with the VBA CicodeCallOpen function.

```
<ReturnValue> = CicodeCallReturn()
```

Where:

- <ReturnValue> represents the return value of the Cicode function specified in the most recent call of the CicodeCallOpen function. Note that the return data type of CicodeCallReturn will depend upon the return data type of the Cicode function called in the most recent call of the CicodeCallOpen function.

No arguments are passed to the CicodeCallReturn function, as it can only return the result of the most recent return-value for the Cicode function called by the VBA CicodeCallOpen function.

**Note:** In the following example, a VBA variable is enclosed in brackets to force the passing of the variable by

---

value. See [Passing Variables Byref and Byval](#).

## VBA

```
' declare modular variant variable to store function results
Dim vntRet as Variant
Function TestCicode() As Integer
    ' declare local variables
    Dim intRet As Integer
    Dim strReply as String
    Dim intMaxScale as Integer
    ' copy current tag value to variable
    ' uses the project variable tag named MAX_SCALE
    intMaxScale = MAX_SCALE
    ' call Cicode function
    ' for example: TrnSetScale( AN, Pen, Percent, Scale)
    intRet = CicodeCallOpen( "TrnSetScale", 53, -1, 100, (IntMaxScale) )
    ' Note the syntax used:
    '- brackets around the VBA function argument list
    '(Only necessary when the VBA function is preceded by an equals (=) sign .)
    '- double quotes around the Cicode function name
    '- no brackets around the Cicode function argument list
    '- brackets around individual variable arguments
    ' test results
    If intRet = 0 Then
        '
        ' insert code for successful completion here
        '

        vntRet = CicodeCallReturn()
        strReply = "CicodeCallOpen Function successfully called"
    Else
        '
        ' insert code for unsuccessful completion here
        '

        Select Case intRet
        Case = 1
            ' assign return comment for this case
            strReply = "CicodeCallOpen Function call general error"

        Case = 2
            ' assign return comment for this case
            strReply = "Cicode Function not found"
        Case = 3
            ' assign return comment for this case
            strReply = "Wrong number of arguments " _
            & "in Cicode CallOpen function call"
        Case Else
            ' assign return comment for this case
            strReply = "Unknown error"
        End Select
    End If
    ' display return comment for your information
    MsgBox strReply
    ' assign return value for this function
    TestCicode = intRet
End Function
```

Alternatively, to call a single VBA function (from within the Plant SCADA Command or Expression field) after you

have already used Cicode in that field, you can wrap the VBA within three nested special Cicode functions: VbCallOpen(), VbCallRun() and VbCallReturn().

## Calling VBA from Cicode

Three new Cicode functions allow VBA code to be called from within Cicode script, and be pre-emptively multitasked by Plant SCADA. These calls VbCallOpen(), VbCallRun(), and VbCallReturn() can be nested to implement the entire function set with a single line of Cicode.

**Note:** When using the CiVBA language override in a Command field, the compiler constructs the nested call for you. The same mechanism is used even though it is not self evident. For details, see .

For information on multithreading in VBA, see [Multithread Considerations with VBA](#).

To call a given VBA function or subroutine from Cicode, use the VbCallOpen function. This returns a handle which can then be used to execute the call by passing it to the VbCallRun function. Upon return from VbCallRun, you can call the VbCallReturn function to get the return value of the VBA function called.

The Cicode VbCallOpen() function is used to initiate a call to the VBA function or subroutine, and returns a handle to the open function.

```
<ReturnValue> = VbCallOpen(<FunctName>, <ArgList>)
```

Where:

- <ReturnValue> represents the handle to the opened function.
- <FunctName> represents the name of the VBA function or subroutine being called.
- <ArgList> represents a comma separated list of arguments to pass to the opened VBA function or subroutine named in <FunctName>.

The Cicode VbCallRun() function is used to execute the VBA function or subroutine (previously opened with the Cicode VbCallOpen function), and requires the handle returned from the VbCallOpen function call. The VbCallRun function provides an opportunity for the opened VBA function to complete and return a value in the multi-threaded Plant SCADA environment. It passes its argument value (of OBJECT data type) through as its return value upon completion.

```
<ReturnValue> = VbCallRun(<CallHandle>)
```

Where:

- <ReturnValue> represents the handle to the opened VBA function passed through for the <CallHandle> argument.
- <CallHandle> represents the handle to the previously opened VBA function as returned by the Cicode VbCallOpenfunction.

The Cicode VbCallReturn() function is used to obtain the return value of the completed VBA function (previously opened with the Cicode VbCallOpen function), and requires the handle returned from the VbCallRun function call.

```
<ReturnValue> = VbCallReturn(<CallHandle>)
```

Where:

- <ReturnValue> represents the value returned by the completed VBA function (which was previously opened by the Cicode VbCallOpen function). The data type of the return value is dependent upon the data type of

the return value for the VBA function opened.

- <CallHandle> represents the handle to the previously opened VBA function as returned by the Cicode VbCallRun function.

## Example

```
FUNCTION
TestCitectVBA()
INT iRet;
STRING sMsg = "Hello";
INT iVal = 123;
iRet = VbCallReturn(VbCallRun(VbCallOpen("CiVBATest", iVal)));
Message("TestCitectVBA Function", "CiVBATest = " + IntToStr(iRet), 0);
END
```

## Example

```
Function CiVBATest(Value As Integer) As Integer
CiVBATest = Value * 2
End Function
```

## See Also

[Integrating VBA into a Project](#)

[Multithread Considerations with VBA](#)

## Multithread Considerations with VBA

Cicode is pre-empted and executed on an instruction-by-instruction basis. This means that execution of a simple unnested Cicode thread can only switch to another thread after the current Cicode instruction has completed execution.

VBA code is pre-empted and executed on a line-by-line basis (as opposed to an instruction-by-instruction basis), and pre-empting can only occur after the current line has completed execution.

Each line of VBA script is handled as a separate thread in Plant SCADA. Therefore multiple procedures placed on one line may not complete before another subsequent thread is processed in a multithreading environment. This could cause unpredictable results and consequences, including data invalidation and corruption.



### UNINTENDED EQUIPMENT OPERATION

- Create your VBA program so that every code statement is positioned on a unique line.
- Do not group more than one code statement on a single line in your program. Grouping VBA statements on a single line can cause data corruption during multithreaded execution.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

If, for example, you were reading or setting some variable or point in a multi-statement thread, and further processing that data in a later thread, that data might become invalid or incorrect. For this reason, you should separate every statement onto separate lines in VBA.

For example, it is better to write:

```
A = Motor1.speed() + Motor4.speed() + Motor5.speed()
```

as

```
A = Motor1.speed()  
A = A + Motor4.speed()  
A = A + Motor5.speed()
```

in situations where the method `speed()` may take a long time to execute.

In the first example above, the VBA thread executes for three times longer before it can be pre-empted than in the latter example.

---

**Note:** This does not apply to Cicode because the Cicode engine can pre-empt aggregated code.

---

## See Also

[Using VBA in Command or Expression Fields](#)

## Understanding VBA Language Basics

This section describes the basics of the VBA programming language.

- [VBA Files](#)
- [Cicode Editor](#)
- [Scope of VBA](#)
- [VBA Statements](#)
- [Comments](#)
- [Labels](#)
- [VBA Line Continuation Character](#)
- [Option Statements](#)
- [VBA Data Types](#)
- [Constants](#)
- [Variables](#)
- [Numbers](#)
- [Date and Time Handling](#)
- [Operators](#)
- [Strings](#)
- [Control Structures](#)
- [Subroutines and Functions](#)
- [DLLs and APIs](#)
- [OLE Services](#)

- [File Input/Output with VBA](#)

## VBA Files

VBA code scripts can be saved to file, can include comments, statements, various representations of numbers, can handle many different data types, and can have multiple and nested control structures. However, VBA is primarily provided with Plant SCADA to interact with ActiveX objects.

VBA files are ASCII text files stored in ANSI format with a BAS extension (*filename.BAS*), and are known as file modules.

VBA file modules can be viewed and edited in any text editor program. They can be used in Plant SCADA, but must be saved as 'text with linebreaks' with a '.BAS' file extension or Plant SCADA will not be able to open the file.

## Cicode Editor

The Cicode Editor is VBA aware and designed to help you create, edit, test, and debug VBA file modules in your Plant SCADA project.

The Cicode Editor has features suitable for use with VBA file modules including:

- Ability to create, open, edit, and save VBA file modules
- Customizable coloration of VBA code syntax structure
- Recognition of predefined keywords with tooltip prompting and auto-completion functionality
- Fully integrated debugging of VBA file modules
- Separate VB Watch window for viewing runtime VBA variable values

A sample VBA file module named Sample.Bas is included in the User\Example subfolder on the drive on which you installed Plant SCADA. This module explains most of the VBA functionality.

VBA file modules will never be compiled into standalone Windows executable files; instead, they're included with the compiled Plant SCADA. As a result, they don't require a Main procedure to be declared. Therefore, VBA file modules are structured to contain only their header information, modular constant and variable declarations, then procedures (subroutines, and functions).

VBA file modules are automatically included with a Plant SCADA project if they are stored in the same file folder as your project. When saving a VBA file module to disk, save it to your project folder.

## Scope of VBA

The scope of an object determines which portions of your code scripts can use that object.

---

**Note:** The use of **Global**, **Public**, and **Private** keywords has no effect on scope in VBA.

---

### Procedural (Local) Level Scope

Variables and constants declared (using the Dim, Static, or Const statements) within a VBA procedure (subroutine or function) have local scope to only that within the procedure. This means that procedural level variables and constants cannot be referenced (accessed and used) from anywhere outside of that procedure.

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use the Global, Public, or Private keywords in your VBA procedures. Using these keywords in procedures can lead to unintended equipment operation when your program is run.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Procedural level variables declared using the Dim statement do not retain their assigned values when dereferenced. Procedural level variables declared using the Static statement, however, do retain their assigned values between references, even after that procedure moves out of scope.

## Modular level scope

Constants declared (using the Const statement) and variables declared (using the Static statement) at the modular level (outside any procedure) in a VBA file have modular scope to all procedures within that same VBA module (file). This means that modular constants and static variables can only be referenced from a procedure located within the same file module, and cannot be referenced from outside of that file module. This has no effect in VBA, even if declared using the Global keyword.

Modular level constants and static variables retain their assigned values for the entire runtime of the project.

## Global Level Scope

Variables declared (using the Dim, Global, or Public statements) at the modular level (outside any procedure) in a VBA module (file), have global scope within the Plant SCADA project. This means that modular VBA variables (except statics) can be referenced from both inside and outside of their file module.

Global level variables can be used directly within Plant SCADA command or expression fields.

Procedures (subroutines or functions) declared within a VBA file module, like global variables, have global scope within a Plant SCADA project. They can be referenced or called from any VBA module, as well as from any Plant SCADA command or expression field.

Equally important, all Plant SCADA variable tags, alarm tags, and ActiveX objects are accessible to all VBA file modules (and their procedures) within that project, in the same manner as they have always been accessible to project Cicode files. For information about referencing Plant SCADA project tags using VBA, see [Integrating VBA into a Project](#).

Global level variables will also retain their assigned values between subsequent references, behaving somewhat similarly to the values stored in Plant SCADA tags. In this regard, Global and Public statements are redundant at the modular (global) level in VBA, as they perform the exact same duty as the Dim statement.

**Note:** If an array is declared with an explicit data type using the Public or Dim keyword at global level, it will result in the error message "Sub or Function not defined". See the example below for more information.

## Example

```
Dim myVarAr(5) As Integer //Results in Error
Dim myVarAr(5) //No Error

Dim daysOfWeek(5) As String //Results in Error
```

```
Dim Intvar(5) As Long //Results in Error
Public myArray(5) //No Error
Public myArray2(5) As Integer //Results in Error

Sub TArrayIn()
Dim myVarAr(5) As Integer //No Error
Dim Intvar(5) As Long //No Error
Dim daysOfWeek(5) As String //No Error

Intvar(1)=1
Print Intvar(1)

daysOfWeek(1)= "Naresh"
Print daysOfWeek(1)

myVarAr(2) = 1
Print myVarAr(2)
myArray(1)=3
Print myArray(1)

myArray2(1)=4
Print myArray2(1)

End Sub
```

## See Also

[Multithread Considerations with VBA](#)

[VBA Files](#)

## VBA Statements

A statement in VBA is an unbroken sequence of syntactically correct code script containing at least one VBA keyword instruction. A single statement in VBA is one complete segment of code script that instructs Plant SCADA to do something.

In VBA there is no statement terminator. As in other BASIC programming languages, the end of the line containing the statement is treated as the statement terminator by default.

Most often, a statement consists of a single line of VBA script. However, more than one statement can be placed on one line of VBA script, provided each statement is separated by a colon character (:); for example:

```
Pump234.AddPoint( 25, 100): Pump234.AddPoint( 0, 75)
```

This is equivalent in VBA to:

```
Pump234.AddPoint( 25, 100)
Pump234.AddPoint( 0, 75)
```

Using complex multi-statement lines of VBA script is not recommended in Plant SCADA. Multithreading should be considered when using more than one statement per line in VBA. For details, see [Multithread Considerations with VBA](#).

## Comments

Comments are non-executed sections of code that are ignored by the VBA compiler. Comments allow programmers to describe the purpose of a section of code to facilitate code maintenance.

As in other BASIC programming languages, both the apostrophe character ( ' ), and the keyword REM are recognized as the start of a comment in VBA. All characters following an apostrophe or the keyword REM are ignored by the VBA compiler until it reaches the end of the line. Line continuation characters do not work inside comments.

REM, like all other keywords and most names in VBA, is not case sensitive.

```
' This whole line is a comment
rem This whole line is a comment
Rem This whole line is a comment
REM This whole line is a comment
```

Both types of comments can be used on their own separate line, or the apostrophe character can be used to start a comment at the end of a statement on the same line as a statement.

```
Pump234.AddPoint( 25, 100 ' Add point to pump 234
```

Everything placed on the same line after an apostrophe is treated by VBA as a comment. If you want to place a comment on the same line as a statement, the comment must be placed last after all statements on that line. Comments cannot be placed between multiple statements on the same line.

Not every line of code requires a comment. In fact, VBA should contain understandable naming structures and be laid out in such a manner as to make comments unnecessary. However, where a complex function, equation, or logic structure is not readily understandable by viewing the code, it is good practice to include a pertinent comment to make the code more understandable when viewed in isolation.

## See Also

[VBA Files](#)

## Header Information

You should include header information with every file you create or edit. Data such as the file name, author name, creation date, update date, editing history, and the like should be included to form the header information. Each function or subroutine should include a brief comment describing the purpose or function of the procedure.

### VBA file header example

```
' FILE IDENTIFICATION
' VBA example named VBA.bas
' Created by Plant SCADA Documentation Team
' Created in April 2001
```

## See Also

[VBA Files](#)

## Labels

Labels can be used to divide a large VBA function or subroutine into logical sub-sections of code script. Labels are often used in association with the GoTo statement. All of the VBA script following the label and extending through to another label, or to the end of the function or subroutine containing the label, is regarded as belonging to that label. Or more appropriately, the label is said to identify, or be attached to, that particular section of VBA script.

Labels must begin with a letter, be no longer than 40 characters, and cannot be a reserved word. Labels must terminate with the colon character (:). Label names can only contain the letters 'A' to 'Z' and 'a' to 'z', the underscore '\_' character, and the digits '0' to '9'. Label names cannot contain the space character.

Label names (once declared), become a keyword in VBA. Like most keywords in VBA, label names are not case sensitive. For example, all of the following label examples are treated identically in VBA:

```
label1:  
Label1:  
LABEL1:
```

**Note:** Labels as used in VBA are not the same as labels used in Plant SCADA.

---

## See Also

[VBA Files](#)

## VBA Line Continuation Character

The underscore is the line continuation character in VBA. There must be a space before and after the line continuation character. Line continuation characters do not work inside comments.

The following sample code statements are treated identically in VBA:

```
Pump234.AddPoint _( 25, 100)  
Pump234.AddPoint( 25, 100)
```

Strings cannot be separated between lines using the line-break character in VBA, unless the strings are properly enclosed within double quotes on each line, and appended together as per the following example:

```
Dim strSample as String  
strSample = "This sentence on the first line in my code. " _  
& "This sentence is on the second line in my code. " _  
& "Yet all would display on the same line " _  
& "if the display were wide enough."
```

## See Also

[VBA Files](#)

## Naming

Function, subroutine, variable, constant, and label naming in VBA must begin with a letter, be no longer than 40 characters, and cannot be a reserved word. Name field are not case-sensitive and can only contain the letters 'A' to 'Z' and 'a' to 'z', the underscore '\_' character, and the digits '0' to '9'. Names cannot contain the space

character. You cannot use the name of a VBA predefined function as a name. For a list of predefined functions, see .

Function, subroutine, variable, constant, and label object names (once declared), become a keyword in VBA. Like most keywords in VBA, these names are not case sensitive. For example, all of the following examples are treated identically in VBA:

```
pump234.addpoint(25, 100)  
Pump234.AddPoint(25, 100)  
PUMP234.ADDPOINT(25, 100)
```

When naming in VBA, make the name an appropriately descriptive term that is easily recognizable. For example:

```
X.addpoint(25, 100)
```

This does not make as much sense as:

```
Pump234.AddPoint(25, 100)
```

Combining upper- and lowercase letters between words in the name is an acceptable common programming practice, and aids in readability.

Identically named objects cannot be declared more than once per Plant SCADA project, even though they may exist in different VBA code file modules. However, if an object declared locally within a procedure has the same name as an object declared in a module, VBA will reference the local procedure scope object instead of the modular scope object.

## See Also

[Scope of VBA](#)

[VBA Files](#)

[Integrating VBA into a Project](#)

## Option Statements

VBA supports the use of file scope Option statements which determine the default behaviour of some VBA functions. For instance, the Option Explicit statement causes the VBA compiler to produce compile errors whenever it encounters the use of previously undeclared variables. The Option Compare statement sets the default comparison method for string comparisons. The Option Base statement sets the default base number for VBA variable arrays to either zero or one.

clare all option statements in VBA at the beginning of your VBA code files.

### Option Explicit Statement

As in other BASIC programming languages, VBA supports the declaration of variables both implicitly and explicitly. An unfortunate consequence of implicit variable declaration is the possible misspelling of the variable name in subsequent code writing, with unreliable program behaviour and unpredictable consequences.

To minimize implicit declaration, and to foster good, consistent programming standards, use the option explicit statement at the beginning of all your VBA files:

```
Option Explicit
```

This causes the VBA compiler to produce a compile error whenever it encounters an undeclared variable. This can be useful in locating and identifying variable name typing errors in your VBA code at compile time, thus

trapping and minimizing the likelihood of runtime errors caused by such mistakes.

## Option Compare Statement

The Option Compare statement determines how strings are compared within a VBA file, and like other Option statements in VBA, should be declared at the beginning of your VBA code files.

When strings are compared using VBA functions such as StrComp() or InStr(), VBA determines whether they contain equivalent characters and how they differ if they do not match.

---

**Note:** When comparing strings, VBA compares the ANSI values of each character in the strings. For example, the character capital 'A' has the ANSI value of 65, and the character lowercase 'a' has the ANSI value of 97. For a listing of ANSI character values, see [ASCII/ANSI Character Code Listings](#).

---

You can use the Option Compare statement to specify the default case-sensitivity behavior for VBA functions when making string comparisons.

The Option Compare statement in VBA has two settings:

- **Option Compare Binary:** String comparisons are case-sensitive, and this is the default string-comparison setting.
- **Option Compare Text:** String comparisons are case-insensitive.

## Option Base Statement

The Option Base statement determines the default base number for the indexing of variable arrays created within a VBA file, and like other Option statements in VBA, should be declared at the beginning of your VBA code files.

There are two settings for the Option Base statement in VBA:

- **Option Base 0:** Variable arrays are indexed from number zero, and this is the default setting.
- **Option Base 1:** Variable arrays are indexed from number one.

For an example of using the Option Base statement, see [Fixed Size Arrays](#).

## See Also

[VBA Function Reference](#)

## VBA Data Types

VBA uses ten predefined data types:

Variable	Char	Type Declaration	Size	Value Range
Byte		Dim bytVar As Byte	1 byte (8 bits)	0 to 255
Boolean		Dim binVar As Boolean	2 bytes	True or False

Variable	Char	Type Declaration	Size	Value Range
String	\$	Dim strVar As String	10 bytes + 1 byte per character	0 to 65,535 characters
Integer	%	Dim intVar As Integer	2 bytes	-32,768 to 32,767
Long Integer	&	Dim lngVar As Long	4 bytes	-2,147,483,648 to 2,147,483,647
Single precision	!	Dim sglVar As Single	4 bytes	3.4E-38 to 3.4E+38
Double Precision	#	Dim dblVar As Double	8 bytes	1.79D-308 to 1.79D+308
Variant		Dim vntVar As Any	16 bytes	Same ranges as data types stored
Object		Dim objVar As Object	4 bytes	Any OLE Object reference
Date/Time		Dim dtmVar As Date	8 bytes	Jan 1, 100 to Dec 31, 9999

**Note:** VBA does not support user-defined data types.

## See Also

[Numeric Data Types](#)

[Numbers](#)

[Variables](#)

[VBA Function Reference](#)

## Constants

Your VBA code may contain frequently recurring constant values like Pi, or may contain numbers that are difficult to remember or have no obvious meaning. You can make your VBA code much easier to read and maintain using constants to represent those values.

Unlike variables, constants can't be changed once your Plant SCADA project is compiled and running. Constants are either symbolic or intrinsic:

- Symbolic or user-defined constants are declared by using the `const` statement.
- Intrinsic constants are provided in object libraries of ActiveX objects and you cannot use them in VBA: they cause compile errors as there is no way to provide early-binding to the object type library.

You can create a constant in VBA named `Pi`, assign it the numeric value once in your code, then refer to it by using the constant name, as shown here:

```
'modular level constant declaration
Const Pi = 3.1415926
Function CircleArea(Byval Radius)
    ' calculate and return area of circle
    ' using radius passed in as argument
    CircleArea = Pi * (Radius * Radius)
End Function
Function CircleCircumference(Byval Radius)
    ' calculate and return circumference of circle
    ' using radius passed in as argument
    CircleCircumference = Pi * Radius * 2
End Function
```

These VBA functions would be called from a Plant SCADA command or expression field like this:

```
CivBA
TestTag_1 = CircleArea(TestTag_1)
```

or

```
CivBA
TestTag_1 = CircleCircumference(TestTag_1)
```

## See Also

[Declaration of Constants](#)  
[Passing Variables Byref and Byval](#)  
[Integrating VBA into a Project](#)  
[Intrinsic Constants](#)  
[Scope of VBA](#)  
[VBA Function Reference](#)

## Declaration of Constants

VBA constants can only be declared and referenced within VBA file modules. VBA modular constants have modular scope and cannot be referenced (accessed and used) from outside their VBA module (file).

**Note:** VBA constants cannot be used directly in Plant SCADA command or expression fields.

Once declared within a VBA module, VBA constants can be referenced and used in any procedure within the same code module. A constant declared outside a procedure has modular scope to all procedures within that same VBA module (file). See [Scope of VBA](#). Constants declared in a Sub or Function procedure have local scope only within that procedure.

VBA constants are declared with the Const statement in the following format.

```
Const <ConstantName> [ As <DataType> ] = <expression>
```

Where:

- Const is the required constant declaration statement BASIC keyword
- <ConstantName> represents the required name of the constant being declared
- <DataType> represents the optional VBA data type of the constant being declared
- <expression> represents the required value being assigned to the constant

---

**Note:** Do not include the brackets from the explanation in the actual code statement.

---

If no data type is declared, VBA automatically assigns one of the following data types to the constant:

- Long (if it is a long or integer).
- Double (if a decimal place is present).
- String (if it contains quote marks).

Constant statements can only be declared and assigned using simple expressions. Constants cannot be assigned values from variables, user-defined functions, intrinsic VBA functions (such as Chr), or from any expression that involves an operator. A constant needs to be defined before it can be used.

## Example

```
' Correct declaration examples
Const Seven = 7

' long assignment
Const Pi = 3.14159

' double assignment
Const Lab = "Laboratory"

' string assignment
' Incorrect declaration examples. Note that the following declarations demonstrate
incorrect assignments because each contains an operator
Const conPi = 4 * Atn(1)
' will cause a VBA compile error
Const conDegToRad = (conPi / 180)
' will cause a VBA compile error
```

For an example of using constants in VBA, see [Constants](#).

---

**Note:** The use of Global, Public, and Private keywords has no effect on scope in VBA.

---

## See Also

[Intrinsic Constants](#)

[Variables](#)

[VBA Data Types](#)

[VBA Function Reference](#)

## Intrinsic Constants

VBA has no predefined intrinsic (built-in and declared) constants, however, does provide limited support for intrinsic constants provided in object libraries of ActiveX objects when the object they refer to is loaded using the predefined VBA CreateObject() function.

## See Also

[Declaration of Constants](#)

[Constants](#)

## Variables

Variables are used in VBA to temporarily store data values. Variables let you assign a descriptive name to the data you are working with. You can create a variable once only in your code, and reference (refer to) it thereafter as many times as you like, by using its name in your code in place of the data value. Unlike constants, the value that a variable holds can be changed during the runtime of the project.

All variables declared within a VBA procedure (subroutine or function) have local scope to that procedure only. Procedural level variables declared using the `Dim` statement do not retain their assigned values when dereferenced. Procedural level variables declared using the `Static` statement, however, retain their assigned values between references, even after that procedure moves out of scope.

VBA code used within a Plant SCADA command or expression field is treated as if the command or expression is a separate VBA procedure. Variables declared within such a command procedure have procedural scope and lifetime, as described above.

Variables declared using the `static` statement at the modular level (outside any procedure) in a VBA file, have modular scope to all procedures within that same VBA module (file). Modular level `static` variables retain their assigned values for the entire runtime of the project.

Variables declared (using the `dim`,`global`,or `public` statements) at the modular level (outside any procedure) in a VBA file do, however, have global scope within the Plant SCADA project.

---

**Note:** Global and public statements are redundant at the modular (global) level in VBA, as they perform the exact same duty as the `dim` statement.

## See Also

[Variant Declaration](#)

[Variable Initialization Values](#)

## Variable Declaration

In VBA, variables are declared (dimensioned) with the `dim` statement in the following format.

`Dim <VariableName> [ As <DataType> ]`

Where:

- `Dim` is the required Variable declaration statement BASIC keyword
- `<VariableName>` represents the required name of the variable being declared (dimensioned)
- `<DataType>` represents the optional VBA data type of the variable being declared

In the variable declaration statement:

- Every placeholder shown inside arrow brackets (`<placeholder>`) should be replaced in any actual code with

the value of the item that it describes. The arrow brackets and the word they contain should not be included in the statement, and are shown here only for your information.

- Statements shown between square brackets ( [ ] ) are optional. The square brackets should not be included in the statement, and are shown here only for your information.

If no data type is declared, the data type is Variant by default. To declare a variable other than a Variant, the variable declaration needs to be immediately followed by As <datatype> (where <datatype> represents one of the 10 data types), or appended by a type declaration character such as a \$, %, &, !, or # for string, integer, long, single, or double data types respectively. For example:

```
Dim intVar As Integer
Dim dblVar As Double
Dim vntVar ' as variant by default
Dim strName$, Age% ' multiple declarations on one line
```

Be aware that multiple declarations in the same statement require individual data type assignment if you want them be other than the variant type. In the following example, only the first variable is not a variant. For example:

```
Dim strName As String, vntAge, vntAddress
```

The same statement with data type assignment for every variable would look like the following example:

```
Dim strName As String, intAge As Integer, strAddress As String
```

## See Also

[VBA Data Types](#)

[Variable Initialization Values](#)

[Constants](#)

[Variant Declaration](#)

[Arrays of Variables](#)

[VBA Function Reference](#)

## Variable Initialization Values

VBA variables are initialized when first declared. Numeric variables are initialized to 0 (zero). Variable-length strings are initialized to zero-length strings (""). Fixed length strings are filled with zeros. Variant variables are initialized to Empty.

To be sure of the contents of a variable, a valid value needs to be assigned to it before it is used as an operand in a VBA statement. For details, see [Assignment Operator](#).

---

**Note:** Only implicitly declared variables can be assigned an initial value in the declaration. However, as explicit declaration is preferred practice in VBA, explicit variables need to be declared before they can be assigned a value.

---

Every call to a procedure will reinitialize the value of all objects (except static variables) declared within that procedure.

---

**Note:** In VBA, use a static variable, a modular variable, or a Plant SCADA tag to store variable values between procedures. For details, see [Scope of VBA](#).

---

Objects (including variables) declared in VBA are only initialized when referenced by a running piece of code, and

are removed from memory when all references are closed.

In the Plant SCADA multithreaded environment, VBA remains active in memory only so long as a procedure is being processed. At the completion of a VBA procedure, all objects no longer referenced by that procedure are removed from memory. For details, see [Multithread Considerations with VBA](#).

## See Also

[VBA Data Types](#)

[Variable Initialization Values](#)

[Constants](#)

[Variant Declaration](#)

[Arrays of Variables](#)

## Arrays of Variables

Arrays of variables allow you to group like variables together, somewhat similar to the grouping of like items in fields of a database. An array is an ordered group of variables of the same name, containing values of the same data type. Individual member elements of the array are identified by a separate index number. Arrays in VBA start their indexing sequence by default at zero. This default base value can be changed in a VBA file module by using the option base statement.

VBA supports single and multi-dimension arrays of variables. VBA creates single dimension arrays by default. Multi-dimension arrays must be specifically declared.

VBA allocates memory space for each element of the array. To minimize the amount of memory used storing arrays, and to minimize the time required to access array data, arrays should not be declared any larger than required.

All elements in an array must be of the same data type. VBA supports arrays of bytes, booleans, longs, integers, singles, doubles, strings, and variants. For details about VBA data types, see [VBA Data Types](#).

Arrays declared in a sub or function procedure have local scope only within that procedure. An array declared outside a procedure has modular (global) scope to all procedures within the project.

---

**Note:** VBA arrays cannot be used directly in Plant SCADA command or expression fields. Also, VBA does not support user-defined data types.

---

Arrays declared (using the dim statement within procedures) do not retain their values between procedure calls in VBA.

## See Also

[Fixed Size Arrays](#)

[Multi-Dimensional Arrays](#)

[Dynamic Size Arrays](#)

## Variable Array Declaration

Arrays of variables are declared within a VBA file module, function, or subroutine, using the dim statement with parentheses positioned after the array name, in the following syntax:

Dim <ArrayName>( [<Subscripts>] ) [As <DataType>]

Where:

- dim is the required variable declaration statement BASIC keyword.
- <ArrayName> represents the required name of the array being declared (dimensioned).
- ( ) are the required parentheses to hold the array subscript range (dimensions).
- <Subscripts> represents the optional subscript ranges and dimensions for the array.
- As is the optional As statement keyword declaring the array data type.
- <DataType> represents the optional VBA data type declaration for the array.

In the variable array declaration statement:

- Every placeholder shown inside arrow brackets (<placeholder>) should be replaced in any actual code with the value of the item that it describes. The arrow brackets and the word they contain should not be included in the statement, and are shown here only for your information.
- Statements shown between square brackets ( [ ] ) are optional. The square brackets should not be included in the statement, and are shown here only for your information.

## See Also

[Fixed Size Arrays](#)

[Multi-Dimensional Arrays](#)

[Dynamic Size Arrays](#)

[Array Subscripts](#)

[Arrays of Variables](#)

[Dim](#)

## Array Subscripts

Arrays can be declared with default or defined boundaries known as bounds. Unless specifically defined in the array declaration statement, default lower bound settings are used. The default lower bound is zero, unless set by the module option base statement setting.

VBA does not have an arbitrary upper bound on array dimensions. The upper bound of the array dimension must be defined before the array can be used. All bound values must be whole integers.

Subscripts are contained within one set of parentheses positioned immediately after the array name in the array declaration statement.

Subscripts are used to specify the bounds of each dimension of an array when the array is declared. If a single value is used, for instance (5), this represents the upper bound for that dimension of the array. If a range is specified, for instance (1 to 9), this specifies both the lower and upper bounds for that dimension of the array. If more than one subscript is used, for instance ( 5, 1 To 9), each subscript must be separated by a comma, and each subscript represents a separate dimension of the array.

The syntax of an array subscript range consists of a numeric value range separated by the to clause:

(<LowerBound> To <UpperBound>)

Where:

- ( ) are the required parentheses to hold an array subscript range (dimensions).
- <LowerBound> represents the lower bound of the subscript range for the array dimension.
- To is the clause linking the lower and upper bounds of the subscript range.
- <UpperBound> represents the upper bound of the subscript range for the array dimension.

## See Also

[Fixed Size Arrays](#)

[Multi-Dimensional Arrays](#)

[Dynamic Size Arrays](#)

[Arrays of Variables](#)

[Dim](#)

### Fixed Size Arrays

To declare a fixed size array, the array name must be followed by the upper bound subscript enclosed within parentheses. The upper bound must be an integer.

```
Dim ArrayName(10) As Integer  
Dim Sum(20) As Double
```

Unless specifically defined in the array declaration statement, default lower bound settings are used. The default lower bound is zero, unless set by the module option base statement setting. For details, see [Array Subscripts](#).

The first declaration in the previous example creates an array with 11 elements, with index numbers running from 0 to 10. The second creates an array with 21 elements (if base 0). One way to specify the lower bound is to provide it explicitly (as an integer in the range -32,768 to 32,767) using the To clause within the subscript:

```
Dim intCounters (1 To13) As Integer  
Dim strSums (100 To126) As String
```

In the preceding example, the index numbers of intCounters run from 1-13, and the index numbers of strSums run from 100-126.

**Note:** An array in VBA must be declared before it can be referenced.

Loops often provide an efficient way to manipulate arrays. For example, the following for loop initializes all elements in the array to 5:

```
Dim int As IntegerDim Counters(1 To 20) As Integer  
For int = 1 To 20  
Counters(int) = 5  
Next int
```

Arrays declared (using the dim statement within procedures) do not retain their values between procedure calls in VBA.

## See Also

[Multi-Dimensional Arrays](#)

[Dynamic Size Arrays](#)

[Arrays of Variables](#)

[Array Subscripts](#)

## Option Statements

### Multi-Dimensional Arrays

VBA supports multi-dimensional arrays, declared using multiple subscripts. Each subscript must be separated by a comma, and each subscript represents a separate dimension of the array.

The following example declares a two-dimensional array.

```
Dim dblMat(20, 20) As Double
```

Unless specifically defined in the array declaration statement, default lower bound settings are used. The default lower bound is zero, unless set by the module option base statement setting. For more information on bounds, see "Array Subscripts in VBA."

Reusing the previous example, either or both dimensions can be declared with explicit lower bounds.

```
Dim dblMat(1 To 10, 1 To 10) As Double
```

Arrays can be more than two dimensional. This declaration creates an array that has three dimensions with sizes 6 elements by 4 elements by 3 elements, using base 0:

```
Dim ArrTest(5, 3, 2)
```

You can efficiently process a multi-dimensional array with the use of for loops. In the following statements the elements in a multi-dimensional array are set to a value.

```
Dim L As Integer, J As Integer
Dim TestArray(1 To 10, 1 to 10) As Double
For L = 1 to 10
For J = 1 to 10
TestArray(L,J) = I * 10 + J
Next J
Next L
```

Arrays declared (using the dim statement within procedures,) do not retain their values between procedure calls in VBA.

## See Also

[Dynamic Size Arrays](#)

[Arrays of Variables](#)

[Array Subscripts](#)

[Option Statements](#)

[Fixed Size Arrays](#)

### Dynamic Size Arrays

To declare a dynamic sized array, the array must first be declared using the dim statement with an empty pair of parentheses following the array name. For example:

```
Dim ArrayName( ) As Integer
```

Once declared as dynamic in this manner, the array can then ONLY be resized within a function or subroutine using the redim statement.

```
ReDim ArrayName(20) As Integer
```

---

**Note:** You cannot resize an array whose size was predefined in its initial declaration.

In the above examples, the first declaration creates an array with 0 elements. The second recreates the array to contain 21 elements, with index numbers running from 0 to 20, unless the option base statement has been set previously in the code module (file), in which case the array will contain 20 elements with index numbering ranging from 1 to 21.

Unless specifically defined in the array declaration statement, default lower bound settings are used. The default lower bound is zero, unless set by the module option base statement setting. For more information on bounds, see "Array Subscripts in VBA."

Redim erases all values the array may have held. To preserve the contents of the array when resizing, precede the Redim statement with the preserve keyword.

```
Preserve ReDim ArrayName(20) As Integer
```

Redimensioning an array to a smaller value, will erase any values it may have contained in the removed portions.

Arrays declared (using the dim statement within procedures,) do not retain their values between procedure calls in VBA.

## See Also

[Multi-Dimensional Arrays](#)

[Arrays of Variables](#)

[Array Subscripts](#)

[Option Statements](#)

[Fixed Size Arrays](#)

## Variant Declaration

As is the case with Visual Basic, when a variable is introduced in VBA, it is not necessary to declare it first (see for an exception to this rule). When a variable is used but not declared, it is implicitly created as a variant data type. Variants can also be declared explicitly using As Variant. Both of the following example declarations are treated identically in VBA:

```
Dim vntVar ' implicit variant declaration
Dim vntVar As Variant ' explicit variant declaration
```

The IsEmpty( ) function can be used to find out if a variant variable has been previously assigned a value.

## Variant Data Types and Coercion

The variant data type is capable of storing numbers, strings, dates, and times. When using a variant, you do not have to explicitly convert a variable from one data type to another. This data type conversion is handled automatically, and is termed data type coercion.

```
' declares variant variable
Dim vntVar
' assign numeric 10 to variant
vntVar = 10
' add numeric 8 to numeric variant value
vntVar = vntVar + 8
' convert variant to string value and concatenates strings
vntVar = "F" & vntVar
```

```
' print string "F18"  
print vntVar
```

Numeric characters inside quotes ("567") will be stored and treated as a string in a variant data type variable. If this string (containing numeric characters) is subsequently used in a numeric operation, it will be coerced into a numeric data type and treated as a number in the operation. Conversely, numeric characters stored as a number data type in a variant variable, and subsequently used in an operation with a string, will be coerced into a string data type, and treated as a string value in the operation.

**Note:** To determine the type of a variant variable, use the function `VarType( )`, which returns a value that corresponds to the explicit data types. See [VarType](#) for return values.

## Numbers in Variants

When storing numbers in variant variables, the data type used is the most compact type possible. For example, if you first assign a whole number to the variant it will be stored as an integer or long. If you assign a number with a fractional component, it is stored as a single or double.

For doing numeric operations on a variant variable, it is sometimes necessary to determine if the value stored is a valid numeric, thus avoiding an error. This can be done with the `IsNumeric( )` function.

## See Also

[VBA Data Types](#)

[Variables](#)

[Constants](#)

[Strings](#)

[Numbers](#)

[VBA Function Reference](#)

## Numbers

VBA supports three representations of numbers: decimal, octal, and hexadecimal.

To indicate the use of octal (base 8) or hexadecimal (base 16) numbers, prefix the number with &O or &H respectively. If no prefix is included with a number, it is treated as decimal (base 10). For example:

```
Dim vntVar as Variant  
vntVar = 12345 ' assign decimal value  
vntVar = &o12345 ' assign octal value  
vntVar = &h12345 ' assign hexadecimal value
```

Most numbers used in VBA formulas are decimal numbers. Decimal numbers consist of integral values (known as integers) positioned to the left of the decimal point, and fractional values (known as fractions) positioned to the right of the decimal point. If the decimal point is omitted, the number is treated as an integer (whole number with no fraction).

When using numbers in VBA, consideration needs to be given to the data type of the variables that hold and store the numbers, as well as to the behaviour of VBA when dealing with numbers. For details, see [Numeric Data Types](#), [Floating Point Calculation Rules](#), and [Rounding Numbers](#).

## See Also

[Variant Declaration](#)  
[Strings](#)  
[Variables](#)  
[Constants](#)  
[VBA Data Types](#)  
[VBA Function Reference](#)

## Numeric Data Types

Numbers are expressed in VBA in decimal format by default, and can be stored in variables of different numeric data types—integer, long, single, double, and variant—providing different levels of numeric accuracy.

- **Integer** (Integer data type) variables can only store whole number values (no decimal or fraction values) within the range -32,000 to +32,000. If you use a number outside this range, the long integer (Long data type) can store whole number values in the range -2.1 million to +2.1 million.
- **Floating point** numbers contain both integer and fractional values with a floating decimal point. VBA provides both single precision (Single data type) and Double Precision (Double data type) variables for handling floating-point numbers.
- **Single-precision** variables can store floating-point numbers within the range of approximately 3.4E-38 to 3.4E+38, with 7 significant digits and occupying 4 bytes of memory.
- **Double-precision** variables can store floating-point numbers within the range of approximately 1.79D-308 to 1.79D+308, with 15 significant digits and occupying 8 bytes of memory.

For an explanation of exponential notation, see [Exponential Notation](#).

The principal differences between single and double data types, are the significance they can represent, the storage they require, and their range. Double data type variables have a smaller range, but hold more precision and occupy more memory than single data type variables.

If precision is less of a concern than storage, consider using single for floating-point variables. Conversely, if precision is the most important criterion, use double.

Variant (variant data type) variables in VBA can store numbers by storing them as integer, long, single, or double data types within the variant structure. See [Variant Declaration](#).

## See Also

[Numbers](#)

## Exponential Notation

VBA can handle very large numbers, up to a value of 1.7976931486232 raised to the power of 308, (1.7 308). To represent very large numbers such as these, exponential notation is used.

Commonly, exponential notation uses the letter 'E' in the number, followed by the sign of the number (+ or -), and then the exponential value (power) of the number. VBA uses the letter 'E' for Single data type exponential

values, and the letter 'D' for Double data type exponential values. The maximum size number for a double precision data type, as quoted above, would be represented using exponential notation as 1.7976931486232D+308, or abbreviated to 1.79D+308.

You can use exponential notation in your VBA calculations, so long as the data types of all the variables in the calculation are capable of storing floating point values; i.e.: Single, Double or Variant.

For details about precision, accuracy, and rounding issues with using floating point variables in VBA, see [Numeric Data Types](#), [Floating Point Calculation Rules](#), and [Rounding Numbers](#).

## See Also

[Numbers](#)

## Floating Point Calculation Rules

Often precision, rounding, and accuracy in floating-point calculations can generate unexpected results. To avoid this situation, follow these rules:

- In a calculation involving both single and double precision, the result will not usually be any more accurate than single precision. If double precision is required, be certain all terms in the calculation, including constants, are specified in double precision.
- Never assume that a simple numeric value is accurately represented in the computer. Most floating-point values can't be precisely represented as a finite binary value. For example .1 is .0001100110011... in binary (it repeats forever), so it can't be represented with complete accuracy on a computer using binary arithmetic, which includes all PCs.
- Never assume that the result is accurate to the last decimal place. There are always small differences between the "true" answer and what can be calculated with the finite precision of any floating point processing unit.
- Never compare two floating-point values to see if they are equal or not-equal. This is a corollary to rule three. There are almost always going to be small differences between numbers that "should" be equal. Instead, always check to see if the numbers are nearly equal. That is, check to see if the difference between them is very small or insignificant.

## See Also

[Numbers](#)

[Date and Time Handling](#)

## Rounding Numbers

Rounding occurs when you convert a number of greater precision into a number of lesser precision. For instance, when converting a floating-point number (single precision, double precision, or variant data types) into an integer or long data type number. The possible ways of numeric rounding are discussed below.

## Rounding Down

The simplest form of rounding is truncation. Any digits after the desired precision are ignored and dropped. The VBA Fix() function is an example of truncation. For example, Fix(3.5) is 3, and Fix(-3.5) is -3.

The Int() function rounds down to the highest integer less than the value. Both Int() and Fix() act the same way with positive numbers (truncating), but give different results for negative numbers: Int(-3.5) gives -4.

The Fix() function is an example of symmetric rounding because it affects the magnitude (absolute value) of positive and negative numbers in the same way. The Int() function is an example of asymmetric rounding because it affects the magnitude of positive and negative numbers differently.

## Rounding Up

VBA does not have a specific round-up function. However, for negative numbers, both Fix() and Int() can be used to round upward, in different ways:

- Fix() rounds towards 0 (up in the absolute sense, but down in terms of absolute magnitude). For example: Fix(-3.5) is -3.5.
- Int() rounds away from 0 (up in terms of absolute magnitude, but down in the absolute sense). For example: Int(-3.5) is -4.

## Arithmetic Rounding

When continually rounding in one direction (down or up), the resulting number is not necessarily the closest to the original number. For example, if you round 1.9 down to 1, the difference is a lot larger than if you round it up to 2. It is easy to see that numbers from 1.6 to 2.4 should be rounded to 2. However, what about 1.5, which is equidistant between 1 and 2? By mathematical convention, the half-way number is rounded up.

To implement rounding half-way numbers in a symmetric fashion, -.5 is rounded down to -1, or in an asymmetric fashion, where -.5 is rounded up to 0.

VBA does not have a function for arithmetic rounding.

## Banker's Rounding

When you add rounded values together, always rounding .5 in the same direction results in a bias that grows with the more numbers you add together. One way to minimize the bias is with banker's rounding.

Banker's rounding rounds .5 up sometimes and down sometimes. The convention is to round to the nearest even number, so that both 1.5 and 2.5 round to 2, and 3.5 and 4.5 both round to 4. Banker's rounding is symmetric.

In VBA, the CByte(), CInt(), CLng(), CCur(), and Round() numeric functions perform banker's rounding.

## Random Rounding

Even banker's rounding can bias totals. You can take an extra step to remove bias by rounding .5 up or down in a truly random fashion. This way, even if the data is deliberately biased, bias might be minimized. However, using random rounding with randomly distributed data might result in a larger bias than banker's rounding. Random rounding could result in two different totals on the same data.

VBA does not have a function for random rounding.

## Alternate Rounding

Alternate rounding is rounding between .5 up and .5 down on successive calls. VBA does not have a function for alternate rounding.

## See Also

[Numbers](#)

## Date and Time Handling

VBA has several pre-defined date and time functions that return date/time values. There are three functions enabling you to determine the current date and time set in Windows. The Now function, Date function, and Time function check your system clock and return all or part of the current setting.

To retrieve the date portion of a date/time value, use the pre-defined DateValue function. This function takes in either a string or a date value and returns only the date portion.

Using DateValue, you can compare the date portion of a date variable to a specific date value, like this:

```
If DateValue(dtmSomeDate) = #5/14/70# Then
    ' You know the date portion of dtmSomeDate is 5/14/70
End If
```

If you need just the time portion of a date variable, use the TimeValue function. Using this function, you could write code that checks the time portion of a date variable against a particular time, like this:

```
If TimeValue(dtmSomeDate) > #1:00 PM# Then
    ' You know the date variable contained a date portion
    ' with a time after 1:00 PM.
End If
```

You can perform arithmetic or mathematics (math) on date/time values because VBA stores dates internally as serial values. Adding or subtracting integers adds or subtracts days, while adding or subtracting fractions adds or subtracts time. Therefore, adding 20 to a date value in VBA adds 20 days, while subtracting 1/24 subtracts one hour. For example, to get tomorrow's date, you could just add 1 to today's date, like this:

```
dtmTomorrow = Date() + 1
```

Date is a built-in VBA function that returns the date portion (the integer part) of the current date and time retrieved from the Windows operating system. Adding 1 to that value returns a date that represents the next day.

The same mechanism works for subtracting two dates. Although VBA supplies the DateDiff function for finding the interval spanned by two date/time values, if you just need to know the number of days between the two dates, you can simply subtract one from the other.

## See Also

[Date and Time Functions](#)  
[Date Constants](#)  
[Formatting Date Values](#)  
[VBA Function Reference](#)

## Date Constants

You can use date/time literals in VBA code by enclosing them with the hash sign (#), in the same way you enclose string literals with double quotation marks (""). This is commonly known as declaring date constants.

For example, #2/6/10# represents the Australian date value of 2nd June, 2010 if the short date setting for the locale was set to d/MM/yyyy. The same date constant would represent the American date value of February 6, 2010 if the short date setting for the locale was set to MM/d/yyyy. See [Formatting Date Values](#).

**Note:** The system locale settings are determined using Regional Settings in Windows Control Panel.

Similarly, you can compare a date/time value with a complete date/time literal:

```
If SomeDate > #3/6/99 1:20pm# Then
```

If you don't include a time in a date/time literal, VBA sets the time part of the value to midnight (the start of the day). If you don't include a date in a date/time literal, VBA sets the date part of the value to December 30, 1899.

VBA accepts a wide variety of date and time formats in literals. These are all valid date/time values:

```
SomeDate = #3-6-93 13:20#
SomeDate = #March 27, 1993 1:20am#
SomeDate = #Apr-2-93#
SomeDate = #4 April 1993#
```

In the same way that you can use the IsNumeric function to determine if a Variant variable contains a value that can be considered a valid numeric value, you can use the IsDate function to determine if a variant contains a value that can be considered a valid date/time value. You can then use the CDate function to convert the value into a date/time value.

For example, the following code tests the Text property of a text box with IsDate. If the property contains text that can be considered a valid date, VBA converts the text into a date and computes the days left until the end of the year:

```
Dim SomeDate, daysleft
If IsDate(Text1.Text) Then
    SomeDate = CDate(Text1.Text)
    daysleft = DateSerial(Year(SomeDate)) + _
    1, 1, 1) - SomeDate
    Text2.Text = daysleft & " days left in the year."
Else
    MsgBox Text1.Text & " is not a valid date."
End If
```

## See Also

[Date and Time Functions](#)  
[Formatting Date Values](#)  
[VBA Function Reference](#)  
[Date and Time Handling](#)

## Formatting Date Values

Date values in VBA can be formatted and displayed as strings containing any combination of the following lists of values, all of which are case-sensitive.

Most VBA date and time functions are locale-aware and recognize the order for day, month, and year according

to the short date format in the Regional Settings section of Windows Control Panel.

**Note:** Always use long date format whenever possible. Also, please ensure that you enter and display dates in an unambiguous format. For example, dates in short date format might be misinterpreted in queries if the year or the day of the month are 12 or less (for example, 3/11/10). Dates in medium date format display only the first few characters of the month name, which can create ambiguity or an undesirable appearance.

## Text

All strings should be surrounded by single quotes, and any single quotes should be entered as four single quotes in a row:

Value	Example
it'''s	it's
'Today is 'M/dd/yy' and it'''s 'h:mm	Today is 01/22/99 and it's 8:18

## Day

The day can be displayed in one of four formats using a lowercase "d".

Value	Meaning	Example
d	Day of the month as digits without leading zeros for single digit days.	5
dd	Day of the month as digits with leading zeros for single digit days.	05
ddd	Day of the week as a three letter abbreviation.	Mon
dddd	Day of the week as its full name.	Monday

## Month

The month can be displayed in one of four formats using capital "M". The letter "M" must be uppercase to distinguish months from minutes.

Value	Meaning	Example
M	Month as digits without leading zeros for single digit months.	1
MM	Month as digits with leading zeros for single digit months.	01
MMM	Month as a three letter abbreviation.	Jan

Value	Meaning	Example
MMMM	Month as its full name.	January

## Year

The year can be displayed in one of three formats using lowercase "y".

Value	Meaning	Example
y	Year represented only by the last digit, if the year is less than 10. Years greater than 10 will be given the value of yy.	9
yy	Year represented only by the last two digits.	09
yyyy	Year represented by the full 4 digits.	1909

## Period/Era

The period/era string can be displayed in a single format using the letter "g". The letter "g" must be lowercase. If you include the gg in a date string that does not have any associated Era string, the gg is ignored.

Value	Meaning
(Null)	Gregorian dates are used. Does nothing if Gregorian is value of iCalendarType
gg	Period/era string. This is used by Windows to calculate the year when an optional calendar is selected. See iCalendarType for optional Calendars.

## Time

The time can be displayed in one of many formats using the letter "h" or "H" to denote hours, the letter "m" to denote minutes, the letter "s" to denote seconds and the letter "t" to denote the time marker. The lowercase "h" denotes the 12 hour clock and uppercase "H" denotes the 24 hour clock. The "m" must be lowercase to denote minutes as opposed to Months. The "s" for seconds and "t" for the time marker string must also be lowercase.

Value	Meaning	Example
h	Hours without leading zeros for single digit hours (12 hour clock).	1
hh	Hours with leading zeros for single	01

Value	Meaning	Example
	digit hours (12 hour clock).	
H	Hours without leading zeros for single digit hours (24 hour clock).	1
HH	Hours with leading zeros for single digit hours (24 hour clock).	01
m	Minutes without leading zeros for single digit minutes.	9
mm	Minutes with leading zeros for single digit minutes.	09
s	Seconds without leading zeros for single digit seconds.	5
ss	Seconds with leading zeros for single digit seconds.	05
t	One character time marker string. This will be the first letter of the values in the AM symbol or PM symbol boxes in Regional Options	A
tt	Multi-character time marker string. This will be values in the AM symbol or PM symbol boxes in Regional Options	AM

## See Also

[Date and Time Handling](#)

[Date Constants](#)

[Date and Time Data Constraints](#)

## Date and Time Data Constraints

VBA has several constraints on date and time values based upon the Gregorian Calendar:

- The value of the month field must be between 1 and 12, inclusive.
- The value of the day field must be in the range from 1 through the number of days in the month. The number of days in the month is determined from the values of the year and months fields and can be 28, 29, 30, or 31. (The number of days in the month can also depend on whether it is a leap year.)
- The value of the hour field must be between 0 and 23, inclusive.
- The value of the minute field must be between 0 and 59, inclusive.

- For the trailing seconds field of interval data types, the value of the seconds field must be between 0 and  $59.9(n)$ , inclusive, where  $n$  is the number of digits in the fractional seconds precision.
- For the trailing seconds field of datetime data types, the value of the seconds field must be between 0 and  $61.9(n)$ , inclusive, where  $n$  specifies the number of "9" digits and the value of  $n$  is the fractional seconds precision. (The range of seconds allows as many as two leap seconds to maintain synchronisation of sidereal time.)

## See Also

- [Date and Time Handling](#)  
[Date Constants](#)  
[Formatting Date Values](#)  
[Date Data Type Structure](#)  
[Dates in Databases Using Different Calendars](#)

## Date Data Type Structure

The date data type structure in VBA is a junction of two very different methods of data storage. It is formed from the serial concatenation of a date-value followed by a time-value, separated by a floating point, and stored in an 8-byte floating-point date data type value.

- The integer (whole) portion of the value represents the number of days elapsed since December 30, 1899.
- The remainder (fractional) portion of the value represents the elapsed portion of the day since midnight.

The integer (date-value) portion (to the left of the floating point) and the remainder (time-value) portion (to the right of the floating point) must be handled differently as they are structured very differently. VBA has a number of pre-defined date and time functions to convert between the internal floating-point date data type format and visibly recognizable dates and times.

---

**Note:** Dates in VBA are based upon the Gregorian Calendar.

For example, the date and time of 5/22/97 at 3:00 p.m. would be stored in VBA as 35572.625 representing the 35572 days since 12/30/1899, and 3:00 p.m. as 625/1000 of a full day.

---

**Note:** Don't confuse Date data types used in VBA with date and time values used in Windows, DLLs, Plant SCADA, or in Cicode. For instance, Plant SCADA stores time/date-related variables as a single integer representing the number of seconds since 01/01/1970.

---

## Date-values

A date-value in VBA is a count of the number of days from December 30, 1899. December 31, 1899 has the date-value of 1, and the 1st January 1900 is 2. December 30, 1899 has the date value of zero. Negative date-values represent dates prior to December 30, 1899.

A date-value in VBA can actually range from January 1, 0100, to December 31, 9999 inclusive, which is a integer value ranging from -657434 up to +2958465 respectively. Using values outside this range will cause compiler errors in VBA.

The pre-defined VBA Year, Month, and Day functions calculate and return the appropriate year, month or day value (as an integer) from a date-value.

## Time-values

A time-value in VBA represents the fractional time of day since the previous midnight. Unlike a date-value which is simply a count of the number of days, the time-value is a decimal fraction of a day.

As every day invariably consists of 24 hours, or 1440 minutes, or 86,400 seconds, the time of day can be readily determined from a time-value using simple math. An hour has the time-value of one twenty-fourth of one day (0.0416'), one minute has the time-value of 0.000694', and a second has the time-value of 0.000011574'0'7'. Midday has the time-value of 0.50. 1AM has the time-value of 0.0416'. 1PM has the time value of 0.5416'.

The pre-defined VBA Hour function, Minute function, and Second function calculates and returns the appropriate hour, minute or second value (as an integer) from a time-value.

## See Also

[Date and Time Handling](#)

[Date Constants](#)

[Formatting Date Values](#)

[Dates in Databases Using Different Calendars](#)

[Date and Time Data Constraints](#)

## Dates in Databases Using Different Calendars

If you use an existing database with date references of a different calendar type than your current operation system locality settings, VBA could report a variety of errors or perform in an unexpected manner at runtime. For example, if you have used Hijri Calendar dates in your database, a syntax error message will occur if VBA makes reference to Gregorian dates that are invalid Hijri dates (for example, the 31st of any month will produce a syntax error because no Hijri month has 31 days).



### UNINTENDED EQUIPMENT OPERATION

Always confirm that calendar types in existing databases are compatible with the locality (regional and language) options of the operating system.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

To avoid problems of this sort, all date references in an external database should be based on the Gregorian Calendar, or the database tables can be exported to text files before use in Plant SCADA. Dates in Microsoft Access database tables exported as text files are always stored as Gregorian values. If the database calendar is set to Hijri for example, automatic Hijri to Gregorian conversion is performed during the export process.

You can't set a database calendar programmatically using VBA.

---

**Note:** When you want to use characters for Baltic, Central European, Cyrillic, Greek, Turkish, and Asian languages, or right-to-left languages (Arabic, Hebrew, Farsi, and Urdu) the operating system would need to have the corresponding language version of Windows, or have installed system support for that language.

---

## See Also

[Date and Time Handling](#)

## Operators

Variables can be manipulated in VBA using assignment, arithmetic, relational, and logical operators.

- The assignment operator is used to assign a value to a variable or constant (that equals this).
- Arithmetic operators are used to mathematically manipulate numeric variables and numbers (addition, subtraction, multiplication, division, etc.).
- Relational Operators are used to compare the relationship between variables (less than, greater than, not equal to, etc.).
- Logical operators are used to perform digital logic operations on variables ( AND, OR, NOT, etc.).

When using multiple operators in a VBA statement, you need to ensure the proper execution of your code by observing order of precedence rules.

The string concatenation operator is used to join strings together.

## See Also

[Constants](#)

[Variables](#)

[Numbers](#)

[Strings](#)

[Date and Time Handling](#)

[VBA Function Reference](#)

## Assignment Operator

The VBA assignment operator uses the equals character ( = ) in a VBA statement. The variable named to the left side of the assignment operator is assigned the operand value from the right side of the assignment operator, as shown here:

```
' declares integer variable named X
Dim X As Integer
' declares integer variable named Y
Dim Y as Integer
X = 123 ' assigns numeric value 123 to variable X
Y = X ' assigns value of variable X to variable Y
```

Only one variable can be assigned at any one time with the assignment operator.

There must be a space on either side of the assignment operator, or the equals character may be confused with either the variable name or the value being assigned, and a compile error may occur.

Unless the variable is a variant data type, the value being assigned must be the same data type as the variable receiving the assigned value. For instance, if you assign a text string into a long data type variable, you'll cause an

error to occur.

The variable must be previously declared before being assigned a value. The value of a variable can be changed any number of times in a later statements, as in the following VBA example:

```
' declare integer variable named X
' and assign an initial numeric value of 123 to it
Dim X = 123 As Integer
' <statement>
' <statement>
' <statement>
' reassign X to store the numeric value 456
X = 456
' <statement>
' <statement>
' <statement>
' reassign X to store the numeric value 789
X = 789
```

## See Also

[Operators](#)

### Arithmetical (Math) Operators

VBA arithmetic operators are used in VBA statements to mathematically manipulate numeric variables and numbers. The resultant is often assigned to a third variable using the assignment operator.

The arithmetic operation as determined by the arithmetic operator is performed between the values of the operands (variables or numbers positioned immediately on either side of the arithmetic operator).

Operator	Function	Usage
$^$	Exponentiation	$X = Y ^ 2$
$-$	Negation	$X = - 2$
$*$	Multiplication	$X = 2 * 3$
$/$	Division	$X = 10 / 2$
Mod	Modulo	$X = Y \text{ MOD } Z$
$+$	Addition	$X = 2 + 3$
$-$	Addition	$X = 6 - 4$

## See Also

[Operators](#)

## Relational Operators

VBA Relational Operators are used in VBA statements to compare the relationship between operands (values positioned immediately on either side of the Relational Operator). The boolean result is either True or False.

Operator	Function	Usage
<	Less than	X < Y
<=	Less than or equal to	X <= Y
=	Equals	X = Y
>=	Greater than or equal to	X >= Y
>	Greater than	X > Y
<>	Not equal to	X <> Y

## See Also

[Operators](#)

## Logical Operators

Logical (boolean) operators are used to perform digital logic operations on variables. All logical operations result in either a boolean True or False.

Operator	Function	Usage
Not	logical negation	if not X
And	logical And	If (X> Y) And (X < Z)
Or	logical Or	If (X = Y) Or (X = Z)

## See Also

[Operators](#)

## Operator Precedence

The operator precedence in VBA runs like this:

Operator	Description	Order
( )	Parenthesis	Highest
^	Exponentiation	

Operator	Description	Order
-	Unary minus	
/, *	division/multiplication	
Mod	Modulo	
+, -, &	addition, subtraction, concatenation	
=, <>, <, >, <=, >=	Relational	
Not	Logical negation	
And	Logical conjunction	
Or	logical disjunction	
Xor	logical exclusion	
Eqv	logical equivalence	
Imp	logical implication	Lowest

## See Also

[Operators](#)

## Strings

Strings can be stored in variables of string and variant. When using variant strings, be aware of type coercion in VBA. Strings can be compared with each other in VBA to determine whether they contain the same characters or not. Strings can be joined together to create longer strings in VBA using the concatenation operator.

Strings can be searched using the:

- [InStr](#) function, which returns the character position of the first occurrence of a string within another;
- [Left](#), [Left\\$\(\)](#) function or [Right\(\)](#) function which return a copy of the left or right most characters of a string respectively; and
- [Mid\(\)](#) function, which returns the copy of a substring from within another string.

To determine the length of a string, use the [Len\(\)](#) function which returns a Long variable containing the number of characters in the string.

String characters can be converted to ASCII values using the [Asc\(\)](#) function, and ASCII values can be converted to their string characters using the [Chr\(\)](#) function.

String characters can be converted to all lowercase or all uppercase using the [LCase\(\)](#) Function or the [UCase\(\)](#) Function respectively.

Leading or trailing spaces can be stripped off strings using the [LTrim\(\)](#) function or the [RTrim\(\)](#) function respectively.

Strings can be created consisting of a specified number of spaces or characters using the [Space\(\)](#) function or the [String\(\)](#) function respectively.

For syntax details of using string functions, see [String Functions](#).

## See Also

[Operators](#)

[Numbers](#)

[Control Structures](#)

## String Comparisons

VBA compares ANSI values of characters when comparing strings. For example, the character capital 'A' has the ANSI value of 65, and the character lowercase 'a' has the ANSI value of 97. For a listing of ANSI characters values, see [ASCII/ANSI Character Code Listings](#).

You can use VBA relational operators (less than, greater than, equal to, not equal to, and so on) to compare string variables. All relational operators return either `true` or `false`.

With comparisons made using relational operators, the result depends on the option `compare string-comparison` option setting of the VBA file. Consider the following example:

`"vba" > "VBA"`

If the file Option string-comparison setting is `Option Compare Binary` (or not set at all), the comparison returns `true`. VBA compares the binary (ANSI) values for each corresponding position in the string until it finds two that differ. In this example, the lowercase letter "v" corresponds to the ANSI value 118, while the uppercase letter "V" corresponds to the ANSI value 86. Because 118 is greater than 86, the comparison returns `true`.

If the file Option string-comparison setting is `Option Compare Text`, `"vba" > "VBA"` returns `false`, because the strings are equivalent apart from case.

The built-in VBA `StrComp()` Function returns a variant containing a value representing the result of the comparison of two strings. It has an optional third argument `Comp` which can override the file Option string-comparison setting.

## See Also

[Option Statements](#)

[Strings](#)

## String Concatenation

To concatenate strings in VBA, use the ampersand ( `&` ) concatenation operator between the strings. Multiple concatenations can occur in the same VBA statement.

```
Dim strFirstName As String
Dim strLastName As String
Dim strFullName As String
```

```
Const strSpaceChar = " "
' note the space character between the quotes
strFirstName = "Colin"
strLastName = "Ramsden"
strFullName = strFirstName &strSpaceChar &strLastName
' concatenates string values
```

The & concatenation operator does not perform arithmetic, and will convert variable data types to strings for concatenation. For instance, if a variant string and a variant number are concatenated, the result is a string. For more details of variant data types, see [Variant Declaration](#) and [VBA Data Types](#).

## See Also

[Strings](#)  
[Operators](#)  
[Control Structures](#)

## Control Structures

VBA provides conditional control functionality, which can be used to conditionally perform VBA statements or blocks of statements dependant upon the result of the condition tested. This is known as logical decision making.

The logical decision-making control structures available in VBA consist of three conditional looping or repetitive statements (Do Statement, While Statement, and For Statement), and two conditional flow control sequence statements (Select Case Statement, and variations of the If Statement). In addition, VBA provides one unconditional branching GoTo Statement.

---

**Note:** In the control structure syntax examples, every placeholder shown inside arrow brackets ( <placeholder> ) should be replaced in any actual code with the value of the item that it describes. The arrow brackets and the word they contain should not be included in the statement, and are shown here only for your information. Also, statements shown between square brackets ( [ ] ) are optional. The square brackets should not be included in the statement, and are shown here only for your information.

## See Also

[Operators](#)  
[GoTo Statement](#)  
[Do Statement](#)  
[While Statement](#)  
[For Statement](#)  
[If Statement](#)  
[Select Case Statement](#)  
[End Statement](#)  
[Exit Statement](#)  
[OnError Statement](#)  
[Stop Statement](#)  
[With Statement](#)

[VBA Function Reference](#)

## GoTo Statement

The GoTo conditional statement branches unconditionally and without return to the label specified in the GoTo statement. The label must be located in the same subroutine or function as the Goto statement.

```
<statement/s>
If <condition> then GoTo Label1
Else GoTo Label2
End If
Label1:
<statement/s>
GoTo Label3
Label2:
<statement/s>
GoTo Label3
Label3:
<statement/s>
```

In this example, VBA tests the If condition, and jumps to the part of the script that begins with the label "Label1:" if the condition was true, or jumps to the part of the script that begins with the label "Label2:" if the condition was false. This could be anywhere in the same subroutine or function.

## See Also

[Control Structures](#)

## Do Statement

The Do...Loop conditional statement allows you to execute a block of statements an indefinite number of times. The variations of the Do...Loop are Do While, Do Until, Do Loop While, and Do Loop Until.

```
Do While|Until <condition>
    <statement/s>
Loop

Do Until <condition>
    <statement/s>
Loop

Do
    <statement/s>
Loop While <condition>

Do
    <statement/s>
Loop Until <condition>
```

Do While and Do Until check the condition before entering the loop, thus the block of statements inside the loop are only executed when those conditions are met. Do Loop While and Do Loop Until check the condition after having executed the block of statements so that the block of statements is executed at least once.

Any Do statement can be exited using the Exit Do statement.

## See Also

[Control Structures](#)

### While Statement

The While...Wend loop conditional statement is similar to the Do While loop statement. The condition is checked before executing the block of statements comprising the loop.

```
While <condition>
    <statement/s>
Wend
```

## See Also

[Control Structures](#)

### For Statement

The For...Next loop conditional statement repeats its block of statements a set number of times as determined by the values used with the To clause.

```
For <CounterName> = <BeginValue> To <EndValue> [Step <StepValue>]
    <statement/s>
Next
```

The counter variable is increased or decreased (by the value stated in the optional Step parameter), with each reiteration of the loop. The counter default is to increment by one if the Step parameter is omitted.

## See Also

[Control Structures](#)

### If Statement

The If statement in VBA tests an initial condition and then either performs or omits to perform the statements it contains, dependant upon the logical result of the test condition. The condition can be a comparison or an expression, and must logically evaluate to either True or False. The If statement has both single line and multiple line syntax structure.

The single line syntax uses the If <TestCondition> Then <StatementToPerformIfTrue> structure, however, can only perform a single statement if and only if the test condition result is True. No 'End If' statement is required:

```
If <Condition> Then <Statement>
```

If the result of the If test condition was True, the program flow continues into and performs the statement following the Then statement, until it reaches the end of the line.

To perform a single statement conditionally upon a False result, use the NOT logical operator:

```
If NOT <Condition> Then <Statement>
```

To perform multiple statements, use the multiple line syntax structure which ends with the 'End If' statement:

```
If <Condition> Then
```

```
' Then statement block
' perform only if true
<Statement/s>
End If
```

If the result of the If test condition was True, the program flow continues into the Then statement block, and performs the statements following the Then statement, until it reaches the End If statement.

If the result of the If test condition was False, the program flow jumps over the Then statement block, which in this case exits the If structure (without performing any statements other than the initial test condition).

The multiple line If structure can perform different blocks of statements dependant upon EITHER a True OR a False result to the test condition, through the use of the Else statement block:

```
If <Condition> Then
' Then statement block
' perform only if true
<Statement/s>
Else
' Else statement block
' perform only if false
<Statement/s>
End If
```

If the result of the If test condition was True, the program flow performs the Then block statements, until it reaches the Else statement. It then jumps over the Else statement block and exits the If structure (without performing any of the Else statement block statements).

If the result of the If test condition was False, the program flow jumps over the Then statement block (without performing any of those statements) to the Else statement to perform the statements in the Else statement block until it reaches the End If statement.

Further test conditions can be placed into an If structure through the use of the optional Else If <Condition> statement block. ElseIf statement blocks can only be positioned within an If structure before the Else statement block.

```
If <Condition> Then
' Then statement block
' perform only if true
<Statement/s>
ElseIf <Condition>
' Else If statement block
' perform only if true
<Statement/s>
Else
' Else statement block
' perform only if false
<Statement/s>
End If
```

The ElseIf test condition is only evaluated after the initial If structure test condition results in False.

If the result of the ElseIf test condition was True, the statements within the ElseIf statement block are performed. The program flow then jumps over the Else statement block and exits the If structure (without performing any of the Else statement block statements).

If the result of the ElseIf test condition was False, the program flow jumps over the ElseIf statement block (without performing any of those statements) to the Else statement to perform the statements in the Else statement block until it reaches the End If statement.

There is no apparent limit to the number of Else If statement blocks that any one If structure can hold, however,

the Select Case Statement structure handles multiple condition result alternatives much more efficiently.

## See Also

[Control Structures](#)

### Select Case Statement

The Select Case statement tests the same variable for many different conditions. The test value provided with the initial Select Case statement is logically tested against the Case test condition.

The Select Case structure can perform different blocks of statements dependant upon whichever Case statement test condition (if more than one) first results as True, through the use of the Case statement block:

```
Select Case <TestValue>
Case <Condition>
  ' Case statement block
  ' perform only if case true
<Statement/s>
Case Else
  ' Else statement block
  ' perform only if all cases false
<Statement/s>
End Select
```

If the result of the Case test condition was True, the program flow performs the statements contained within that Case statement block, and will then exit the Select Case structure (without performing any of the Else statement block statements).

If the result of the Case test condition was False, the program flow jumps over the Case statement block (without performing any of those statements) to the Case Else statement to perform the statements in the Else statement block until it reaches the End Select statement.

Further test conditions can be placed into a Select Case structure through the optional use of further Case statement blocks. Case statement blocks can only be positioned within a Select Case structure before the Case Else statement block.

```
Select Case <TestValue>
Case <Condition>
  ' Case statement block
  ' perform only if case true
<Statement/s>
Case <Condition>
  ' Case statement block
  ' perform only if case true
<Statement/s>
Case Else
  ' Else statement block
  ' perform only if all cases false
<Statement/s>
End Select
```

Each Case statement block is evaluated in order until the test condition of one results as True. The program flow performs the statements contained within that Case statement block, and will then exit the Select Case structure (without performing any other statements).

The statements of ONLY one Case statement block are ever performed, unless all result in False and there is no

Case Else block declared, in which case no Case statement blocks are performed at all.

The following example may help clarify the logic testing being performed in a Select Case structure. Lets say that we have a variable named (intDayOfWeek) containing an integer (ranging from 1 to 7) representing the day of the week, and we wished to display that value as a string (named strDayOfWeek) containing the name of the day of the week, assuming in this example, that Sunday is the first day of the week (1). The Select Case structure would look like this:

```
Dim strDayOfWeek As String
Select Case intDayOfWeek
Case = 1
StrDayOfWeek = "Sunday"
Case = 2
StrDayOfWeek = "Monday"
Case = 3
StrDayOfWeek = "Tuesday"
Case = 4
StrDayOfWeek = "Wednesday"
Case = 5
StrDayOfWeek = "Thursday"
Case = 6
StrDayOfWeek = "Friday"
Case = 7
StrDayOfWeek = "Saturday"
Case Else
StrDayOfWeek = "Invalid"
End Select
```

The Select Case structure tends to be easier to read, understand, and follow and should be used in place of a complicated multi-nested If...ElseIf structure.

## See Also

[Control Structures](#)

## End Statement

The End statement Ends a block of statements such as a Sub procedure or Function.

```
End[{Function | If | Sub}]
```

## Example

```
Dim Var1 as String
Var1 = "hello"
' Calling Test
Test Var1
MsgBox Var1

Sub Test(wvar1 as string)
MsgBox wvar1
wvar1 = "goodbye"
End
End Sub
```

## See Also

[Control Structures](#)

## Exit Statement

Exits a loop or procedure

```
Exit {Do | For | Function | Sub }
```

## Example

```
' This sample shows Do ... Loop with Exit Do to get out.  
Dim Value, Msg ' Declare variables  
Do  
Value = InputBox("Enter a value from 5 to 10.")  
If Value >= 5 And Value <= 10 Then ' Check range  
Exit Do ' Exit Do...Loop  
Else  
Beep ' Beep if not in range  
End If  
Loop
```

## See Also

[Control Structures](#)

## OnError Statement

VBA's error-handling routine and specifies the line label of the error-handling routine. The line parameter refers to a label. That label needs to be present in the code or an error is generated.

## Syntax

```
On Error {GoTo line | Resume Next | GoTo 0}
```

## Example

```
On Error GoTo errHandler  
Dim x as object  
x.draw ' Object not set  
..  
Exit Sub  
errHandler:  
Print Err.Number, Err.Description  
Resume Next
```

## See Also

[Control Structures](#)

### Stop Statement

Ends execution of the program. The Stop statement can be placed anywhere in your code.

## Example

```
Dim x,y,z
For x = 1 to 5
For y = 1 to 5
For z = 1 to 5
Print "Looping",z,y,x
Next z
Next y
Stop
Next x
```

## See Also

[Control Structures](#)

### With Statement

The With Statement is not supported in VBA. When performing a series of commands on an object, you must explicitly refer to the name of the object with each command.

## See Also

[Control Structures](#)

## Subroutines and Functions

Commonly used sequences of VBA statements can be grouped together into functions and subroutines, so that they can be keyed in once, and used many times in many places, by 'calling' the name of the subroutine or function in code.

A subroutine or function is a group or list of sequential statements that VBA can perform (execute) in the logical order that they exist within the subroutine or function from top to bottom in the order they are listed within the function or subroutine.

If the group of statements returns a value, it must be declared as a function. If it does not return a value, it must be declared as a subroutine. A subroutine or function is called by placing the name of the subroutine or function in a code statement where you want the action of the subroutine or function to occur.

**Note:** Subroutines and functions can contain statements that call other subroutines or functions (to perform, before returning to the following statements within the calling subroutine or function).

Both subroutines and functions can similarly be passed values as arguments when they are called:

- Arguments are passed to **subroutines** in VBA code following the subroutine name and separated by space characters.
- Arguments are passed to **functions** enclosed within parentheses in VBA code, similarly following the subroutine name and separated by space characters.

**Note:** Plant SCADA tag values must be declared by value when passed as argument values to a VBA procedure from within a Plant SCADA command or expression field (see [Passing Variables Byref and Byval](#)).

## See Also

[Subroutines](#)  
[Functions](#)  
[Arguments](#)

## Subroutines

A VBA subroutine starts with the SUB statement and finishes with the END SUB statement. All other statements that lie between the SUB and END SUB statements, will be executed by the subroutine, when called to do so.

In the following subroutine syntax example:

- Every placeholder shown inside arrow brackets (**<placeholder>**) should be replaced in any actual code with the value of the item that it describes. The arrow brackets and the word they contain should not be included in the statement, and are shown here only for your information.
- Statements shown between square brackets ([ ]) are optional. The square brackets should not be included in the statement, and are shown here only for your information.

In VBA, Subroutines are created with the SUB statement in the following format.

```
Sub <SubName> ( [ Byval ] [ <Argument/s> ] [ <As Data Type> ] )
<statement>
<statement>
<statement>
End Sub
```

Where:

- **[Byval]** is the optional parameter for the argument;
- **Sub** is the required subroutine statement basic keyword
- **<SubName>** represents the required name of the subroutine being created
- **<Argument/s>** represents the optional argument/s of the subroutine
- **<statement>** represents the executable VBA script statement/s
- **End Sub** is the subroutine terminating statement

The name given to the subroutine immediately follows the SUB keyword, and is used to identify the subroutine to VBA. This name is referred to when the subroutine is called upon (called) to be executed (perform the statements it contains) by some other procedure in VBA.

Subroutine names can contain the letters 'A' to 'Z' and 'a' to 'z', the underscore '\_' and digits '0' to '9'. The subroutine name must begin with a letter, be no longer than 40 characters, cannot contain the space character, and cannot be a reserved word. Subroutine names (once declared), become a keyword in VBA. Like most keywords in VBA, these names are not case sensitive.

The subroutine name always ends with a pair of parentheses ( ) which may or may not contain one or more arguments required by (necessary for use in) the subroutine . Multiple arguments if used, are separated by commas ( , ). See [Arguments](#) for more details and argument syntax.

All the lines located between the SUB and the END SUB statements, contain the statements that will be executed when the subroutine is called in VBA. These statements will be executed one at a time in logical order from top to bottom within the subroutine.

## See Also

[Subroutines and Functions](#)

[Functions](#)

[Arguments](#)

## Functions

A VBA function starts with the FUNCTION statement and finishes with the END FUNCTION statement. All other statements that lie between the FUNCTION and END FUNCTION statements, will be executed by the function, when called to do so.

In the following function syntax example:

- Every placeholder shown inside arrow brackets ( <placeholder>) should be replaced in any actual code with the value of the item that it describes. The arrow brackets and the word they contain should not be included in the statement, and are shown here only for your information.
- Statements shown between square brackets ( [ ] ) are optional. The square brackets should not be included in the statement, and are shown here only for your information.

A typical VBA function is structured like in the following example:

```
Function <FunctionName> ( [ ByVal ] [ <Argument/s> ] ) [ As <ReturnDataType> ]
<statement>
<statement>
<statement>
[ <FunctionName> = <value> ]
End Function
```

Where:

- **Function** is the required function statement basic keyword
- **[ ByVal ]** is the optional parameter for the argument;
- **<FunctionName>** represents the required name of the function being created
- **<Argument/s>** represents the optional argument/s of the function
- **<ReturnDataType>** represents the optional return data type of the function
- **<statement>** represents the executable VBA script statement/s

- **= <value>** represents the optional assignment of the return value for the function
- **End Function** is the function terminating statement

The name given to the function, immediately follows the FUNCTION keyword, and is used to identify the function to VBA. This name is referred to when the function is called upon (called) to be executed (perform the statements it contains) by some other procedure in VBA.

Function names can contain the letters 'A' to 'Z' and 'a' to 'z', the underscore '\_' and digits '0' to '9'. The function name must begin with a letter, be no longer than 40 characters, cannot contain the space character, and cannot be a reserved word. Function names (once declared), become a keyword in VBA. Like most keywords in VBA, these names are not case sensitive.

The function name always ends with a pair of parentheses ( ) which may or may not contain one or more arguments required by (necessary for use in) the function. Multiple arguments if used, are separated by commas ( , ). See the section titled 'Arguments in VBA' for more details and argument syntax.

All the lines located between the FUNCTION and the END FUNCTION statements, contain the statements that will be executed when the function is called in VBA. These statements will be executed one at a time in logical order from top to bottom within the function.

The return value of the function is optionally assigned within the function in a statement using the function name. This value is often used within the calling procedure to determine the status of the function. Commonly, this value may be a Boolean True or False to indicate the successful completion or not of the function.

## See Also

[Subroutines and Functions](#)  
[Arguments](#)  
[Subroutines](#)  
[Accessing Functions in DLLs](#)

## Arguments

Arguments are used in VBA to pass values into subroutines and functions when they are being called. Arguments are positioned between parentheses '( )' immediately after the subroutine or function name in the subroutine or function declaration. If no arguments are required for the subroutine or function, the parentheses must be included and left empty in the declaration.

Arguments are optional in the sense that subroutines and functions do not require them. However, if arguments are to be used in a subroutine or function, the arguments must first be declared with the subroutine or function declaration, before they can be used. If declared, they must be used whenever the subroutine or function is called.

VBA does NOT support named arguments so all arguments must be used in declaration order. If omitted, strings default to an empty string (""), and numeric values default to zero (0). Boolean values in VBA are represented with -1 for TRUE, and 0 for FALSE.

Multiple arguments must be separated by a comma ( , ) placed between the arguments. The number of arguments that can be used in any single subroutine or function is not stated, (but likely limited to something like 255). If you are declaring a subroutine or function with that many arguments, you should probably split your subroutine or function into smaller separate logical routines with less arguments for each routine. If an argument is omitted, its place must be declared by the use of a comma in the call.

If you want to use the value in a Plant SCADA tag as an argument to a function or subroutine, you must assign the value of the tag to a VBA variable, and then pass the variable as the argument. You cannot pass a Plant SCADA tag name as an argument to a function or subroutine.

Each argument declaration in a subroutine or function must be structured using the proper VBA argument syntax as described below.

VBA argument structure syntax in the declaration of functions or subroutines is as follows:

( [ Byval ] <Argument/s> [ As <DataType> ] )

Where:

- [ **Byval** ] is the optional parameter for the argument.
- <Argument/s> represents the argument/s required by the function or subroutine.
- [ As <DataType> ] represents the optional data type declaration of the argument.

The optional 'Byval' parameter specifies that the variable is passed by value instead of by reference (see the section titled 'Passing Variables Byref and Byval with VBA').

---

**Note:** Plant SCADA tag values MUST be declared by value when passed as argument values to a VBA procedure from within a Plant SCADA command or expression field. This is best done by declaring a variable, assigning it the tag value, then passing the variable by value.

The function or subroutine name always ends with a pair of parentheses ( ) which may or may not contain one or more arguments required by (necessary for use in) the function or subroutine. Multiple arguments if used, are separated by commas ( , ).

The optional 'As <DataType>' parameter is used to specify the data type of the argument variable. The argument data types must be individually declared, or will be of Variant data type by default. Valid data types for arguments in VBA are: String, Integer, Double, Long, and Variant (see the section titled 'VBA\_Data\_Types' for descriptions of data types in VBA).

## Example

```
' Arguments are declared with the function or subroutine
' The function is called from the subroutine highlighted below
Function longArea(Byval longLength As Long, _
Byval longWidth As Long) As Long
    ' multiplies arguments and
    ' assigns result to return value
longArea = longLength * longWidth
End Function
Sub FindArea
    ' declare long variables X Y and Z
Dim longX As Long
Dim longY As Long
Dim longZ As Long

    ' assign numeric value 12 to variable X
X = 12
    ' assign numeric value 34 to variable Y
Y = 34

    ' call function named longArea,
    ' passing in values of X and Y variables
```

```
' as arguments
'store result in variable Z
Z = longArea(X, Y)
' copy result Z to tag
TestTag_1 = Z
End Sub
```

Granted, that's not likely the way you'd actually calculate an area given two fixed values in a subroutine that calls a function. You could just as easily do the calculation within the subroutine. However, this example does demonstrate the passing of values from a subroutine to a function, and the retrieval of a return value from the function back to the calling subroutine.

Note in the previous example, that the argument names ('longLength' and 'longWidth') are only used within the function in which they were declared. The values they represented were passed in with the call to the function in the statement line:

```
Z = longArea(X, Y)
```

The values of the variables 'X' and 'Y' were passed into the function 'longArea' and were handled within the function as its argument names 'longLength' and 'longWidth'. The result was returned and stored in the variable named 'Z'.

## See Also

[Subroutines and Functions](#)

[Subroutines](#)

[Functions](#)

## DLLs and APIs

Dynamic Linked Libraries (DLLs) are files that contain functions which can be called from any application running under Microsoft Windows. At run time, a function in a DLL is dynamically linked into an application that calls it. No matter how many applications call a function in a DLL, that function exists in only a single file on the computer, and the DLL is loaded only once in memory.

An application programming interface (API) is a set of functions you can use to work with a component, application, or the operating system. Typically an API consists of one or more DLLs that provide some specific functionality.

For example, the Windows API includes the DLLs that make up the Windows Operating System (OS). Every Windows application interacts with the Windows API directly or indirectly. The Windows API is provided so that all applications running under Windows can behave in a consistent manner.

---

**Note:** Plant SCADA itself provides an API for external access to Plant SCADA I/O variable tags via a DLL interface.

APIs are traditionally written for C and C++ programmers who are building Windows applications, however, the functions in a DLL can also be called by other programming languages, including VBA. Because most DLLs are written and documented primarily for C/C++ programmers, calling a DLL function may differ somewhat from calling a VBA function. In order to work with an API, you need to understand how to pass arguments from VBA to a DLL function. See [Passing Arguments to DLL Functions from VBA](#).

## See Also

[Passing Variables Byref and ByVal](#)

[VBA Function Reference](#)

## Accessing Functions in DLLs

To be able to call and use an external DLL function using VBA, you must have previously provided VBA with details about the function being called. VBA requires information like the name of the function, where that function is located, what arguments it expects, and what type of data it returns. VBA uses the Declare statement to detail that information.

The Declare statement must be positioned in the VBA file in your project above and before any code that calls that declared function of the DLL.

### Declare Statement Structure

The Declare statement consists of the required Declare keyword, followed by the required Function statement, the required Lib statement, the optional Alias statement, the optional Argument statement(s) contained within braces, and the optional return data type statement.

---

**Note:** The use of the OPTIONAL components of the **Declare** statement syntax indicates that they may not be required in all DLL functions. It is not up to you whether you can optionally use them or not. If included in a DLL function, they **MUST** be used when declaring that function to VBA.

---

The Declare statement in VBA details the name, file location, arguments, intrinsic constants, and type definitions that the DLL function requires. Here's an example of the Declare statement for the Windows API GetTempPathA function, which returns the path to the Windows system temporary folder:

```
Declare Function GetTempPathA Lib "kernel32" _
(Byval nBufferLength As Long, _
Byval lpBuffer As String) As Long
```

The Declare keyword indicates to VBA that you intend to call a function belonging to an external DLL. The Declare keyword must be used first in the declaration statement.

## Declare - Function Statement

The Function statement consists of the Function keyword, followed by the name that you will use when calling this function from VBA.

## Declare - Lib Statement

The Lib statement specifies which DLL contains the function you wish to use. The Lib statement consists of the Lib keyword, followed by the name of the DLL contained within string double quotes. Some commonly used DLLs in the Windows API for example, are Kernel32.dll - which performs low level OS functions like memory management and resource handling, the User32.dll - which performs Windows message handling, timers, menus and communication functions, and the GDI32.dll - which performs the graphics display and font management functions.

# Declare - Alias Statement

In the previous Declare statement example, the name of the declared function in VBA is the same as the name of the actual function within the DLL. This does not necessarily have to be the case. There are some instances where the name of the function in the DLL is incompatible with the naming structure of VBA, and cannot be used as a declared function name in VBA. An example would be those DLL function names that start with an underscore.

To overcome such incompatibilities, the VBA Declare statement supports the use of an alias name for the DLL function, through the use of the optional Alias statement . The Alias statement consists of the Alias keyword, followed by the actual name of the DLL function contained within string double quotes. The Alias statement must be positioned within the Declare statement between the Lib statement and the Argument statement.

Here's an example of the Declare statement for the Windows API GetTempPathA function as used above, however, this time using the optional Alias statement:

```
Declare Function GetWinTempPath Lib "kernel32" _  
    (Byval nBufferLength As Long, _ Alias "GetTempPathA" _  
    Byval lpBuffer As String) As Long
```

In this example, the name of the API function in the DLL is GetTempPathA, and the name by which you would call this function from VBA is GetWinTempPath. Note that the actual name of the DLL function appears contained within string double quotes positioned after the Alias keyword. This instructs VBA to use the alias function name when calling the DLL.

Because an alias allows you to name a declared DLL function anything you want in VBA, you can make the function name conform to your own naming standards.

---

**Note:** DLL functions are case sensitive; VBA function names are not. When declaring DLL functions in VBA, be careful to accurately remain case sensitive in the declaration.

---

## See Also

[Functions](#)

[Passing Variables Byref and Byval](#)

[Passing Arguments to DLL Functions from VBA](#)

[DLLs and APIs](#)

## Passing Variables Byref and Byval

Passing an argument by reference (using the Byref parameter) passes a pointer to the memory location of that argument. A pointer is just a memory address that indicates where the value is stored. If the procedure modifies that argument's value, it modifies the source of that argument, so when execution returns to the calling procedure, the source contains the modified value.

Passing an argument to a function by value (using the Byval parameter), on the other hand, passes a copy of the value as the argument. This prevents that function from modifying the source of the argument. When execution returns to the calling procedure, the source contains the same value it did before the function was called.

The **Byref** parameter is the default in VBA and does not need to be used explicitly within VBA. **Byref** gives other subroutines and functions permission to make changes to the source of the values that are passed in **Byref**. The keyword **Byval** denies this permission so the argument source cannot be altered.

There are two possible methods for indicating to VBA that you wish to pass an argument by value:

- When declaring the argument in the subroutine or function declaration statement, by using the **Byval** keyword placed immediately before the argument name. This forces the subroutine or function to use a copy of the argument passed in and not modify the source. For example, the following function TestPassArg has declared its first argument intVal as being requested **Byval**.

```
Function TestPassArg(ByvalintVal As Integer, varVal, strVal as String)
```

- When passing an argument to a subroutine or function, by enclosing the individual argument within parentheses. Only the value of the argument, and not its address in memory, is passed to the subroutine or function, so that the source of the argument is not modified. For example, only the variable var3 is passed by value to the subroutine TestPassArg (because only that argument is enclosed within parentheses in the subroutine call).

```
TestPassArg var1, var2,(var3)
```

- In the next example, the parameter iVar is passed by value to the function TestFunction. Since arguments passed to functions must be enclosed in parentheses, an extra pair is used to force the argument to be passed by value.

```
TestFunction((iVar))
```

---

**Note:**

- Plant SCADA does not support passing by reference, so Plant SCADA tag values need to be declared by value when passed as arguments to a VBA procedure from within a Plant SCADA command or expression field. This is best done by declaring the variable, assigning it the tag value, then passing the variable by value. (See the example below.)
  - Passing arrays to functions is not supported.
- 

## Example

Suppose you had a variable tag of integer type named "iTag1" and you need to pass it to a function. From within a VBA script, or Plant SCADA command or expression field, you would use the following code example to pass the variable tag value to a function named TagArgumentTest:

```
CivBA
Dim iVar1 as Integer
iVar1 = iTag1
TagArgumentTest(iVar1)
```

---

**Note:** CivBA does not support passing by reference, so VBA variables passed to CivBA functions using the CivBAFunctionCallOpen function must be enclosed in brackets to force the passing of those variables by value.

---

## See Also

[Passing Arguments to DLL Functions from VBA](#)  
[DLLs and APIs](#)  
[Arguments](#)

## Passing Arguments to DLL Functions from VBA

Many arguments to DLL functions are passed by value. By default, arguments in VBA are passed by reference, so

It's important that you include the `Byval` keyword in the function definition when the DLL function requires that an argument be passed by value. See [Passing Variables Byref and Byval](#).

**Note:** Although the `Byval` keyword appears in front of some arguments of type `String`, strings are always passed to Windows API functions by reference, therefore any DLL function can always modify a string source directly.

DLL functions don't return strings in the same way that VBA functions do. Because strings are always passed to DLL functions by reference, the DLL function can modify the value of the string argument. Rather than returning a string as the return value for the function, as you would probably do in VBA, a DLL function returns a string into an argument of type `String` that was passed to the function. The actual return value for the function is often a long integer specifying the number of bytes that were written into the string argument.

To call a DLL function that writes to a `String` variable, you need to take additional steps to format the string properly. First of all, the `String` variable must be a null-terminated string. A null-terminated string ends in a special null character. Secondly, a DLL function can't change the size of a string once it has been created.

Therefore, you need to make sure that the string that you pass to a function is large enough to hold the entire return value, and that it terminates with a Null character. When you pass a string to a DLL function, you'll usually need to specify the size of the string that you've passed in another argument. Windows keeps track of the length of the string so that it doesn't overwrite any memory that the string is using.

**Note:** It's only necessary to pass in a null-terminated string and its size if you're returning a string from a function. If the function does not return a string into a string argument, but instead takes a string that provides information to the function, you can simply pass in a normal VBA `String` variable.

A `Nullstring` is a string of value 0 [no Character code]; note that this is not the same as an empty string ("").

## See Also

[DLLs and APIs](#)

[Arguments](#)

[Passing Variables Byref and Byval](#)

## OLE Services

OLE (Object Linking and Embedding) services is the term used to generally describe the integrated use of separate software components (applications) working together to provide custom software solutions based upon the Microsoft Component Object Model (COM) architecture.

**Note:** When considering the use of OLE services, you should be aware that there are different uses of OLE which have developed over the years and which may be confused with one another. Examples of different OLE services include: object linking, object embedding, visual editing, drag-and-drop, ActiveX Controls, OLE Automation, OLE DB, OLE Messaging, and OLE Networking services. See [OLE Terminology](#).

Plant SCADA supports linked and embedded OLE objects in its graphics pages with the use of ActiveX Controls. See [Accessing ActiveX Objects with VBA](#) in the topic [Integrating VBA into a Project](#).

Plant SCADA can use VBA to perform as an OLE Automation controller. See [OLE Automation Objects](#). Plant SCADA can also exchange data with other applications using other data transfer technologies.

## OLE Terminology

OLE superceded the Dynamic Data Exchange protocol. Network DDE was introduced to afford the same data

transfer facility between Windows applications connected across the same network. Plant SCADA supports both DDE and Network DDE connectivity.

## OLE Linking and Embedding

The differences between linked objects and embedded objects which may affect you, concern where the data is stored, and how it is updated after you place it in the destination file. With linked OLE objects, the source of the OLE object data remains in the original data file of the application that was used to create it, and only a copy of the data is ever displayed in the destination document. The data is updated only when the source file is modified. Embedded OLE objects duplicate and store a local copy of the source file data within the destination document data file, and are not linked to the source file. That is, the data copy in the destination file does not change when you modify the source file.

With both linked and embedded OLE objects, when the OLE object in the destination document is double-clicked, the original application (that was used to create the data) of the OLE object is launched to permit editing of the data using that source program's editor. Linked OLE objects store their data back in the original source data files, while embedded OLE objects store their data in the destination program data files.

## OLE Automation

'OLE Automation' was developed to permit the (remote) control of other applications on the same computer. Applications which expose their functionality using OLE Automation are known as OLE Automation servers, and could be automated by code running in a completely separate application, known as OLE Automation clients or controllers.

OLE Automation servers exposed their functionality through structured object models, which are listings of the internal functions, methods and properties of the application object. All Microsoft Office applications are OLE Automation servers to some extent, and can be subsequently controlled by any OLE Automation compliant controller, using the appropriate syntax to manipulate and control the relevant application object model.

Not all applications that support OLE services support OLE Automation. For example, many products support drag-and-drop, and object linking and embedding, but do not support OLE Automation. Linking and embedding allow the user to access the object, whereas OLE Automation allows one application to control another application, possibly with minimal or no user interaction.

## See Also

[OLE Services](#)

## OLE Automation Objects

VBA supports the referencing and control of OLE Automation objects of external applications, permitting you to use the properties, methods and events of those objects from within Plant SCADA.

To access an OLE Automation object using VBA, you must first declare an object variable in your VBA code, then assign an OLE Automation reference to the variable. See Declaration of OLE Automation objects in VBA, and Assigning references to OLE Automation objects in VBA.

Objects declared in a VBA Sub or Function procedure are local to that procedure, and their lifetime ends along with the end of the procedure. An object declared outside a procedure has modular scope to all procedures within the same VBA file module and lasts for the lifetime of the variable that retains the reference to the object.

All object references must be deleted when they are no longer required, to release the memory they were using. When considering the use of OLE Automation, you should be aware that there are different uses of OLE which have developed over the years and which may be confused with one another.

## See Also

[OLE Services](#)

### Declaration of OLE Automation Objects

VBA objects can only be declared and referenced within VBA file modules. VBA modular objects have modular scope and cannot be referenced (accessed and used) from outside their VBA module (file).

---

**Note:** VBA objects cannot be used directly in Plant SCADA command or expression fields.

Once declared within a VBA module, VBA objects can be referenced and used in any procedure within the same code module. An object declared outside of a procedure has modular scope to all procedures within that same VBA module (file). Objects declared within a Sub or Function procedure have local scope only within that procedure.

The object variable must be declared before it can be assigned an object reference. Object variables are declared by the Dim Statement with the As Object VBA data type using the following syntax:

Dim <VariableName> As Object

Where:

- Dim is the required Variable declaration statement BASIC keyword
- <VariableName> represents the required name of the variable being declared (dimensioned)
- As Object declares the variable as a VBA 'object' data type

---

**Note:** The placeholder shown inside arrow brackets (<placeholder>) should be replaced in any actual code with the value of the item that it describes. The arrow brackets and the word they contain should not be included in the statement, and are shown here only for your information.

For example:

```
' create local variables to store object references
Dim objExcelApp As Object
Dim objWordApp As Object
```

Once declared, you can then assign an OLE Automation reference to the object variable in VBA.

## See Also

[Deleting OLE Automation Objects](#)

[Using OLE Automation Objects](#)

### Assigning References to OLE Automation Objects

An OLE Automation object MUST be defined before it can be used. Once defined (see Declaration of OLE Automation objects in VBA), you assign an OLE Automation reference to the object variable in VBA using the VBA

CreateObject function within a VBA Set statement in the following syntax:

```
Set <objVarName> = CreateObject(<objClassName>)
```

Where:

- Set is the required reference assignment statement keyword
- <objVarName> represents the required name of the variable receiving the reference
- CreateObject() function creates the object of the class type specified in the argument
- <objClassName> represents the required name of the class providing the object

The object class name passed as the argument to the CreateObject function usually consists of the fully qualified class name of the object being created, for example "Word.Application" or "Excel.Application".

## Example

```
' create variable to store object reference
Dim objExcelApp As Object
' create the app object and assign the reference
Set objExcelApp = CreateObject("Excel.Application")
' or
' create variable to store object reference
Dim objWordApp As Object
' create the app object and assign the reference
Set objWordApp = CreateObject("Word.Application")
```

Once assigned, you can then use that object variable in your VBA code to manipulate the referenced object model. See [Using OLE Automation Objects](#).

Dependant objects (which cannot be created independantly) can be "drilled-down" to and subsequently assigned from existing (externally creatable) independant object s, by using a method of the higher level object. See Understanding object models in OLE Automation.

For examples of independant objects, Microsoft Excel provides the "Excel.Application", "Excel.Sheet ", and "Excel.Chart" externally creatable objects amongst others, (two of which are demonstrated in OLE Automation example using the Microsoft Excel object), and Microsoft Word provides the "Word.Application", "Word.Document", and "Word.Picture" externally creatable objects amongst others (and is demonstrated in OLE Automation example using the Microsoft Word object).

## See Also

[OLE Automation Objects](#)

### Using OLE Automation Objects

The trick with successfully using OLE Automation is determining what you can and can't do with it. In theory, you can do anything the OLE Automation server application can do. However, in practice, not every OLE Automation server application exposes all of its functionality through its OLE Automation interface.

You have to be able to use the native programming language of the OLE Automation server application in your code. You also need to know about the limitations imposed by the Plant SCADA operating environment, and its implementation of the VBA programming language.

VBA does not support early binding of OLE Automation objects, as there is no mechanism for providing a reference to the object type library (like you can do in Microsoft Visual Studio) until runtime. So, VBA compile errors can occur with valid VBA code which may work well in other VBA supporting applications. Most ported VBA code will require some modification to compile and perform as expected in VBA. For example, VBA does not support the use of "With" statements concerning properties or methods of an object, yet does support the use of "For Each" statements with objects in a collection.

VBA does not support the use of named arguments using the ":=" named argument operator (colon followed by an equal sign). Nor does it support the use of missing arguments using placeholder commas, however, VBA does support the use of the "empty" keyword in place of missing arguments.

VBA does not support the passing of SCADA variable tags by reference, however, the tag value can be copied to a VBA variable, and it can be passed by value. See [Passing Variables Byref and Byval with VBA](#).

To help manage these considerations, you should know how to access the object model of the OLE Automation server applications. VBA does not support the use of application-defined object types nor intrinsic constants due to late-binding of the object model. VBA supports only 10 data-types, so be aware of the possibility of data being lost due to rounding when converting between different data types. See [Rounding Numbers in VBA](#).

To make full use of the OLE Automation object models, you should make yourself familiar with Object related terms. See [Understanding object models in OLE Automation](#).

## See Also

[OLE Automation Objects](#)

### Accessing the Object Model of OLE Automation Server Applications

During the development stage of your project, to access the object model of any OLE Automation server applications, you must have a copy of the appropriate application program installed on the computer you are developing the OLE Automation controller with.

Equally, during Plant SCADA runtime, there must be a copy of the appropriate application program installed on the computer you are running the OLE Automation controller from. If, for example, you were calling the code which creates the object from say a button on a graphics page on a Plant SCADA Client machine, the appropriate application program must be installed on every Client machine with access to that graphics page, for the code to work (if called) on that Client machine.

All of the Microsoft Office suite of products support the VBA language in some manner, and export an OLE Automation object type library which you can view and use. See [How to view an OLE Automation object type library from a Microsoft Office product](#).

Also, the VB programming IDE within Visual Studio can be referenced to load the appropriate type library as required. See [How to view an OLE Automation object type library in VB](#).

Both these suites provide an object browser which you can use to explore the object models. You use the structure of the object model to access, manipulate and control the OLE Automation object using VBA. See [Understanding object models in OLE Automation](#).

## See Also

[OLE Automation Objects](#)

## Understanding Object Models in OLE Automation

Objects are the fundamental building blocks of OLE Automation, and object models are a roadmap to the object structure. OLE Automation using VBA involves creating and modifying the objects provided by other applications (external to the Plant SCADA application). For instance, every element of Microsoft Word ( documents, tables, paragraphs, bookmarks, fields and so on) can be represented by an object in VBA using OLE automation with the Word object model.

### What are objects and collections?

An object represents an element of the OLE Automation application. A collection is an object that contains several other objects, usually of the same type. For example, all the bookmark objects in a document are contained in a single Bookmarks collection object of the Word application. Using appropriate properties and methods, OLE Automation permits the modification of a single object or an entire collection of objects.

### What is a property?

A property is an attribute of an object or an aspect of its behavior. For example, properties of a Word document include its name, its content, and its save status, as well as whether change tracking is turned on. To change the characteristics of any referenced object, you change the values of its properties.

To set the value of a property, follow the reference to an object with a period, the property name, an equal sign, and the new property value. The following example turns on change tracking in the Word document named "MyDoc.doc."

```
objWordApp.Documents("MyDoc.doc").TrackRevisions = True
```

In this example, Documents refers to the collection of open documents, and the name "MyDoc.doc" identifies a single document in the collection. The TrackRevisions property is set for that single document.

You can also return information about an object by returning the value of one of its properties. The following example returns the name of the active Word document.

```
docName = objWordApp.ActiveDocument.Name
```

In this example, ActiveDocument refers to the document in the active window in Word. The name of that document is assigned to the variable "docName".

---

**Note:** Some properties cannot be set. The Help topic for each property indicates whether you can set that property (read-write), only read the property (read-only), or only write the property (write-only). Also the Object Browser in the Visual Basic Editor displays the read-write status at the bottom of the browser window when the property is selected.

---

### What is a method?

A method is an action that an object can perform. For example, just as a Word document can be printed, the Document object has a PrintOut method. Methods often have arguments that qualify how the action is performed. The following example prints the first three pages of the active Word document.

```
objWordApp.ActiveDocument.PrintOut From:=1, To:=3
```

In most cases, methods are actions and properties are qualities. Using a method causes something to happen to an object, while using a property returns information about the object or it causes a quality about the object to change.

## Returning an object

Most objects return a single object from the collection. For example, the Documents collection contains the currently open Word documents. You use the Documents property of the Application object (the object at the top of the Word object hierarchy) to return the Documents collection.

After you've accessed the collection, you can return a single object by using an index value in parentheses (this is similar to how you work with VBA arrays). The index value can be either a number or a name.

The following example uses the Documents property to access the Document collection. The index number is used to return the first document in the Documents collection. The Close method is then applied to the Document object to close the first document in the Documents collection.

```
objWordApp.Documents(1).Close
```

The following example uses a name (specified as a string) to identify a Document object within the Documents collection.

```
objWordApp.Documents("Sales.doc").Close
```

Collection objects often have methods and properties which you can use to modify the entire collection of objects. The Documents object has a Save method that saves all the documents in the collection. The following example saves the open documents by applying the Save method.

```
objWordApp.Documents.Save
```

The Document object also has a Save method available for saving a single document. The following example saves the document named Report.doc.

```
objWordApp.Documents("Report.doc").Save
```

To return an object that is further down in the Word object hierarchy, you must "drill down" to it by using properties and methods to return objects.

To see how this is done, in Word, open the Visual Basic Editor and click Object Browser on the View menu. Click Application in the Classes list on the left. Then click ActiveDocument from the list of members on the right. The text at bottom of the Object Browser indicates that ActiveDocument is a read-only property that returns a Document object. Click Document at the bottom of the Object Browser; the Document object is automatically selected in the Classes list, and the Members list displays the members of the Document object. Scroll through the list of members until you find Close. Click the Close method. The text at the bottom of the Object Browser window shows the syntax for the method. For more information about the method, press F1 or click the Help button to jump to the Close method Help topic.

Given this information, you can write the following instruction to close the active document.

```
objWordApp.ActiveDocument.Close SaveChanges:=wdSaveChanges
```

The following example maximizes the active document window.

```
objWordApp.ActiveDocument.ActiveWindow.WindowState = wdWindowStateMaximize
```

The ActiveWindow property returns a Window object that represents the active window. The WindowState property is set to the maximize constant (wdWindowStateMaximize).

The following example creates a new document and displays the Save As dialog box so that a name can be provided for the document.

```
objWordApp.Documents.Add.Save
```

The Documents property returns the Documents collection. The Add method creates a new document and returns a Document object. The Save method is then applied to the Document object.

As you can see, you use methods or properties to drill down to an object. That is, you return an object by

applying a method or property to an object above it in the object hierarchy. After you return the object you want, you can apply the methods and control the properties of that object.

## Using the Microsoft Word Object Model

You should use the associated online help documentation that came with the object application to obtain details of the object model. The help is quite easy to use. Each of the classes and collections can be clicked to jump to its page.

In VBA, you must use the full Application object qualifier when referencing the properties and methods of the object. For example, you must use the full syntax "Application.ActiveDocument.PrintOut", instead of "ActiveDocument.PrintOut".

## See Also

[OLE Services](#)

## OLE Automation Example Using the Microsoft Word Object

All commands in Word are directed to the active document, which may be changed in code. It is recommended to use named arguments, as the argument sequences are recorded incorrectly in some documentation, including the type library and what the recorder writes to macros.

```
Sub runWord()
    ' demonstrating the use of OLE Automation
    ' to manipulate Word

    ' create local variables
    Dim objWordApp As Object
    ' create the app object and assign the reference
    Set objWordApp = CreateObject("Word.Application")

    ' manipulate the app object
    ' insert appropriate VBA code here to manipulate Word object

    ' close Word
    objWordApp.Quit

    ' delete the object
    Set objWordApp= Nothing

End Sub
```

## See Also

[Using OLE Automation Objects](#)

## Using the Microsoft Excel Object Model

You should use the associated online help documentation that came with the object application to obtain details of the object model.

## See Also

[Using OLE Automation Objects](#)

### Deleting OLE Automation Objects

All object references must be deleted when they are no longer required, to release the memory they were using.

You delete an OLE Automation reference to the object variable in VBA using the VBA Nothing keyword within a VBA Set statement in the following syntax:

Set <objVarName> = Nothing

Where:

- Set is the required reference assignment/release statement keyword.
- <objVarName> represents the required name of the variable holding the reference.
- Nothing is the keyword used to release the object reference.

When several object variables refer to the same object, they also refer to the memory and system resources associated with the object. These resources are released only after all of them have been set to Nothing, either explicitly using Set, or implicitly after the last object variable set to Nothing goes out of scope.

## Example

```
' Word example
' create variable to store object reference
Dim objWord as Object
' create object and assign reference to variable
Set objWord = CreateObject( "Word.Document" )
' insert appropriate VBA code here to manipulate Word object
' release reference
Set objWord = Nothing
' Excel example
' create local variables
Dim objExcelApp As Object
Dim objExcelCht As Object
' create the app object and assign the reference
Set objExcelApp = CreateObject("Excel.Application")
' create a chart and assign the reference
Set objExcelCht = objExcelApp.Charts.Add()
' insert appropriate VBA code here to manipulate Excel objects
' delete the objects
Set objExcelApp = Nothing
Set objExcelCht = Nothing
```

## See Also

[Using OLE Automation Objects](#)

## File Input/Output with VBA

VBA supports full sequential and binary file Input/Output (I/O).

Files stored on disk, can contain text (ASCII) characters or binary (ones and zeros) digits. All VBA files that contain VBA code are stored as text files. However, you can use VBA to store and retrieve files in either format, using VBA file I/O functions and statements.

The File I/O functions predefined in VBA are:

ChDir, ChDrive, Close, CurDir, Dir, EOF, FileCopy, FileLen,  
FreeFile, Get #, GetAttr, Input #, Kill, Line Input #, Loc, LOF,  
MkDir, RmDir, Name, Open, Print #, Put, Seek, SetAttr, Write #.

For details of all predefined VBA functions, see [VBA Function Reference](#).

## VBA Function Reference

VBA includes the following function categories.

Array Functions	File I/O Functions
Conditional Statements	Math/Trigonometry Functions
Conversion Functions	Miscellaneous Functions
Declarations	Procedural Statements
Date and Time Functions	String Functions

## Array Functions

VBA array functions are provided to allow you to declare, resize, initialize, populate, and erase arrays and their elements.

The array functions predefined in VBA are:

Dim	Allocates storage for, and declares the data type of, variables and arrays in a module.
Erase	Reinitializes the elements of a fixed array.
Lbound	Returns the smallest available subscript for the dimension of the indicated array.
Option Base	Declares the default lower bound for array subscripts.
ReDim	Used to size or resize a dynamic array that has already been declared using the Dim statement with empty parentheses.
Ubound	Returns the value of the largest usable subscript for

	the specified dimension of an array.
--	--------------------------------------

## Dim

The Dim statement allocates storage for, and declares the data type of, variables and arrays in a module.

The To clause in the array subscript range of a Dim statement provides a more flexible way to control the lower bound of an array. If you don't explicitly set the lower bound with a To clause, the [Option Base](#) setting (if used) comes into affect, or defaults to zero (if not used).

## Syntax

**Dim** *VariableName*[(*Subscripts*)] [As *DataType*]

*VariableName*:

The name of the variable or array being declared (dimensioned).

*Subscripts*:

The optional subscript range (dimensions) for an array in parentheses.

*DataType*:

The optional data type declaration for the variable or array.

## Related Functions

[Const](#) | [ReDim](#) | [Static](#)

## Example

```
Dim bytVar As Byte
Dim binVar As Boolean
Dim strVar As String
Dim intVar As Integer
Dim lngVar As Long
Dim sngVar As Single
Dim dblVar As Double
Dim vntVar As Variant
Dim objVar As Object
Dim dtmVar As Date

Dim daysOfWeek() As String ' declares an array variable to hold strings

Dim monthsOfYear(12) As Date ' declares an array variable to hold 12 strings
Dim users(,) As String ' declares a two dimensional array to hold strings
Dim usernames(5,5) As String ' declares a two dimensional 5 x 5 array to hold strings

Dim MyArray(1 To 10, 5 To 15, 10 To 20) ' declares the three dimensional array
MyArray and specifies the upper and lower bounds of each dimension
```

## Erase

Reinitializes the elements of a fixed array specified in the *ArrayList* parameter.

## Syntax

**Erase**(*ArrayList*)

*ArrayList*:

A comma delimited list of valid variable array names.

## Related Functions

[Dim](#) | [ReDim](#)

## Example

```
Option Base 1
Dim Num(10) As Integer ' Integer array.
Dim StrVarArray(10) As String ' Variable-string array.
Dim StrFixArray(10) As String * 10 ' Fixed-string array.
Dim VarArray(10) As Variant ' Variant array.
Dim DynamicArray() As Integer ' Dynamic array.
ReDim DynamicArray(10) ' Allocate storage space.
Erase Num ' Each element set to 0.
Erase StrVarArray ' Each element set to zero-length string ("").
Erase StrFixArray ' Each element set to 0.
Erase VarArray ' Each element set to Empty.
Erase DynamicArray ' Free memory used by array.
Erase StrVarArray,StrFixArray,VarArray ' Reset three arrays at the same time.
```

## Lbound

Determines the value of the lower bound for the dimension of the array specified in the arguments.

*Lbound* expects the required argument *ArrayName* to be a valid variable array name. The optional argument *ArrayDimension* must be a whole long number indicating which dimension's lower bound is to be returned. Use 1 for the first dimension, 2 for the second, and so on.

## Syntax

**Lbound**(*ArrayName*, *ArrayDimension*)

*ArrayName*:

The name of the array.

*ArrayDimension*:

The dimension of the array for which you want to the lower bound. If *ArrayDimension* is omitted, 1 is assumed.

## Return Value

Returns a number of Long data type.

## Related Functions

[Ubound](#)

## Example

```
Dim Lower
Dim MyArray(1 To 10, 5 To 15, 10 To 20) ' Declare array variables.
Dim AnyArray(10)
Lower = LBound(MyArray, 1) ' Returns 1.
Lower = LBound(MyArray, 2) ' Returns 5.
Lower = LBound(AnyArray) ' Returns 1.
```

## Option Base

Declares the default lower bound for array subscripts.

The Option Base statement is optional. If used, it can appear only once in a VBA file, and must be used before you declare the dimensions of any arrays.

The To clause in the array subscript range of a [Dim](#) statement provides a more flexible way to control the lower bound of an array. If you don't explicitly set the lower bound with a To clause, the Option Base setting (if used) comes into affect, or defaults to zero (if not used).

## Syntax

### Option Base Num

*Num:*

An Integer or expression representing a valid numeric value. The value of the 'number' parameter must be either 0 or 1. The default is 0.

## Related Functions

[Dim | ReDim](#)

## Example

The example below uses the Option Base statement to override the default base array subscript value of 0.

```
' Module level statement
Option Base 1

' Create the array
Dim Arr(20)
' Declare message variables
```

```
Dim Msg As String
Dim NL as String
' Define newline
NL = Chr(10) & Chr(13)
' Create message
Msg = "The lower bound is " & LBound(Arr) & "."
Msg = Msg & NL & "The upper bound is " & UBound(Arr) & "."
' Display message
MsgBox Msg
```

## ReDim

Used to size or resize a dynamic array that has already been declared using the [Dim](#) statement with empty parentheses.

Use the ReDim statement to change the number of elements in an array, but not to change the number of dimensions in an array or the type of the elements in the array.

## Syntax

**ReDim** *VariableName*(*Subscripts*)

*VariableName*:

The name of the variable or array being redimensioned.

*Subscripts*:

An Integer or expression representing a valid To numeric value range when declaring the dimensions of an variable array. Up to 60 multiple dimensions may be declared.

The subscripts argument uses the following syntax:

[lower To] upper [, [lower To] upper] . . .

When not explicitly stated in **lower**, the **lower** bound of an array is controlled by the [Option Base](#) statement. The lower bound is zero if no Option Base statement is present in the VBA file.

## Related Functions

[Dim](#) | [Const](#) | [Static](#)

## Example

```
Dim TestArray() As Integer
Dim I
ReDim TestArray(10)
For I = 1 To 10
    TestArray(I) = I + 10
    Print TestArray(I)
Next I
```

## Ubound

Determines the value of the largest subscript for the *ArrayDimension* of the *ArrayName* provided in the

argument. Ubound expects the required argument *ArrayName* to be a valid variable array name.

The optional argument *ArrayDimension* must be a whole long number indicating which dimension's lower bound is to be returned. Use 1 for the first dimension, 2 for the second, and so on. If *ArrayDimension* is omitted, 1 is assumed.

## Syntax

**Ubound(*ArrayName*, *ArrayDimension*)**

*ArrayName*:

A string or expression that can represent a valid variable array name.

*ArrayDimension*:

A numeric value or expression that can represent a valid long data type value.

## Return Value

Returns a number of Long data type.

## Related Functions

[Lbound](#)

## Example

```
Dim Upper
Dim MyArray(1 To 10, 5 To 15, 10 To 20) ' Declare array variables.
Dim AnyArray(10)
Upper = UBound(MyArray, 1) ' Returns 10.
Upper = UBound(MyArray, 3) ' Returns 20.
Upper = UBound(AnyArray) ' Returns 10.
```

## Conditional Statements

<a href="#">Do Loop</a>	Allows you to execute a block of statements an indefinite number of times.
<a href="#">End Function</a>	Ends a block of statements such as a Sub procedure or function.
<a href="#">Exit</a>	Exits a loop or procedure.
<a href="#">For</a>	Repeats its block of statements a set number of times as determined by the values used with the To clause.
<a href="#">GoTo</a>	Branches unconditionally and without return to the label specified in the GoTo statement.

If	Tests an initial condition and then either performs or omits to perform the statements it contains, dependant upon the logical result of the test condition.
OnError Statement	VBA's error-handling routine and specifies the line label of the error-handling routine.
Select	Tests the same variable for many different conditions.
Stop	Ends execution of the program.
While...Wend	Similar to the Do While loop statement.
With	Not supported in VBA.

## Do Loop

The Do...Loop conditional statement allows you to execute a block of statements an indefinite number of times. The variations of the Do...Loop are Do...While, Do...Until, Do...Loop While, and Do...Loop Until.

```
Do While <condition>
    <statement/s>
Loop
Do Until <condition>
    <statement/s>
Loop
Do
    <statement/s>
Loop While <condition>
Do
    <statement/s>
Loop Until <condition>
```

Do...While and Do...Until check the condition before entering the loop, thus the block of statements inside the loop are only executed when those conditions are met. Do...Loop While and Do...Loop Until check the condition after having executed the block of statements so that the block of statements is executed at least once.

Any Do statement can be exited using the [Exit](#) statement.

## End Function

The End Function statement ends a program or a block of statements within a function. A VBA function starts with the FUNCTION statement and finishes with the END FUNCTION statement. All other statements that lie between the FUNCTION and END FUNCTION statements will be executed by the function when called to do so.

## Syntax

**End {Function | Sub | If}**

## Related Functions

[Call](#) | [Sub](#) | [End Sub](#) | [Exit](#)

## Example

```
Function GetColor2( c% ) As Long
    GetColor2 = c% * 25
    If c% > 2 Then
        GetColor2 = 255 ' 0x0000FF - Red
    End If
    If c% > 5 Then
        GetColor2 = 65280 ' 0x00FF00 - Green
    End If
    If c% > 8 Then
        GetColor2 = 16711680 ' 0xFF0000 - Blue
    End If
End Function
Sub TestColor2
    Dim I as integer
    For I = 1 to 10
        Print GetColor2(I)
    Next I
End Sub
```

## Exit

Exits a loop or procedure.

## Syntax

**Exit {Do | For | Function | Sub}**

## Example

```
' This sample shows Do ... Loop with Exit Do to get out.
Dim Value, Msg ' Declare variables
Do
    Value = InputBox("Enter a value from 5 to 10.")
    If Value >= 5 And Value <= 10 Then ' Check range
        Exit Do ' Exit Do...Loop
    Else
        Beep ' Beep if not in range
    End If
Loop
```

## For

Repeats its block of statements a set number of times as determined by the values used with the To clause.

## Example

```
For <CounterName> = <BeginValue> To <EndValue> [Step <StepValue>]  
    <statement/s>  
Next
```

## GoTo

The GoTo conditional statement branches unconditionally and without return to the label specified in the GoTo statement. The label must be located in the same subroutine or function as the GoTo statement.

## Example

```
<statement/s>  
If <condition> then  
    GoTo Label1  
Else  
    GoTo Label2  
End If  
  
Label1:  
<statement/s>  
GoTo Label3  
  
Label2:  
<statement/s>  
GoTo Label3  
  
Label3:  
<statement/s>
```

In this example, VBA tests the If condition, and jumps to the part of the script that begins with the label "Label1:" if the condition was true, or jumps to the part of the script that begins with the label "Label2:" if the condition was false. This could be anywhere in the same subroutine or function.

## If

Tests an initial condition and then either performs or omits to perform the statements it contains, dependant upon the logical result of the test condition. The condition can be a comparison or an expression, and must logically evaluate to either True or False. The If statement has both single line and multiple line syntax structure.

The single line syntax uses the If <TestCondition> Then <StatementToPerformIfTrue> structure, however, can only perform a single statement if and only if the test condition result is True. No 'End If' statement is required:

## Example

If<Condition>Then<Statement>

If the result of the If test condition was True, the program flow continues into and performs the statement following the Then statement, until it reaches the end of the line.

To perform a single statement conditionally upon a False result, use the NOT logical operator:

```
If NOT <Condition> Then <Statement>
```

To perform multiple statements, use the multiple line syntax structure which ends with the 'End If' statement:

```
If <Condition> Then
    ' Then statement block
    ' perform only if true
    <Statement/s>
End If
```

If the result of the If test condition was True, the program flow continues into the Then statement block, and performs the statements following the Then statement, until it reaches the End If statement.

If the result of the If test condition was False, the program flow jumps over the Then statement block, which in this case exits the If structure (without performing any statements other than the initial test condition).

The mutiple line If structure can perform different blocks of statements dependant upon EITHER a True OR a False result to the test condition, through the use of the Else statement block:

```
If <Condition> Then
    ' Then statement block
    ' perform only if true
    <Statement/s>
Else
    ' Else statement block
    ' perform only if false
    <Statement/s>
End If
```

If the result of the If test condition was True, the program flow performs the Then block statements, until it reaches the Else statement. It then jumps over the Else statement block and exits the If structure (without performing any of the Else statement block statements).

Further test conditions can be placed into an If structure through the use of the optional Else If <Condition> statement block. Elseif statement blocks can only be positioned within an If structure before the Else statement block. If the result of the If test condition was False, the program flow jumps over the Then statement block (without performing any of those statements) to the Else statement to perform the statements in the Else statement block until it reaches the End If statement.

```
If <Condition> Then
    ' Then statement block
    ' perform only if true
    <Statement/s>
ElseIf <Condition>
    ' Else If statement block
    ' perform only if true
    <Statement/s>
Else
    ' Else statement block
    ' perform only if false
    <Statement/s>
End If
```

The Elseif test condition is only evaluated after the initial If structure test condition results in False.

If the result of the Elseif test condition was True, the statements within the Elseif statement block are performed. The program flow then jumps over the Else statement block and exits the If structure (without performing any of the Else statement block statements).

If the result of the Elseif test condition was False, the program flow jumps over the Elseif statement block (without performing any of those statements) to the Else statement to perform the statements in the Else

statement block until it reaches the End If statement.

There is no apparent limit to the number of Else If statement blocks that any one If structure can hold, however, the Select Case Statement structure handles multiple condition result alternatives much more efficiently.

## OnError

VBA's error-handling routine and specifies the line label of the error-handling routine. The line parameter refers to a label. That label needs to be present in the code or an error is generated.

## Syntax

```
On Error { GoTo line | Resume Next | GoTo 0}
```

## Example

```
On Error GoTo errHandler
Dim x as object
x.draw ' Object not set
..
Exit Sub
errHandler:
Print Err.Number, Err.Description
Resume Next
```

## Select

The Select Case statement tests the same variable for many different conditions. The test value provided with the initial Select Case statement is logically tested against the Case test condition.

The Select Case structure can perform different blocks of statements dependant upon whichever Case statement test condition (if more than one) first results as True, through the use of the Case statement block:

```
Select Case <TestValue>
Case <Condition>
    ' Case statement block
    ' perform only if case true
    <Statement/s>
Case Else
    ' Else statement block
    ' perform only if all cases false
    <Statement/s>
End Select
```

If the result of the Case test condition was True, the program flow performs the statements contained within that Case statement block, and will then exit the Select Case structure (without performing any of the Else statement block statements).

If the result of the Case test condition was False, the program flow jumps over the Case statement block (without performing any of those statements) to the Case Else statement to perform the statements in the Else statement block until it reaches the End Select statement.

Further test conditions can be placed into a Select Case structure through the optional use of further Case statement blocks. Case statement blocks can only be positioned within a Select Case structure before the Case

Else statement block.

```
Select Case <TestValue>
Case <Condition>
    ' Case statement block
    ' perform only if case true
    <Statement/s>
Case <Condition>
    ' Case statement block
    ' perform only if case true
    <Statement/s>
Case Else
    ' Else statement block
    ' perform only if all cases false
    <Statement/s>
End Select
```

Each Case statement block is evaluated in order until the test condition of one results as True. The program flow performs the statements contained within that Case statement block, and will then exit the Select Case structure (without performing any other statements).

The statements of ONLY one Case statement block are ever performed, unless all result in False and there is no Case Else block declared, in which case no Case statement blocks are performed at all.

The following example may help clarify the logic testing being performed in a Select Case structure. Let's say that we have a variable named (intDayOfWeek) containing an integer (ranging from 1 to 7) representing the day of the week, and we wished to display that value as a string (named strDayOfWeek) containing the name of the day of the week, assuming in this example, that Sunday is the first day of the week (1). The Select Case structure would look like this:

```
Dim strDayOfWeek As String
Select Case intDayOfWeek
Case = 1
    StrDayOfWeek = "Sunday"
Case = 2
    StrDayOfWeek = "Monday"
Case = 3
    StrDayOfWeek = "Tuesday"
Case = 4
    StrDayOfWeek = "Wednesday"
Case = 5
    StrDayOfWeek = "Thursday"
Case = 6
    StrDayOfWeek = "Friday"
Case = 7
    StrDayOfWeek = "Saturday"
Case Else
    StrDayOfWeek = "Invalid"
End Select
```

The Select Case structure tends to be easier to read, understand, and follow and should be used in place of a complicated multi-nested If...ElseIf structure.

## Stop

Ends execution of the program. The Stop statement can be placed anywhere in your code.

## Example

```
Dim x,y,z
For x = 1 to 5
    For y = 1 to 5
        For z = 1 to 5
            Print "Looping",z,y,x
        Next z
    Next y
    Stop
Next x
```

## While...Wend

The While...Wend loop conditional statement is similar to the Do While loop statement. The condition is checked before executing the block of statements comprising the loop.

## Example

```
While <condition>
    <statement/s>
Wend
```

## With

---

**Note:** The With statement is not supported in VBA.

---

When performing a series of commands on an object, you must explicitly refer to the name of the object with each command.

## Conversion Functions

VBA conversion functions are provided to assist with data manipulation and calculation in your formulas. Conversion functions can be used in VBA statements, and will (like all other functions), return a value to the caller.

### ASCII Character Code Conversion

Plant SCADA uses the following character code conversion functions:

Asc	Returns the numeric ASCII value of a string.
Chr	Returns the string ASCII value of a number.

#### Asc

Converts a text string character to its numeric ASCII code value. The Asc function expects the argument *Str* to be a valid string expression. If *Str* contains no characters, a runtime error occurs. The Asc function performs the

opposite of the [Chr](#) function, which converts a number into its string character ASCII code value.

## Syntax

**Asc(*Str*)**

*Str*:

A string or expression that can represent a valid text value.

## Return Value

Returns the numeric ASCII code value of the first character in *Str* provided in the argument.

## Related Functions

[Chr](#)

## Example

```
Dim vntVar ' declare result holder variable
vntVar = Asc("A")' returns 65
vntVar = Asc("Z")' returns 90
vntVar = Asc("a")' returns 97
vntVar = Asc("z")' returns 122
vntVar = Asc("Apple")' returns 65
vntVar = Asc("Zoe")' returns 90
```

## Chr

Converts a number into its string character ASCII code value.

The Chr function expects the argument *Num* to be a valid numeric integer (whole positive number within the range 0 to 255 inclusive). If *Chr* contains no number, a runtime error occurs.

**Note:** Values 8, 9, 10, and 13 convert to backspace, tab, linefeed, and carriage return characters respectively.

The Chr function performs the opposite of the [Asc](#) function, which converts a text string character to its numeric ASCII code value.

## Syntax

**Chr(*Num*)**

*Num*:

An integer or expression representing a valid numeric value.

## Return Value

Returns a single character string representing the ASCII character code value of the number *Num* provided in the argument.

## Related Functions

[Asc](#)

## Example

```
Dim vntVar ' declare result holder variable
vntVar = Chr(65) ' returns "A"
vntVar = Chr(97) ' returns "a"
vntVar = Chr(90) ' returns "Z"
vntVar = Chr(122) ' returns "z"
```

## Date Conversion

Plant SCADA uses the following date conversion functions:

<a href="#">CDate</a>	Converts an expression to a variant of date data type.
<a href="#">CDbl</a>	Converts an expression to a double data type.
<a href="#">Clnt</a>	Converts an expression to a integer data type.
<a href="#">CLng</a>	Converts an expression to a long data type.
<a href="#">CSng</a>	Converts an expression to a single data type.
<a href="#">CStr</a>	Converts an expression to a string data type.
<a href="#">CVar</a>	Converts an expression to a variant data type.

### CDate

Converts any valid date expression to a Date data type.

The CDate function expects the argument Date to be a date expression (limited to numbers or strings in any combination) that can represent a date from January 1, 100 through December 31, 9999.

## Syntax

**CDate(Date)**

*Date:*

A string or expression that can represent a date value. This includes any combination of date literals, numbers that look like dates, strings that look like dates, and dates from functions.

## Return Value

Returns the value of the expression Date provided in the argument as a variant with a vartype of 7 (date data type).

## Related Functions

[CDbl](#) | [CInt](#) | [CLng](#) | [CSng](#) | [CStr](#) | [CVar](#)

## Example

```
Dim MybDate, MDate, MTime, MSTime
' Define date.
MybDate = "May 29, 1959"
' Convert to Date data type.
MDate = CDate(MybDate)
' Define time.
MTime = "10:32:27 PM"
' Convert to Date data type.
MSTime = CDate(MTime)
```

## CDbl

Converts expressions to a double data type.

## Syntax

**CDbl(***Exp***)**

*Exp*:

A valid string, number or Variant containing a value recognizable as a string or a number.

## Return Value

Returns the value of the expression *Exp* provided in the argument as a double data type.

## Related Functions

[CDate](#) | [CInt](#) | [CLng](#) | [CSng](#) | [CStr](#) | [CVar](#)

## Example

```
Dim x as integer
Dim z as double
z = CDbl(x)'Converts the integer value of x to a double value in z
```

## CInt

Converts expressions to an integer data type.

## Syntax

**CInt(Exp)**

*Exp:*

A valid string, number or Variant containing a value recognizable as a string or number.

## Return Value

Returns the value of the expression *Exp* provided in the argument as an integer data type.

## Related Functions

[CDate](#) | [CDbl](#) | [CLng](#) | [CSng](#) | [CStr](#) | [CVar](#)

## Example

```
Dim x as integer
Dim y as long
x = CInt(y) 'Converts the long value of y to an integer value in x
```

## CLng

Converts expressions to a long data type.

## Syntax

**CLng(Exp)**

*Exp:*

A valid string, number or Variant containing a value recognizable as a string or number.

## Return Value

Returns the value of the expression *Exp* provided in the argument as a long data type.

## Related Functions

[CDate](#) | [CDbl](#) | [CInt](#) | [CSng](#) | [CStr](#) | [CVar](#)

## Example

```
Dim x as integer
Dim y as long
y = CLng(x) 'Converts the integer value of x to an long value in y
```

## CSng

Converts expressions to a single data type.

### Syntax

**CSng(*Exp*)**

*Exp*:

A valid string, number or Variant containing a value recognizable as a string or number.

### Return Value

Returns the value of the expression *Exp* provided in the argument as a single data type.

### Related Functions

[CDate](#) | [CDbl](#) | [CInt](#) | [CLng](#) | [CStr](#) | [CVar](#)

## Example

```
Dim x as integer
Dim s as single
s = CSng(x) 'Converts the integer value of x to a single value in s
```

## CStr

Converts expressions to a string data type.

### Syntax

**CStr(*Exp*)**

*Exp*:

A valid string, number or Variant containing a value recognizable as a string or number.

### Return Value

Returns the value of the expression *Exp* provided in the argument as a string data type.

### Related Functions

[CDate](#) | [CDbl](#) | [CInt](#) | [CLng](#) | [CStr](#) | [CVar](#) | [CSng](#)

## Example

```
Dim x as integer
```

```
Dim t as string
t = CStr(x) 'Converts the integer value of x to a string value in t
```

## CVar

Converts expressions to a variant data type.

## Syntax

**CVar(Exp)**

*Exp:*

A valid string, number or Variant containing a value recognizable as a string or number.

## Return Value

Returns the value of the expression (*Exp*) provided in the argument as a variant data type.

## Related Functions

[CDate](#) | [CDbl](#) | [CInt](#) | [CLng](#) | [CSng](#) | [CStr](#)

## Example

```
Dim x as integer
Dim v as variant
v = CVar(x) 'Converts the integer value of x to a variant value in v
```

## Date and Time Formatting/Conversion

Plant SCADA uses the following formatting/conversion functions:

<a href="#">DateSerial</a>	Constructs a date value.
<a href="#">TimeSerial</a>	Constructs an time value.

## DateSerial

Constructs a date value from the given Year, Month, and Day arguments passed to the function. The DateSerial function expects all three parameters to be valid. Date values in VBA are evaluated using the Gregorian Calendar.

## Syntax

**DateSerial(year,month,day)**

*year, month, day:*

The year, month and day as integers.

## Return Value

Returns a Variant (of date data type) containing a date value corresponding to the Year, Month and Day values that were passed in to the function.

## Related Functions

[TimeSerial](#)

## Example

```
Dim varMyBDate
varMyBDate = DateSerial(1958, 7, 08)
' constructs and stores date value.
```

## TimeSerial

Constructs a time value serially from the given Hrs, Mins, and Secs arguments passed to the function. The TimeSerial Function expects all three arguments to be valid.

## Syntax

**TimeSerial(*hours,minutes,seconds*)**

*hours, minutes, seconds:*

The hours, minutes and seconds to be converted to serial form as integers.

## Return Value

Returns a Variant (of date data type) containing a time value corresponding to the Hrs, Mins, and Secs values that were passed in to the function.

## Related Functions

[DateSerial](#)

## Example

```
Dim varMyTime
varMyTime = TimeSerial(14, 35, 17)
' stores time as 2:35:17 PM
```

## Number and String Conversion

Plant SCADA uses the following functions for converting and formatting numbers and strings:

<a href="#">Format</a>	Formats a string, number, or variant to the format expression (fmt ).
<a href="#">Hex</a>	Converts a value to a string representing the hex value.
<a href="#">Oct</a>	Converts a value to a string representing the octal value.
<a href="#">Str</a>	Converts a value to a string containing numeric characters.
<a href="#">Val</a>	Converts a string containing numeric characters to a numeric value.

## Format

Formats a string, number, or variant to the format expression *fmt*. The Format function expects the argument *Exp* to be a valid expression to be formatted.

The Format function expects the argument *fmt* to be a string of characters that specify how the expression is to be displayed, or the name of a commonly used format that has been predefined in VBA. Do not mix different type format expressions in a single *fmt* parameter.

If the *fmt* parameter is omitted or is zero-length and the expression parameter is a numeric, Format[\$] provides the same functionality as the Str[\$] function by converting the numeric value to the appropriate return data type. Positive numbers convert to strings using Format[\$] lack the leading space reserved for displaying the sign of the value, whereas those converted using Str[\$] retain the leading space.

To format numbers, you can use the commonly used formats that have been predefined in VBA, or you can create user-defined formats with standard characters that have special meaning when used in a format expression.

## Syntax

**Format(*Exp* [,*fmt*])**

*Exp*:

A valid string, number or Variant containing a value recognizable as a string or number.

## Return Value

Returns a formatted string.

### Predefined numeric format names

- *General* - Display the number as is, with no thousand Separators
- *Fixed* - Display at least one digit to the left and two digits to the right of the decimal separator.
- *Standard* - Display number with thousand separator, if appropriate; display two digits to the right of the decimal separator.

- *Percent* - Display number multiplied by 100 with a percent sign (%) appended to the right' display two digits to the right of the decimal separator.
- *Scientific* - Use standard scientific notation.
- *True/False* - Display False if number is 0, otherwise display True.

### User-defined number formats

- *Null string* - Display the number with no formatting.
- *0* - Digit placeholder.

### Display a digit or a zero

If the number being formatted has fewer digits than there are zeros (on either side of the decimal) in the format expression, leading or trailing zeros are displayed. If the number has more digits to the right of the decimal separator than there are zeros to the right of the decimal separator in the format expression, the number is rounded to as many decimal places as there are zeros. If the number has more digits to left of the decimal separator than there are zeros to the left of the decimal separator in the format expression, the extra digits are displayed without modification.

- **Digit placeholder(#)**. Displays a digit or nothing. If there is a digit in the expression being formatted in the position where the # appears in the format string, displays it; otherwise, nothing is displayed.
- **Decimal placeholder(.)**. The decimal placeholder determines how many digits are displayed to the left and right of the decimal separator.
- **Percentage placeholder.(%)** The percent character (%) is inserted in the position where it appears in the format string. The expression is multiplied by 100.
- **Thousand separator(,)**. The thousand separator separates thousands from hundreds within a number that has four or more places to the left of the decimal separator. Use of this separator as specified in the format statement contains a comma surrounded by digit placeholders(0 or #). Two adjacent commas or a comma immediately to the left of the decimal separator (whether or not a decimal is specified) means "scale the number by dividing it by 1000, rounding as needed."
- **Scientific format(E-E+e-e+)**. If the format expression contains at least one digit placeholder (0 or #) to the right of E-,E+,e- or e+, the number is displayed in scientific formatted E or e inserted between the number and its exponent. The number of digit placeholders to the right determines the number of digits in the exponent. Use E- or e- to place a minus sign next to negative exponents. Use E+ or e+ to place a plus sign next to positive exponents.
- **Time separator(:)**. The actual character used as the time separator depends on the Time Format specified in the International section of the Control Panel.
- **Date separator(/)**. The actual character used as the date separator in the formatted out depends on Date Format specified in the International section of the Control Panel.
- Display a literal character (- + \$ ( )). To display a character other than one of those listed, precede it with a backslash (\).
- Display the next character in the format string (\). The backslash itself isn't displayed. To display a backslash, use two backslashes (\\\).

---

**Note:** Examples of characters that can't be displayed as literal characters are the date- and time- formatting characters (a,c,d,h,m,n,p,q,s,t,w,y, and /:), the numeric formatting characters(#,0,%,E,e,comma, and period), and the string- formatting characters (@,&,<,>, and !).

---

- Display the string inside the double quotation marks ("String"). To include a string in fmt from within VBA, you need to use the ANSI code for a double quotation mark Chr(34) to enclose the text.
- Display the next character as the fill character (\*). Any empty space in a field is filled with the character following the asterisk.

Unless the *fmt* argument contains one of the predefined formats, a format expression for numbers can have from one to four sections separated by semicolons.

If you use	The result is
One section	The format expression applies to all values.
Two	The first section applies to positive values, the second to negative sections values.
Three	The first section applies to positive values, the second to negative sections values, and the third to zeros.
Four	The first section applies to positive values, the second to negative section values, the third to zeros, and the fourth to Null values.

The following example has two sections: the first defines the format for positive values and zeros; the second section defines the format for negative values.

"\$#,##0; (\$#,##0)"

If you include semicolons with nothing between them, the missing section is printed using the format of the positive value. For example, the following format displays positive and negative values using the format in the first section and displays "Zero" if the value is zero.

"\$#,##0; ;\Z\ e\r\o"

Some sample format expressions for numbers are shown below. (These examples assume the Country is set to United States in the International section of the Control Panel.) The first column contains the format strings. The other columns contain the output the results if the formatted data has the value given in the column headings.

Format (fmt)	Positive 3	Negative 3	Decimal .3	Null
Null string	3	-3	0.3	
0	3	-3	1	
0.00	3.00	-3.00	0.30	
#,##0	3	-3	1	
#,##0.00;;Nil	3.00	-3.00	0.30	Nil
\$#,##0;(\$#,##0)	\$3	(\$3)	\$1	
\$#,##0.00;(\$#,##0.0)	\$3.00	(\$3.00)	\$0.30	

<b>Format (fmt)</b>	<b>Positive 3</b>	<b>Negative 3</b>	<b>Decimal .3</b>	<b>Null</b>
0%	300%	-300%	30%	
0.00%	300.00%	-300.00%	30.00%	
0.00E+00	3.00E+00	-3.00E+00	3.00E-01	
0.00E-00	3.00E00	-3.00E00	3.00E-01	

Numbers can also be used to represent date and time information. You can format date and time serial numbers using date and time formats or number formats because date/time serial numbers are stored as floating-point values.

To format dates and times, you can use either the commonly used format that have been predefined or create user-defined time formats using standard meaning of each:

General	Display a date and/or time. for real numbers, display a date and time.(e.g. 4/3/93 03:34 PM); If there is no fractional part, display only a date (e.g. 4/3/93); if there is no integer part, display time only (e.g. 03:34 PM).
Long Date	Display a Long Date, as defined in the International section of the Control Panel.
Medium	Display a date in the same form as the Short Date, as defined in the international section of the Control Panel, except spell out the month abbreviation.
Short Date	Display a Short Date, as defined in the International section of the Control Panel.
Long Time	Display a Long Time, as defined in the International section of the Control panel. Long Time includes hours, minutes, seconds.
Medium	Display time in 12-hour format using hours and minuets and the Time AM/PM designator.
Short Time	Display a time using the 24-hour format (e.g. 17:45)
c	Display the date as dddd and display the time as tttt. in the order.
d	Display the day as a number without a leading zero (1-31).
dd	Display the day as a number with a leading zero (01-31).
ddd	Display the day as an abbreviation (Sun-Sat).

ddddd	Display a date serial number as a complete date (including day , month, and year).
w	Display the day of the week as a number (1- 7 ).
ww	Display the week of the year as a number (1-53).
m	Display the month as a number without a leading zero (1-12). If m immediately follows h or hh, the minute rather than the month is displayed.
mm	Display the month as a number with a leading zero (01-12). If mm immediately follows h or hh, the minute rather than the month is displayed.
mmm	Display the month as an abbreviation (Jan-Dec).
mmmm	Display the month as a full month name (January-December).
q	display the quarter of the year as a number (1-4).
y	Display the day of the year as a number (1-366).
yy	Display the day of the year as a two-digit number (00-99)
yyyy	Display the day of the year as a four-digit number (100-9999).
h	Display the hour as a number without leading zeros (0-23).
hh	Display the hour as a number with leading zeros (00-23).
n	Display the minute as a number without leading zeros (0-59).
nn	Display the minute as a number with leading zeros (00-59).
s	Display the second as a number without leading zeros (0-59).
ss	Display the second as a number with leading zeros (00-59).
tttt	Display a time serial number as a complete time (including hour, minute, and second) formatted using the time separator defined by the Time Format in the

	International section of the Control Panel. A leading zero is displayed if the Leading Zero option is selected and the time is before 10:00 A.M. or P.M. The default time format is h:mm:ss.
AM/PM	Use the 12-hour clock and display an uppercase AM/PM
am/pm	Use the 12-hour clock display a lowercase am/pm
A/P	Use the 12-hour clock display a uppercase A/P
a/p	Use the 12-hour clock display a lowercase a/p
AMPM	Use the 12-hour clock and display the contents of the 11:59 string(s1159) in the WIN.INI file with any hour before noon; display the contents of the 2359 string (s2359) with any hour between noon and 11:59 PM. AMPM can be either uppercase or lowercase, but the case of the string displayed matches the string as it exists in the WIN.INI file. The default format is AM/PM.
m/d/yy	2/26/65
d-mmmm-yy	26-February-65
d-mmmm	26 February
mmmm-yy	February 65
hh:nn	AM/PM 06:45 PM
h:nn:ss a/p	6:45:15 p
h:nn:ss	18:45:15
m/d/yy/h:nn	2/26/65 18:45

Strings can also be formatted with Format[\$]. A format expression for strings can have one section or two sections separated by a semicolon.

If you use	The result is
One section only	The format applies to all string data.
Two sections	The first section applies to string data, the second to Null values and zero-length strings.

The following characters can be used to create a format expression for strings:

@	Character placeholder. Displays a character or a space.
---	---

	Placeholders are filled from right to left unless there is an !character in the format string.
&	Character placeholder. Display a character or nothing.
<	Force lowercase.
>	Force uppercase.
!	Force placeholders to fill from left to right instead of right to left.

## Related Functions

[Hex](#) | [Oct](#) | [Str](#) | [Val](#)

## Example

```
' Format Function Example
' This example shows various uses of the Format function to format values
' using both named and user-defined formats. For the date separator (/),
' time separator (:), and AM/ PM literal, the actual formatted output
' displayed by your system depends on the locale settings on which the code
' is running. When times and dates are displayed in the development
' environment, the short time and short date formats of the code locale
' are used. When displayed by running code, the short time and short date
' formats of the system locale are used, which may differ from the code
' locale. For this example, English/United States is assumed.

' MyTime and MyDate are displayed in the development environment using
' current system short time and short date settings.
MyTime = "08:04:23 PM"
MyDate = "03/03/95"
MyDate = "January 27, 1993"
MsgBox Now
MsgBox MyTime
MsgBox Second( MyTime ) & " Seconds"
MsgBox Minute( MyTime ) & " Minutes"
MsgBox Hour( MyTime ) & " Hours"
MsgBox Day( MyDate ) & " Days"
MsgBox Month( MyDate ) & " Months"
MsgBox Year( MyDate ) & " Years"
' Returns current system time in the system-defined long time format.
MsgBox Format(Time, "Short Time")
MyStr = Format(Time, "Long Time")
' Returns current system date in the system-defined long date format.
MsgBox Format(Date, "Short Date")
MsgBox Format(Date, "Long Date")
MyStr Format(MyTime, "h:n:s") ' Returns "17:4:23".
MyStr Format(MyTime, "hh:nn:ss")' Returns "20:04:22 ".
MyStr Format(MyDate, "dddd, mmm d yyyy")' Returns "Wednesday, Jan 27 1993".
' If format is not supplied, a string is returned.
MsgBox Format(23) ' Returns "23".
```

```
' User-defined formats.  
MsgBox Format(5459.4, "##,##0.00") ' Returns "5,459.40".  
MsgBox Format(334.9, "####0.00") ' Returns "334.90".  
MsgBox Format(5, "0.00%") ' Returns "500.00%".  
MsgBox Format("HELLO", "<") ' Returns "hello".  
MsgBox Format("This is it", ">") ' Returns "THIS IS IT".
```

## Hex

Converts a numeric value to a text string representing the hexadecimal value of the number.

The Hex function expects the argument *Num* to be a valid numeric value. It is rounded to nearest whole number before evaluation.

## Syntax

**Hex(*Num*)**

*Num*:

An Integer or expression representing a valid numeric value.

## Return Value

Returns a text string containing the hexadecimal value of the numeric *Num* value provided in the argument.

## Related Functions

[Format](#) | [Oct](#) | [Str](#) | [Val](#)

## Example

```
Dim MyHex as String  
MyHex = Hex(5) 'returns "5"  
MyHex = Hex(10) 'returns "A"  
MyHex = Hex(459) 'returns "1CB"
```

## Oct

Converts a numeric value to a text string representing the octal value of the number.

The Oct function expects the argument *Num* to be a valid numeric value. It is rounded to nearest whole number before evaluation.

## Syntax

**Oct(*Num*)**

*Num*:

An Integer or expression representing a valid numeric value.

## Return Value

Returns a text string containing the octal value of the numeric *Num* value provided in the argument.

## Related Functions

[Format](#) | [Hex](#) | [Str](#) | [Val](#)

## Example

```
Dim MyOct as String
MyOct = Oct(4) 'returns "4"
MyOct = Oct(8) 'returns "10"
MyOct = Oct(459) 'returns "713"
```

## Str

Converts a numeric value to a text string containing numeric characters. The Str function expects the argument *Num* to be a valid numeric value.

The Str function is often used to prepare a numerical value for display as a string in a caption, label, string field, or string expression.

The Str function performs the opposite of the Val function, which converts a text string containing numeric characters to a numeric value.

---

**Note:** Please remember the data type coercion considerations with variant data types. See Variants.

---

## Syntax

**Str(*Num*)**

*Num*:

An Integer or expression representing a valid numeric value.

## Return Value

Returns a string containing the numeric character representation of the numeric *Num* value provided in the argument.

The Str function reserves the first return string character for the sign of *Num*. If *Num* is positive, a leading space is used and the plus sign is implied.

## Related Functions

[Format](#) | [Hex](#) | [Oct](#) | [Val](#)

## Example

```
Dim vntVar ' declare result holder variable
```

```
vntVar = Str() ' returns " "
vntVar = Str(65) ' returns " 65"
vntVar = Str(97.578) ' returns " 97.578"
vntVar = Str(-97.578) ' returns "-97.578"
```

## Val

Converts a text string containing numeric characters to a numeric value. The Val function expects the argument *Str* to be a valid string expression. The Val function stops reading the string when it reaches a non numeric character.

Symbols such as dollar signs and commas are not recognized; however, radix prefixes for octal (&O) and hexadecimal (&H) are. Blanks, tabs and line feeds are stripped out from the return.

The Val function performs the opposite of the Str function, which converts a numeric value to a text string containing numeric characters.

## Syntax

**Val(*Str*)**

*Str*:

A string or expression that can represent a valid text value.

## Return Value

Returns the numeric value of a string of characters extracted from the *Str* provided in the argument.

## Related Functions

[Format](#) | [Hex](#) | [Oct](#) | [Str](#)

## Example

```
Dim vntVar ' declare result holder variable
vntVar = Val("65") ' returns 65
vntVar = Val("90 Main St.") ' returns 90
vntVar = Val("12+34+56") ' returns 12
vntVar = Val(" 12 34 56 ") ' returns 123456
vntVar = Val("&0FF") ' returns
vntVar = Val("Zoe") ' returns 0
```

## Declarations

VBA declarations allow you to manipulate and control variables and constants. The Declaration functions and statements predefined in VBA are:

<a href="#">CreateObject</a> function	Creates an OLE Automation object reference
<a href="#">Const</a> statement	Assigns a symbolic name to a constant value.
<a href="#">Declare</a> statement	Declare references to external procedures in a DLL.
<a href="#">Dim</a> statement	Allocates storage for, and declares the data type of, variables and arrays.
<a href="#">IsDate</a>	Determines if a Variant parameter can be converted to a date.
<a href="#">IsEmpty</a>	Determines if a Variant parameter has been initialized.
<a href="#">IsNull</a>	Determines if a Variant contains NULL.
<a href="#">IsNumeric</a>	Determines if a Variant can be converted to a numeric data type.
<a href="#">Nothing</a> keyword	Releases an OLE Automation object reference from a variable of object type.
<a href="#">Option Base</a> statement	Declares the default lower bound for array subscripts.
<a href="#">Option Compare</a> statement	Determines the default string comparison method. Forces explicit declaration of all variables.
<a href="#">ReDim</a> statement	Used to size or resize a dynamic array.
<a href="#">Set</a> statement	Assigns an OLE Automation object reference to a variable of object type.
<a href="#">Static</a> statement	Allocates storage for, and declares the data type of, variables and arrays.
<a href="#">VarType</a>	Indicates the data type used within the Variant.

## CreateObject

Creates a new OLE Automation object and assigns a reference to the object.

## Syntax

Set *objVarName* = CreateObject(*objClassName*)

*objVarName*:

The required name of the variable receiving the reference.

*objClassName*:

The required class name of the object to be created.

The object variable *objVarName* must be declared before it can be set to reference an OLE Automation object.

## Related Functions

[Dim](#) | [Set](#)

## Example

```
' create variable to store object reference
Dim objWord as Object
' create object and assign reference to variable
Set objWord = CreateObject( "Word.Document" )
' insert appropriate VBA code here to manipulate Word object
' release reference
Set objWord = Nothing.
```

## Const

Assigns a symbolic name to a constant value using the following syntax:

```
Const VarName [As DataType] = Expression
```

A constant must be defined before it is used. Unlike variables, constants are assigned values when initialized and retain that same value during the life of the constant.

Constant statements can only be declared and assigned using simple expressions. Constants can NOT be assigned values from variables, user-defined functions, intrinsic VBA functions (such as Chr), or from any expression that involves an operator.

Constants declared in a Sub or Function procedure are local to that procedure. A constant declared outside a procedure has modular scope to all procedures within the same VBA file module. See Scope of VBA.

Constants can be used anywhere in your VBA code where you could use a VBA expression.

If you use Const outside a procedure its scope becomes global.

A type declaration character may also be used. However if none is used, VBA will automatically assign one of the following data types to the constant: long (if it is a long or integer); Double (if a decimal place is present); or String (if it is a string).

## Syntax

**Const**(*VarName*, *Exp*)

*VarName*:

A string representing a valid variable name.

*Exp*:

A valid string, number or Variant containing a value recognizable as a string or number.

## Related Functions

[Dim](#) | [ReDim](#) | [Static](#)

## Example

```
' Correct declaration examples
' long assignment
Const Seven = 7
' double assignment
Const Pi = 3.14159
' string assignment
Const Lab = "Laboratory"

' Incorrect declaration examples
' NOTE that the following declarations demonstrate incorrect
assignments
' because each contains an operator
Const conPi = 4 * Atn(1)
' will cause a VBA compile error
Const conDegToRad = (conPi / 180)
' will cause a VBA compile error
```

## Declare

The Declare statement is used at module (file) level to declare references to external procedures in a dynamic-link library (DLL).

## Syntax

**Declare Function**<FunctionName> Lib "<LibName>" [Alias "<AliasName>"] [(<ArgList>)] [As <ReturnType>]

*FunctionName*:

The required name of the function being declared.

*LibName*:

The required DLL filename containing the function being called.

*AliasName*:

The optional function name within the DLL being called.

*ArgList*:

The optional argument/s of the function.

*ReturnType*:

The optional return data type.

## Related Functions

[Dim](#)

## Example

```
Declare Function GetWinTempPath Lib "kernel32" _
Alias "GetTempPathA" _
(ByVal nBufferLength As Long, _
```

```
ByVal lpBuffer As String) As Long
```

## Dim

The Dim statement allocates storage for, and declares the data type of, variables and arrays in a module.

The To clause in the array subscript range of a Dim statement provides a more flexible way to control the lower bound of an array. If you don't explicitly set the lower bound with a To clause, the [Option Base](#) setting (if used) comes into affect, or defaults to zero (if not used).

## Syntax

**Dim** *VariableName*[(*Subscripts*)] [As *DataType*]

*VariableName*:

The name of the variable or array being declared (dimensioned).

*Subscripts*:

The optional subscript range (dimensions) for an array in parentheses.

*DataType*:

The optional data type declaration for the variable or array.

## Related Functions

[Const](#) | [ReDim](#) | [Static](#)

## Example

```
Dim bytVar As Byte
Dim binVar As Boolean
Dim strVar As String
Dim intVar As Integer
Dim lngVar As Long
Dim sngVar As Single
Dim dblVar As Double
Dim vntVar As Variant
Dim objVar As Object
Dim dtmVar As Date

Dim daysOfWeek() As String ' declares an array variable to hold strings

Dim monthsOfYear(12) As Date ' declares an array variable to hold 12 strings
Dim users(,) As String ' declares a two dimensional array to hold strings
Dim usernames(5,5) As String ' declares a two dimensional 5 x 5 array to hold strings

Dim MyArray(1 To 10, 5 To 15, 10 To 20) ' declares the three dimensional array
MyArray and specifies the upper and lower bounds of each dimension
```

## IsDate

Determines if an expression can be converted to a date.

The required *Date* argument is a Variant containing a date expression or string expression recognizable as a date or time value.

## Syntax

**IsDate(*Date*)**

*Date*:

A string or expression that can represent a date value. This includes any combination of date literals, numbers that look like dates, strings that look like dates, and dates from functions.

## Return Value

Returns a Boolean True or False.

## Related Functions

[IsEmpty](#) | [IsNull](#) | [IsNumeric](#) | [VarType](#)

## Example

```
Dim x As String
Dim MArray As Integer, MCheck
MArray = 345
x = "January 1, 1987"
MCheck = IsDate(MArray)
MChekk = IsDate(x)
MArray1 = CStr(MArray)
MCheck1 = CStr(MCheck)
Print MArray1 & " is a date " & Chr(10) & MCheck
Print x & " is a date" & Chr(10) & MChekk
```

## IsEmpty

Determines if a variant parameter has been initialised.

The required *Exp* argument is a variant containing a numeric or string expression. However, because **IsEmpty** is used to determine if individual variables are initialised, the *Exp* argument is most often a single variable name.

**IsEmpty** returns **True** if the variable is un-initialised, or is explicitly set to **Empty**; otherwise, it returns **False**. **False** is returned if *Exp* contains more than one variable.

**Note:** **IsEmpty** only returns meaningful information for variants.

## Syntax

**IsEmpty(*Exp*)**

*Exp*

A valid string, number or Variant containing a value recognizable as a string or number.

## Related Functions

Returns a Boolean **True** or **False**.

## Related Functions

[IsDate](#) | [IsNull](#) | [IsNumeric](#) | [VarType](#)

## Example

```
Dim x ' Empty
x = 5 ' Not Empty - Long
x = Empty ' Empty
y = x ' Both Empty
MsgBox "x" & " IsEmpty: " & IsEmpty(x)
```

## IsNull

Determines if a Variant contains **Null**.

**IsNull** returns **True** if expression is **Null**; otherwise, **IsNull** returns **False**. If *Exp* consists of more than one variable, **Null** in any constituent variable causes **True** to be returned for the entire expression.

The **Null** value indicates that the Variant contains no valid data. **Null** is not the same as **Empty**, which indicates that a variable has not yet been initialized. It is also not the same as a zero-length string (" "), which is sometimes referred to as a null string.

---

**Note:** Use the **IsNull** function to determine whether *VarName* contains a **Null** value. Expressions that you might expect to evaluate to **True** under some circumstances, such as If *Var* = **Null** and If *Var* <> **Null**, are always **False**. This is because any expression containing a **Null** is itself **Null** and, therefore, **False**.

---

## Syntax

**IsNull**(*Exp*)

*Exp*

A valid string, number or Variant containing a value recognizable as a string or number.

## Return Value

Returns a Boolean **True** or **False**.

## Related Functions

[IsDate](#) | [IsEmpty](#) | [IsNumeric](#) | [VarType](#)

## Example

```
Dim MyVar, MyCheck
MyCheck = IsNull(MyVar) ' Returns False.

MyVar = ""
MyCheck = IsNull(MyVar) ' Returns False.

MyVar = Null
MyCheck = IsNull(MyVar) ' Returns True.
```

## IsNumeric

Determines if a variant can be evaluated as a number.

The required *Exp* argument is a variant containing a numeric expression or string expression that can be evaluated as a number.

IsNumeric returns **False** if *Exp* is a date expression.

## Syntax

**IsNumeric(*Exp*)**

*Exp*

A valid string, number or Variant containing a value recognizable as a string or number.

## Return Value

Returns a Boolean **True** or **False**.

## Related Functions

[IsDate](#) | [IsEmpty](#) | [IsNull](#) | [VarType](#)

## Example

```
Dim TestVar ' Declare variable.
TestVar = InputBox("Please enter a number, letter, or symbol.")
If IsNumeric(TestVar) Then ' Evaluate variable.
    MsgBox "Entered data is numeric." ' Message if number.
Else
    MsgBox "Entered data is not numeric." ' Message if not.
End If
```

## Nothing

Releases an OLE Automation object reference from a variable of object type. The Nothing keyword is used in a Set statement.

In the following declaration syntax example, each placeholder shown inside arrow brackets ( <placeholder> )

should be replaced in any actual code with the value of the item that it describes. The arrow brackets and the word they contain should not be included in the statement, and are shown here only for your information.

## Syntax

**Set** *objVarName* = **Nothing**

*objVarName*:

The required name of the variable receiving the reference.

The **nothing** keyword should be used when you are finished with an object, to clear any variables that reference the object, so the object can be released from memory.

## Related Functions

[CreateObject](#) | [Function](#) | [Set](#)

## Example

```
' create variable to store object reference
Dim objWord as Object
' create object and assign reference to variable
Set objWord = CreateObject( "Word.Document" )
' insert appropriate VBA code here to manipulate Word object
' release reference
Set objWord = Nothing
```

## Option Base

Declares the default lower bound for array subscripts.

The Option Base statement is optional. If used, it can appear only once in a VBA file, and must be used before you declare the dimensions of any arrays.

The To clause in the array subscript range of a **Dim** statement provides a more flexible way to control the lower bound of an array. If you don't explicitly set the lower bound with a To clause, the Option Base setting (if used) comes into affect, or defaults to zero (if not used).

## Syntax

**Option Base** *Num*

*Num*:

An Integer or expression representing a valid numeric value. The value of the 'number' parameter must be either 0 or 1. The default is 0.

## Related Functions

[Dim](#) | [ReDim](#)

## Example

The example below uses the Option Base statement to override the default base array subscript value of 0.

```
' Module level statement
Option Base 1

' Create the array
Dim Arr(20)
' Declare message variables
Dim Msg As String
Dim NL as String
' Define newline
NL = Chr(10) & Chr(13)
' Create message
Msg = "The lower bound is " & LBound(Arr) & "."
Msg = Msg & NL & "The upper bound is " & UBound(Arr) & "."
' Display message
MsgBox Msg
```

## Option Compare

Determines how strings are compared within a VBA module. The optional Option Compare statement if used, must be placed at the top of the VBA file along with any other Option declarations.

If an Option Compare statement is not included, the default text comparison method is Binary.

## Syntax

**Option Compare {Binary | Text}**

## Related Functions

[InStr](#) | [StrComp](#)

## Example

```
Option Compare Binary
Dim vntResult as Variant
vntResult = StrComp("VBA rules!", "vba Rules!")
' returns 1 (strings unequal)
```

## Example

```
Option Compare Text
Dim vntResult as Variant
vntResult = StrComp("VBA rules!", "vba Rules!")
' returns 0 (strings equal)
```

## ReDim

Used to size or resize a dynamic array that has already been declared using the [Dim](#) statement with empty parentheses.

Use the ReDim statement to change the number of elements in an array, but not to change the number of dimensions in an array or the type of the elements in the array.

## Syntax

**ReDim** *VariableName(Subscripts)*

*VariableName*:

The name of the variable or array being redimensioned.

*Subscripts*:

An Integer or expression representing a valid To numeric value range when declaring the dimensions of an variable array. Up to 60 multiple dimensions may be declared.

The subscripts argument uses the following syntax:

[lower To] upper [, [lower To] upper] . . .

When not explicitly stated in **lower**, the **lower** bound of an array is controlled by the [Option Base](#) statement. The lower bound is zero if no Option Base statement is present in the VBA file.

## Related Functions

[Dim](#) | [Const](#) | [Static](#)

## Example

```
Dim TestArray() As Integer
Dim I
ReDim TestArray(10)
For I = 1 To 10
    TestArray(I) = I + 10
    Print TestArray(I)
Next I
```

## Set

Assigns an OLE Automation object reference to a variable of object type.

## Syntax

**Set** *objVarName* = [CreateObject](#)(*objClassName*) | [Nothing](#)

*objVarName*:

The required name of the variable receiving the reference.

*objClassName*:

The required class name of the object to be created.

Use the **Nothing** keyword to release the object reference.

The object variable *objVarName* must be declared before it can be set to reference an OLE Automation object.

## Related Functions

[CreateObject](#) | [Nothing](#)

## Example

```
' create variable to store object reference
Dim objWord as Object
' create object and assign reference to variable
Set objWord = CreateObject( "Word.Document" )
' insert appropriate VBA code here to manipulate Word object
' release reference
Set objWord = Nothing
```

## Static

The Static statement allocates storage for-and declares the data type of-variables and arrays that will retain their values between subsequent references. Static variables are more commonly used within procedures (subroutines and functions), and have local scope.

## Syntax

**Static** *VariableName*[(*Subscripts*)] [As *DataType*]

*VariableName*:

The required name of the variable being declared (dimensioned).

*Subscripts*:

The optional subscript range for an array.

*DataType*:

The optional VBA data type declaration for the variable.

## Related Functions

[Const](#) | [Dim](#) | [ReDim](#)

## Example

```
Static bytVar As Byte
Static binVar As Boolean
Static strVar As String
Static intVar As Integer
Static lngVar As Long
Static sngVar As Single
```

```
Static dblVar As Double
Static vntVar As Variant
Static objVar As Object
Static dtmVar As Date
Static udtVar As <UserDefinedTypeName>
```

## VarType

Determines the data type of a Variant variable.

The required VarName argument is a Variant containing any variable (except user-defined type).

## Syntax

**VarType(VarName)**

*VarName:*

A string representing a valid variable name.

## Return Value

These are the return values:

- 0 - Empty
- 1 - Null
- 2 - Integer
- 3 - Long
- 4 - Single
- 5 - Double
- 6 - Not Applicable
- 7 - Date/Time
- 8 - String

## Related Functions

[IsDate](#) | [IsEmpty](#) | [IsNull](#) | [IsNumeric](#)

## Example

```
Dim IntVar, StrVar, DateVar, MyCheck
' Initialize variables.
IntVar = 459
StrVar = "Hello World"
DateVar = #2/12/69#
MyCheck = VarType(IntVar) ' Returns 2.
MyCheck = VarType(DateVar) ' Returns 7.
MyCheck = VarType(StrVar) ' Returns 8.
```

## Date and Time Functions

VBA date and time functions let you make use of your computer's system time and date.

The date and time functions predefined in VBA are:

<a href="#">Date function</a>	Determines the current system date according to the setting of the computer's system time.
<a href="#">Date Statement</a>	Sets the current system date.
<a href="#">DateSerial function</a>	Constructs a date value.
<a href="#">DateValue function</a>	Calculates a date.
<a href="#">Day function</a>	Calculates the day.
<a href="#">Hour function</a>	Extracts the hours value from an expression (Time ).
<a href="#">Minute function</a>	Extracts the minutes value from an expression (Time ).
<a href="#">Month function</a>	Calculates the month.
<a href="#">Now function</a>	Determines the current date and time according to the setting of the computer's system date and time.
<a href="#">Second function</a>	Extracts the seconds value from an expression (Time ).
<a href="#">Time function</a>	Determines the current time according to the setting of the computer's system time.
<a href="#">Time (statement)</a>	Sets the current system time.
<a href="#">Timer event</a>	Used to track elapsed time.
<a href="#">TimeSerial function</a>	Constructs an time value.
<a href="#">TimeValue function</a>	Calculates a time.
<a href="#">WeekDay function</a>	Calculates the weekday value of a date.
<a href="#">Year function</a>	Calculates the year.

### Date

Gets the current date in string format.

Time/Date functions can only be used with dates between 1980 and 2035. You should check that the date you are using is valid with Cicode similar to the following:

```
IF StrToDate(Arg1) > 0 THEN
...
ELSE
...

```

END

## Syntax

**Date([Format])**

*Format:*

The format required:

2 - Short date format, dd/mm/yy

3 - Long date format, day month year

9 - Extended date format, dd/mm/yyyy

If omitted, the default Format is 2. All of these formats follow the Regional Settings found in the Windows Control Panel.

## Return Value

The current date (in string format).

## Related Functions

[Time](#)

## Example

```
/* If the current system date is 3rd November 1991 and the Windows date format is
dd/mm/yy; */
str = Date();
! Sets str to "3/11/91".
str = Date(2);
! Sets str to "3/11/91".
str = Date(3);
! Sets str to "3rd November 1991".
```

## See Also

[Date and Time Functions](#)

## Date Statement

Sets the current system date.

The DateValue literal is displayed in short date format using the locale settings of your development system. To view the locale settings for your system, in Windows, select Start, Settings, Control Panel, Regional Options, Date. For example: in Australia, the short date format is represented as d/MM/yyyy.

## Syntax

**Date** = *dateVariable*

*dateVariable*:

You must enclose a Date literal within number signs (# #), for example #31/5/1993#.

## Related Functions

[Time \(statement\)](#)

## Example

```
Dim varVBAReleaseDate
varVBAReleaseDate = #01/07/2001#
Date = varVBAReleaseDate
' sets system date to VBA Release Date
```

## DateValue

Calculates a date from the given date argument passed to the function. Date values in VBA are evaluated using the Gregorian Calendar. The DateValue function expects the argument value (*Date*) to be a string or any expression that can represent a date.

## Syntax

**DateValue**(*Date*)

*Date*:

A string or expression that can represent a date value. This includes any combination of date literals, numbers that look like dates, strings that look like dates, and dates from functions.

## Return Value

Returns a variant (of date data type) corresponding to the string date expression that was passed in.

## Related Functions

[TimeValue](#)

## Example

```
Dim varMyBDate
varMyBDate = DateValue("1958/07/08")
' stores date value.
```

## Day

Calculates the day from the given date argument passed to the function using the Gregorian Calendar.

## Syntax

**Day(*Date*)**

*Date*:

A string or expression that can represent a date value. This includes any combination of date literals, numbers that look like dates, strings that look like dates, and dates from functions.

## Return Value

Returns a variant date corresponding to the date expression that was passed in.

## Related Functions

[Date](#) | [Year](#) | [Month](#) | [WeekDay](#)

## Example

```
Dim varMyBDate, varMyDay
varMyBDate = #July 8, 1958#
varMyDay = Day(varMyBDate)
' stores 8 for day value.
```

## Hour

Calculates the hour value from the given time argument passed to the function.

## Syntax

**Hour(*Time*)**

*Time*:

A string or expression that can represent a time value. This includes and combination of time literals, numbers that look like times, strings that look like times, and times from functions.

## Return Value

Returns an integer between 0 and 23 that is the hour of the parameter (*Time*).

## Related Functions

[Minute](#) | [Second](#)

## Example

```
Dim varMyHour, varMyTime
varMyTime = "08:04:23 PM"
varMyHour = Hour(varMyTime)
' stores hours value.
```

## Minute

Calculates the minute value from the given time argument passed to the function.

## Syntax

**Minute(*Time*)**

*Time*:

A string or expression that can represent a time value. This includes any combination of time literals, numbers that look like times, strings that look like times, and times from functions.

## Return Value

Returns an integer between 0 and 59 representing the minute of the parameter (*Time*).

## Related Functions

[Hour](#) | [Second](#)

## Example

```
Dim varMyMin, varMyTime
varMyTime = "08:04:23 PM"
varMyMin = Minute(varMyTime)
' stores minutes value.
```

## Month

Calculates the month from the given date argument passed to the function using the Gregorian Calendar.

## Syntax

**Month(*Date*)**

*Date*:

A string or expression that can represent a date value. This includes any combination of date literals, numbers that look like dates, strings that look like dates, and dates from functions.

## Return Value

Returns an integer between 1 and 12 inclusive, that represents the month of the year.

## Related Functions

[Date](#) | [Year](#) | [WeekDay](#) | [Day](#)

## Example

```
Dim varMyBDate, varMyMonth
varMyBDate = "08 July 1958"
varMyMonth = Month(varMyBDate)
' returns 7 for July
```

## Now

Determines the current date and time according to the setting of the computer's system date and time using the Gregorian Calendar. Unlike other functions, Now does not require trailing parentheses.

## Syntax

**Now()**

## Return Value

The Now function returns a Variant data type containing a date and time value that is stored internally as a double data type.

The number represents a date and time from January 1, 100 through December 31, 9999. Numbers to the left of the decimal point represent the date and numbers to the right represent the time.

## Related Functions

[Date](#) | [Time](#) | [Timer](#)

## Example

```
Dim vntToday
vntToday = Now
' stores current system date and time.
```

## Second

Calculates the second value from the given time argument passed to the function.

## Syntax

**Second**(*Time*)

*Time*:

A string or expression that can represent a time value. This includes and combination of time literals, numbers that look like times, strings that look like times, and times from functions.

## Return Value

Returns an integer that is the second portion of the parameter (*Time*).

## Related Functions

[Hour](#) | [Minute](#)

## Example

```
Dim varMySec, varMyTime
varMyTime = "08:04:23 PM"
varMySec = Second(varMyTime)
' stores seconds value.
```

## Time

Gets the current time in string format.

Time/date functions can only be used with dates from 1980 to 2035. You should check that the date you are using is valid with Cicode similar to the following:

```
IF StrToDate(Arg1) > 0 THEN
...
ELSE
...
END
```

## Syntax

**Time**([*Format*])

*Format*:

The format of the time:

0 - Short time format, hh:mm

1 - Long time format., hh:mm:ss

If omitted, the default *Format* is 0.

## Return Value

The current time (as a string).

## Related Functions

[Date](#)

## Example

```
! If the current time is 10:45:30;  
Variable=Time();  
! Sets Variable to "10:45".  
Variable=Time(0);  
! Sets Variable to "10:45".  
Variable=Time(1);  
! Sets Variable to "10:45:30".
```

## See Also

[Date and Time Functions](#)

## Time (statement)

Sets the system time.

## Syntax

**Time** = *timeVariable*

*timeVariable*:

You must enclose a Time literal within number signs (# #), for example #12:14:00 PM#.

## Related Functions

[Date Statement](#)

## Example

```
' Time statement example  
Dim varMyTime  
' Assign a time.  
varMyTime = #4:35:17 PM#  
' Set system time to variant varMyTime.  
Time = varMyTime
```

## Timer

The Timer event is used to track elapsed time or can be displayed as a stopwatch in a dialog.

## Syntax

**Timer()**

## Return Value

The number of seconds since midnight.

## Related Functions

[Date](#) | [Time](#) | [Now](#)

## Example

```
Dim TS As Single
Dim TE As Single
Dim TEL As Single

TS = Timer
MsgBox "Starting Timer"
TE = Timer
TT = TE - TS
Print TT
```

## TimeValue

Calculates a time. The TimeValue function expects the argument value (*Time*) to be a string or any expression that can represent a time value.

## Syntax

**TimeValue(*Time*)**

*Time*:

A string or expression that can represent a time value. This includes and combination of time literals, numbers that look like times, strings that look like times, and times from functions.

## Return Value

Returns a variant (of date data type) corresponding to the parameter (*Time*).

## Related Functions

[DateValue](#)

## Example

```
Dim varMyTime
varMyTime = TimeValue("2:35:17 PM")
' stores time as 14:35:17
```

## WeekDay

Calculates the weekday value of the given date argument passed to the function. Date values in VBA are evaluated using the Gregorian Calendar.

## Syntax

**WeekDay(Date)**

*Date:*

A string or expression that can represent a date value. This includes any combination of date literals, numbers that look like dates, strings that look like dates, and dates from functions.

## Return Value

Returns an integer between the range of 1-7 inclusive representing the whole number for the weekday:

Return Value	Description
1	Sunday
2	Monday
3	Tuesday
4	Wednesday
5	Thursday
6	Friday
7	Saturday

## Related Functions

[Date](#) | [Year](#) | [Month](#) | [Day](#)

## Example

```
Dim varMyBDate, varMyWeekDay
varMyBDate = #8/07/1958#
varMyWeekDay = WeekDay(varMyBDate)
' returns 3 (Tuesday)
```

## Year

Calculates the year from the given date argument passed to the function. Date values in VBA are evaluated using the Gregorian Calendar.

## Syntax

**Year(*Date*)**

*Date*:

A string or expression that can represent a date value. This includes any combination of date literals, numbers that look like dates, strings that look like dates, and dates from functions.

## Return Value

Returns an integer representing a year 1930-2029 inclusive.

## Related Functions

[Date](#) | [Month](#) | [WeekDay](#) | [Day](#)

## Example

```
Dim varMyBDate, varMyYear
varMyDate = "08/07/58"
varMyYear = Year(varMyBDate)
' returns 1958
```

## File I/O Functions

VBA file Input/Output (I/O) functions are provided to enable read and write disk file functionality.

The file I/O functions predefined in VBA are:

<a href="#">ChDir</a>	Changes the system environment current directory on the specified drive.
<a href="#">ChDrive</a>	Changes the system environment current drive to the specified drive.
<a href="#">Close</a>	Closes the file/s previously opened with the Open statement.
<a href="#">CurDir, CurDir\$</a>	Returns the current system environment path for the specified drive (Drv).
<a href="#">Dir</a>	Returns a file or directory name that matches the given File and Attrib arguments.

<a href="#">EOF</a>	Returns a boolean True or False value during file access that indicates whether the current position of an open file has reached the end of the file.
<a href="#">FileCopy</a>	Copies a file from Src to Dest.
<a href="#">FileLen</a>	Determines the byte length of a file.
<a href="#">FreeFile</a>	Retrieves the next sequential system file number available for association with a file.
<a href="#">Get #</a>	Reads data from a disk file into a variable.
<a href="#">GetAttr</a>	Returns an Integer representing the attribute settings of a file, directory, or volume.
<a href="#">Input</a>	Reads data from a Sequential file and assigns that data to variables. Input function returns characters from a file opened in Input or Binary mode.
<a href="#">Kill</a>	Deletes files from disk.
<a href="#">Line Input #</a>	Reads a single line from an open sequential file and assigns it to a String variable.
<a href="#">Loc</a>	Returns a number indicating the current position within a file opened using the Open statement.
<a href="#">LOF</a>	Returns a number indicating the byte length of a sequential file opened using the Open statement.
<a href="#">MkDir</a>	Creates the directory specified in the Path parameter.
<a href="#">Name</a>	Renames the disk file specified in the OldFileName parameter, to the name specified in the NewFileName parameter.
<a href="#">Open</a>	Enables input/output (I/O) to a disk file.
<a href="#">Print (function)</a>	Displays a message in the Plant SCADA Kernel and the Cicode Editor output window.
<a href="#">Print #</a>	Reads data from OutputList and writes that data to a sequential file.
<a href="#">Put #</a>	Writes data from a variable to a disk file.
<a href="#">RmDir</a>	Deletes the directory specified in the Path parameter.
<a href="#">Seek</a>	Sets the current position within a file opened using the Open statement, ready for the next read or write action.

Write #	Writes data to a Sequential file opened in output or append mode and reads that data from a list of variables.
---------	--

## ChDir

ChDir statement changes the system environment current directory on the specified drive.

The parameter Path must be a string or expression that can represent a valid DOS file structure path value. The parameter Dir must be a string or expression that can represent a valid DOS file structure directory name. The Path and Dir parameters together, must be limited to less than 128 characters. The Path drive letter is optional, unless the directory is on another drive. The required Dir parameter must be a valid directory name.

**Note:** The file system keeps track of the current drive, and the current directory of every drive. Use the [CurDir](#), [CurDir\\$](#) statement to determine the current directory. The current drive letter can be extracted from the Left character returned in the CurDir statement.

The ChDir statement changes the current directory but not the current drive. To change the current drive, use the [ChDrive](#) statement.

## Syntax

**ChDir** *Path* *Dir*

*Path:*

A string or expression that can represent a valid DOS file structure path value. This includes a directory name, and may include a relative or static directory or folder structure and drive letter, in the order:

[<driveletter>:] [\<rootdirectoryname>] [\<subdirectory> ... \<subdirectory>\] directoryname

Note that the path can be relative to the current directory. A single period represents the current directory (.), and two periods represent the parent directory of the current directory (..). For example:

chdir .. ' changes to the parent directory of the current directory

chdir ..\test ' changes to the test subdirectory of the parent directory

*Dir:*

A string or expression that can represent a valid DOS file structure directory name. Dir is not case sensitive. Dir is often used with the Path parameter.

## Related Functions

[ChDrive](#) | [CurDir](#), [CurDir\\$](#) | [Dir](#) | [MkDir](#) | [RmDir](#)

## Example

```
Dim strPath as String
strPath = CurDir()' store current path
ChDir "\"' change to root dir on current drive
<statements>' do stuff in root directory
ChDir strPath' change back to previous path
```

## ChDrive

Changes the system environment current drive to the specified drive.

The parameter *Drv* must be a string or expression that can represent a valid DOS file structure drive letter. The *Drv* may be local to the computer, or mapped from anywhere on the network connected to the computer. If *Drv* contains more than one letter, only the first character is used.

---

**Note:** The file system keeps track of the current drive, and the current directory of every drive. Use the [CurDir](#), [CurDir\\$](#) statement to determine the current directory. The current drive letter can be extracted from the Left character returned in the CurDir statement.

The ChDrive statement changes the current drive but not the current directory on any drive. To change the current directory, use the [ChDir](#) statement.

## Syntax

**ChDrive***Drv*

*Drv*:

A string or expression that can represent a valid DOS file structure drive letter. *Drv* is case insensitive and must end with a colon (:). The *Drv* may be local to the computer, or mapped from anywhere on the network connected to the computer. *Drv* is often included as part of the Path parameter.

## Related Functions

[ChDir](#) | [CurDir](#), [CurDir\\$](#) | [Dir](#) | [RmDir](#) | [MkDir](#)

## Example

```
Dim strCurPath as String
strCurPath = CurDir$()' store current path as string
ChDir "\'" change to root directory of current drive
<statements>' do stuff in root directory
ChDrive "C'" change to C drive (if not already there)
<statements>' do stuff in current directory on C drive
ChDrive strCurPath' change back to previous drive
ChDir strCurPath' change back to previous path
```

## Close

Closes the file(s) previously opened with the Open statement.

The optional *FileNumList* parameter can contain one or more valid file associated reference numbers using the following syntax:

[[#]FileNum] [, [#]FileNum] ...

Where *Filenum* is any valid number associated with an open file.

If the Close statement is used without any arguments it closes all open files. When the Close statement is executed, the association of a file with its file number ends.

## Syntax

### **CloseFileNumList**

*FileNumList:*

Must contain one or more valid integer or numeric expression values representing associated file numbers using the following syntax:

[[#]filenumber] [, [#]filenumber] ... where filenumber is any valid number associated with an open file.

## Related Functions

[FileCopy](#) | [FreeFile](#) | [Name](#) | [Open](#)

## Example

```
Dim strFileContents As String
Dim strTemp As String
Open "c:\test.txt" For Input As #1 ' open file.
Do While Not EOF(1) ' loop until end of file
    strTemp = Input(10, #1) ' read next ten characters
    strFileContents = strFileContents & strTemp ' join strings
Loop
Close #1
```

## CurDir, CurDir\$

Both CurDir and CurDir\$ functions return the current system environment path for the specified drive (*Drv*).

The parameter *Drv* must be a string or expression that can represent a valid DOS file structure drive letter. The *Drv* may be local to the computer, or mapped from anywhere on the network connected to the computer. If *Drv* contains more than one letter, only the first character is used.

---

**Note:** The file system keeps track of the current drive, and the current directory of every drive. Use the CurDir statement to determine the current directory. The current drive letter can be extracted from the Left character returned in the CurDir statement.

---

If no *Drv* is specified or if *Drv* is a zero-length string (" "), CurDir functions return the system environment path for the current drive.

## Syntax

### **CurDir(*Drv*)**

*Drv:*

A string or expression that can represent a valid DOS file structure drive letter. *Drv* is case insensitive and must end with a colon (:). The *Drv* may be local to the computer, or mapped from anywhere on the network connected to the computer. *Drv* is often included as part of the Path parameter.

## Return Value

CurDir returns a Variant containing a string; CurDir\$ returns a String.

## Related Functions

[ChDir](#) | [ChDrive](#) | [Dir](#) | [MkDir](#) | [RmDir](#)

## Example

```
Dim vntCurPath As Variant
Dim strCurPath As String
vntCurPath = CurDir() ' store current path as variant
strCurPath = CurDir$() ' store current path as string
```

## Dir

Dir function returns a file or directory name that matches the given *File* and *Attrib* arguments.

- The *File* argument is optional, and represents a string expression that specifies a valid file name, and may include a DOS path structure including directory or folder names, and a drive letter. You must specify *File* the first time you call the Dir function, or an error occurs.
- The *Attrib* argument is optional, and can be a constant or numeric expression whose sum specifies file attribute values. If you specify file attributes in the function call, *File* must be included. If the Volume attribute value (8) is specified, all other attribute values are ignored.

Dir supports the use of multiple-character (\*) and single-character (?) wildcards to specify multiple files.

Dir returns the first file name that matches both *File* and *Attrib*. To get any additional file names that match, call Dir again with no arguments. When no more file names match, Dir returns a zero-length string (""). Once a zero-length string is returned, you must specify argument/s in the next call (to reset the function), or an error occurs.

Calling Dir with any argument will reset the function, and it will treat the call as a new call. Previous arguments passed to the Dir function are overwritten and forgotten (reset). You can reset the function (by supplying arguments in the function call) at any time, even if it has not yet returned every possible argument match result.

Calling Dir with the Directory attribute (16) does not continually return Directory names. You will need to check the attribute value of every return result to determine if the return is a valid directory name. To do so, use the [GetAttr](#) function. Because file names are retrieved in no particular order, you may want to store returned file names in an array and then sort the array.

---

**Note:** The file system keeps track of the current drive, and the current directory of every drive. Use the [CurDir](#), [CurDir\\$](#) statement to determine the current directory. The current drive letter can be extracted from the Left character returned in the CurDir statement.

---

## Syntax

**Dir(*File*, *Attrib*)**

*File*:

A string or expression that can represent a valid file name, and may include a DOS path structure including

directory or folder names, and a drive letter.

*Attrib:*

A number or expression that can represent a sum of the attribute values of a file . This can be a constant or a numeric expression which includes any combination of attribute numeric values, whose sum specifies all relevant attributes of a file. Where:

0 = Normal

1 = Read Only

2 = Hidden

4 = System

8 = Volume

16 = Directory or Folder

32 = Archive

Possible combinations of values can sum to 0, 1, 2, 3, in fact every number from 0 to 64, each representing a unique combination of attribute values. For example, a file attribute value of 6 represents that the file has both System (4) and Hidden (2) attributes set.

## Return Value

Returns a String representing the name of a file, directory, or folder that matches a specified pattern or file attribute, or the volume label of a drive. If *File* is not found, a zero-length string (" ") is returned. If *Attrib* is omitted, all files are returned that match *File*.

## Related Functions

[ChDir](#) | [ChDrive](#) | [CurDir, CurDir\\$](#) | [MkDir](#) | [RmDir](#)

## Example

```
Dim strCurPath As String ' declare string to store current path
Dim strFileName As String ' declare string to store retrieved file name
Dim intFileCount As Integer ' declare integer to keep count of retrieved files
Dim arrFileList() As String ' declare string array to store file names
strCurPath = CurDir$() ' store current path for later restoration
ChDir "\" ' change to root directory of current drive
strFileName = Dir (*.dat) ' retrieve file name with .dat extension
Do ' initialize loop
    If strFileName = "" Then ' check to see if valid filename returned
        exit do ' exit from loop
    Else
        intFileCount = intFileCount + 1 ' increment file counter variable
        arrFileList(intFileCount) = strFileName ' store file name in array Redim
        arrFileList(intFileCount) ' increase array size to count value
        strFileName = Dir() ' retrieve next file name to match original Dir call
    EndIf
Loop Until strFileName = "" ' loop again
ChDir strCurPath ' restore previous current directory
```

## EOF

EOF function returns a Boolean True or False value during file access that indicates whether the current position of an open file has reached the end of the file. The required FileNum argument must contain an Integer representing any valid system file number associated with an open file.

---

**Note:** The file system keeps track of all open files and the current position of access within every file. Every statement or function that accesses the data within a file, alters the current position within that file. The Loc function can be used to determine the current position within an open file.

---

Use the LOF and Loc functions instead of EOF when reading binary files with the Input function, or use Get when using the EOF function.

---

**Note:** An error occurs with files opened for Binary access, when the file is read using the Input function until EOF returns True.

---

## Syntax

**EOF(FileNum)**

*FileNum:*

An Integer or numeric expression representing any valid number in the range 1 to 511 inclusive, which is referenced by the file system to be associated with an open file.

## Return Value

Returns an Integer containing the Boolean value False until the end of the file has been reached. Returns True when the end of a file opened for Random or sequential Input has been reached.

## Related Functions

[FileLen](#) | [Loc](#) | [LOF](#) | [Seek](#)

## Example

```
Dim strFileContents as String, strTemp as String
Open "c:\test.txt" For Input As #1 ' open file
Do While Not EOF(1) ' loop until end of file
    strTemp = Input(10, #1) ' read next ten characters
    strFileContents = strFileContents & strTemp ' join strings
Loop
Close #1
```

## FileCopy

Copies a file from *Src* to *Dest*.

The required source file (*Src*) and destination file (*Dest*) arguments must be valid string expressions representing valid file names. *Src* is the file name of the file to copy from. *Dest* is the file name to be copied to. Both *Src* and *Dest* arguments may contain a DOS path structure including directory or folder names, and a drive letter.

If the *Dest* file does not exist, it will be created by the FileCopy statement. If the *Dest* file already exists, it will be overwritten.

The FileCopy statement does not work on a currently open file. Both the *Src* and *Dest* files must be closed before using the FileCopy statement. To close an open file, use the Close statement.

**Note:** The file system keeps track of the current drive, and the current directory of every drive. Use the [CurDir](#), [CurDir\\$](#) statement to determine the current directory. The current drive letter can be extracted from the Left character returned in the CurDir statement.

## Syntax

**FileCopy***Src*, *Dest*

*Src*:

A string or expression that can represent a valid DOS file structure FileName. Src is case insensitive. This may include a relative or static Path including directory or folder structure and drive letter. To specify multiple files, the FileName may consist of multiple-character ( \* ) and single-character ( ? ) wildcards in the file name.

*Dest*:

A string or expression that can represent a valid DOS file structure FileName. Dest is case insensitive. This may include a relative or static Path including directory or folder structure and drive letter. To specify multiple files, the FileName may consist of multiple-character ( \* ) and single-character ( ? ) wildcards in the file name.

## Related Functions

[Close](#) | [FreeFile](#) | [Kill](#) | [Name](#) | [Open](#)

## Example

```
Dim SourceFile as String, DestinationFile as String
SourceFile = "SRCFILE.Dat" ' Define source file name.
DestinationFile = "DESTFILE.Dat" ' Define target file name.
FileCopy SourceFile, DestinationFile ' Copy source to target.
```

## FileLen

FileLen function determines the byte length of a file. The required *File* argument must be valid string expression representing a valid file name. *File* may contain a DOS path structure including directory or folder names, and a drive letter.

The FileLen function returns the size of a file immediately before it was most recently opened. To obtain the length of a file that is already open, use the LOF function.

## Syntax

**FileLen**(*File*)

*File*:

A string or expression that can represent a valid file name, and may include a DOS path structure including directory or folder names, and a drive letter.

## Return Value

Returns a Long value representing the length of the file measured in bytes.

## Related Functions

[EOF](#) | [Loc](#) | [LOF](#) | [Seek](#)

## Example

```
Dim lonFileSize As Long
lonFileSize = FileLen("C:\TESTFILE.txt") ' returns length of file
in bytes
```

## FreeFile

Retrieves the next sequential system file number available for association with a file. Use the FreeFile function to retrieve an unassociated file number from the file system. This number can be used by the Open statement to be associated with a file.

## Syntax

**FreeFile**

## Return Value

Returns an Integer reference number ready for being associated with a file.

## Related Functions

[Close](#) | [FileCopy](#) | [Kill](#) | [Name](#) | [Open](#)

## Example

```
Dim intFileNum as Integer
intFileNum = FreeFile 'retrieve next free file number
Open "c:\TEST.txt" For Output As #intFileNum
Write #intFileNum, "This is a sample line of text."
Close #intFileNum
```

## Get #

Get statement reads data from a disk file into a variable.

The required *FileNum* argument is a system reference number associated with an open file. The optional *RecNum* argument is the byte position where the read starts for files opened in Binary mode. If you omit *RecNum*, the next record or byte following the last Get or Put statement (or pointed to by the last Seek function) is read. You

must include delimiting commas.

The required *VarName* is the name of the variable where the file data is read (copied) to.

### Random mode

For files opened in Random mode, the following rules apply:

- If the length of the data being read is less than the length specified in the *Len* clause of the Open statement, Get reads subsequent records on record-length boundaries. The space between the end of one record and the beginning of the next record is padded with the existing contents of the file buffer. Because the amount of padding data can't be determined with any certainty, it is generally a good idea to have the record length match the length of the data being read.
- If the variable being read into is a variable-length string, Get reads a 2-byte descriptor containing the string length and then reads the data that goes into the variable. Therefore, the record length specified by the *Len* clause in the Open statement must be at least 2 bytes greater than the actual length of the string.
- If the variable being read into is a Variant of numeric type, Get reads 2 bytes identifying the VarType of the Variant and then the data that goes into the variable. For example, when reading a Variant of VarType 3, Get reads 6 bytes: 2 bytes identifying the Variant as VarType 3 (Long) and 4 bytes containing the Long data. The record length specified by the *Len* clause in the Open statement must be at least 2 bytes greater than the actual number of bytes required to store the variable.

---

**Note:** You can use the Get statement to read a Variant array from disk, but you can't use Get to read a scalar Variant containing an array. You also can't use Get to read objects from disk.

---

- If the variable being read into is a Variant of VarType 8 (String), Get reads 2 bytes identifying the VarType, 2 bytes indicating the length of the string, and then reads the string data. The record length specified by the *Len* clause in the Open statement must be at least 4 bytes greater than the actual length of the string.
- If the variable being read into is a dynamic array, Get reads a descriptor whose length equals 2 plus 8 times the number of dimensions, that is,  $2 + 8 * \text{NumberOfDimensions}$ . The record length specified by the *Len* clause in the Open statement must be greater than or equal to the sum of all the bytes required to read the array data and the array descriptor. For example, the following array declaration requires 118 bytes when the array is written to disk.
- If the variable being read into is a fixed-size array, Get reads only the data. No descriptor is read.
- If the variable being read into is any other type of variable (not a variable-length string or a Variant), Get reads only the variable data. The record length specified by the *Len* clause in the Open statement must be greater than or equal to the length of the data being read.

Get reads elements of user-defined types as if each were being read individually, except that there is no padding between elements. On disk, a dynamic array in a user-defined type (written with Put) is prefixed by a descriptor whose length equals 2 plus 8 times the number of dimensions, that is,  $2 + 8 * \text{NumberOfDimensions}$ . The record length specified by the *Len* clause in the Open statement must be greater than or equal to the sum of all the bytes required to read the individual elements, including any arrays and their descriptors.

### Binary mode

For files opened in Binary mode, all of the Random rules apply, except:

- The *Len* clause in the Open statement has no effect. Get reads all variables from disk contiguously; that is, with no padding between records.
- For any array other than an array in a user-defined type, Get reads only the data. No descriptor is read.

Get reads variable-length strings that aren't elements of user-defined types without expecting the 2-byte length

descriptor. The number of bytes read equals the number of characters already in the string.

## Syntax

**Get #(*FileNum*, *RecNum*, *VarName*)**

*FileNum*:

An Integer or numeric expression representing any valid number in the range 1 to 511 inclusive, which is referenced by the file system to be associated with an open file.

*RecNum*:

The byte position where the read starts for files opened in Binary mode. If you omit RecNum, the next record or byte following the last Get or Put statement (or pointed to by the last Seek function) is read.

*VarName*:

A string representing a valid variable name.

## Related Functions

[GetAttr](#) | [Input](#) | [Line Input #](#) | [Print #](#) | [Put #](#) | [Write #](#)

## Example

```
Type Record ' Define user-defined type.  
ID As Integer  
Name As String * 20  
End Type  
  
Dim recRecord As Record  
Dim intPosition As Integer  
Dim intFileNum as Integer  
intFileNum = FreeFile 'retrieve next free file number  
' Open sample file for random access.  
Open "TESTFILE.txt" For Random As #intFileNum  
' Read the sample file using the Get statement.  
intPosition = 3 ' Define third record number.  
Get #intFileNum, intPosition, recRecord ' Read third record.  
Close #intFileNum ' Close file.
```

## GetAttr

GetAttr function returns an Integer representing the attribute settings of a file, directory, or volume.

The required *File* argument must be valid string expression representing a valid file name. *File* may contain a DOS path structure including directory or folder names, and a drive letter.

To determine which attributes are set, use the **AND** operator to perform a bitwise comparison of the value returned by the GetAttr function and the value of the individual file attribute you want. If the result is not zero, that attribute is set for the named file. For example, the return value of the following **AND** expression is zero if the Archive attribute is not set:

```
Const AttrArchive = 32  
Result = GetAttr(FileName) And AttrArchive ' A nonzero value is returned if the Archive
```

attribute is set.

## Syntax

### **GetAttr(*File*)**

*File*:

A string or expression that can represent a valid file name, and may include a DOS path structure including directory or folder names, and a drive letter.

## Return Value

Returns an Integer number indicating the sum Attribute value of a file, directory, or folder for the *File* argument, where:

- 0 = Normal
- 1 = Read Only
- 2 = Hidden
- 4 = System
- 8 = Volume
- 16 = Directory or Folder
- 32 = Archive

## Related Functions

[Get #](#) | [Line Input #](#) | [Put #](#)

## Example

```
Dim intAttrVal
' Assume file TESTFILE has hidden attribute set.
intAttrVal = GetAttr("TESTFILE.txt") ' Returns 2.
' Assume file TESTFILE has hidden and read-only attributes set.
intAttrVal = GetAttr("TESTFILE.txt") Returns 3.
' Assume MYDIR is a directory or folder.
intAttrVal = GetAttr("MYDIR") ' Returns 16.
```

## Input

*Input #* statement reads data from a Sequential file and assigns that data to variables. *Input* function returns characters from a file opened in Input or Binary mode.

The *Input #* statement has two parameters *FileNum* and *VarList*. The required *FileNum* argument is the associated file number used in the *Open* statement when the file was opened. The required *VarList* argument is a comma delimited list of variables that are assigned values read from the file.

The *Input* function has two parameters: *Num* and *FileNum*. The required *Num* argument is a number or valid numeric expression specifying the number of characters (bytes) to be read from the file. *FileNum* is the

associated file number used in the Open statement when the file was opened.

The file system tracks all open files and the current position of access within every file. Every statement or function that accesses the data within a file, alters the current position within that file. The Loc function can be used to determine the current position within an open file.

Use the LOF and Loc functions instead of EOF when reading binary files with the Input function, or use Get when using the EOF function.

An error occurs with files opened for Binary access, when the file is read using the Input function until EOF returns True.

Data read with the Input # statement has usually been written to a file with the Write # statement. Data read with the Input function has usually been written to a file with the Print # or Put statements.

When saving data to a file for future reading with the Input # statement, use the Write # statement instead of the Print # statement to write the data to the file. Using Write # properly delimits each separate data field, so it can be read back in using Input #. Using Write # also formats the data in a manner that will allow correct read operations in most locales.

## Syntax

**Input #(*FileNum*, *VarList*)**

*FileNum*:

An Integer or numeric expression representing any valid number in the range 1 to 511 inclusive, which is referenced by the file system to be associated with an open file.

*VarList*:

A predefined valid VBA variable name or comma delimited list of valid variable names.

## Return Value

Input # statement returns data record by record from a file opened in Input or Binary mode. Data items in a file must appear in the same order as the variables in *VarList* and match variables of the same data type. If a variable is numeric and the data is not numeric, a value of zero is assigned to the variable.

Input function returns a String containing characters from a file opened in Input or Binary mode. The Input function returns all of the characters it reads, including commas, carriage returns, linefeeds, quotation marks, and leading spaces.

## Related Functions

[Get #](#) | [GetAttr](#) | [Line Input #](#) | [Print #](#) | [Put #](#) | [Write #](#)

## Example

```
Dim strFileContents As String
Dim strTemp As String
Dim strString As String
Dim intFileNum as Integer
Dim intNumber as Integer
intFileNum = FreeFile 'retrieve next free file number
Open "c:\test.txt" For Input As #intFileNum ' open file.
```

```
Do While Not EOF(intFileNum) ' loop until end of file
    strTemp = Input(10, #intFileNum) ' read next ten characters
    strFileContents = strFileContents & strTemp ' join strings
Loop
Input #intFileNum, strString, intNumber ' Read data into two variables.
Close #intFileNum
```

## Kill

Kill statement deletes files from disk.

The required *File* argument must be valid string expression representing a valid file name. *File* may contain a DOS path structure including directory or folder names, and a drive letter.

Kill supports the use of multiple-character (\*) and single-character (?) wildcards to specify multiple files. The Kill statement does not work on a currently open file. To remove a directory use the [RmDir](#) statement.

The file system tracks the current drive and the current directory of every drive. Use the [CurDir](#), [CurDir\\$](#) statement to determine the current directory. The current drive letter can be extracted from the Left character returned in the CurDir statement.

## Syntax

**Kill** *File*

*File*:

A string or expression that can represent a valid file name, and may include a DOS path structure including directory or folder names, and a drive letter.

## Related Functions

[Close](#) | [FileCopy](#) | [FreeFile](#) | [Name](#) | [Open](#)

## Example

```
' Assume TESTFILE is a file containing some data.
Kill "TestFile"

' Delete all Dat files in current directory.
Kill "*.*.Dat"
```

## Line Input #

Line Input # statement reads a single line from an open sequential file and assigns it to a String variable.

The required *FileNum* argument is a system reference number associated with an open file. The required *VarName* is the name of the variable where the file data is read (copied) to.

---

**Note:** The number sign (#) preceding *FileNum* is not optional.

The Line Input # statement reads from a file one character at a time until it encounters a carriage return (Chr(13)) or carriage return-linefeed (Chr(13) + Chr(10)) sequence. Carriage return - linefeed sequences are skipped rather than appended to the character string.

---

**Note:** The file system keeps track of all open files and the current position of access within every file. Every statement or function that accesses the data within a file, alters the current position within that file. The Loc function can be used to determine the current position within an open file.

---

Data read with the Line Input # statement has usually been written to a file with the Print # statement.

## Syntax

**Line Input # FileNum, VarName**

*FileNum:*

An Integer or numeric expression representing any valid number in the range 1 to 511 inclusive, which is referenced by the file system to be associated with an open file.

*VarName:*

A string representing a valid variable name.

## Related Functions

[Get #](#) | [GetAttr](#) | [Input](#) | [Print #](#) | [Put #](#) | [Write #](#)

## Example

```
Dim strTextLine As String
Dim intFileNum As Integer
Open "c:\TEST.txt" For Input As #intFileNum intFileNum = FreeFile
'retrieve next free file number
Do While Not EOF(intFileNum) ' Loop until end of file.
    Line Input #intFileNum, strTextLine ' Read line into variable.
    Print TextLine ' Print line.
Loop
Close #intFileNum
```

## Loc

Loc function returns a number indicating the current position within a file opened using the Open statement.

The required *FileNum* argument must contain an Integer representing any valid number associated with an open file.

## Syntax

**Loc(*FileNum*)**

*FileNum:*

An Integer or numeric expression representing any valid number in the range 1 to 511 inclusive, which is referenced by the file system to be associated with an open file.

## Return Value

Returns a Long representing the current position within a file, the value dependant upon which file access mode the file was opened with:

- If the file was opened in Random mode, the Loc function will return a number representing the last record read from or written to the file.
- If the file was opened in Sequential mode, the Loc function will return a number representing the current byte position in the file divided by 128. (However, information returned by Loc for Sequential files is neither used nor required.)
- If the file was opened in Binary mode, the Loc function will return a number representing the position of the last byte read from or written to the file.

## Related Functions

[EOF](#) | [FileLen](#) | [LOF](#) | [Seek](#)

## Example

```
Dim lonLoc As Long
Dim strLine As String
Open "TESTFILE.txt" For Binary As #1 ' Open file
Do While lonLoc < LOF(1) ' Loop until end of file
    strLine = strLine & Input(1, #1) ' Read character into variable
    lonLoc = Loc(1) ' Get current position within file
Loop
<statements> ' Do stuff with retrieved data
Close #1 ' Close file
```

## LOF

LOF function returns a number indicating the byte length of a sequential file opened using the Open statement.

The required *FileNum* argument must contain an Integer representing any valid number associated with an open file.

---

**Note:** The file system keeps track of all open files and the current position of access within every file. Every statement or function that accesses the data within a file, alters the current position within that file. The Loc function can be used to determine the current position within an open file.

---

The LOF function returns the size of a file that is already open. To obtain the length of a file that is not open, use the FileLen function.

Use the LOF and Loc functions instead of EOF when reading binary files with the Input function.

## Syntax

**LOF(*FileNum*)**

*FileNum*:

An Integer or numeric expression representing any valid number in the range 1 to 511 inclusive, which is

referenced by the file system to be associated with an open file.

## Return Value

Returns a Long representing the size of a file in bytes.

## Related Functions

[EOF](#) | [FileLen](#) | [Loc](#) | [Seek](#)

## Example

```
Dim lonFileSize As Long  
lonFileSize = LOF "C:\TESTFILE.txt" ' returns length of file in bytes
```

## MkDir

The MkDir statement creates the directory specified in the Path parameter.

The required parameter Path must be a string or expression that can represent a valid DOS file structure path value, must contain a directory name, may contain a relative path structure, and may contain a drive letter. The Path parameter must be limited to less than 128 characters.

The MkDir statement is relative to the current directory. If no path structure is provided, the directory is created in the current directory. If no drive is specified, the MkDir statement creates the directory on the current drive.

---

**Note:** The file system keeps track of the current drive, and the current directory of every drive. Use the CurDir statement to determine the current directory. The current drive letter can be extracted from the Left character returned in the CurDir statement.

---

## Syntax

**MkDir Path**

*Path:*

A string or expression that can represent a valid DOS file structure path value. This includes a directory name, and may include a relative or static directory or folder structure and drive letter, in the order:

```
[<driveletter>:] [\<rootdirectoryname>] [\<subdirectory> ...  
\<subdirectory>\] directoryname
```

The path can be relative to the current directory. A single period represents the current directory (.). Two periods represent the parent directory of the current directory(..). For example:

```
chdir .. ' changes to the parent directory of the current directory  
chdir ..\test ' changes to the test subdirectory of the parent directory
```

## Related Functions

[ChDir](#) | [ChDrive](#) | [CurDir, CurDir\\$](#) | [Dir](#) | [RmDir](#)

## Example

```
Dim strPath As String
Dim strDir As String
strPath = CurDir() ' store current path
strDir = "Temp"
ChDir "\" ' change to root dir on current drive
MkDir strDir ' create new directory
ChDir strPath ' change back to previous path
```

## Name

The Name statement renames the disk file specified in the *OldFileName* parameter, to the name specified in the *NewFileName* parameter.

The required parameter *OldFileName* must be valid existing file name, may contain a path structure, and may contain a drive letter.

The *NewFileName* parameter must be a string or expression that can represent a valid DOS file name value, may contain a relative path structure, and may contain a drive letter. The *NewFileName* parameter must be limited to less than 128 characters.

The Name statement uses the file system relative to the current directory. If no path structure is provided, the *NewFileName* file is expected to be in the current directory. If no drive is specified, the Name statement expects the file to be on the current drive.

Using Name, you can move a file from one directory or folder to another. If the path in *NewFileName* exists and is different from the path in *OldFileName*, the Name statement moves the file to the new directory or folder and renames the file, if necessary. If *NewFileName* and *OldFileName* have different paths and the same file name, Name moves the file to the new location and leaves the file name unchanged.

Name does not support the use of multiple-character ( \* ) and single-character (?) wildcards to specify multiple files.

The Name statement does not work on a currently open file. You must close an open file before renaming it.

---

**Note:** The file system keeps track of the current drive, and the current directory of every drive. Use the [CurDir](#), [CurDir\\$](#) statement to determine the current directory. The current drive letter can be extracted from the Left character returned in the CurDir statement.

---

## Syntax

**Name** *OldFileName NewFileName*

*OldFileName*:

A string or expression that can represent a valid file name, and may include a DOS path structure including directory or folder names, and a drive letter.

*NewFileName*:

A string or expression that can represent a valid file name, and may include a DOS path structure including directory or folder names, and a drive letter.

## Related Functions

[Close](#) | [FileCopy](#) | [FreeFile](#) | [Kill](#) | [Open](#)

## Example

```
Dim strNewFileName As String
Dim strOldFileName As String
strOldFileName = "c:\temp\oldfile.txt"
strNewFileName = "newfile.txt"
ChDir "\" ' change to root dir on current drive
Name strOldFileName strNewFileName ' moves file to root dir and renames it
ChDir strPath ' change back to previous path
```

## Open

Open statement enables input/output (I/O) to a disk file.

The required *File* argument must be a valid string expression representing a valid file name. *File* may contain a DOS path structure including directory or folder names, and a drive letter.

The required *Mode* argument must be a valid keyword specifying the file I/O mode: Append, Binary, Input, Output, or Random. If unspecified, the file is opened for Random access.

The optional *Access* argument must be a valid keyword specifying the operations permitted on the open file: Read, Write, or Read Write.

The optional *Lock* argument must be a valid keyword specifying the operations permitted on the open file by other processes: Shared, Lock Read, Lock Write, and Lock Read Write.

The required *FileNum* argument must contain an Integer representing the number that will be associated with the file. This is the file system reference number supplied by the FreeFile statement that can be used in functions such as Get #, Input #, Line Input #, Print #, and Write #. In Binary, Input, and Random modes, you can open a file using a different file number without first closing the file. In Append and Output modes, you must close a file before opening it with a different file number.

---

**Note:** The file system tracks all open files and the current position of access within every file. Every statement or function that accesses the data within a file, alters the current position within that file. The Loc function can be used to determine the current position within an open file.

---

The optional *RecLen* argument must be a number less than or equal to 32,767 (bytes). For files opened for Random access, this value is the record length. For sequential files, this value is the number of characters buffered. The Len clause is ignored if mode is Binary.

You must open a file before any I/O operation can be performed on it. Open allocates a buffer for I/O to the file and determines the mode of access to use with the buffer. If the file is already opened by another process and the specified type of access is not allowed, the Open operation will not succeed and an error message will be generated.

If the file specified by pathname doesn't exist, it is created when a file is opened for Append, Binary, Output, or Random modes.

## Syntax

**Open(*File For Mode Access Lock As #FileNum Len=RecLen*)**

***File:***

A string or expression that can represent a valid file name, and may include a DOS path structure including directory or folder names, and a drive letter.

***Mode:***

A VBA keyword specifying the file I/O mode: Append, Binary, Input, Output, or Random.

***Lock:***

A VBA keyword specifying the operations permitted on the open file by other processes: Shared, Lock Read, Lock Write, and Lock Read Write.

***Access:***

A VBA keyword specifying the operations permitted on the open file: Read, Write, or Read Write.

***FileNum:***

An Integer or numeric expression representing any valid number in the range 1 to 511 inclusive, which is referenced by the file system to be associated with an open file.

***RecLen:***

An Integer or numeric expression representing the byte length of a file record as a number less than or equal to 32,767.

## Related Functions

[Close](#) | [FileCopy](#) | [FreeFile](#) | [Kill](#) | [Name](#)

## Example

```
' The following code opens the file TESTFILE in sequential-input
mode.
Open "TESTFILE" For Input As #1
' Close before reopening in another mode.
Close #1

' This example opens the file in Binary mode for writing
operations only.
Open "TESTFILE" For Binary Access Write As #1
' Close before reopening in another mode.
Close #1

' The following example opens the file in Random mode. The file
contains records of the user-defined type Record.
Type Record ' Define user-defined type.
ID As Integer
Name As String * 20
End Type
Dim recRecord As Record ' Declare variable.
Open "TESTFILE" For Random As #1
' Close before reopening in another mode.
Close #1

' This code example opens the file for sequential output; any
process can read or write to file.
Open "TESTFILE" For Output Shared As #1
```

```
' Close before reopening in another mode.  
Close #1  
  
' This code example opens the file in Binary mode for reading;  
other processes can't read file.  
Open "TESTFILE" For Binary Access Read Lock Read As #1  
' Close before reopening in another mode.  
Close #1
```

## Print (function)

Displays a message in the Runtime Kernel, and the Cicode Editor output window if you are in debug mode.

**Note:** Do not confuse this function with the [Print #](#) statement, which prints data to disk.

## Related Functions

[TraceMsg](#) (Cicode function)

## Print #

Print # statement reads data from *OutputList* and writes that data to a sequential file.

The Print # statement has two parameters *FileNum* and *OutputList*. The required *FileNum* argument is the associated file number used in the Open statement when the file was opened. The required *OutputList* argument is a delimited list of expressions whose values are written to the file.

**Note:** The number sign hash character (#) preceding *FileNum* is not optional. This character indicates disk file access with the file referenced by the system file number that follows it. Do not confuse Print # which prints to disk, with Print which displays data on the screen.

Data written with Print # is usually read from a file with Line Input # or Input.

**Note:** If you want to read the data from a file using the Input # statement, use the Write # statement instead of the Print # statement to write the data to the file. Using Write # properly delimits each separate data field, so it can be read back in using Input #. Using Write # also formats the data in a manner that will allow correct read operations in most locales.

If you omit expressionlist, the Print # statement prints a blank line in the file, but you must include the comma. Because Print # writes an image of the data to the file, you must delimit the data so it is printed correctly. If you use commas as delimiters, Print # also writes the blanks between print fields to the file.

The Print # statement usually writes Variant data to a file the same way it writes any other data type. However, there are some exceptions:

If the data being written is a Variant of VarType 0 (Empty), Print # writes nothing to the file for that data item.

If the data being written is a Variant of VarType 1 (Null), Print # writes the literal #NULL# to the file.

If the data being written is a Variant of VarType 7 (Date), Print # writes the date to the file using the Short Date format defined in the WIN.INI file. When either the date or the time component is missing or zero, Print # writes only the part provided to the file.

## Syntax

**Print #**FileNum, *OutputList*

*FileNum*:

An Integer or numeric expression representing any valid number in the range 1 to 511 inclusive, which is referenced by the file system to be associated with an open file.

*OutputList*:

One or more formatted numeric and/or string expressions to be written to the file using the following syntax:

[ {Spc( s ) | Tab [( n ) ] } ] [expression] [charpos]

Where:

[ ] square brackets are used for illustrative purposes to indicate in the code that the arguments they enclose are optionally used in the *OutputList*. Do not use the square brackets themselves in your code.

{ } curly braces are required to encompass and delineate the arguments they enclose, and to separate their contents from the other arguments in the *OutputList*.

( | ) vertical line are used for illustrative purposes to indicate in the code that either side of the line is an alternative argument. You can use the argument provided on one of the line or the other, but not both arguments at the same time within the same set of curly braces. Do not include the vertical line in your code.

{Spc(s)} argument is optionally used to insert 's' number of space characters in the output file at the position of the argument in the *OutputList*. The *Spc* argument must be enclosed by curly braces to delineate it from an *expression*. The *Spc* argument can be repeated any number of times to insert spaces in the file between *expressions*. The *Spc* argument is mutually exclusive with the *Tab* argument when used within the same set of curly braces.

{Tab(n)} argument is optionally used to position the insertion point to an absolute column number in the output file at the position of the argument in the *OutputList*, where 'n' is the column number. Use *Tab*with no argument to position the insertion point at the beginning of the next print zone. The *Tab* argument must be enclosed by curly braces to delineate it from an *expression*. The *Tab* argument can be repeated any number of times to insert tabs in the file between *expressions*. The *Tab* argument is mutually exclusive with the *Spc* argument when used within the same set of curly braces.

*expression* argument represents a valid numeric or string expression to output to the file. The *expression* argument can be repeated any number of times.

*charpos* is the character that determines the position of the next character in the output. A semicolon means the next character is printed immediately after the last character; a comma means the next character is printed at the start of the next print zone. Print zones begin every 14 columns. If neither character is specified, the next character is printed on the next line.

## Return Value

Input # statement returns data record by record from a file opened in Input or Binary mode. Data items in a file must appear in the same order as the variables in *VarList* and match variables of the same data type. If a variable is numeric and the data is not numeric, a value of zero is assigned to the variable.

## Related Functions

[Get #](#) | [GetAttr](#) | [Input](#) | [Line Input #](#) | [Put #](#) | [Write #](#)

## Example

The following example writes data to a test file.

```
Dim I, FNum, FName ' Declare variables.  
For I = 1 To 3  
FNum = FreeFile ' Determine next file number.  
FName = "TEST" & FNum  
Open FName For Output As FNum ' Open file.  
Print #1, "This is test #" & I ' Write string to file.  
Print #1, "Here is another "; "line"; I  
Next I  
Close ' Close all files.  
The following example writes data to a test file and reads it back.  
Dim FileData, Msg, NL ' Declare variables.  
NL = Chr(10) ' Define newline.  
Open "TESTFILE" For Output As #1 ' Open to write file.  
Print #1, "This is a test of the Print # statement."  
Print #1, ' Print blank line to file.  
Print #1, "Zone 1", "Zone 2" ' Print in two print zones.  
Print #1, "With no space between" ; "." ' Print two strings together.  
Close #1  
Open "TESTFILE" for Input As #2 ' Open to read file.  
Do While Not EOF(2)  
Line Input #2, FileData ' Read a line of data.  
Msg = Msg & FileData & NL ' Construct message.  
MsgBox Msg  
Loop  
Close #2 ' Close all open files.  
Kill "TESTFILE" ' Remove file from disk.
```

## Put #

Put # statement writes data from a variable to a disk file.

The required *FileNum* argument is a system reference number associated with an open file.

---

**Note:** The number sign (#) preceding *FileNum* is not optional.

The optional *RecNum* argument is the byte position where the read starts for files opened in Binary mode. The first record or byte in a file is at position 1, the second record or byte is at position 2, and so on. If you omit *RecNum*, the next record or byte following the last Get or Put statement (or pointed to by the last Seek function) is read. You must include delimiting commas.

---

**Note:** The file system keeps track of all open files and the current position of access within every file. Every statement or function that accesses the data within a file, alters the current position within that file. The Loc function can be used to determine the current position within an open file.

The required *VarName* is the name of the variable where the file data is read (copied) from.

Data written with the Put # statement is usually read from a file with the Get # statement.

### Random mode

For files opened in Random mode, the following rules apply:

- If the length of the data being written is less than the length specified in the Len clause of the Open statement, Put writes subsequent records on record-length boundaries. The space between the end of one record and the beginning of the next record is padded with the existing contents of the file buffer. Because the amount of padding data can't be determined with any certainty, it is generally a good idea to have the record length match the length of the data being written. If the length of the data being written is greater than the length specified in the Len clause of the Open statement, an error occurs.
- If the variable being written is a variable-length string, Put writes a 2-byte descriptor containing the string length and then the variable. The record length specified by the Len clause in the Open statement must be at least 2 bytes greater than the actual length of the string.
- If the variable being written is a Variant of a numeric type, Put writes 2 bytes identifying the VarType of the Variant and then writes the variable. For example, when writing a Variant of VarType 3, Put writes 6 bytes: 2 bytes identifying the Variant as VarType 3 (Long) and 4 bytes containing the Long data. The record length specified by the Len clause in the Open statement must be at least 2 bytes greater than the actual number of bytes required to store the variable.

---

**Note:** You can use the Put statement to write a Variant array to disk, but you can't use Put to write a scalar Variant containing an array to disk. You also can't use Put to write objects to disk.

---

- If the variable being written is a Variant of VarType 8 (String), Put writes 2 bytes identifying the VarType, 2 bytes indicating the length of the string, and then writes the string data. The record length specified by the Len clause in the Open statement must be at least 4 bytes greater than the actual length of the string.
- If the variable being written is a dynamic array, Put writes a descriptor whose length equals 2 plus 8 times the number of dimensions, that is,  $2 + 8 * \text{NumberOfDimensions}$ . The record length specified by the Len clause in the Open statement must be greater than or equal to the sum of all the bytes required to write the array data and the array descriptor. For example, the following array declaration requires 118 bytes when the array is written to disk.

```
Dim MyArray(1 To 5,1 To 10) As Integer
```

The 118 bytes are distributed as follows: 18 bytes for the descriptor ( $2 + 8 * 2$ ), and 100 bytes for the data ( $5 * 10 * 2$ ).

- If the variable being written is a fixed-size array, Put writes only the data. No descriptor is written to disk.
- If the variable being written is any other type of variable (not a variable-length string or a Variant), Put writes only the variable data. The record length specified by the Len clause in the Open statement must be greater than or equal to the length of the data being written.

Put writes elements of user-defined types as if each were written individually, except there is no padding between elements. On disk, a dynamic array in a user-defined type written with Put is prefixed by a descriptor whose length equals 2 plus 8 times the number of dimensions, that is,  $2 + 8 * \text{NumberOfDimensions}$ . The record length specified by the Len clause in the Open statement must be greater than or equal to the sum of all the bytes required to write the individual elements, including any arrays and their descriptors.

### Binary mode

For files opened in Binary mode, all of the Random rules apply, except:

- The Len clause in the Open statement has no effect. Put writes all variables to disk contiguously; that is, with no padding between records.
- For any array other than an array in a user-defined type, Put writes only the data. No descriptor is written.
- Put writes variable-length strings that aren't elements of user-defined types without the 2-byte length descriptor. The number of bytes written equals the number of characters in the string. For example, the following statements write 10 bytes to file number 1:

```
VarString$ = String$(10, " ")
```

Put writes variable-length strings that are not elements of user-defined types without the 2-byte length descriptor.

```
Put #1,,VarString$
```

## Syntax

**Put #** *FileNum*, *RecNum*, *VarName*

*FileNum*:

An Integer or numeric expression representing any valid number in the range 1 to 511 inclusive, which is referenced by the file system to be associated with an open file.

*RecNum*:

The byte position where the read starts for files opened in Binary mode. The first record or byte in a file is at position 1, the second record or byte is at position 2, and so on. If you omit RecNum, the next record or byte following the last Get or Put statement (or pointed to by the last Seek function) is read.

*VarName*:

A string representing a valid variable name.

## Related Functions

[Get #](#) | [GetAttr](#) | [Input](#) | [Line Input #](#) | [Put #](#) | [Write #](#)

## Example

```
' This example uses the Put statement to write data to a file.  
' Five records of the user-defined type Record are written to the file.  
Type Record ' Define user-defined type.  
ID As Integer  
Name As String * 20  
End Type  
  
Dim MyRecord As Record, RecordNumber ' Declare variables.  
' Open file for random access.  
Open "TESTFILE" For Random As #1 Len = Len(MyRecord)  
For RecordNumber = 1 To 5 ' Loop 5 times.  
    MyRecord.ID = RecordNumber ' Define ID.  
    MyRecord.Name = "My Name" & RecordNumber ' Create a string.  
    Put #1, RecordNumber, MyRecord ' Write record to file.  
Next RecordNumber  
Close #1 ' Close file.
```

## RmDir

The RmDir statement deletes the directory specified in the *Path* parameter.

The required parameter *Path* must be a string or expression that can represent a valid DOS file structure path value, must contain a directory name, may contain a relative path structure, and may contain a drive letter. The

*Path* parameter must be limited to less than 128 characters.

The RmDir statement is relative to the current directory. If no path structure is provided, the directory is expected to be a subdirectory of the current directory. If no drive is specified, the RmDir statement deletes the directory on the current drive.

The current directory cannot be deleted. To change the current directory to another directory, use the ChDir statement. The directory to be deleted must be empty and contain no files or sub-directories. To delete files in a directory, use the Kill statement.

---

**Note:** The file system keeps track of the current drive, and the current directory of every drive. Use the [CurDir](#), [CurDir\\$](#) statement to determine the current directory. The current drive letter can be extracted from the Left character returned in the CurDir statement.

---

## Syntax

**RmDir** *Path*

*Path*:

A string or expression that can represent a valid DOS file structure path value. This includes a directory name, and may include a relative or static directory or folder structure and drive letter, in the order:

[<driveletter>:] [\<rootdirectoryname>] [\<subdirectory> ...  
\<subdirectory>\] directoryname

---

**Note:** The path can be relative to the current directory. A single period represents the current directory (.). Two periods represent the parent directory of the current directory (..). For example, chdir .. changes to the parent directory of the current directory. chdir ..\test changes to the test subdirectory of the parent directory

---

## Related Functions

[ChDir](#) | [ChDrive](#) | [CurDir](#), [CurDir\\$](#) | [Dir](#) | [MkDir](#)

## Example

```
Dim strDir As String
strDir = CurDir ' retrieve current directory name
Kill "*.*" ' delete all files from current directory
ChDir "\" ' change to root dir on current drive
RmDir strDir ' delete directory
```

## Seek

Sets the current position within a file opened using the Open statement, ready for the next read or write action.

The required *FileNum* argument must contain an Integer representing any valid system file number associated with an open file.

The required *Position* argument must contain an Integer or expression representing a valid number.

---

**Note:** The file system keeps track of all open files and the current position of access within every file. Every statement or function that accesses the data within a file, alters the current position within that file. The [Loc](#) function can be used to determine the current position within an open file.

---

## Syntax

**Seek** *FileNum*, *Position*

*FileNum*:

An Integer or numeric expression representing any valid number in the range 1 to 511 inclusive, which is referenced by the file system to be associated with an open file.

*Position*:

An Integer or expression representing a valid numeric value.

## Related Functions

[EOF](#) | [FileLen](#) | [Loc](#) | [LOF](#)

## Example

```
Open "TESTFILE" For Input As #1 ' Open file for reading.  
For i = 1 To 24 Step 3 ' Loop until end of file.  
    Seek #1, i ' Seek to byte position  
    MyChar = Input(1, #1) ' Read next character of data.  
    Print MyChar 'Print character of data  
Next i  
Close #1 ' Close file.
```

## Write #

Write # statement writes data to a Sequential file opened in output or append mode and reads that data from a list of variables.

The Write # statement has two parameters *FileNum* and *VarList*. The required *FileNum* argument is the associated file number used in the Open statement when the file was opened. The required *VarList* argument is a comma delimited list of variables that are assigned values read from the file.

---

**Note:** The file system keeps track of all open files and the current position of access within every file. Every statement or function that accesses the data within a file, alters the current position within that file. The [Loc](#) function can be used to determine the current position within an open file.

---

Data written to a file with the Write # statement is usually read with the Input # statement.

---

**Note:** When saving data to a file for future reading with the Input # statement, use the Write # statement instead of the Print # statement to write the data to the file. Using Write # properly delimits each separate data field , so it can be read back in using Input #. Using Write # also formats the data in a manner that will allow correct read operations in most locales.

---

## Syntax

**Write #***FileNum*, *VarList*

*FileNum*:

An Integer or numeric expression representing any valid number in the range 1 to 511 inclusive, which is referenced by the file system to be associated with an open file.

**VarList:**

A predefined valid VBA variable name or comma delimited list of valid variable names.

## Related Functions

[Get #](#) | [GetAttr](#) | [Input](#) | [Line Input #](#) | [Print #](#) | [Put #](#)

## Example

```
Dim strFileContents As String
Dim strTemp As String
Dim strString As String
Dim intFileNum as Integer
Dim intNumber as Integer
intFileNum = FreeFile 'retrieve next free file number
Open "c:\test.txt" For Output As #intFileNum ' open file.
Write #intFileNum, "This is a test of the Write # statement."
Close #intFileNum
```

## Math/Trigonometry Functions

VBA math functions are provided to assist with number manipulation and calculation in your formulas.

Mathematical functions can be used in VBA statements, and will (like all other functions), return a value to the caller.

## Numeric Functions

Plant SCADA uses the following predefined numeric functions:

<a href="#">Abs</a>	returns the absolute value of a number (Num ).
<a href="#">Exp</a>	returns base log (e) to the power of (Num ).
<a href="#">Fix</a>	returns the Integer value of a number (Num ).
<a href="#">Int</a>	returns the Integer value of a number (Num ).
<a href="#">Log</a>	returns the natural log of a number (Num ).
<a href="#">Rnd</a>	returns a random value influenced by (Num ).
<a href="#">Sgn</a>	returns a value indicating the Sign of (Num ).
<a href="#">Sqrt</a>	returns the square root value of a number (Num ).

### Abs

Calculates the absolute (positive) value of a number. The absolute value of a number is the number without its sign. Abs does not round the number, and ignores the fractional value of the number.

## Syntax

**Abs(Num)**

*Num:*

An integer or expression representing a valid numeric value.

## Return Value

Returns the absolute value of the number (*Num*) provided in the argument.

The data type of the return value is the same as that of the number argument. However, if the number argument is a Variant of VarType (String) and can be converted to a number, the return value will be a Variant of VarType (Double). If the numeric expression results in a null, Abs returns a null.

## Related Functions

[Sgn](#)

## Example

```
Variable=Abs(-67); ! Sets Variable to 67.  
Variable=Abs(67); ! Sets Variable to 67.
```

## Exp

Calculates the exponential of a number. The exponential is the base of the natural logarithm  $e$  raised to a power ( $e^{\text{Num}}$ ). The Exp function complements the Log function and is sometimes referred to as the antilogarithm.

---

**Note:** The value of the constant  $e$  is approximately 2.71828.

---

## Syntax

**Exp(Num)**

*Num:*

An Integer or expression representing a valid numeric value.

## Return Value

Returns the value equivalent to the base of the natural logarithm ( $e$ ) raised to the power of the number (*Num*) provided in the argument.

## Related Functions

[Log](#)

## Example

```
Variable=Exp(1); ! Sets Variable to 2.7182...
```

## Fix

Calculates the integer portion of a number. Fix does not round the number, and ignores the fractional value of the number.

Fix expects the argument (*Num*) to be a valid numeric value. If the argument value is positive, rounds the *Num* down by dropping any fractional value. If the argument value is negative, rounds the *Num* up to the next integer number greater than or equal to *Num*.

Do not confuse Fix with Int , which rounds a negative argument value (*Num*) down to the next integer number less than or equal to *Num*.

## Syntax

**Fix(*Num*)**

*Num*:

An Integer or expression representing a valid numeric value.

## Return Value

Returns the Integer value of the number (*Num*) provided in the argument.

## Related Functions

[Abs](#) | [Int](#) | [Sgn](#) | [Sqrt](#)

## Example

```
Dim vntVar
vntVar = Fix(99.2) ' returns 99
vntVar = Fix(99.8) ' returns 99
vntVar = Fix(-99.8) ' returns -99
vntVar = Fix(-99.2) ' returns -99
```

## Int

Calculates the integer portion of a number. Int does not round the number, and ignores the fractional value of the number.

Int expects the argument (*Num*) to be a valid numeric value. If the argument value is positive, rounds the *Num* down by dropping any fractional value. If the argument value is negative, rounds the *Num* down to the next integer number less than or equal to *Num*.

Do not confuse Int with Fix, which rounds a negative argument value (*Num*) up to the next integer number greater than or equal to *Num*.

## Syntax

**Int(Num)**

*Num:*

An Integer or expression representing a valid numeric value.

## Return Value

Returns the integer value of the number (*Num*) provided in the argument. If *Num* contains a Null, Int returns a Null.

## Related Functions

[Abs](#) | [Fix](#) | [Rnd](#) | [Sgn](#) | [Sqrt](#)

## Example

```
Dim vntVar
vntVar = Int(99.2) ' returns 99
vntVar = Int(99.8) ' returns 99
vntVar = Int(-99.8) ' returns -100
vntVar = Int(-99.2) ' returns -100
```

## Log

Calculates the natural logarithm of a number.

Log expects the argument (*Num*) to be a valid numeric value. The argument value must be greater than zero.

The natural logarithm is the logarithm to the base *e*. You can calculate the base-*n* logarithms for any number *X* by dividing the natural logarithm of *X* by the natural logarithm of *n* as follows:

$$\text{Log}_n(X) = \text{Log}(X) / \text{Log}(n)$$

---

**Note:** The value of the constant *e* is approximately 2.71828.

---

## Syntax

**Log(Num)**

*Num:*

An Integer or expression representing a valid numeric value.

## Return Value

Returns the natural log of the number (*Num*) provided in the argument.

## Related Functions

[Exp](#)

## Example

```
Variable=Log(100); ! Sets Variable to 2 (i.e. 100=10 to the power of 2).
```

## Rnd

Generates a decimal fraction number using the optional argument value (*Num*) to determine the sequence of the (random) number generation.

Rnd expects the argument (*Num*) if supplied, to be a valid numeric value.

If *Num* is less than zero, Rnd generates the same number every time, using *Num* as the seed. If *Num* is equal than zero, Rnd repeats the most recently generated number. If *Num* is greater than zero, Rnd generates the next random number in the sequence. If *Num* is not supplied, Rnd generates the next random number in the sequence.

Before calling Rnd, use the Randomize statement without an argument to initialise the random-number generator with a seed based on the system timer.

---

**Note:** The square brackets [ ] in the syntax indicate that the argument is optional.

Do NOT include the square brackets in your code.

---

## Syntax

**Rnd[(*Num*)]**

*Num*:

An Integer or expression representing a valid numeric value.

## Return Value

Returns a (random) decimal fraction number influenced by the (*Num*) provided in the argument. The return value lies in the range of less than 1 but greater than or equal to 0.

## Related Functions

[Randomize](#)

## Example

```
Dim vntRndValue
Randomize ' Initialize random-number generator.
vntRndValue = Int((6 * Rnd) + 1) ' returns a value between 1 and 6
```

## Sgn

Indicates the sign of a number. Sgn does not round the number, and ignores the fractional value of the number. Sgn expects the argument (*Num*) to be a valid numeric value. If *Num* is greater than zero, Sgn returns the value of 1. If *Num* is equal to zero, Sgn returns the value of 0. If *Num* is less than zero, Sgn returns the value of -1.

## Syntax

**Sgn(*Num*)**

*Num*:

An Integer or expression representing a valid numeric value.

## Return Value

Returns a value indicating the Sign (+ or -) value of the (*Num*) provided in the argument.

## Related Functions

[Abs](#) | [Fix](#) | [Int](#) | [Sqrt](#)

## Example

```
Dim vntVal
vntVal = Sgn(99.8) ' returns 1
vntVal = Sgn(-99.8) ' returns -1
vntVal = Sgn(0) ' returns 0
```

## Sqrt

Calculates the square root of a number. Sqrt expects the argument (*Num*) to be a valid numeric value greater than or equal to 0.

## Syntax

**Sqrt(*Num*)**

*Num*:

An Integer or expression representing a valid numeric value.

## Return Value

Returns the square root value of the (*Num*) provided in the argument.

## Related Functions

[Abs](#) | [Fix](#) | [Int](#) | [Sgn](#)

## Example

```
Variable=.Sqrt(4);  
! Sets Variable to 2.
```

## Trigonometric Functions

Plant SCADA uses the following trigonometric functions:

Atn	returns the Arctangent value of a number (Num ).
Cos	returns the Cosine value of angle (Rad ).
Sin	returns the Sine value of angle (Rad ).
Tan	returns the Tangent value of angle (Rad ).

Trigonometry uses angles and ratios, axes, degrees, Pi, radians and angular conversions. VBA supports the use of Decimal numbers by default, as well as Hexadecimal and Octal numbers. See [Numbers](#).

When using numbers in VBA, you must consider the data type of the variables that hold and store the numbers, as well as the behaviour of VBA when dealing with numbers. See .

### Atn

Calculates the trigonometric Arctangent value of a Tangent number.

The Atn function expects the argument (*Num*) to be a valid tangent value between the range of - Pi/2 to + Pi/2 (representing the ratio of the two sides of a right-angle triangle), and calculates the corresponding angle in radians.

Atn is the inverse trigonometric function of Tan (which takes an angle as its argument, and returns the ratio of two sides of a right-angle triangle). Do not confuse Atn with the Cotangent, which is the inverse of a Tangent (1/tangent).

## Syntax

**Atn(*Num*)**

*Num*:

An integer or expression representing a valid numeric value.

## Return Value

Returns the Arctangent value of the angle (*Num*) provided in the argument.

## Related Functions

[Cos](#) | [Sin](#) | [Tan](#)

## Example

```
Dim Msg, Pi' Declare variables.  
Pi = 4 * Atn(1)' Calculate Pi
```

### Cos

Calculates the trigonometric Cosine value of an angle.

The Cos function expects the argument (*Rad*) to be a valid angle value in radians, and calculates the ratio of the two sides of a right-angle triangle on either side of the angle. The ratio is the length of the side adjacent to the angle divided by the length of the hypotenuse.

**Note:** To convert degrees to radians, multiply degrees by Pi/180. To convert radians to degrees, multiply radians by 180/Pi.

## Syntax

**Cos(*Rad*)**

*Rad*:

An angle expressed in radians. It must be a valid numeric value.

## Return Value

Returns the Cosine value of the angle (*Rad*) provided in the argument.

The result lies in the range - 1 to +1.

Cos will return a double.

## Related Functions

[Atn](#) | [Sin](#) | [Tan](#)

## Example

```
Variable=Cos(0.7854); ! Sets Variable to 0.7071...
```

### Sin

Calculates the trigonometric Sine value of an angle. The Sin function expects the argument (*Rad*) to be a valid angle value in radians, and calculates the ratio of two sides of a right-angle triangle. The ratio is the length of the side opposite to the angle divided by the length of the hypotenuse.

To convert degrees to radians, multiply degrees by Pi/180 . To convert radians to degrees, multiply radians by 180/Pi. For more information, see Circle Maths.

## Syntax

**Sin(*Rad*)**

*Rad:*

An angle expressed in radians. Must be a valid numeric value.

## Return Value

Returns the Sine value of the angle (*Rad*) provided in the argument. The result lies in the range - 1 to + 1.

## Related Functions

[Atn](#) | [Cos](#) | [Tan](#)

## Example

```
Variable=Sin(0.7854); ! Sets Variable to 0.7071
```

## Tan

Calculates the trigonometric Tangent value of an angle. The Tan function expects the argument (*Rad*) to be a valid angle value in radians, and calculates the ratio of two sides of a right-angle triangle. The ratio is the length of the side opposite to the angle divided by the length of the side adjacent to the angle.

**Note:** To convert degrees to radians, multiply degrees by Pi/180. To convert radians to degrees, multiply radians by 180/Pi.

## Syntax

**Tan(*Rad*)**

*Rad:*

An angle expressed in radians. Must be a valid numeric value.

## Return Value

Returns the Tangent value of the angle (*Rad*) provided in the argument. Tan will return as a double.

## Example

```
Variable=Tan(1); ! Sets Variable to 1.5574...
```

## Miscellaneous Functions

The miscellaneous functions predefined in VBA are:

<a href="#">Beep statement</a>	Sounds a tone through the computer's speaker.
<a href="#">Randomize statement</a>	Initializes the random number generator.

Rem statement	Used to include explanatory remarks in a program.
SendKeys statement	Sends keystrokes to the active window as if entered at the keyboard.

## Beep

The Beep statement sounds a tone through the computer's speaker. The frequency and duration of the beep depends on the computer's hardware.

## Syntax

Beep

## Related Functions

[SendKeys](#)

## Example

```
If (TestTag_1 <1) OR (TestTag_1 > 100) Then
    Beep
Else
    Startup_AN38.Value = TestTag_1
End If
```

## Randomize

The Randomize statement initializes the random number generator.

It has one optional parameter number. This parameter can be any valid number and is used to initialize the random number generator. If you omit the parameter then the value returned by the Timer event is used as the default parameter to seed the random number generator.

## Syntax

`Randomize[number]`

## Related Functions

[Timer](#)

## Example

```
Dim MValue
' Initialise random-number generator
Randomize
```

```
MValue = Int((6 * Rnd) + 1)  
Print MValue
```

## Rem

Used to include explanatory comments in a program.

## Syntax

**Rem Comment**

*Comment:*

The text of any comment you want to include in the code.

## Example

```
' This is another way to comment  
Rem This is a remark
```

## SendKeys

Sends one or more keystrokes to the active window of the active application as if they had been entered at the keyboard.

The value of the *Wait* argument determines when the SendKeys function completes and returns control to VBA. If omitted, *Wait* is treated as FALSE by default.

---

**Note:** You can't use SendKeys to send keystrokes to an application that is not designed to run in Microsoft Windows. Sendkeys also can't send the PRINT SCREEN key {PRTSC} to any application..

---

## Syntax

**SendKeys(keys, wait)**

*keys:*

The string that is sent to the active window.

*wait:*

Enter TRUE or FALSE.

If *wait* is true the keystrokes must be processed before control is returned to the calling procedure. This argument is optional. If you omit it, it is assumed to be false.

## Return Value

None

## Example

```
Dim intCounter As Integer ' Declare variables.
```

```

Dim dblProgID As Double, ' Launch Windows Calculator program.
dblProgID = Shell("Calc.exe", 1) ' Set up counting loop.
For intCounter = 1 To 5 ' Send keystrokes to Calculator
    SendKeys intCounter & "{+}", True ' to add the value of intCounter each time
Next intCounter ' Return focus to Calculator.
AppActivate "Calculator" ' Send keystrokes to Close Calculator.
SendKeys "%{F4}", True

```

## Procedural Statements

VBA procedural function statements are provided to assist with conditional code execution and program flow:

<a href="#">Call statement</a>	Transfers control to a Sub procedure, function procedure, or dynamic-link library (DLL) procedure.
<a href="#">Function statement</a>	Declares and defines a procedure that can receive arguments and return a value of a specified data type.
<a href="#">End Function statement</a>	Ends a program or a block of statements within a function.
<a href="#">Sub statement</a>	Declares and defines a Sub procedures name, parameters and code.
<a href="#">End Sub statement</a>	Ends a program or a block of statements within a subroutine.
<a href="#">CicodeCallOpen function</a>	Calls a Cicode function from VBA.
<a href="#">CicodeCallReturn function</a>	Obtains the return value of the most recently completed Cicode function opened with the VBA CicodeCallOpen function.

Cicode functions used to handle VBA functions and statements:

<a href="#">VbCallOpen Function function</a>	Opens a VBA function or subroutine from Cicode.
<a href="#">VbCallRun Function function</a>	Runs the opened VBA function or subroutine from Cicode.
<a href="#">VbCallReturn Function function</a>	Obtains the return value of the completed VBA function previously opened with the Cicode VbCallOpen function.

### Call

The Call Statement transfers control to a Sub procedure, Function procedure, or dynamic-link library (DLL) procedure.

The required *ProcedureName* is the name of the function or subroutine to call. The optional *Parameters* is the list of arguments to pass to the called function or subroutine.

You are not required to use the Call statement when calling an VBA subroutine or a DLL function. Parentheses must be used in the argument list if the Call statement is being used.

## Syntax

**Call ProcedureName[Parameter(s)]**

## Related Functions

[End Function](#) | [Sub](#) | [End Sub](#) | [Exit](#)

## Example

```
Call Beep
```

### CicodeCallOpen

The CicodeCallOpen function is used to call a Cicode function from VBA. It is used to initiate and execute a call to the Cicode function and returns an integer value representing either an error code or the success of this VBA function making the call.

**Note:** This VBA function does not return the actual return-value of the Cicode function being called. You can obtain that return value by using the associated [CicodeCallReturn](#) function.

#### **WARNING**

##### **UNINTENDED EQUIPMENT OPERATION**

Do not nest the CicodeCallOpen and CicodeCallReturn functions. Nesting these functions can lead to unintended equipment operation when your program is run.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

For details, see *Calling Cicode from VBA* in the topic Using VBA in Command or Expression Fields.

## Syntax

*ReturnValue* = **CicodeCallOpen(FunctName, ArgList)**

*ReturnValue*:

The return value for the function in the range of 0 to 3.

*FunctName*:

The name of the Cicode function being called.

*Arglist*:

A variable length comma separated argument list of all the arguments to be passed to the Cicode function being opened (dependant upon which Cicode function is being called and the arguments that Cicode function requires). The argument list should not be enclosed within brackets, although when using variable names as arguments, those variable arguments within the list need to be individually enclosed within brackets to force the

passing of the variable to Cicode by value.

## Return Value

CicodeCallOpen returns a integer data type containing a value in the range of 0 to 3:

- 0 if CicodeCallOpenfunction was successful
- 1 for CicodeCallOpenfunction general error
- 2 for specified Cicode function not found
- 3 for incorrect number of arguments for specified Cicode function passed in <ArgList>.

## Related Functions

[CicodeCallReturn](#)

## Example

In the following example, a VBA variable is enclosed in brackets to force the passing of the variable by value. See [Passing Variables Byref and Byval](#).

```
Dim vntRet as Variant
' declare modular variant variable to store function results
Function TestCicode() As Integer
' declare local variables
Dim intRet As Integer
Dim strReply as String
Dim intMaxScale as Integer
' copy current tag value to variable
' uses the project variable tag named MAX_SCALE
intMaxScale = MAX_SCALE
' call Cicode function
' for example: TrnSetScale( AN, Pen, Percent, Scale)
intRet = CicodeCallOpen( "TrnSetScale", 53, -1, 100, (IntMaxScale)
)
' Note the syntax used:
' - brackets around the VBA function argument list.
' (This is only necessary when the VBA function is preceded
by an equals (=) sign .)
' - double quotes around the Cicode function name
' - no brackets around the Cicode function argument list
' - brackets around individual variable arguments
' test results
If intRet = 0 Then
'
    ' insert code for successful completion here
'

    vntRet = CicodeCallReturn()
    strReply = "CicodeCallOpen Function successfully called"
Else
'
    ' insert code for unsuccessful completion here
'
```

```
Select Case intRet
Case = 1
    ' assign return comment for this case
    strReply = "CicodeCallOpen Function call error (unsuccessful)"
Case = 2
    ' assign return comment for this case
    strReply = "Cicode Function not found"
Case = 3
    ' assign return comment for this case
    strReply = "Wrong number of arguments "_
    & "in Cicode CallOpen function call"
Case Else
    ' assign return comment for this case
    strReply = "Unknown error"
End Select
End If
' display return comment for your information
MsgBox strReply
' assign return value for this function
TestCicode = intRet
End Function
```

## CicodeCallReturn

The CicodeCallReturn function is used to obtain the return value of the most recently completed Cicode function opened and run with the VBA CicodeCallOpen function.

No arguments are passed to the CicodeCallReturn function, as it will return the result of the most recent return-value for the Cicode function called by the VBA CicodeCallOpen function.

The CicodeCallReturn function should be used in its own separate line of VBA code and must not be nested with the CicodeCallOpen function. For details, see *Calling Cicode from VBA* in the topic Using VBA in Command or Expression Fields.

## Syntax

*ReturnValue* = **CicodeCallReturn()**

*ReturnValue*:

The return value of the Cicode function specified in the most recent call of the CicodeCallOpen function. Note that the return data type of CicodeCallReturn will depend upon the return data type of the completed Cicode function most recently called by the CicodeCallOpen function.

## Return Value

CicodeCallReturn returns the return-value of the completed Cicode function most recently called by the CicodeCallOpen function.

## Related Functions

[CicodeCallOpen](#)

## Example

```
' declare modular variant variable to store function results
Dim vntRet as Variant
Function TestCicode() As Integer
    ' declare local variables
    Dim intRet As Integer
    Dim strReply as String
    Dim intMaxScale as Integer
    ' copy current tag value to variable
    ' uses the project variable tag named MAX_SCALE
    intMaxScale = MAX_SCALE
    ' call Cicode function
    ' for example: TrnSetScale( AN, Pen, Percent, Scale)
    intRet = CicodeCallOpen( "TrnSetScale", 53, -1, 100, (IntMaxScale)
)
    ' Note the syntax used:
    ' - brackets around the VBA function argument list
    ' - double quotes around the Cicode function name
    ' - no brackets around the Cicode function argument list
    ' - brackets around individual variable arguments
    ' test results
    If intRet = 0 Then
        '
        ' insert code for successful completion here
        '
        vntRet = CicodeCallReturn()
        strReply = "CicodeCallOpen Function successfully called"
    Else
        '
        ' insert code for unsuccessful completion here
        '

        Select Case intRet
        Case = 1
            ' assign return comment for this case
            strReply = "CicodeCallOpen Function call error (unsuccessful)"
        Case = 2
            ' assign return comment for this case
            strReply = "Cicode Function not found"
        Case = 3
            ' assign return comment for this case
            strReply = "Wrong number of arguments " _
                & "in Cicode CallOpen function call"
        Case Else
            ' assign return comment for this case
            strReply = "Unknown error"
        End Select
    End If
    ' display return comment for your information
    MsgBox strReply
    ' assign return value for this function
    TestCicode = intRet
End Function
```

## End Function

The End Function statement ends a program or a block of statements within a function. A VBA function starts with the FUNCTION statement and finishes with the END FUNCTION statement. All other statements that lie between the FUNCTION and END FUNCTION statements will be executed by the function when called to do so.

## Syntax

**End {Function | Sub | If}**

## Related Functions

[Call](#) | [Sub](#) | [End Sub](#) | [Exit](#)

## Example

```
Function GetColor2( c% ) As Long
    GetColor2 = c% * 25
    If c% > 2 Then
        GetColor2 = 255 ' 0x0000FF - Red
    End If
    If c% > 5 Then
        GetColor2 = 65280 ' 0x00FF00 - Green
    End If
    If c% > 8 Then
        GetColor2 = 16711680 ' 0xFF0000 - Blue
    End If
End Function
Sub TestColor2
    Dim I as integer
    For I = 1 to 10
        Print GetColor2(I)
    Next I
End Sub
```

## End Sub

The End Sub statement ends a program or a block of statements within a subroutine. A VBA subroutine starts with the SUB statement and finishes with the END SUB statement. All other statements that lie between the SUB and END SUB statements, will be executed by the subroutine, when called to do so.

## Syntax

**End Sub**

## Related Functions

[Call](#) | [End Function](#) | [Sub](#)

## Example

```
Function GetColor2( c% ) As Long
GetColor2 = c% * 25
If c% > 2 Then
    GetColor2 = 255 ' 0x0000FF - Red
End If
If c% > 5 Then
    GetColor2 = 65280 ' 0x00FF00 - Green
End If
If c% > 8 Then
    GetColor2 = 16711680 ' 0xFF0000 - Blue
End If
End Function
Sub TestColor2
    Dim I as integer
    For I = 1 to 10
        Print GetColor2(I)
    Next I
End Sub
```

## Function

The **Function** statement declares and defines a function procedure, its name, parameters, and code to be enacted upon when the subroutine is called. Functions differ from subroutines in that functions return a value, whereas subroutines do not.

The required *FunctionName* is the name of the function being declared. The optional *ArgList* is the list of arguments used within the function.

A VBA function starts with the **FUNCTION** statement and finishes with the **END FUNCTION** statement. All other statements that lie between the **FUNCTION** and **END FUNCTION** statements will be executed by the function when called to do so.

## Syntax

**Function** *FunctionName* *[(ArgList)]* *[As type]*

## Related Functions

[Call](#) | [End Function](#) | [Sub](#) | [End Sub](#) |

## Example

```
Function GetColor2( c% ) As Long
GetColor2 = c% * 25
If c% > 2 Then
    GetColor2 = 255 ' 0x0000FF - Red
End If
If c% > 5 Then
    GetColor2 = 65280 ' 0x00FF00 - Green
End If
```

```
If c% > 8 Then
    GetColor2 = 16711680 ' 0xFF0000 - Blue
End If
End Function
Sub TestColor2
    Dim I as integer
    For I = 1 to 10
        Print GetColor2(I)
    Next I
End Sub
```

## Sub

Declares and defines a subroutine procedure, its name, parameters, and code to be enacted upon when the subroutine is called. Subroutines differ from functions in that functions return a value, whereas subroutines do not.

The required *SubroutineName* is the name of the subroutine being declared.

The optional *ArgList* is the list of arguments used within the subroutine.

A VBA subroutine starts with the SUB statement and finishes with the END SUB statement. All other statements that lie between the SUB and END SUB statements, will be executed by the subroutine, when called to do so.

## Syntax

**Sub**

## Related Functions

[Call](#) | [End Function](#) | [End Sub](#) | [Exit](#)

## Example

```
Function GetColor2( c% ) As Long
GetColor2 = c% * 25
If c% > 2 Then
    GetColor2 = 255 ' 0x0000FF - Red
End If
If c% > 5 Then
    GetColor2 = 65280 ' 0x00FF00 - Green
End If
If c% > 8 Then
    GetColor2 = 16711680 ' 0xFF0000 - Blue
End If
End Function
Sub TestColor2
    Dim I as integer
    For I = 1 to 10
        Print GetColor2(I)
    Next I
End Sub
```

## VbCallOpen Function

The VbCallOpen function is a Cicode function used to call a VBA function or subroutine from Cicode. It is used to initiate a call to the VBA function or subroutine and returns a handle (of OBJECT data type) to that opened function call.

VbCallOpen is used in conjunction with VbCallRun and VbCallReturn functions, which can all be nested to implement the entire function set with a single line of Cicode. For further information, see the section "Calling VBA from Cicode".

## Syntax

*ReturnValue* = **VbCallOpen**(*FunctName*, *ArgList*)

*ReturnValue*:

The handle to the opened VBA function.

*FunctName*:

The name of the VBA function or subroutine being called.

*ArgList*:

A comma separated list of arguments to pass to the function or subroutine being called.

## Return Value

VbCallOpen returns an Object data type containing a handle to the VBA function being called. If the function cannot open the VBA function or subroutine the return value is zero.

## Related Functions

[VbCallRun Function](#) | [VbCallReturn Function](#)

## Example

```
FUNCTION
TestCitectVBA()
INT iRet;
STRING sMsg = "Hello";
INT iVal = 123;

iRet = VbCallReturn(VbCallRun(VbCallOpen("CiVBATest",
iVal)));
Message("TestCitectVBA Function", "CiVBATest = " +
IntToStr(iRet), 0);
END
```

## Example

```
Function CiVBATest(Value As Integer) As Integer
CiVBATest = Value * 2
End Function
```

## VbCallReturn Function

Used to obtain the return value of the completed VBA function (previously opened with the Cicode VbCallOpen function), and requires the handle returned from the VbCallRun function call.

VbCallReturn is used in conjunction with VbCallOpen and VbCallRun functions, which can all be nested to implement the entire function set with a single line of Cicode. For details, see *Calling VBA from Cicode* in the topic Using VBA in Command or Expression Fields.

## Syntax

*ReturnValue* = **VbCallReturn**(*CallHandle*)

*ReturnValue*:

The value returned by the completed VBA function (which was previously opened by the Cicode VbCallOpen function). The data type of the return value is dependent upon the data type of the return value for the VBA function opened.

*CallHandle*:

The handle to the previously opened VBA function as returned by the Cicode VbCallRun function

## Return Value

VbCallReturn returns the completed return value for the VBA function.

## Related Functions

[VbCallOpen Function](#) | [VbCallRun Function](#)

## Example 1

```
FUNCTION
TestCitectVBA()
INT iRet;
STRING sMsg = "Hello";
INT iVal = 123;

iRet = VbCallReturn(VbCallRun(VbCallOpen("CiVBATest",
iVal)));
Message("TestCitectVBA Function", "CiVBATest = " +
IntToStr(iRet), 0);
END
```

## Example 2

```
Function CiVBATest(Value As Integer) As Integer
CiVBATest = Value * 2
End Function
```

## Example 3

```
FUNCTION VBA_from_Cicode()
    INT iRet;
    WHILE TRUE DO
        iRet = TestFunc("TestVBA")
        Sleep(5);
    END
END
INT FUNCTION TestFunc(STRING sFun)
    RETURN VbCallReturn(VbCallRun(VbCallOpen(sFun)));
END
```

## VbCallRun Function

Used to execute the VBA function or subroutine (previously opened with the Cicode VbCallOpen function), and requires the handle returned from the VbCallOpen function call.

The VbCallRun function provides an opportunity for the opened VBA function to complete and return a value in the multi-threaded Plant SCADA environment. It passes its argument value (of OBJECT data type) through as its return value upon completion.

VbCallRun is used in conjunction with VbCallOpen and VbCallReturn functions, which can all be nested to implement the entire function set with a single line of Cicode. For details, see *Calling VBA from Cicode* in the topic Using VBA in Command or Expression Fields.

## Syntax

*ReturnValue* = **VbCallRun**(*CallHandle*)

*ReturnValue*:

The handle to the opened VBA function passed in as *CallHandle*.

*CallHandle*:

The handle to the previously opened VBA function as returned by the VbCallOpen function.

## Return Value

VbCallRun (passes through and) returns a Object data type containing a handle to the VBA function being called.

## Related Functions

[VbCallOpen Function](#) | [VbCallReturn Function](#)

## Example

```
FUNCTION
TestCitectVBA()
INT iRet;
STRING sMsg = "Hello";
INT iVal = 123;
```

```
iRet = VbCallReturn(VbCallRun(VbCallOpen("CiVBATest",
iVal)));
Message("TestCitectVBA Function", "CiVBATest = " +
IntToStr(iRet), 0);
END
```

## Example

```
Function CiVBATest(Value As Integer) As Integer
CiVBATest = Value * 2
End Function
```

## String Functions

VBA strings functions are provided to create, edit and implement strings within VBA code. The strings functions predefined in VBA are:

<a href="#">Asc</a>	Returns a numeric value that is the ASCII code for the first character in a string.
<a href="#">Chr</a>	Converts an ASCII number to a one character string.
<a href="#">InStr</a>	Returns the character position of the first occurrence of string2 within string1.
<a href="#">LCase</a>	Returns a copy of string in which all characters have been converted to lowercase.
<a href="#">Left, Left\$</a>	Returns the left most characters of a string parameter.
<a href="#">Len</a>	Determines the number of characters in the string argument.
<a href="#">Line Input #</a>	Reads a single line from an open sequential file and assigns it to a string variable.
<a href="#">LTrim</a>	Strips any leading spaces from a string variable.
<a href="#">Mid</a>	Returns a substring within a string.
<a href="#">Option Compare</a>	Determines the default string comparison method.
<a href="#">Option Explicit</a>	Forces explicit declaration of all variables.
<a href="#">Right</a>	Returns the right most characters of a string parameter.
<a href="#">RTrim</a>	Strips any trailing spaces from a string variable.
<a href="#">Space</a>	Adds a specified number of spaces in a print statement.

<a href="#">StrComp</a>	Returns a variant that is the result of the comparison of two strings.
<a href="#">String</a>	Create a string that consists of one character repeated a specific number of times.
<a href="#">Trim</a>	Strips any leading and trailing spaces from Str variable.
<a href="#">UCase</a>	Returns a copy of string in which all characters have been converted to uppercase.

## Asc

Converts a text string character to its numeric ASCII code value. The Asc function expects the argument *Str* to be a valid string expression. If *Str* contains no characters, a runtime error occurs. The Asc function performs the opposite of the [Chr](#) function, which converts a number into its string character ASCII code value.

## Syntax

**Asc(*Str*)**

*Str*:

A string or expression that can represent a valid text value.

## Return Value

Returns the numeric ASCII code value of the first character in *Str* provided in the argument.

## Related Functions

[Chr](#)

## Example

```
Dim vntVar ' declare result holder variable
vntVar = Asc("A")' returns 65
vntVar = Asc("Z")' returns 90
vntVar = Asc("a")' returns 97
vntVar = Asc("z")' returns 122
vntVar = Asc("Apple")' returns 65
vntVar = Asc("Zoe")' returns 90
```

## Chr

Converts a number into its string character ASCII code value.

The Chr function expects the argument *Num* to be a valid numeric integer (whole positive number within the range 0 to 255 inclusive). If *Chr* contains no number, a runtime error occurs.

---

**Note:** Values 8, 9, 10, and 13 convert to backspace, tab, linefeed, and carriage return characters respectively.

The Chr function performs the opposite of the [Asc](#) function, which converts a text string character to its numeric ASCII code value.

## Syntax

**Chr(Num)**

*Num:*

An integer or expression representing a valid numeric value.

## Return Value

Returns a single character string representing the ASCII character code value of the number *Num* provided in the argument.

## Related Functions

[Asc](#)

## Example

```
Dim vntVar ' declare result holder variable
vntVar = Chr(65) ' returns "A"
vntVar = Chr(97) ' returns "a"
vntVar = Chr(90) ' returns "Z"
vntVar = Chr(122) ' returns "z"
```

## InStr

Returns the character position of the first occurrence of *String2* within *String1*.

## Syntax

**InStr(StartPos, StringToSearch, StringToMatch )**

*StartPos:*

A numeric expression that sets the starting position for the search. If omitted, search begins at the first character position. If Num contains Null, an error occurs. An Integer or expression representing a valid numeric value.

*StringToSearch:*

The string expression being searched. A string or expression that can represent a valid text value.

*StringToMatch:*

The string expression being searched for. A string or expression that can represent a valid text value.

## Return Value

Returns a variant containing a Long data type indicating the result of the string search. Returns 0 if:

- *StringToSearch* is of zero length.
- *StringToMatch* is not found.
- *StartPos* is longer than *StringToMatch*.

Returns a value representing the count position where character match was first found.

Returns Null if *StringToSearch* or *StringToMatch* contains null.

## Related Functions

[IsNull](#) | [Left, Left\\$](#) | [Mid](#) | [Right](#) | [StrComp](#)

## Example

```
Dim strToSearch as String
Dim strToFind as String
Dim lngPosition as Long
strToSearch = "Good Bye"
' note this has an uppercase "B"
strToFind = "bye"
' note this has a lowercase "b"
lngPosition = InStr(1, strToSearch, strToFind, 0)
' returns 0 (Did not find match)
lngPosition = InStr(1, strToSearch, strToFind, 1)
' returns 6 (Position of first character in match)
```

## LCase

Converts all uppercase letters in *Str* to lowercase letters. All lowercase letters and non-letter characters remain unchanged.

## Syntax

**LCase(*Str*)**

*Str*:

A string or expression that can represent a valid text value.

## Return Value

Returns a string.

## Related Functions

[UCase](#)

## Example

```
Dim strMixedCase as String
Dim strLowerCase as String
Dim strUpperCase as String
strMixedCase = "AbCdE"
strLowerCase = LCase(strMixedCase) ' returns "abcde"
strUpperCase = UCase(strMixedCase) ' returns "ABCDE"
```

## Left, Left\$

Returns the left most *Num* characters of *Str*.

The required *Str* argument is a String expression from which the leftmost characters are returned. If *Str* contains **Null**, **Null** is returned.

The required *Num* argument is a Variant (Long) numeric expression indicating how many characters to return. If 0, a zero-length string (" ") is returned. If greater than or equal to the number of characters in string, the entire string is returned.

## Syntax

**Left**(*Str*, *Num*)

*Str*:

A string or expression that can represent a valid text value.

*Num*:

An Integer or expression representing a valid numeric value.

## Return Value

The Left function returns a variant containing a String data type. The Left\$ function returns a String.

## Related Functions

[InStr](#) | [Mid](#) | [Right](#)

## Example

```
Dim strGreeting as String
Dim strTest
strGreeting = "Hello World"
strTest = Left(strGreeting, 1) ' Returns "H".
strTest = Left(strGreeting, 7) ' Returns "Hello W".
strTest = Left(strGreeting, 20) ' Returns "Hello World".
```

## Len

The Len function determines the number of characters in the *Str* argument. The LenB function determines the number of bytes in the *VarName* argument.

- The *Str* argument can be any valid string expression. If *Str* contains **Null**, **Null** is returned.
- The *VarName* argument can be any valid variable name. If *VarName* contains **Null**, **Null** is returned. If *VarName* is a Variant, LenB treats it the same as a String and returns the number of characters it contains.

## Syntax

**Len(*Str*)**

*Str*:

A string or expression that can represent a valid text value.

## Return Value

Returns a Long.

## Related Functions

[InStr](#) | [Left](#), [Left\\$](#) | [Mid](#) | [Right](#)

## Example

```
Dim strTest as String
Dim lngStringLength as Long
strTest = "VBA"
lngStringLength = Len(strTest) ' returns 3
```

## Line Input #

Line Input # statement reads a single line from an open sequential file and assigns it to a String variable.

The required *FileNum* argument is a system reference number associated with an open file. The required *VarName* is the name of the variable where the file data is read (copied) to.

---

**Note:** The number sign (#) preceding *FileNum* is not optional.

---

The Line Input # statement reads from a file one character at a time until it encounters a carriage return (Chr(13)) or carriage return-linefeed (Chr(13) + Chr(10)) sequence. Carriage return - linefeed sequences are skipped rather than appended to the character string.

---

**Note:** The file system keeps track of all open files and the current position of access within every file. Every statement or function that accesses the data within a file, alters the current position within that file. The Loc function can be used to determine the current position within an open file.

---

Data read with the Line Input # statement has usually been written to a file with the Print # statement.

## Syntax

**Line Input # FileNum, VarName**

*FileNum:*

An Integer or numeric expression representing any valid number in the range 1 to 511 inclusive, which is referenced by the file system to be associated with an open file.

*VarName:*

A string representing a valid variable name.

## Related Functions

[Get #](#) | [GetAttr](#) | [Input](#) | [Print #](#) | [Put #](#) | [Write #](#)

## Example

```
Dim strTextLine As String
Dim intFileNum As Integer
Open "c:\TEST.txt" For Input As #intFileNum intFileNum = FreeFile
'retrieve next free file number
Do While Not EOF(intFileNum) ' Loop until end of file.
    Line Input #intFileNum, strTextLine ' Read line into variable.
    Print TextLine ' Print line.
Loop
Close #intFileNum
```

## LTrim

Strips any leading spaces from *Str* variable.

## Syntax

**LTrim(*Str*)**

*Str:*

A string or expression that can represent a valid text value.

## Return Value

Returns a string.

## Related Functions

[Trim](#)

## Example

```
Dim strTest as String
Dim strResult as String
Dim lngStartLength as Long
Dim lngFinishLength as Long
strTest = " VBA"
lngStartLength = Len(strTest) ' returns 9
strResult = LTrim(strTest) ' returns "VBA"
lngStringLength = Len(strResult) ' returns 3
```

## Mid

The Mid Function extracts a portion of a string from Str.

---

**Note:** To determine the number of characters in a string, use the Len function.

The *Str* argument can be any valid string expression. If *Str* contains Null, Null is returned.

The required *Num* argument is a Long numeric expression that sets the starting position for the extraction. If *Num* is greater than the number of characters in string, Mid returns a zero-length string ("").

The optional *Len* argument is a Variant containing a Long data type representing the number of characters to return. If omitted or if there are fewer than *Len* characters in *Str* (including the character at position *Num*), all characters from the *Num* position to the end of the string are returned.

## Syntax

**Mid(*Str*, *Num*, *Len*)**

*Str*:

A string or expression that can represent a valid text value. If *Str* contains Null, Null is returned.

*Num*:

A Long numeric expression that sets the starting position for the extraction. If *Num* is greater than the number of characters in string, Mid returns a zero-length string ("").

*Len*:

A Variant containing a Long data type representing the number of characters to return. If omitted or if there are fewer than *Len* characters in *Str* (including the character at position *Num*), all characters from the *Num* position to the end of the string are returned.

## Return Value

The Mid function returns a Variant (containing a String data type).

## Related Functions

[InStr](#) | [Left](#), [Left\\$](#) | [Right](#)

## Example

```
Dim strSource as String
Dim strFirstWord as String
Dim strSecondWord as String
Dim strThirdWord as String
Dim lngPosition as Long
Dim lngNextPosition as Long
Dim lngWordLength as Long
strSource = "Mid Function Demo" ' Create test string.
lngPosition = 1 ' Start at character position 1
lngNextPosition = Instr(lngPosition, strSource, " ") ' Locate first space character
lngWordLength = lngNextPosition - lngPosition ' calculate word length
strFirstWord = Mid(strSource, lngPosition, lngWordLength) ' Returns first word "Mid"
lngPosition = lngNextPosition + 1 ' Move to next word position
lngNextPosition = Instr(lngPosition, strSource, " ") ' Locate next space character
lngWordLength = lngNextPosition - lngPosition ' calculate word length
strSecondWord = Mid(strSource, lngPosition, lngWordLength) ' Returns second word
"Function"
lngPosition = lngNextPosition + 1 ' Move to next word position
lngNextPosition = Instr(lngPosition, strSource, " ") ' Locate next space character
lngWordLength = lngNextPosition - lngPosition ' calculate word length
strThirdWord = Mid(strSource, lngPosition, lngWordLength) ' Returns third word
"Demo"
```

## Option Compare

Determines how strings are compared within a VBA module. The optional Option Compare statement if used, must be placed at the top of the VBA file along with any other Option declarations.

If an Option Compare statement is not included, the default text comparison method is Binary.

## Syntax

**Option Compare {Binary | Text}**

## Related Functions

[InStr](#) | [StrComp](#)

## Example

```
Option Compare Binary
Dim vntResult as Variant
vntResult = StrComp("VBA rules!", "vba Rules!")
' returns 1 (strings unequal)
```

## Example

```
Option Compare Text
Dim vntResult as Variant
```

```
vntResult = StrComp("VBA rules!", "vba Rules!")
' returns 0 (strings equal)
```

## Option Explicit

Forces explicit declaration of all variables.

The optional Option Explicit statement if used, must be placed at the top of the VBA file. This causes a check of variable declarations at compile time. Setting this option is a good way to catch misspelling of variables in your code.

## Syntax

**Option Explicit**

## Related Functions

[Const](#) | [Dim](#)

## Example

```
Option Explicit
' various statements go here
' a compile error will occur with the following line
strMyVar = "This string assignment won't work"
'because strMyVar is not explicitly declared
```

## Right

Returns the right most *Num* characters of *Str*. The required *Str* argument is a String expression from which the rightmost characters are returned. If *Str* contains **Null**, **Null** is returned.

The required *Num* argument is a Variant (Long) numeric expression indicating how many characters to return. If 0, a zero-length string (" ") is returned. If greater than or equal to the number of characters in string, the entire string is returned.

**Note:** To determine the number of characters in a string, use the *Len* function.

## Syntax

**Right(*Str*, *Num*)**

*Str*:

A string or expression that can represent a valid text value.

*Num*:

An Integer or expression representing a valid numeric value.

## Return Value

The Right function returns a variant containing a string data type.  
The Right\$ function returns a string.

## Related Functions

[InStr](#) | [Left](#), [Left\\$](#) | [Mid](#)

## Example

```
Dim strGreeting as String
Dim strTest
strGreeting = "Hello World"
strTest = Right(strGreeting, 1) ' Returns "d"
strTest = Right(strGreeting, 5) ' Returns "World"
strTest = Right(strGreeting, 20) ' Returns "Hello World"
```

## RTrim

Strips any trailing spaces from Str variable.

## Syntax

**RTrim**(*Str*)  
*Str*:  
A string or expression that can represent a valid text value.

## Return Value

Returns a String.

## Related Functions

[LTrim](#) | [Trim](#)

## Example

```
Dim strTest as String
Dim strResult as String
Dim lngStartLength as Long
Dim lngFinishLength as Long
strTest = "VBA "
lngStartLength = Len(strTest) ' returns 9
strResult = RTrim(strTest) ' returns "VBA"
lngStringLength = Len(strResult) ' returns 3
```

## Space

Creates a String consisting of the specified number *Num* of spaces. The Space function is useful for formatting output and clearing data in fixed-length strings.

## Syntax

**Space(Num)**

*Num*:

An Integer or expression representing a valid numeric value.

## Return Value

Returns a Variant containing a String data type.

## Related Functions

[String](#)

## Example

```
Dim strTest as String
' Returns a string with 10 spaces.
strTest = Space(10)

' Insert 10 spaces between two strings.
strTest = "Hello" & Space(10) & "World"
```

## StrComp

Returns an integer that is the result of the comparison of two strings.

The required *String1* argument is any valid string expression. The required *String2* argument is any valid string expression.

The optional *Compare* argument is a numeric expression that specifies the type of string comparison. It can be omitted, 0, or 1. Specify 0 (default) to perform a binary comparison. Specify 1 to perform a textual comparison. If compare is Null, an error occurs.

## Syntax

**StrComp(String1, String2)**

*String1*:

A string or expression that can represent a valid text value.

*String2*:

A string or expression that can represent a valid text value.

## Return Value

Returns a variant containing an integer data type indicating the result of the string compare:

- Returns -1 where *String1* is less than *String2*.
- Returns 0 where *String1* is equal to *String2*.
- Returns 1 where *String1* is greater than *String2*.
- Returns **Null** where *String1* or *String2*s **Null**.

## Example

```
Dim strTest1 as String
Dim strTest2 as String
Dim strTest3 as String
Dim vntComp
strTest1 = "ABCD"
strTest2 = "abcd"
strTest3 = NULL
vntComp = StrComp(strTest1, strTest2) ' Returns -1 (less than)
vntComp = StrComp(strTest1, strTest1) ' Returns 0 (equal to)
vntComp = StrComp(strTest2, strTest1) ' Returns 1 (greater than)
vntComp = StrComp(strTest1, strTest3) ' Returns NULL (strTest3 is NULL)
```

## String

Creates a string that consists of one character repeated a specific number of times.

The required *Num* argument is Long numeric expression indicating how many characters to return. If *Num* contains **Null**, **Null** is returned.

The required *Character* argument is a String expression from which the first character is repeated and returned, or is a Variant (Long) representing a valid character code. If character contains **Null**, **Null** is returned.

## Syntax

**String(Num)**

*Num*:

An Integer or expression representing a valid numeric value.

## Related Functions

[Space](#)

## Example

```
Dim strTest as String
strTest = String(5, "*")
' Returns *****
strTest = String(5, 42)
```

```
' Returns "44444"  
strTest = String(10, "Today")  
' Returns "TTTTTTTTTT"
```

## Trim

Strips any leading and trailing spaces from *Str* variable.

## Syntax

**Trim(*Str*)**

*Str*:

A string or expression that can represent a valid text value.

## Return Value

Returns a String.

## Related Functions

[LTrim](#) | [RTrim](#)

## Example

```
Dim strTest as String  
Dim strResult as String  
Dim lngStartLength as Long  
Dim lngFinishLength as Long  
strTest = " VBA "  
lngStartLength = Len(strTest)  
' returns 9  
strResult = Trim(strTest)  
' returns "VBA"  
lngStringLength = Len(strResult)  
' returns 3
```

## UCase

Converts all lowercase letters in *Str* to uppercase letters. All uppercase letters and non-letter characters remain unchanged.

## Syntax

**UCase(*Str*)**

*Str*:

A string or expression that can represent a valid text value.

## Return Value

Returns a string.

## Related Functions

[LCase](#)

## Example

```
Dim strMixedCase as String
Dim strLowerCase as String
Dim strUpperCase as String
strMixedCase = "AbCdE"
strLowerCase = LCase(strMixedCase) ' returns "abcde"
strUpperCase = UCase(strMixedCase) ' returns "ABCDE"
```

## ASCII/ANSI Character Code Listings

The table below shows the Latin 1 ANSI character set.

Codes 0-31 are control codes. The standard ASCII codes are from 32-127 (decimal) and are common regardless of the ANSI set being used. The remaining codes from 160-255 (decimal) vary between languages dependent upon the ANSI set being used.

Symbol	Decimal	Hex
{NUL}	0	00
{SOH}	1	01
{STX}	2	02
{ETX}	3	03
{EOT}	4	04
{ENQ}	5	05
{ACK}	6	06
{BEL}	7	07
{BS}	8	08
{HT}	9	09
{LF}	10	0A
{VT}	11	0B

Symbol	Decimal	Hex
{FF}	12	0C
{CR}	13	0D
{SO}	14	0E
{SI}	15	0F
{DLE}	16	10
{DC1}	17	11
{DC2}	18	12
{DC3}	19	13
{DC4}	20	14
{NAK}	21	15
{SYN}	22	16
{ETB}	23	17
{CAN}	24	18
{EM}	25	19
{SUB}	26	1A
{ESC}	27	1B
{FS}	28	1C
{GS}	29	1D
{RS}	30	1E
{US}	31	1F
{SPC}	32	20
!	33	21
"	34	22
#	35	23
\$	36	24
%	37	25

Symbol	Decimal	Hex
&	38	26
'	39	27
(	40	28
)	41	29
*	42	2A
+	43	2B
,	44	2C
-	45	2D
.	46	2E
/	47	2F
0	48	30
1	49	31
2	50	32
3	51	33
4	52	34
5	53	35
6	54	36
7	55	37
8	56	38
9	57	39
:	58	3A
;	59	3B
<	60	3C
=	61	3D
>	62	3E
?	63	3F

Symbol	Decimal	Hex
@	64	40
A	65	41
B	66	42
C	67	43
D	68	44
E	69	45
F	70	46
G	71	47
H	72	48
I	73	49
J	74	4A
K	75	4B
L	76	4C
M	77	4D
N	78	4E
O	79	4F
P	80	50
Q	81	51
R	82	52
S	83	53
T	84	54
U	85	55
V	86	56
W	87	57
X	88	58
Y	89	59

Symbol	Decimal	Hex
z	90	5A
[	91	5B
\	92	5C
]	93	5D
^	94	5E
-	95	5F
`	96	60
a	97	61
b	98	62
c	99	63
d	100	64
e	101	65
f	102	66
g	103	67
h	104	68
i	105	69
j	106	6A
k	107	6B
l	108	6C
m	109	6D
n	110	6E
o	111	6F
p	112	70
q	113	71
r	114	72
s	115	73

Symbol	Decimal	Hex
t	116	74
u	117	75
v	118	76
w	119	77
x	120	78
y	121	79
z	122	7A
{	123	7B
	124	7C
}	125	7D
~	126	7E
{Delete}	127	7F
	128	80
	129	81
,	130	82
f	131	83
"	132	84
...	133	85
†	134	86
‡	135	87
^	136	88
%o	137	89
ſ	138	8A
<	139	8B
Œ	140	8C
	141	8D

Symbol	Decimal	Hex
	142	8E
	143	8F
	144	90
`	145	91
'	146	92
"	147	93
"	148	94
.	149	95
-	150	96
-	151	97
~	152	98
™	153	99
š	154	9A
>	155	9B
œ	156	9C
	157	9D
	158	9E
ÿ	159	9F
{NBSP}	160	A0
í	161	A1
¢	162	A2
£	163	A3
¤	164	A4
¥	165	A5
¡	166	A6
§	167	A7

Symbol	Decimal	Hex
..	168	A8
©	169	A9
¤	170	AA
«	171	AB
¬	172	AC
-	173	AD
®	174	AE
-	175	AF
◦	176	B0
±	177	B1
²	178	B2
³	179	B3
'	180	B4
µ	181	B5
¶	182	B6
.	183	B7
,	184	B8
¹	185	B9
º	186	BA
»	187	BB
¼	188	BC
½	189	BD
¾	190	BE
¿	191	BF
À	192	C0
Á	193	C1

Symbol	Decimal	Hex
Â	194	C2
Ã	195	C3
Ä	196	C4
Å	197	C5
Æ	198	C6
Ҫ	199	C7
Ѐ	200	C8
Ӗ	201	C9
Ӯ	202	CA
ӯ	203	CB
Ӱ	204	CC
ӻ	205	CD
Ӵ	206	CE
ӵ	207	CF
ӷ	208	D0
Ӹ	209	D1
ӹ	210	D2
Ӻ	211	D3
ӻ	212	D4
Ӽ	213	D5
ӽ	214	D6
Ӿ	215	D7
ӿ	216	D8
ӻ	217	D9
ӻ	218	DA
ӻ	219	DB

Symbol	Decimal	Hex
Ü	220	DC
Ý	221	DD
Þ	222	DE
ß	223	DF
à	224	E0
á	225	E1
â	226	E2
ã	227	E3
ä	228	E4
å	229	E5
æ	230	E6
ç	231	E7
è	232	E8
é	233	E9
ê	234	EA
ë	235	EB
ì	236	EC
í	237	ED
î	238	EE
ï	239	EF
ð	240	F0
ñ	241	F1
ò	242	F2
ó	243	F3
ô	244	F4
õ	245	F5

Symbol	Decimal	Hex
ö	246	F6
÷	247	F7
ø	248	F8
ù	249	F9
ú	250	FA
û	251	FB
ü	252	FC
ý	253	FD
þ	254	FE
ÿ	255	FF

## Driver Reference Help

The Driver Reference Help is designed to assist you in setting up and configuring device communications for a Plant SCADA system.

The information presented here includes reference and configuration information for each of the drivers included with Plant SCADA.

**Note:** Before you attempt to implement a driver, have a clear understanding of how drivers operate within a Plant SCADA system. Understand the concepts in the [I/O Devices](#) section of the Plant SCADA Help before you start configuring a driver.

Accessing Driver Information		
<b>Locating a particular driver</b> If you already know the driver required to connect to a particular device, you can access the associated reference information by locating the driver's name in the <a href="#">Table Of Contents</a> .	<b>Industry-standard drivers</b> If you are aware that a particular device supports a generic communications protocol, you may find the required configuration information via the list of <a href="#">Industry-Standard Drivers</a> .	<b>Checking for updated drivers</b> To confirm if a new or updated driver is available, go to the <b>Connectivity Hub</b> at the AVEVA Knowledge and Support Center. It is located at <a href="https://softwaresupport.aveva.com">https://softwaresupport.aveva.com</a> .

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

#### DANGER

**DANGER** indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.

#### WARNING

**WARNING** indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.

#### CAUTION

**CAUTION** indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.

#### NOTICE

NOTICE used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.

### Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

## Before You Begin

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

### **WARNING**

#### **LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

## Industry-Standard Drivers

Consider the communication capabilities of a device to determine if it supports any industry-standard protocols, such as Modbus or OPC. If this is the case, you may be able to use one of the following generic Plant SCADA drivers that support these standards.

**Note:** To confirm if a new or updated driver pack is available for a driver, go to the **Communication Drivers** page at the AVEVA Knowledge and Support Center. It is located at <https://softwaresupport.aveva.com/>.

Generic Device Type	Communications Method	Plant SCADA Driver
Modbus (ASCII-based data framing)	Serial	MODBUSA
Modbus (RTU-based data framing)	Serial	MODBUS
Modbus/TCP (Ethernet)	Ethernet	MODNET
Modbus Plus	SA85 Interface Board	MBPLUS
OPC DA Client	OPC via OPC Server (see below)	OPC

**Note:** If you use OPC to connect a device to Plant SCADA, you will need to purchase a third party application to establish an OPC Server. If you take this approach, use an OPC Server application recommended by the manufacturer of the device to which you are trying to connect.

## ABCLX Driver

The ABCLX Driver enables Plant SCADA to connect to [ControlLogix PLCs via Ethernet](#). It can be installed in parallel with a functional ControlLogix system, allowing engineering and testing to be conducted without any impact on operation.

No additional hardware (other than an Ethernet module) is required to connect to the ControlLogix system. This allows you to replace ControlView operator stations on a step-by-step basis.

**Note:** Prior to version 3, the ABCLX driver (then known as the ABLogix driver) was installed as a TCP/IP board driver. Since then, the driver has been installed as an ABCLX-type board driver. If you are upgrading an existing project which used a version 2.x ABLogix driver, you will need to alter the board type in your existing project to ABCLX type. Additionally, the format used for the port definition in the **Special Options** field for the Ports settings has changed.

## See Also

[Data Types](#)

[Adding Tags to Your Project](#)

[Advanced Configuration and Maintenance](#)

[ABCLX Specific Errors](#)

[Logging](#)

## ControlLogix PLCs via Ethernet

Typically, the ControlLogix CPU module you want to connect to will be mounted on the same PLC backplane as Ethernet module your I/O Server communicates with. If not, you must set up a route path to indicate the location of the CPU you want to use.

See [Using a Route Path To Locate a Remote CPU](#).

**Note:** The ABCLX driver will not perform on a slow network, or a network that has an inconsistent connection to an I/O device in the field (for example, a wireless or dial-up connection). This is because the driver downloads all

---

the tags from the PLC and subscribes them before it comes online. Attempting to do this over a slow network can take too long and may cause the driver to time out on startup.

---

## See Also

- [ControlLogix PLCs - Device Address](#)
- [ControlLogix PLCs - Software Requirements](#)
- [ControlLogix PLCs - Communication Settings](#)

### ControlLogix PLCs - Device Address

For ControlLogix PLCs connected via Ethernet, this field must be left blank.

## See Also

- [ControlLogix PLCs via Ethernet](#)

### ControlLogix PLCs - Software Requirements

You will need to have access to Rockwell's RSLogix 5000 Enterprise Series controller configuration software (obtainable from your ControlLogix system supplier) and its associated tools.

This software provides access to controller configuration, project code, tag definitions and data values. Use it to:

- Configure each Ethernet Module IP address
- Create tags for each controller
- Download the projects to the controllers.

---

#### Note:

- To configure Plant SCADA to communicate with ControlLogix system PLCs, you can use RSLogix 5000 to export PLC tag definitions to comma separated value (CSV) files, and use Windows Clipboard to copy them directly into Plant SCADA Studio.
  - Plant SCADA tags can only support arrays of 256 bytes. If your ControlLogix system has arrays larger than this, you might have to subdivide them into segments using the offset and array size in the tag address. See [Mapping to More Than an Individual Element of an Array](#) for more information.
- 

## See Also

- [ControlLogix PLCs via Ethernet](#)

### ControlLogix PLCs - Communication Settings

The following tables show the settings for communication with a ControlLogix device. To establish communication with the device, configure the associated **Board**, **Port** and **I/O Device** in Plant SCADA Studio's **Topology** activity. If you use the Express Communications Wizard to connect to a device, these will be automatically configured for you using the settings outlined below.

## Boards

Field	Value
Board Name	A unique name (up to 16 alphanumeric character) per server for the computer board being used to connect with the device. This is used by Plant SCADA in the Ports settings.
Board Type	The type of board. Select ABCLX from the drop-down list (see Note below).
Address	This field must be "0" (zero).
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.
Comment	Any useful comment.

**Note:** Prior to version 3, the ABCLX driver (then known as the ABLogix driver) was installed as a TCP/IP board driver. Since version 3, the driver has been installed as an ABCLX-type board driver. If you are upgrading an existing project which used a version 2.x ABLogix driver, you will need to alter the board type in your existing project to ABCLX type.

## Ports

Field	Value
Port Name	A unique name (up to 16 alphanumeric character) for the computer port being used to connect with the device. This is used by Plant SCADA in the I/O Device settings.
Port Number	An integer value, unique to the board as defined in the Board name field. The Ethernet port needs no identification; however the Plant SCADA communication database requires a value in this field.
Board Name	The name of the board this port is attached to as defined on the Board settings.
Baud Rate	Leave this field blank.

Field	Value
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank.
Special Options	The IP address of the Ethernet module on the ControlLogix system PLC, appended with a forward slash and the slot number (zero based) of the backplane containing the controller CPU module, in the format: x.x.x.x[/n]. For example: "192.168.0.1/0" (See Note below.)
Comment	Any useful comment.

**Note:** With the release of Version 3 of the ABCLX driver (formerly known as the ABLogix driver), the format used for the port definition in the Special Options field of the Ports settings has changed. If you are upgrading from Version 2.x (ABLogix), you will have to adjust the content of this field.

## I/O Devices

Field	Value
Name	A unique name (up to 16 alphanumeric character) for the I/O Device being identified. Each I/O Device must have a unique name in the Plant SCADA system.
Number	A unique number for the I/O Device (0–16383). Each I/O Device must have a unique number in the Plant SCADA system, (unless redundancy of the I/O Device is being used—See <a href="#">Configuring Redundancy</a> for more information).
Address	Leave this field blank. <b>Note:</b> See <a href="#">Session Control</a> for additional configuration information if you are using the Session Disable feature.
Protocol	This field must be "ABCLX". Default is "ABCLX" when the Plant SCADA Express I/O Device Wizard is used to complete these forms.
Port Name	This field must contain one of the names

Field	Value
	previously defined in the Port Names field of the Port settings. There must be only one I/O device per port.
Comment	Any useful comment. This is viewed in the Devices records database.

## See Also

[ControlLogix PLCs via Ethernet](#)

## Data Types

When configuring your variable tags to communicate with the ControlLogix system, you must select a Plant SCADA data type that is compatible with the data type used by the ControlLogix system controller.

Ideally, you should match the data type of the Plant SCADA variable tag with the data type of the mapped ControlLogix system PLC tag as per the following table:

ControlLogix Data Description	Plant SCADA Address Format	Access rights	Plant SCADA Data Type
BOOL	<TagName>	R/W	DIGITAL
SINT	<TagName>	R/W	BYTE
INT	<TagName>	R/W	INTEGER
DINT	<TagName>	R/W	LONG
REAL	<TagName>	R/W	REAL
STRING	<TagName>	R/W	STRING

For example, if the ControlLogix tag you are addressing uses a "float" data type, you should select "real" in the Data Type field of the Variable Tags settings.

---

**Note:** Plant SCADA reads a string data type from the ControlLogix PLC as a null terminated string. Therefore, if you need to read a 12 character string, you will have to configure the string length on the PLC to at least 13 characters to allow for the null character.

## See Also

[Data Types - Data Coercion](#)

## Data Types - Data Coercion

The ABCLX driver also supports a number of data conversions from a Plant SCADA data type to a native PLC data type. However, as it is possible to coerce from a large data type to a smaller one in both directions, care must be taken so that the range of the smaller type is respected, otherwise the driver will return BAD\_DATA\_TYPE back to Plant SCADA. It is recommended you only rely on these conversions where circumstances demand it.

The following matrix table details the conversions supported by the driver. The plus sign (+) represents those conversions that are impacted by range differences.

PLC DATA TYPES	Plant SCADA DATA TYPES										
	DIGITAL	BYTE	INTEGER	LONG	REAL	DIGITAL ARRAY	BYTE ARRAY	INTEGER ARRAY	LONG ARRAY	REAL ARRAY	STRING
BOOLEAN	YES										
SINT		YES	+ YES	+ YES	+ YES						
INT		+ YES	YES	+ YES	+ YES						
DINT		+ YES	+ YES	YES	+ YES						
REAL		+ YES	+ YES	+ YES	YES						
BOOLEAN ARRAY						YES					
SINT ARRAY							YES	+ YES	+ YES	+ YES	
INT ARRAY							+ YES	YES	+ YES	+ YES	
DINT ARRAY							+ YES	+ YES	YES	+ YES	
FLOAT ARRAY							+ YES	+ YES	+ YES	YES	
STRING											YES

## See Also

[Data Types](#)

## Adding Tags to Your Project

A separate variable tag must be added to your Plant SCADA project for each data-point (memory register) on the I/O devices you want to communicate with.

ControlLogix system PLC controllers use a tag-based addressing system. The tags associated with a controller are each assigned a unique name using the RSLogix 5000 Enterprise Series controller configuration software. (See [ControlLogix PLCs - Software Requirements](#).)

You use the ControlLogix system PLC tag name when you specify the address for a tag in your Plant SCADA project. However, any square braces "[ ]" should be replaced with "{ }", and forward slashes "/" should be used to access bits within words, rather than a period ".".

**Note:** The ABCLX driver supports [Individual Tag Addressing](#) and [Array Tag Addressing](#). [Addressing Program Tags](#) support is also available for both these types of addressing. Local tag addressing is also supported.

Your Plant SCADA variable tags are defined using the **Variable Tags** view. For Plant SCADA to compile correctly, you must include appropriate values for the following fields:

Field	Value
Variable Tag Name	A unique name per project for the variable tag (up to 79 alphanumeric characters in Version 6, 32 characters for earlier versions). This can be used by Plant SCADA graphic pages and in Cicode where defined.
Data Type	The type of variable. Select from the drop-down menu. It must be one of the predefined types used in Plant SCADA.
I/O Device Name	The name of an I/O Device defined in the I/O Devices form which is used to identify the ControlLogix PLC controller (CPU module).
Address	The name of a predefined ControlLogix system PLC tag, either <a href="#">Individual Tag Addressing</a> , <a href="#">Array Tag Addressing</a> or addressed as part of a <a href="#">Addressing Program Tags</a> .

**Note:** Plant SCADA does not indicate any compile errors if an invalid address is entered in the Variable Tag settings. Any variable tag values with an invalid address will display as "#COM" at run time on the graphic pages where they have been positioned.

## See Also

- [Individual Tag Addressing](#)
- [Array Tag Addressing](#)
- [Addressing Program Tags](#)
- [Configuring Quality and Timestamp Tags](#)
- [Addressing Statistics Tags](#)

## Individual Tag Addressing

When declaring a Plant SCADA variable tag using individual tag addressing, type in the ControlLogix tag name, as defined on the ControlLogix PLC, into the **Address** field of the Variable Tag.

## Examples

### Single Byte within a structure

PLC Tag Address	SingleStructure.SingleByte
PLC Tag Type	SINT
Plant SCADA Tag Name	SingleStructure_SingleByte
Plant SCADA Tag Address	SingleStructure.SingleByte

Plant SCADA Tag Type	BYTE
----------------------	------

**Single Digital**

PLC Tag Address	SingleBoolean
PLC Tag Type	BOOLEAN
Plant SCADA Tag Name	SingleBoolean
Plant SCADA Tag Address	SingleBoolean
Plant SCADA Tag Type	DIGITAL

**Single 82nd bit from a digital array**

PLC Tag Address	DigitalArray
PLC Tag Type	BOOL[128]
Plant SCADA Tag Name	SingleDigital_81
Plant SCADA Tag Address	SingleDigital{81}
Plant SCADA Tag Type	DIGITAL

**Single 7th bit from fourth item in a long array**

PLC Tag Address	LongArray
PLC Tag Type	DINT[256]
Plant SCADA Tag Name	SingleBit_3_7
Plant SCADA Tag Address	LongArray{3}/7
Plant SCADA Tag Type	DIGITAL

**See Also**

[Adding Tags to Your Project](#)

**Array Tag Addressing**

Arrays can exist in the ControlLogix system PLC and/or the Plant SCADA tag database.

Within the ControlLogix system PLC, they are defined using the RSLogix 5000 Enterprise Series configuration software. See [ControlLogix PLCs - Software Requirements](#).

Plant SCADA accesses the tag arrays within the ControlLogix system PLCs via the **Address** field of the Variable Tags for each tag or tag array. Equally, the way to define Plant SCADA tag arrays is also to use the Address field of the Plant SCADA Variable Tags for each tag array required in Plant SCADA.

---

**Note:** Plant SCADA tag arrays are simple, not complex, arrays. This means they can only contain elements of the same data type; for example, a digital tag array in Plant SCADA will only include digital type variables.

---

Tag arrays can be extremely flexible and quite complicated. It is possible to declare a tag array in Plant SCADA which maps to a tag array in a ControlLogix system PLC, or which maps to a contiguous sequence of tags (of the same data type) which may or may not be defined as a tag array in the ControlLogix system PLC.

## See Also

- [Mapping to Individual Elements of an Array](#)
- [Individual Tag Address Type Mapping](#)
- [Mapping to More Than an Individual Element of an Array](#)
- [Mapping Data Types for Array Tag Addressing](#)

### Mapping to Individual Elements of an Array

To map to an individual tag in a ControlLogix PLC tag array, use { } brackets to define the array element as a zero-based index. Use the following syntax:

<TagName>{<IndexNumber>}

Where:

<TagName> =	the ControlLogix tag name as defined on the ControlLogix system PLC.
{ } =	the required braces to define the array element index.
<IndexNumber> =	the zero-based index number of the array element.

**Note:** With versions of Citect SCADA prior to version 6, curly brackets { } were required when addressing arrays. This is no longer the case, as Version 6 supports the use of both curly brackets and square brackets [ ].

---

### Individual Tag Address Type Mapping

You should match the data type of the Plant SCADA variable tag with the data type of the mapped ControlLogix system PLC tag as per the following table:

ControlLogix Data Description	Plant SCADA Address Format	Access rights	Plant SCADA Data Type
BOOL	<TagName>{<IndexNumber>}	R/W	DIGITAL
SINT	<TagName>{<IndexNumber>}	R/W	BYTE
INT	<TagName>{<IndexNumber>}	R/W	INTEGER
DINT	<TagName>{<IndexNumber>}	R/W	LONG

	er>}		
REAL	<TagName>{<IndexNumber>}	R/W	REAL
String structure (see below) or STRING (supplied by RSLinx)	<TagName>{<IndexNumber>}	R/W	STRING

The string structure must follow the format:

```
{
DINT LEN
SINT[ ] DATA
}
```

## Examples

### Single integer within an array of integers

PLC Tag Address	IntegerArray
PLC Tag Type	INT[5]
Plant SCADA Tag Name	IntegerArray_5
Plant SCADA Tag Address	IntegerArray{5}
Plant SCADA Tag Type	INTEGER

### Single float within an array of structures

PLC Tag Address	StructureArray[5].SingleFloat
PLC Tag Type	REAL
Plant SCADA Tag Name	StructureArray_5_SingleFloat
Plant SCADA Tag Address	StructureArray{5}.SingleFloat
Plant SCADA Tag Type	REAL

## See Also

[Mapping to Individual Elements of an Array](#)

[Mapping to More Than an Individual Element of an Array](#)

### Mapping to More Than an Individual Element of an Array

Plant SCADA can only handle simple (same-type) arrays of up to 256 bytes in length, so when declaring variable arrays to access ControlLogix system PLC arrays (or data structures larger than 256 bytes), you will need to split

them into smaller structures of less than 256 bytes. For example, you'll need to declare multiple variable arrays each accessing a separate 256 byte portion of the larger structure.

To map to a portion of an array greater than a single tag, you identify the starting position for the array, and the size of the array to map using the syntax:

`<TagName>/<ArrayOffset>!A[<ArraySize>]`

Where:

<code>&lt;TagName&gt; =</code>	The ControlLogix tag name as defined on the PLC.
<code>/&lt;ArrayOffset&gt; =</code>	The zero based offset of the item index used within the PLC (not on bit widths or other sizes) representing the starting point for the tag array value (optional).
<code>!A&lt;ArraySize&gt; =</code>	The zero based array size to return using the item count within the PLC, (not on bit widths or any other sizes).

#### Note:

- The array offset value is a zero-based item count used within the PLC to define the start of the array structure which is being mapped to the Plant SCADA variable tag array being declared. The array size value is a continuation of the same count and is used to define the size of the array structure being mapped to the Plant SCADA variable tag array being declared. They are used together to determine the start and end points of the variable tag array data in the ControlLogix system PLC which will be mapped to for the Plant SCADA variable tag array being declared in the Plant SCADA Variable Tags.
- Plant SCADA ignores all characters following an exclamation mark "!" in the tag address field, so that tag-based drivers like the ABCLX driver can use square brackets in the tag address for device array addressing. Therefore, you must precede an array address schema with an exclamation mark when defining an array in a Plant SCADA variable tag address.

## See Also

[Mapping to Individual Elements of an Array](#)

## Mapping Data Types for Array Tag Addressing

You should match the data type of the Plant SCADA variable tag array with the mapped ControlLogix PLC tag array as per the following table:

ControlLogix Data Description	Plant SCADA Address Format	Access rights	Plant SCADA Data Type
Array of BOOL	<code>&lt;TagName&gt;/&lt;ArrayOffset&gt;!A[&lt;ArraySize&gt;]</code>	R/W	DIGITAL array
Array of SINT	<code>&lt;TagName&gt;/&lt;ArrayOffset&gt;!A[&lt;ArraySize&gt;]</code>	R/W	BYTE array
Array of INT	<code>&lt;TagName&gt;/&lt;ArrayOffset&gt;!A[&lt;ArraySize&gt;]</code>	R/W	INTEGER array

Array of DINT	<TagName>/<ArrayOffset>!A[<ArraySize>]	R/W	LONG array
Array of REAL	<TagName>/<ArrayOffset>!A[<ArraySize>]	R/W	REAL array

## Examples

### Configuration of an array of 128 integers

PLC Tag Address	IntegerArray
PLC Tag Type	INT[128]
Plant SCADA Tag Name	IntegerArray_128
Plant SCADA Tag Address	IntegerArray!A[128]
Plant SCADATag Type	INT

### Configuration of an array of 64 floats offset by 32 items

PLC Tag Address	LargeFloatArray
PLC Tag Type	FLOAT[256]
Plant SCADA Tag Name	LargeFloatArray_32_64
Plant SCADA Tag Address	LargeFloatArray/32!A[64]
Plant SCADA Tag Type	FLOAT

### Configuration of an array of digitals within a structure

PLC Tag Address	SingleStructure.BooleanArray
PLC Tag Type	BOOLEAN[2048]
Plant SCADA Tag Name	SingleStructureBooleanArray
Plant SCADA Tag Address	SingleStructure.BooleanArray!A[2048]
Plant SCADA Tag Type	DIGITAL

### Configuration of an array of 256 bytes, offset by 256 items within an array of structures at element 32

PLC Tag Address	StructureArray.LargeByteArray
PLC Tag Type	SINT[1024]
Plant SCADA Tag Name	StructureArray_32_LargeByteArray_256_256

Plant SCADA Tag Address	StructureArray{32}.LargeByteArray/256!A[256]
Plant SCADA Tag Type	BYTE

## Addressing Program Tags

ControlLogix program tags contain data that is used exclusively by the routines within an individual program. These tags can be viewed as local variables.

Individual tag addressing and array tag addressing are available for program tags by using PROGRAM:<TaskName> as a prefix.

## Examples

### Single Byte within a structure in program MyProgram.

PLC Tag Address	MyProgram.SingleStructure.SingleByte
PLC Tag Type	SINT
Plant SCADA Tag Name	MyProgram_SingleStructure_SingleByte
Plant SCADA Tag Address	PROGRAM:MyProgram.SingleStructure.SingleByte
Plant SCADA Tag Type	BYTE

### Single Digital in program MyProgram

PLC Tag Address	MyProgram.SingleBoolean
PLC Tag Type	BOOLEAN
Plant SCADA Tag Name	MyProgram_SingleBoolean
Plant SCADA Tag Address	PROGRAM:MyProgram.SingleBoolean
Plant SCADA Tag Type	DIGITAL

### Single integer within an array of integers in program MyProgram

PLC Tag Address	MyProgram.IntegerArray[5]
PLC Tag Type	INT
Plant SCADA Tag Name	MyProgram_IntegerArray_5
Plant SCADA Tag Address	PROGRAM:MyProgram.IntegerArray{5}
Plant SCADA Tag Type	INTEGER

## Configuring Quality and Timestamp Tags

The Plant SCADA ABCLX driver can use additional variable tags for quality and time stamp values. Quality or time stamp tags are duplicates of a variable tag, with an appended "!Q" or "!T" or "!M" in the address field to define their purpose. Their data type needs to be Long to enable them to store a quality or time stamp value.

**Note:** The quickest way to create a quality or timestamp tag in Plant SCADA is to locate the relevant variable tag using the Variable Tags dialog box, edit it as required to create a quality or timestamp tag, and save it. This creates a copy of the original variable tag with the required changes saved.

### To create a quality tag:

1. From the Plant SCADA **Studio** select **System Model** activity, then select **Variables**.
2. On the menu below command bar, select **Variables**.
3. Add a row to the grid editor.
4. Type the following information in columns, or in the fields of the **Property Grid**.
  - In the **Tag Name** field, enter an appropriate and unique name for the quality tag. Common practice suggests using a name such as "<Tagname>\_Quality".
  - In the **Data Type** field, select **Long** from the drop-down menu (if not already selected).
  - In the **I/O Device** field, select the appropriate I/O device for the tag (if not already displayed).
  - In the **Address** field, enter the name of the tag you want to associate with the quality value (if not already displayed), and append "!Q" directly to the end of the address (without the quotes and without any spaces).
5. Click **Save**.

**Note:** If there is already an exclamation mark with other declarations included in the tag address, there is no need to include another one. For example, a quality tag monitoring TagName!A[64] can be addressed as TagName!Q.

The quality value generated by the ABCLX driver is a mask of the quality level and quality status field values as per the following table:

Quality level	Quality status	Value (Hex)
QLEVEL_UNCERTAIN	QSTATUS_NONE	0x00
QLEVEL_GOOD	QSTATUS_NOTDEFINED	0x01
QLEVEL_BAD	QSTATUS_INITIAL	0x02
N/A	QSTATUS_STALE	0x03
N/A	QSTATUS_ERRORDATA	0x04
N/A	QSTATUS_WRITE	0x05

**Note:** The quality is calculated by shifting up the quality level by 8 bits and OR masking in the quality status.

## Examples

Quality tag settings for a single long in root of PLC:

Plant SCADA Tag Name	SingleLong_Quality
Plant SCADA Tag Address	SingleLong!Q
Plant SCADA Tag Type	LONG

Quality tag settings for an array of 256 bytes, offset by 256 items within an array of structures at element 32:

Plant SCADA Tag Name	StructureArray_32_.LargeByteArray_256_256_Quality
Plant SCADA Tag Address	StructureArray{32}.LargeByteArray/256!Q
Plant SCADA Tag Type	LONG

### To create a timestamp tag:

- From the Plant SCADA Studio select **System Model** activity, then select **Variables**.
- On the menu below command bar, select **Variables**.
- Add a row to the grid editor.
- Type the following information in columns, or in the fields of the **Property Grid**.
  - In the **Tag Name** field, enter an appropriate and unique name for the timestamp tag. Common practice suggests using a name such as "<Tagname>\_sTimestamp" (for seconds) or "<Tagname>\_msTimestamp" (for milliseconds).
  - In the **Data Type** field, select **Long** from the drop-down menu (if not already selected).
  - In the **I/O Device** field, select the appropriate I/O device for the tag (if not already displayed).
  - In the **Address** field, enter the name of the tag you want to associate with a timestamp value (if not already displayed), and append "!T" or "!M" (as appropriate).
- Click **Save**.

#### Note:

- If there is already an exclamation mark with other declarations included in the tag address, there is no need to include them for timestamping. For example, a timestamp tag monitoring TagName!A[64] can be addressed as TagName!A[64]T.
- Timestamps store the number of seconds since 01/01/1970 when using the "!T" element, or the number of milliseconds since midnight using the "!M" element. If you want to store a complete time in millisecond accuracy, create two separate timestamp tags, one for seconds and one for milliseconds, each addressed appropriately.

## Examples

Timestamp tag settings for a single long in root of PLC:

Plant SCADA Tag Name	SingleLong_msTimestamp
Plant SCADA Tag Address	SingleLong!M
Plant SCADA Tag Type	LONG

Timestamp setting for an array of 256 Bytes offset by 256 items within an array of structures at element 32:

Plant SCADA Tag Name	StructureArray_32_.LargeByteArray_256_256_sTimestamp
Plant SCADA Tag Address	StructureArray{32}.LargeByteArray/256!A[256]T
Plant SCADA Tag Type	LONG

**Note:** A timestamp value is based on the time a value change is detected by the ABCLX driver in UTC time, so the timestamp is only as accurate as the polling cycle of the I/O device. If, for example, the data value changed while the unit was not polling, the timestamp value will not be accurate.

## Addressing Statistics Tags

Statistics tags contain driver statistics data.

Statistics name	Statistics tag address	Data type
AvgGroupPollTime	Statistics_AvgGroupPollTime	LONG
OptimisedBlocks	Statistics_OptimisedBlocks	LONG
SessionThreadState	Statistics_SessionThreadState	LONG
SessionState	Statistics_SessionState	LONG
BackendPendingResp	Statistics_BackendPendingResp	LONG
SocketThreadState	Statistics_SocketThreadState	LONG
CacheDelayedReply	Statistics_CacheDelayedReply	LONG
GroupPollCounter	Statistics_GroupPollCounter	LONG
MissedGroupPolls	Statistics_MissedGroupPolls	LONG
ForwardOpens	Statistics_ForwardOpens	LONG
TagsSubscribed	Statistics_TagsSubscribed	LONG
GoodTags	Statistics_GoodTags	LONG
BadTags	Statistics_BadTags	LONG
QTMTags	Statistics_QTMTags	LONG

SessionDisabled	Statistics_SessionDisabled	LONG
DelayedWrites	Statistics_DelayedWrites	LONG
TagConfigWarnings	Statistics_TagConfigWarnings	LONG
LostMessageCount	Statistics_LostMessageCount	LONG
TimeoutDirWrites	Statistics_TimeoutDirWrites	LONG
InitTimeout	Statistics_InitTimeout	LONG

## Advanced Configuration and Maintenance

This section provides advanced configuration instructions.

- [Using a Route Path To Locate a Remote CPU](#)
- [Configuring Redundancy](#)
- [Performance Tuning](#)
- [Customizing a Project using Citect.ini Parameters](#)
- [Session Control](#)
- [Status Tags](#)
- [Mode Detection](#)
- [Identifying Bad Tags Using 'FailOnBadData'](#)
- [Scan Rates](#)

### Using a Route Path To Locate a Remote CPU

A route path allows you to indicate the location of a particular CPU on a ControlLogix network in relation to the Ethernet module you're using as your initial point of communication. It maps out a communication channel across a network of PLCs by defining a series of ports and addresses that eventually lead to the destination CPU.

This information is configured in the [ABCLX] section of the Citect.ini file, using the following syntax:

*IOServer name.Port name.RoutePath=/P:m/A:n,...,/P:m/A:n*

where:

<i>IOServer name =</i>	I/O server name
<i>Port name =</i>	Port name on I/O server
<i>m =</i>	Port type/number on the module 0 - Reserved 1 - Backplane 2 - Port A 3 - Port B 4 - Port C

*n* =

The slot number if the module is on the backplane, or the address of the module, which will either be a CNET address, IP address or DH address. It can also be an IP address if it is an EIP module

## Example

Starting from the initial Ethernet module, each point in the communication path is defined with a Port value and Address in the following format:

/P:*m*/A:*n*,

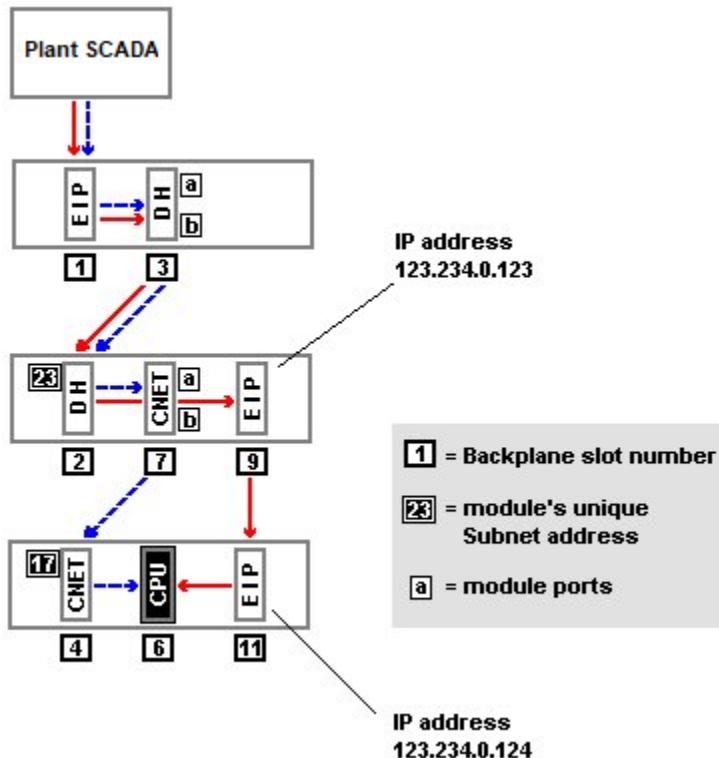
Include an appropriate numeric value for "m" to indicate if the next communication point is on the same backplane, or connected via a particular port. The value for "n" will then become a backplane slot number, or a module's unique address.

If you are attempting to route from one Ethernet module to another, you'll have to use "2" (Port A) for "m", and the destination IP address for "n". For example:

/P:2/A:123.234.0.124,

Simply recreate this syntax for each point in the communication path, separated by commas, to complete the Route Path parameter.

Consider the following diagram, showing a Plant SCADA I/O server, and a network of three ControlLogix PLC backplanes. There are two example route paths represented, one distinguished by a dotted line.



The syntax for the first example (solid line) would be:

IOServerName.PortName.RoutePath=/P:1/A:3,/P:2/A:23,/P:1/A:9,/P:2/A:123.234.0.124,/P:1/A:6

The syntax for the second example would be:

IOServerName.PortName.RoutePath=/P:1/A:3,/P:2/A:23,/P:1/A:7,/P:2/A:17,/P:1/A:6

## Configuring Redundancy

Redundancy is handled in Plant SCADA by the Plant SCADA clients. They determine which I/O server to request I/O device data from. This is done by allowing the scan rate to be specified per I/O device, and then configuring multiple I/O devices to the same PLC.

The inactive unit under redundancy will receive a STATUS\_UNIT command from Plant SCADA every configured WatchTime. The reply sent back to Plant SCADA will be based on the current state of the status tag condition. If the condition isn't met the standby unit will be set to UNIT\_OFFLINE until the status tag condition passes. The status group polling rate remains unchanged on the inactive unit. See [Stop\\_Unit Handling](#).

A status tag can be defined in the Citect.ini file to determine whether or not to set a unit offline or online depending on the condition of the status tag. If the condition is true the unit is allowed to come online, otherwise the unit is set to offline.

The status tag is polled at the same rate configured for the groups of the corresponding unit. The driver supports STATUS\_UNIT commands on the inactive server, so even though a particular unit is not currently servicing Plant SCADA requests, the state of the status condition will be checked periodically allowing the unit to be taken either offline or online.

Both the primary and standby devices will be offline if both have the same status condition and the condition is not passing. All points in the system for that unit would therefore be #COM.

The status tag can be defined either in the main [ABCLX] section to apply to all devices or a status can be defined specifically for a particular device within the [<IOServerName>.<IOPortName>] section.

The inactive unit under redundancy will receive a STATUS\_UNIT command from Plant SCADA every configured WatchTime. The reply sent back to Plant SCADA will be based on the current state of the status tag condition. If the condition isn't met, the standby unit will be set to UNIT\_OFFLINE until the status tag condition passes. The status group polling rate remains unchanged on the inactive unit.

## Stop\_Unit Handling

If the driver receives a **STOP\_UNIT** command from the I/O Server for a particular device, then the driver places the unit offline and adjusts the polling rate based on the inactive scan adjust. When the **INIT\_UNIT** command is received again, the driver will start polling again and place the unit back online.

If the unit is currently offline, the **STOP\_UNIT** command will not be sent down to the driver so the driver will not know to stop polling the PLC.

## Performance Tuning

There are two ways you can fine tune the performance of the ABCLX driver on your Plant SCADA system:

- [Optimizing Driver to PLC Communications](#)
- [Optimizing Client To Server Communications](#)

### Optimizing Driver to PLC Communications

The ABCLX driver connects to the ControlLogix system devices using the Ethernet/IP network. To know the

connection state of all configured PLCs, the driver supports the use of Status Tags for the system and for each device.

To minimize network traffic, the ABCLX driver also supports the use of different scan rates for different devices, and active and inactive devices. See [Status Tags](#) and [Scan Rates](#).

---

**Note:** The ABCLX driver maintains an internal tag cache and services Plant SCADA DCB requests from the internal cache.

You can also adjust the System Overhead Time Slice in the PLC to the highest value possible. The higher you make the number, the better the MSG performance will be. However, this may have an adverse performance impact on the control program being executed by the ControlLogix controller.

The System Overhead Time Slice in the PLC is adjusted from the ControlLogix programming software by right clicking on the controller and navigating to the "Controller Properties" then click on the "Advanced" tab.

It is also good engineering practice to arrange your variable.dbf by IO Device then DataType. This allows Plant SCADA to create requests by blocking the same data types within an IO Device together.

### Optimizing Client To Server Communications

Due to this driver being a front-end/back-end driver, there are no transport delays between Plant SCADA and the driver involved in processing a read request from the driver cache. This means that under heavy loads, the driver could use up most the CPU processing requests. The driver can tune the CPU usage and throughput to help manage the resources of the I/O server. The standard driver parameters MaxPending and Delay can be used as follows:

```
[ABCLX]
MaxPending=x
Delay=y
```

Where:

- $x$  = the maximum number of simultaneous requests the driver will receive from Plant SCADA.
- $y$  = the average delay applied to the requests to control CPU usage.

### Examples

```
[ABCLX]
MaxPending=256
Delay=50
```

The above example would mean that the driver is capable of servicing 256 requests every 50 milliseconds.

```
[ABCLX]
MaxPending=1
Delay=0
```

This example would mean the driver would reply to every request as quickly as possible, which could cause CPU issues.

### Customizing a Project using Citect.ini Parameters

The ABCLX driver supports the following parameter categories:

- ABCLX Specific Parameters
- Logging Parameters.

There are also device-specific parameters you can use to override global parameters. See [Global and Device-Specific Parameters](#).

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

- Do not under any circumstances change or remove any citect.ini parameters without an informed understanding of the possible implications of your actions.
- Do not delete sections of the citect.ini file without an informed understanding of the possible implications of your actions.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** At times during the life cycle of a driver, circumstances may call for the creation of specific citect.ini parameters that remain undocumented. Always seek the advice of Technical Support regarding the possible implementation of undocumented parameters before you make any changes to a citect.ini file.

## **ABCLX Specific Parameters**

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

The following table describes the driver-specific citect.ini parameters.

[ABCLX] Parameter	Allowable Values	Default Value	Description
ConnTimeout	5000 to 30000 (milliseconds)	15000	Timeout for establishing TCP/IP connection.
FailOnBadData	0 or 1	1	Used by the ABCLX driver to determine whether or not to force the display of good data within a block read which contains bad tags during commissioning.  When this parameter is set to 1, a block read

			<p>which contains a bad tag would result in the whole block returning #COM.</p> <p>This can be system wide if placed in the driver's general section, or device specific if placed under a specific device's section.</p> <p>See also <a href="#">Global and Device-Specific Parameters</a>.</p>
ForwardOpenPoolSize	1 - 32	4	<p>Used by the ABCLX driver to determine the number of Forward Open services can be used on the device.</p> <p>This can be system wide if placed in the driver's general section, or device specific if placed under a specific device's section.</p> <p>See also <a href="#">Global and Device-Specific Parameters</a>.</p>
InitTimeOut	0 to 360000 (milliseconds)	180000	Timeout for the retrieval of the tag database and initialization functions.
LostMsgRetry	There is no set limit but the recommended value is between 1-5	3	<p>If the driver does not get a reply to a message sent to the device, this parameter specifies the number of times the driver will retry sending that message before setting the I/O device offline.</p> <p>See Knowledge base article KB5452 for details.</p> <p>This can be system wide if placed in the driver's general section, or device specific if placed under a specific device's section.</p> <p>See also <a href="#">Global and</a></p>

			Device-Specific Parameters.
LostMsgTimeout	There is no set limit but the recommended value is in between 1000 – 2000 if ExclusiveConnection parameter is set to 1, otherwise 300 - 2000 is recommended.	1000 if the ExclusiveConnection parameter is set to 1, otherwise, 300.	This parameter specifies the message timeout (in milliseconds) See Knowledge Base article KB5452 for details. This can be system wide if placed in the driver's general section, or device specific if placed under a specific device's section. See also <a href="#">Global and Device-Specific Parameters</a> .
MaxBackendQueue	1-20	10	Used by the ABCLX driver to restrict the number of writes queued for a device. This can be used in conjunction with the ForwardOpenPoolSize to limit loading on PLCs. This can be system wide if placed in the driver's general section, or device specific if placed under a specific device's section. See also <a href="#">Global and Device-Specific Parameters</a> .

MissedPollTolerance	1 - 5	3	<p>Number of polls that can be missed before the unit is taken offline.</p> <p>This can be system wide if placed in the driver's general section, or device specific if placed under a specific device's section.</p> <p>If this parameter is enabled, the driver will log the following information to the I/O server syslog file:</p> <ul style="list-style-type: none"><li>• All tags downloaded from the PLC (STATISTICS log category).</li><li>• All subscribed Plant SCADA tags (REPORT log category). Note that any tags that do not exist in the PLC are logged under the WARNING debug level.</li><li>• All optimized blocks the driver generates (VALIDATE log category).</li></ul> <p>This can be system wide if placed in the driver's general section, or device specific if placed under a specific device's section.</p> <p>See also <a href="#">Global and Device-Specific Parameters</a>.</p>
---------------------	-------	---	--

OptimizeUserDefinedData Type	0 - 1	1	<p>By default, the optimization for reading structures is enabled (default is 1). In this case ,the driver will retrieve the whole structure in one packet provided the size of the structure is 460 bytes or less.</p> <p>If we disable this INI parameter (set the value to 0), the driver will send the add item request as multiple service requests.</p>
RegisteringTagsToPLCDelayTime	0 to 1000 milliseconds	0	<p>The delay time period (in milliseconds) between requests to register a block of tags.</p> <p>This can be system wide if placed in the driver's general section, or device specific if placed under a specific device's section.</p> <p>See also <a href="#">Global and Device-Specific Parameters</a>.</p>
ScanRate	100 to 30000 milliseconds	500	<p>The time period (in milliseconds) between each poll of the root group.</p> <p>This can be system wide if placed in the driver's general section, or device specific if placed under a specific device's section.</p> <p>See also <a href="#">Global and Device-Specific Parameters</a>.</p>

## Logging Parameters

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Do not under any circumstances change or remove any of the undocumented citect.ini parameters.

Before deleting sections of the citect.ini file, confirm that no undocumented parameters will be deleted.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

[ABCLX] Parameter	Description	Allowable Values	Default Value
DebugLevel	The trace level used for the log file.	Any combination of the following, separated by a pipe character ( ):  ERROR WARN TRACE DEBUG ALL  See <a href="#">Logging</a> for examples.	-
DebugCategory	The categories used for the log file.	Any combination of the following, separated by a pipe character ( ):  TAG= Tag configuration trace SOCK= Socket trace EIP= EIP session trace DCB= Front-end driver trace BUFF= Dump a counted buffer to hex THRD= Thread trace STATISTICS = All tags downloaded from PLC REPORT = All Plant SCADA tags subscribed VALIDATE = All optimized blocks the driver generated ALL = all of the above.  See <a href="#">Logging</a> for examples.	-

LogFileArchive	This parameter has been deprecated.	-	-
LogFileSize	This parameter has been deprecated.	-	-
LogTagInfo	<p>If this parameter is enabled, the driver will log the following information to the I/O server syslog file:</p> <ul style="list-style-type: none"> <li>• All tags downloaded from the PLC (STATISTICS log category).</li> <li>• All subscribed Plant SCADA tags (REPORT log category). Note that any tags that do not exist in the PLC are logged under the WARNING debug level.</li> <li>• All optimized blocks the driver generates (VALIDATE log category).</li> </ul> <p>See also <a href="#">Global and Device-Specific Parameters</a>.</p>	0 or 1	0

## See Also

[Logging](#)

[Global and Device-Specific Parameters](#)

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

Do not change protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

With the current release of the ABCLX driver, there are device-specific parameters you can use to override global

parameters; they are:

- FailOnBadData
- ForwardOpenPoolSize
- InactiveScanAdjust
- LogTagInfo
- LostMsgRetry
- LostMsgTimeout
- MaxBackendQueue
- MissedPollTolerance
- RegisteringTagsToPLCDelayTime
- StatusTag
- ScanRate

For more details, see [ABCLX Specific Parameters](#).

## Examples

Implementation of global parameters:

```
[ABCLX]  
Watchtime=5  
ScanRate=1000
```

Implementation of a device-specific parameter:

```
[CLServer.CLServerPort]  
ScanRate=500
```

## Session Control

You can use session control to release any current references to tags on the ControlLogix PLCs. This is required if you want to modify or delete any of your PLC tags, as the Logix5000 software cannot access a tag if Plant SCADA is currently subscribed to it. You can also use session control to load a refreshed tag list from a PLC.

Session control is achieved by using the **SessionDisable:x** tag. This tag needs to be configured on a virtual device on a virtual channel. The virtual channel is specified by entering 0.0.0.0 as the IP address in the **Special Options** field of the Ports form. There should be a single virtual port and virtual device configured for each I/O server, and the virtual devices should not be configured for redundancy; i.e., the network numbers for the virtual units should not be the same. Otherwise it is not possible to communicate with a specific virtual unit from a Plant SCADA client.

Device Data Description	Plant SCADA Address Format	Access rights	Plant SCADA Data Type
Session Disable and Enable	SessionDisable: x	R/W	INTEGER

Where:

x = the address entered for the specified device in the I/O Devices form. It needs to be unique across all systems. If this tag is written to with a 1, the session is disabled. Writing a 0 to this tag will re-enable the session.

## Status Tags

A status tag can be defined in the Citect.INI file to determine whether or not to set all or any ControlLogix system PLC devices offline or online depending on the condition of the status tag. If the condition is true, the device is allowed to come online; otherwise the device is set to offline by the ABCLX driver.

There can be one general status tag (defined in the general [ABCLX] section of the project Citect.INI file) which determines the online status of all I/O devices using the ABCLX driver. There can also be an individual status tag set for each I/O device (using the syntax [IOServerName.IOPortName]) that will override the general section status tag.

By default, no status tag is created for the driver or a device until defined in the project Citect.INI file.

The status tag is polled at the same rate configured for the groups of the corresponding device. The ABCLX driver supports STATUS\_UNIT commands on the inactive server, so even though a particular device is not currently servicing Plant SCADA requests, the state of the status condition will be checked periodically allowing the device to be taken offline or online. See [Stop\\_Unit Handling](#).

### Note:

- If both the primary and standby devices have the same status condition and the condition does not meet requirements, both will be taken offline. All points in the system for that unit will display "#COM".
- Supported PLC data types for status tags are BOOL, SINT, INT, DINT and REAL. The STRING type is not supported.
- PLC data type REAL cannot be used with bit operator's '&', '!&' to form a status tag expression. The driver will ignore decimal point values in REAL condition checks, the maximum REAL number that can be correctly read is 2^32 for the purposes of the condition check

The syntax for a status tag is:

**StatusTag=<TagName><Operator><Operand>**

where:

<TagName> = the name of the PLC tag address whose value is to be used in the comparison operation;

<Operator> = the comparison operator (see operator list below);

<Operand> = the value to be used in the comparison operation against the value of <TagName>.

The comparison operator can be one of the following:

Operator	Description
>	<TagName> is greater than <Operand>
>=	<TagName> is greater than or equal to <Operand>
<	<TagName> is less than <Operand>
<=	<TagName> is less than or equal to <Operand>
&	<TagName> has common asserted bit(s) with

	<Operand>
=	<TagName> is equal to <Operand>
!&	<TagName> has no common asserted bit(s) with <Operand>
!=	<TagName> is not equal to <Operand>

## Examples

Bring all ABCLX driver devices online only when the integer tag named "TankLevel" is greater than 100:

[ABCLX]

StatusTag=TankLevel>100

Bring any device on the port named "Port1" on the I/O server named "CLServer" online only when the digital tag named "isReady" is true (where 0=false and 1=true):

[CLServer.Port1]

StatusTag=isReady=1

Bring the device named "Port123" on the I/O server named "CLServer" online only when the current value shares common asserted bits with "5".

[CLServer.Port123]

StatusTag=TagName&5

If the current value for the tag called "TagName" was 3, the example above would set the status to true and bring the device online, as 3 and 5 share a common bit (3 = 0011 in binary, and 5 = 0101). If the value for TagName were 2, the device would remain off line as 2 and 5 do not share common bits (2 = 0010, 5 = 0101).

With the example below, the port named "Port123" on "CLServer" will come online only when the current value for TagName does not share common asserted bits with "5".

[CLServer.Port123]

StatusTag=TagName!&5

## Mode Detection

The PLC mode is determined by two things. Firstly, the physical key position in the PLC; and secondly, when in REMOTE mode, whether or not the PLC has been set to RUN mode or PROGRAM mode. This state is represented in the PLC by a numeric value that can be queried.

The following table sets out the different values and the corresponding PLC mode:

Mode Value	Key Position	Mode
0x1060	RUN	RUN
0x3060	REMOTE	RUN
0x3070	REMOTE	PROGRAM
0x2070	PROGRAM	PROGRAM

These values can be used to set up a status tag to determine what mode the PLC is currently in, as shown below.

Status Tag	Configuration Description
@Mode<0x2000	Key position in RUN
@Mode>0x3000	Key position in REMOTE
@Mode!&0x1000	Key position in PROGRAM
@Mode!&0x10	PLC in RUN mode
@Mode&0x10	PLC in PROGRAM mode

Hence the unit can be taken offline or back online depending on the key position and the current mode of the PLC to display #COM for the points on this unit.

### Identifying Bad Tags Using 'FailOnBadData'

A bad tag is defined as any tag that does not exist within the ControlLogix system, which is something that can occur when developing large systems.

During the commissioning phase a project, you need to clearly identify any blocks of data that include bad tags. The FailOnBadData parameter helps achieve this by allowing you to run your project in two different modes:

- When **FailOnBadData** is **off**, the ABCLX driver will not return "#COM" for a block of data if at least one valid value is in the read block.
- If **FailOnBadData** is **on**, if any tag values in a blocked read are missing or inaccessible, the whole block is rejected and will return "#COM". This highlights the presence of bad tags and identifies where corrective action is required. This default behavior is the preferred mode of operation when you are running a project.

By default FailOnBadData is on (1). See [ABCLX Specific Parameters](#).

### Example

Display reads that include a bad tag in the block with #COM:

```
[ABCLX]
FailOnBadData=1
```

Ignore any bad tags in the block reads and just show 0 for those tags:

```
[ABCLX]
FailOnBadData=0
```

### Scan Rates

The Plant SCADA ABCLX driver can define I/O devices with different scan rates. This provides you with the ability to better tune your system, by ordering the tag database into logical groups and defining different scan rates for each group. For example, tags that are being read frequently can be placed in a group that is scanned frequently, and tags that are read infrequently can be placed in a group that is scanned infrequently.

By default, all tags are scanned at the default scan rate defined by the ScanRate parameter in the project

Citect.INI file.

**Note:** If you have configured I/O server redundancy in your project, it may be considered an inefficient use of network bandwidth to be continually polling devices non-critical to the currently inactive server. To help reduce unnecessary network traffic, the ABCLX driver supports the use of the InactiveScanAdjust parameter, which applies a delay factor to the scan rate of currently inactive I/O Servers.

See [ABCLX Specific Parameters](#) for details.

## Examples

Set the project scan rate to 1 second (1000 milliseconds):

```
[ABCLX]  
ScanRate=1000
```

Set the project scan rate to 1 second (1000 milliseconds), and the inactive server scan adjust rate to a factor of 5:

```
[ABCLX]  
ScanRate=1000  
InactiveScanAdjust=5
```

Set a faster (half second) scan rate for a specific port named "Port1" on the I/O server named "CLServer":

```
[CLServer.Port1]  
ScanRate=500
```

Set a slower (5 second) scan rate for a specific port named "Port123" on the I/O port named "CLServer":

```
[CLServer.Port123]  
ScanRate=5000
```

## ABCLX Specific Errors

The following errors, listed in (hexadecimal) order, are specific to the Plant SCADA ABCLX driver:

Error Value (in Hex)	Error Description
0x20000006	Failed to read tag
0x20000007	Failed to write to tag
0x20000009	Failed to initialize channel and download PLC tag database
0x2000000B	Failed to initialize unit and start polling tags
0x2000000C	The configured status condition is currently false
0x20000010	Failed to access the Citect Variable.DBF file
0x20000011	Failed to access the Citect Units.DBF file
0x20000015	Unit initialization is still in progress
0x20000016	The PLC session or TCP/IP connection has failed

0x20000002	Connect Error
0x20000019	Database initialization in progress
0x2000001a	CTAPI server unavailable
0x2000001b	Failed to open Variable.DBF, this error can be logged before Citect V6.
0x2000001c	Failed to open Units.DBF, this error can be logged before Citect V6.
0x2000001d	Direct Read Failure, Time Out
0x2000001e	Write failure, Time Out

## Logging

The ABCLX driver can log combinations of trace levels across different categories. This is achieved by setting the following INI parameters.

### [ABCLX]DebugLevel

This parameter allows you to define the trace level. The options include:

Option	Description
WARN	Output warning messages.
ERROR	Output error messages.
TRACE	Output trace messages.
DEBUG	Output debug messages.
ALL	All of the above.

### [ABCLX]DebugCategory

This parameter allows you to enable logging for a particular category of trace. The options include:

Option	Description
TAG	Tag configuration trace.
SOCK	Socket trace.

EIP	EIP session trace.
DCB	Front end driver trace.
BUFF	Dump a counted buffer to hex.
THRD	Thread trace.
STATISTICS	All tags downloaded from PLC.
REPORT	All subscribed Plant SCADA tag.
VALIDATE	All optimized blocks generated by the driver.
ALL	All of the above.

In both cases, you can use any combination of the available options separated by a pipe character ( | ).

#### Example

```
[ABCLX]
DebugLevel=WARNING | ERROR | TRACE
DebugCategory=ALL
```

In most cases, you can set the DebugLevel and DebugCategory parameters to ALL. If this results in a large, cumbersome log file that causes older data to be overwritten, you can consider refining the debug options.

---

**Note:** These parameters do not have a value specified by default. To enable logging for this driver, you need to manually configure these parameters in your INI file.

The events generated by the ABCLX driver are logged in the following Plant SCADA syslog file:

```
syslog.IOServer.<Cluster name>.<IO Server name>.dat
```

This file is located in the directory specified by the Citect.ini parameter **[CtEdit]Logs**. By default, this location will be:

```
%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Logs
```

The size of the syslog file and its rollover process is configured via the **[Debug]SysLogSize** and **[Debug]SysLogArchive** INI parameters. For more information, see the topic *Debug Parameters* in the Plant SCADA documentation.

---

**Note:** If the number of messages being processed for a driver exceeds the value specified by the Plant SCADA parameter **[Kernel]ErrorBuffers**, the pending messages will not be logged. If this occurs, you can increase the value of this parameter so that all the messages generated by the driver are reported. The number of messages that are not logged can be monitored via the Kernel's General window under the item "Lost Errors". If you adjust **[Kernel]ErrorBuffers** and Lost Errors are still occurring, you may need to adjust the level of system logging.

#### See Also

[Logging Parameters](#)

## Driver Kernel Additional Fields

The Kernel Driver window, launched via the Page Driver Kernel command, displays information about each driver in the Plant SCADA system. The statistics it presents include read requests, physical reads, digital reads per second, register reads, cache reads, error count, timeouts, and so on. See the Plant SCADA User Guide for details on the general statistics it presents.

In the case of the ABCLX driver, the Kernel Driver window includes additional fields that are not normally supported. Press "V" to see these additional fields. They are:

Field	Description
Session Thread State	Current state of the driver EIP session thread: 0 – Thread is stopped 1 – Thread is starting 2 – Thread is running 3 – Thread is stopping
Session State	Current state of the driver EIP session to the PLC: 0 – The session has not been initialized yet 1 – The session is connecting to the PLC and downloading the PLC tag database 2 – The session has connected to the PLC and downloaded the PLC tag database 3 – The session is subscribing the tags to the PLC 4 – The session is currently polling the subscribed tags 5 – The session has terminated unexpectedly and will be restarted
Socket Thread State	Current state of the socket to the PLC (same thread states as Session Thread State above)
Backend Pending Resp	Number of requests currently awaiting response from the PLC
Cache Delayed Reply	Number of cache read requests currently being delayed by driver
Group Poll Counter	Number of polls sent out by the driver
Missed Group Polls	Number of polls that were skipped by the driver
Avg Group Poll Time	Average time in milliseconds for a complete scan of the tags
Forward Opens	Number of Forward Open sessions to the PLC

Optimized Blocks	Number of optimized blocks created by the driver. Should match number of Forward Opens.
Tags Subscribed	Number of non Quality and non Timestamp Plant SCADA tags for this unit
Good Tags	Number of tags that were successfully resolved in the PLC
Bad Tags	Number of tags that were not successfully resolved in the PLC
QTM Tags	Number of Quality and Timestamp tags subscribed to
Session Disabled	Indicates whether session is currently disabled. Value of 1 indicates session is disabled.
Tag Config Warnings	Number of tags that still are considered good but have some issue that needs to be looked at
Init Timeout	Number of connection attempts that have been timed out

## Frequently Asked Questions

### Why do I get a "Channel Offline Cannot Talk" error message when the ABCLX driver is coming online?

There can be several reasons why ABCLX driver cannot bring its devices online at startup. You should initially confirm that the Boards, Ports and I/O Devices dialog boxes are filled out correctly. You should also confirm that you have enough idle time in your PLC program to allow communication requests to take place.

For details on how to diagnose this problem, search the **Tech Notes, FAQ** page of the AVEVA Knowledge and Support Center for article **#4009, ABCLX Does Not Come Online**. This tech note also includes information about parameter settings and adjustments that might impact this process.

### Why does it take so long for the ABCLX driver to come online?

When the ABCLX driver attempts to bring devices online, it initiates a process of downloading and subscribing the required tags from the associated devices. If a project has many tags, this process can overlap with driver polling, causing unnecessary delays. Insufficient idle time in the PLC processor can also cause Plant SCADA driver requests to be prolonged unnecessarily.

For details on how Plant SCADA brings the ABCLX driver online, search the **Tech Notes, FAQ** page of the AVEVA Knowledge and Support Center for article **#4001, How Does Plant SCADA Bring ABCLX Online?**.

## Every time I start Plant SCADA, I see a nuisance hardware alarm for my ABCLX device that becomes inactive after about 30 seconds. How do I stop this happening?

Hardware alarms are generated by Plant SCADA whenever InitUnit times out or reports an error. In the case of ABCLX driver, this can occur because the required tags have not downloaded and completed subscription.

As a result, there is no workaround or solution for suppressing this hardware alarm. However, the speed at which a driver comes online can be impacted by network bandwidth, PLC CPU utilization, and the number of tags and the number of bad tags.

Search the **Tech Notes, FAQ** page of the AVEVA Knowledge and Support Center for articles #4009 and #4001 for details on how to diagnose and fine-tune the startup process.

## Using CTAPI to read tags from an ABCLX device causes blank tags to appear.

Using CTAPI to read tags from a device using the ABCLX driver causes the first tag to display its value correctly, but the rest to appear blank. This is caused by request block changes in CTAPI.

To resolve this, change the Protdir.dbf entry for ABCLX from a Max\_Length of 2048 to 2032, and Bit\_Block from 2048 to 2032 on all Plant SCADA computers on the system. This allows the requested tags to be read correctly; however this reduces the maximum request length from 256 bytes to 254 bytes, which reduces the maximum size of arrays and strings.

Make this change to the protdir.dbf file in the \AVEVA Plant SCADA 2020 R2\bin and the \AVEVA Plant SCADA 2020 R2\user\include directories.

For more details, search the **Tech Notes, FAQ** page of the AVEVA Knowledge and Support Center for article #4069.

## Is it possible to address ControlLogix multidimensional arrays in Plant SCADA if the total length does not exceed 256 bytes?

The ABCLX driver supports multidimensional arrays of up to three dimensions.

## The I/O device does not come online and the "Channel Offline" error is reported

The v18 and v19 firmware revision onwards by Rockwell for its Logix5000 series processors contain changes in the communications protocol between the SCADA and the PLCs. In some circumstances these changes result in the inability for Plant SCADA to communicate with the controller (PLC) and bring tags online using the ABCLX driver version v3.04.16.000 or later. In certain situations, where ABCLX driver configuration is all correct and during runtime Kernel -> Driver page (ABCLX) confirms that there is no bad tags at all and all configured tags are successfully subscribed, still ABCLX fails to bring the device (unit) online and keeps flickering between Starting and Offline states with Init Timeout count increments.

### Solution:

In this situation, make sure External Access is granted (Read or Read/Write) for all tags including Add-On Instructions (AOI) and User-Defined data type (UDT) structure elements in the PLC project. If this is not the case or this cannot be cross-checked, please follow these steps to resolve the issue.

1. In RSLogix5000 project, select Tools > Export > Tags. Ensure that the file name follows the naming convention of PLC\_PROJECT\_NAME-Tags.csv and that the Tags and Logic Comments fields are both set to All.

2. Copy the tag export CSV file into the main SCADA project directory. The ABCLX driver will search for this file in that location only.
3. Once the CSV file is put inside the SCADA project folder, compile the project and run.

Please note that despite the fact External Access is granted (not set to None) for all the tags including Add-On Instructions (AOI) and User-Defined data type (UDT) structure elements in the PLC project, the ABCLX driver log still records following traces (error):

```
2020-12-06 14:14:03.081 +10:00 [TRACE] [EIP ] [0x00005b78] [ProcessPLCProjectName()] Attempting to
open tag export CSV file C:\ProgramData\AVEVA Plant SCADA 2020 R2\User\UDT_Test1\NewTest-Tags.csv
2020-12-06 14:14:03.081 +10:00 [ERROR] [EIP ] [0x00005b78][ProcessPLCProjectName()] Tag export CSV file
C:\ProgramData\AVEVA Plant SCADA <VersionNumber> \User\UDT_Test1\NewTest-Tags.csv could not be
open, the I/O device may not be able to connect due to tags with no external access.
```

Please ignore these messages as long as it is confirmed that External Access is granted (not set to None) to all tags including Add-On Instructions (AOI) and User-Defined data type (UDT) structure elements in the PLC project. If this is not the case, please follow the above-mentioned steps.

For more details, search the **Tech Notes**, **FAQ** page of the AVEVA Knowledge and Support Center for article #6037.

## ABMLXEIP Driver

The ABMLXEIP driver was developed to enable communications between Plant SCADA and Allen-Bradley MicroLogix devices. It is an ABMLXEIP-type board driver, which supports communication using the Encapsulation protocol over EtherNet/IP.

This driver supports reading and writing values for data files of various types, addressed in accordance with the DF1 protocol conventions.

---

**Note:** To communicate with MicroLogix devices via serial, use the **ABDF1FD** driver.

---

## See Also

[Allen-Bradley MicroLogix via Ethernet](#)

[Tag Addressing](#)

[Advanced Configuration and Maintenance](#)

[Troubleshooting](#)

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.
<b>⚠ DANGER</b>	<b>DANGER</b> indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.
<b>⚠ WARNING</b>	<b>WARNING</b> indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.
<b>⚠ CAUTION</b>	<b>CAUTION</b> indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.
<b>NOTICE</b>	NOTICE used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.

## Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

## Before You Begin

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ WARNING****LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

## Allen-Bradley MicroLogix via Ethernet

The ABMLXEIP Driver connects to the Allen-Bradley MicroLogix 1000, 1100, 1200, and 1500 devices.

An Ethernet card and a properly configured EtherNet/IP Interface module is required to enable the driver to communicate with a device.

**Note:** The MicroLogix 1100 device does not require an Ethernet/IP module since it comes with a built-in Ethernet port.

## See Also

[Device Address](#)

[Software Requirements](#)

[Communication Settings](#)

## Device Address

For this device, an address is not required.

The address information is placed in the **Special Options** field in the **Ports** settings by the Express Communications Wizard. In this case, the field is left blank.

## See Also

[Allen-Bradley MicroLogix via Ethernet](#)

## Software Requirements

The following software is required when using the ABMLXEIP driver:

- RSLogix 500 software to create and configure various data files within a MicroLogix device
- ENI utility to configure the EtherNet/IP Interface Module.

---

**Note:** The MicroLogix 1100 requires Rockwell's BOOTP-DHCP server to configure the IP address for the built-in Ethernet port.

---

## See Also

[Allen-Bradley MicroLogix via Ethernet](#)

## Communication Settings

To establish communication with a device, the associated **Board**, **Port** and **I/O Device** need to be correctly configured in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure them, use the information outlined below.

## Boards

Field	Value
Board Name	A unique name (up to 16 alphanumeric characters) per server for the computer board being used to connect with the device. This is used by Plant SCADA in the Ports settings.
Board Type	The type of board. Input ABMLXEIP.
Address	This field must be "0" (zero).
I/O Port	Leave this field blank.

Interrupt	Leave this field blank.
Special Options	Leave this field blank.
Comment	Any useful comment.

## Ports

Field	Value
Port Name	A unique name (up to 16 alphanumeric character) for the computer port being used to connect with the device. This is used by Plant SCADA in the I/O Devices settings.
Port Number	An integer value, unique to the board as defined in the Board Name field. The Ethernet port needs no identification; however, the Plant SCADA communication database requires a value in this field.
Board Name	The name of the board this port is attached to as defined in the Boards settings.
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank.
Special Options	IP Address of EtherNet/IP interface module
Comment	Any useful comment.

## Devices

Field	Value
Name	A unique name (up to 16 alphanumeric character) for the I/O Device being identified. Each I/O Device must have a unique name in the Plant SCADA system.
Number	A unique number for the I/O Device (0–16383).

	Each I/O Device must have a unique number in the Plant SCADA system, (unless redundancy of the I/O Device is being used).
Address	Leave this field blank.
Protocol	Enter ABMLXEIP.
Port Name	This field must contain one of the names previously defined in the Port Names field of the Ports settings. There must be only one I/O device per port.
Comment	Any useful comment.

## See Also

[Allen-Bradley MicroLogix via Ethernet](#)

## Tag Addressing

The address format for reading and writing values to and from MicroLogix PLC controllers depends on which MicroLogix data file you would like to address.

For each data file type, the following conventions apply:

f	file number
e	element number (or slot number for input and output data files)
s	sub-element number (or word number for input and output data files)
b	bit number (or terminal number for input and output data files)

For the allowable range of values, the following conventions apply:

[y - z]	the number range from y to z inclusive
x, [y - z]	the number x and the number range from y to z inclusive.

Each data file type also has correct address formats and a range of allowable values. The file types are:

- [Output Data File](#)
- [Input Data File](#)
- [Status Data File](#)

- Bit Data File
- Timer Data File
- Counter Data File
- Control Data File
- Integer Data File
- Float Data File
- Long Word Data File
- String Data File

## Output Data File

Plant SCADA Type	Tag Address Rule	Allowable Value Ranges
INT	Of:e.s	f - file number: 0, optional e - slot number: [0-30] s - word number: [0-255]
DIGITAL	Of:e.s/b	f - file number: 0, optional e - slot number: [0-30] s - word number: [0-255] b - terminal number: [0-15]

## Examples

Examples	Meaning
O:1.3	Output data file, slot 1, word 3
00:1.3	Output data file 0, slot 1, word 3
O:1.3/2	Output data file, slot 1, word 3, terminal 2
00:1.3/2	Output data file 0, slot 1, word 3, terminal 2

## See Also

[Tag Addressing](#)

## Input Data File

Plant SCADA Type	Tag Address Rule	Allowable Value Ranges
INT	If:e.s	f - file number: 1, optional e - slot number: [0-30] s - word number: [0-255]
DIGITAL	If:e.s/b	f - file number: 1, optional e - slot number: [0-30] s - word number: [0-255] b - terminal number: [0-15]

## Examples

Examples	Meaning
I:1.3	Input data file, slot 1, word 3
I1:1.3	Input data file 1, slot 1, word 3
I:1.3/2	Input data file, slot 1 word 3, terminal 2
I1:1.3/2	Input data file 1, slot 1, word 3, terminal 2

## See Also

[Tag Addressing](#)

## Status Data File

Plant SCADA Type	Tag Address Rule	Allowable Value Ranges
INT	Sf:e	f - file number: 2, optional e - element number: [0-64]
DIGITAL	Sf:e/b	f - file number: 2, optional e - element number: [0-64] b - bit number: [0-15]

## Examples

Examples	Meaning
S:4	Status data file, element 2

S2:4	Status data file 2, element 4
S:54/6	Status data file, element 54, bit 6
S2:54/6	Status data file 2, element 54, bit 6

## See Also

[Tag Addressing](#)

## Bit Data File

Plant SCADA Type	Tag Address Rule	Allowable Value Ranges
INT	Bf:e	f - file number: 3, optional e - element number: [0-64]
DIGITAL	Bf:e/b	f - file number: 3, optional e - element number: [0-64] b - bit number: [0-15]
DIGITAL	Bf/b	f - file number: 3,[9-255] b - bit number: [0-4095]

## Examples

Examples	Meaning
B3:4	Bit data file 3, element 4 (i.e. no bit number indicates the entire word of bits)
B3:4/7	Bit data file 3, element 4, bit 7
B10/1001	Bit data file 10, bit 1001

## See Also

[Tag Addressing](#)

## Timer Data File

Plant SCADA Type	Tag Address Rule	Allowable Value Ranges
INT	Tf:e.s Tf:e.PRE/0 Tf:e.ACC/0	f - file number: 4,[9-255] e - element number: [0-255] s - sub-element number: [0-2] "/0" is optional
DIGITAL	Tf:e/b Tf:e/DN Tf:e/TT Tf:e/EN	f - file number: 4,[9-255] e - element number: [0-255] b - bit number: [0-15]

## See Also

[Tag Addressing](#)

## Counter Data File

Plant SCADA Type	Tag Address Rule	Allowable Value Ranges
INT	Cf:e.s Cf:e.PRE/0 Cf:e.ACC/0	f - file number: 5,[9-255] e - element number:[0-255] s - sub-element number: [0-2] "/0" is optional
DIGITAL	Cf:e/b Cf:e/UA Cf:e/UN Cf:e/OV Cf:e/DN Cf:e/CD Cf:e/CU	f - file number: 5,[9-255] e - element number:[0-255] b - bit number: [0-15]

## Examples

Examples	Meaning
C5:1.0	Counter data file 5, element 1, sub-element 0
C5:1.PRE	Counter data file 5, element 1, sub-element 1

	(equivalent to PRE)
C5:1.ACC/0	Counter data file 5, element 1, sub-element 2 (equivalent to ACC)
C5:2/11	Counter data file 5, element 2, bit number 11
C5:2/UN	Counter data file 5, element 2, bit number 11 (equivalent to UN)

## See Also

[Tag Addressing](#)

## Control Data File

Plant SCADA Type	Tag Address Rule	Allowable Value Ranges
INT	Rf:e.s Rf:e.LEN/0 Rf:e.POS/0	f - file number: 6,[9-255] e - element number: [0-255] s - sub-element number: [0-2] "/0" is optional
DIGITAL	Rf:e/b Rf:e/FD Rf:e/IN Rf:e/UL Rf:e/ER Rf:e/EM Rf:e/DN Rf:e/EU Rf:e/EN	f - file number: 6,[9-255] e - element number: [0-255] b - bit number: [0-15]

## Examples

Examples	Meaning
R6:1.0	Control data file 6, element 1, sub-element 0
R6:1.LEN	Control data file 6, element 1, sub-element 1 (equivalent to LEN)
R6.2.POS/0	Control data file 6, element 2, sub-element 2

	(equivalent to POS)
R6:2/15	Control data file 6, element 2, bit 15
R6:2/EN	Control data file 6, element 2, bit 15 (equivalent to EN)

## See Also

[Tag Addressing](#)

## Integer Data File

Plant SCADA Type	Tag Address Rule	Allowable Value Ranges
INT	Nf:e	f - file number: 7,[9-255] e - element number: [0-255]
DIGITAL	Nf:e/b	f - file number: 7,[9-255] e - element number: [0-255] b - bit number: [0-15]

## Examples

Examples	Meaning
N22:5	Integer data file 22, element 5
N22:5/7	Integer data file 22, element 5, bit 7

## See Also

[Tag Addressing](#)

## Float Data File

Plant SCADA Type	Tag Address Rule	Allowable Value Ranges
REAL	Ff:e	f - file number: 8,[9-255] e - element number: [0-255]

## Examples

Examples	Meaning
F8:0	Float data file 8, element 0

## See Also

[Tag Addressing](#)

## Long Word Data File

Plant SCADA Type	Tag Address Rule	Allowable Ranges
LONG	Lf:e	f - file number: [9-255] e - element number:[0-255]

## Examples

Examples	Meaning
L16:10	Long Word data file 16, element 10

## See Also

[Tag Addressing](#)

## String Data File

Plant SCADA Type	Tag Address Rule	Allowable Value Ranges
INT	STf:e.s STf:e.LEN STf:e.DATA[n]	f - file number: [9-255] e - element number: [0-255] s - sub-element number: [0-41] n - a pair of characters (high & low bytes): [0-40]
STRING	STf:e.DATA	f - file number: [9-255] e - element number: [0-255] <b>Note:</b> This tag would cover the entire string contents (excluding length byte)

## Examples

Examples	Meaning
ST9:1.0	String data file 9, element 1, sub-element 0
ST9:1.LEN	String data file 9, element 1, sub-element 1 (which is LEN)
ST9:3.DATA	String data file 9, element 3, entire string contents
ST9:1.DATA[2]	String data file 9, element 1, characters 5 and 6 [*see table below]

The DATA[x] string can specify certain characters only as shown in the following table:

DATA[0]	Indicates characters 1 and 2
DATA[1]	Indicates characters 3 and 4
DATA[2]	Indicates characters 5 and 6
etc...	etc...

## See Also

[Tag Addressing](#)

## Advanced Configuration and Maintenance

This section provides instructions for advanced configuration and maintenance tasks. It includes the following topics:

[Configuring Redundancy](#)

[Optimizing Communications](#)

[Customizing a Project using Citect.ini Parameters](#)

## Configuring Redundancy

In order to support Plant SCADA redundancy, at least two I/O devices need to be configured for the same PLC, i.e. one as Primary and all others as Standby. While the primary I/O device services Plant SCADA requests, all standby I/O devices are polled for their status every configured **WatchTime** to determine whether they are online.

Both the Primary and Standby devices will be offline if both have the same inoperable status condition. All tags in the system for that unit would be #COM.

## Optimizing Communications

As the ABMLXEIP driver is a front-end/back-end driver, there are no transport delays when processing read requests between Plant SCADA and the driver cache. If many requests are being serviced simultaneously, the CPU of the I/O Server may be pushed to its performance limitations.

This may lead to communication problems, evidenced by #COM appearing occasionally on graphics pages, or "Command canceled" errors appearing in the Kernel Window.

If this happens, you can tune the driver throughput to help manage the resources of the I/O server. The standard driver parameters you can use to achieve this are **MaxPending** and **Delay**.

MaxPending limits the number of requests handled simultaneously, and can be used as follows:

[ABF1FD]MaxPending=x

where:

x = the maximum number of requests the driver can service simultaneously.

You can also implement a time delay to steady the flow of requests to the driver, rather than having them sent in bursts. The standard driver parameter Delay can be used as follows:

[ABMLXEIP]Delay=x

where:

x = the time delay (in milliseconds) between the delivery of each message.

The two can be used in tandem to maximize the number of requests managed simultaneously, while improving system stability.

## Examples

[ABF1FD]MaxPending=1

The above example would mean the driver would service just one request at a time, and would wait for a reply to each request before the next one is sent.

[ABF1FD]MaxPending=256

The above example would mean that the driver is capable of servicing up to 256 requests at the same time. This means up to 256 requests can be sent before waiting for a response.

[ABMLXEIP]Delay=20

This example would result in a 20 millisecond delay between each request sent, slowing the rate of delivery to a manageable pace.

---

**Note:** MaxPending=1 would make the Delay parameter redundant, as only one request would be handled at a time.

## Customizing a Project using Citect.ini Parameters

The ABMLXEIP driver supports the following parameter categories:

- Common Parameters
- Specific Parameters
- Logging Parameters.

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

- Do not under any circumstances change or remove any citect.ini parameters without an informed understanding of the possible implications of your actions.
- Do not delete sections of the citect.ini file without an informed understanding of the possible implications of your actions.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** At times during the life cycle of a driver, circumstances may call for the creation of specific citect.ini parameters that remain undocumented. Always seek the advice of Technical Support regarding the possible implementation of undocumented parameters before you make any changes to a citect.ini file.

**Common Parameters****⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

Parameters grouped beneath the [ABLMXEIP] category in the citect.ini file are global and impact MicroLogix system PLCs.

The following table describes the common citect.ini driver parameters.

[ABLMXEIP] Parameter	Allowable Values	Default Value	Description
Block	1 to 248	248	A value (bytes) used by the Plant SCADA I/O server to determine if two or more packets can be blocked into one data request before being sent to the ABMLXEIP driver. Do not make this value greater than 248, as this is the maximum number of bytes that can be requested from the device. For this setting to have a bearing, the PROTDIR.DBF file will

			need to be altered.
Delay	0 to 300	20	The time delay (in milliseconds) between the delivery of each message. Use this parameter to avoid messages being sent in bursts.
MaxPending	1 to 256	5	The maximum number of pending commands that the Plant SCADA ABMLXEIP driver holds ready for immediate execution.
Retry	0 to 5	3	The number of times to retry a command after a timeout. This is also the maximum number of total retries before the device is considered offline.
Timeout	100 to 30000	3000	Specifies how many milliseconds to wait for a response before considering a write or polling request to be invalid and displaying an error message in Plant SCADA. Connection timeout will be TimeOut * 3.
Watchtime	5 to 30	30	The period (in seconds) that the Plant SCADA ABMLXEIP driver uses to check the communications link to the MicroLogix system and attempts to bring offline units back online. Also determines the rate of STATUS_UNIT commands to the driver. See <a href="#">Configuring Redundancy</a> .

## See Also

[Customizing a Project using Citect.ini Parameters](#)

### Specific Parameters

#### **WARNING**

##### **UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

The following table describes the driver-specific Citect.ini parameters.

[ABLMXEIP] Parameter	Allowable Values	Default Value	Description
ENIOnlyMode	1 - enabled 0 - disabled	0	This parameter enables a communication method that works exclusively with an ENI Ethernet/IP module. It does not support communication with the built-in port on a MicroLogix 1100.  You should only enable this parameter if you are experiencing difficulties communicating with a device (other than an MicroLogix 1100) via an Ethernet/IP module.

## See Also

[Customizing a Project using Citect.ini Parameters](#)

### Logging Parameters

#### **WARNING**

##### **UNINTENDED EQUIPMENT OPERATION**

Do not under any circumstances change or remove any of the undocumented citect.ini parameters.

Before deleting sections of the citect.ini file, confirm that no undocumented parameters will be deleted.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

[ABMLXEIP] Parameter	Description	Allowable Values	Default Value
DebugLevel	The trace level used for the log file.	Any combination of the following, separated by a pipe character ( ): ERROR WARN TRACE ALL See <a href="#">Logging</a> for examples.	-
DebugCategory	The categories used for the log file.	Any combination of the following, separated by a pipe character ( ): TAG= Tag configuration trace SOCK= Socket trace EIP= EIP session trace DCB= Front-end driver trace BUFF= Dump a counted buffer to hex THRD= Thread trace ALL = all of the above. See <a href="#">Logging</a> for examples.	-

## See Also

[Customizing a Project using Citect.ini Parameters](#)

## Troubleshooting

The following topics provide information about the tools available to diagnose and resolve any unexpected behavior from the ABMLXEIP driver at runtime:

- [Logging](#)
- [Specific Errors](#)
- [Kernel Diagnostics.](#)

## Logging

The ABMLXEIP driver can log combinations of trace levels across different categories. This is achieved by setting the following INI parameters.

### [ABMLXEIP]DebugLevel

This parameter allows you to define the trace level. The options include:

Option	Description
WARN	Output warning messages.
ERROR	Output error messages.
TRACE	Output trace messages.
ALL	All of the above.

### [ABMLXEIP]DebugCategory

This parameter allows you to enable logging for a particular trace category. The options include:

Option	Description	Notes
TAG	Tag configuration trace.	N/A for ABMLXEIP driver
SOCK	Socket trace.	
EIP	EIP session trace.	
DCB	Front end driver trace.	
BUFF	Dump a counted buffer to hex.	
THRD	Thread trace.	
ALL	All of the above.	

In both cases, you can use any combination of the available options separated by a pipe character ( | ).

#### Example

```
[ABMLXEIP]
DebugLevel=WARNING|ERROR|TRACE
DebugCategory=ALL
```

In most cases, you can set the DebugLevel and DebugCategory parameters to ALL. If this results in a large, cumbersome log file that causes older data to be overwritten, you can consider refining the debug options.

---

**Note:** These parameters do not have a value specified by default. To enable logging for this driver, you need to manually configure these parameters in your INI file.

The events generated by the ABMLXEIP driver are logged in the following Plant SCADA syslog file:

syslog.IOServer.<Cluster name>.<IO Server name>.dat

This file is located in the directory specified by the Citect.ini parameter [**CtEditLogs**]. By default, this location will be:

%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Logs

The size of the syslog file and its rollover process is configured via the [**DebugSysLogSize**] and [**DebugSysLogArchive**] INI parameters. For more information, see the topic *Debug Parameters* in the Plant SCADA documentation.

**Note:** If the number of messages being processed for a driver exceeds the value specified by the Plant SCADA parameter [Kernel]ErrorBuffers, the pending messages will not be logged. If this occurs, you can increase the value of this parameter so that all the messages generated by the driver are reported. The number of messages that are not logged can be monitored via the Kernel's General window under the item "Lost Errors". If you adjust [Kernel]ErrorBuffers and Lost Errors are still occurring, you may need to adjust the level of system logging.

## See Also

[Troubleshooting](#)

## Specific Errors

The following errors, listed in (hexadecimal) sequence, are specific to the Plant SCADA ABMLXEIP protocol:

Error Value	Error Definition
0x01000	Bad request from SCADA – device cannot service the request (device error)
0x10001	Bad response to ECHO command from the device (driver error)
0x10002	Bad response to READ command from the device (driver error)
0x10003	Bad response to WRITE command from the device (driver error)
0x10004	Unrecognized message from the device (driver error)
0x10006	Device cannot process requests at the current rate of delivery. Increase the Delay parameter value, or decrease the MaxPending parameter value. See <a href="#">Optimizing Communications</a> .

You may require additional information to enable you to rectify an error. This information should be detailed in the documentation that accompanied the MicroLogix system.

## See Also

[Troubleshooting](#)

## Kernel Diagnostics

The Plant SCADA Kernel Driver window, launched via the Page Driver Kernel command, displays information about each driver in the Plant SCADA system. The statistics it presents include: read requests; physical reads; digital reads per second; register reads; cache reads; error count; timeouts; etc. See the Plant SCADA User Guide for details on the general statistics it presents.

The following ABMLXEIP driver statistics are maintained and displayed per channel:

Driver Statistics Entry	Description
Total Requests	Total requests made to the device
Total Req Bytes	Length of all requests made (in bytes)
Min Bytes per Req	Minimum request length (in bytes)
Max Bytes per Req	Maximum request length (in bytes)
Ave Bytes per Req	Average request length (in bytes)
Total Replies	Total replies received from the device
Total Rep Bytes	Length of all replies received (in bytes)
Min Bytes per Rep	Minimum reply length (in bytes)
Max Bytes per Rep	Maximum reply length (in bytes)
Ave Bytes per Rep	Average reply length (in bytes)
Total Connects	Total number of successful connects and reconnects to the device
Last Req in Queue	Number of requests sent to the device during the last attempt
Max Req in Queue	Maximum number of requests sent to the device at one time
Min Wait (ms)	Minimum wait for a response from the device (in milliseconds)
Max Wait (ms)	Maximum wait for a response from the device (in milliseconds)
Ave Wait (ms)	Average wait for a response from the device (in milliseconds)

## See Also

[Troubleshooting](#)

## ABTCP Driver

The ABTCP protocol supports TCP/IP and Data Highway Plus communication with Allen-Bradley PLC-5/250, PLC-2, PLC-3, SLC-500, and PLC-5 series PLCs.

No additional software is required by when Data Highway Plus is used with the KT, KTX, KTXD, PKTX cards, or the 5136-SD-ISA or 5136-SD-PCI boards. Unlike the ABEI protocol, the Allen-Bradley Interchange software is not required.

When using TCP/IP, the physical communication layer is usually Ethernet. This requires an Ethernet board installed in your computer, and an Ethernet Interface with a transceiver installed at the PLC end. It is recommended you setup your specific Ethernet card as required and confirm basic network communications before proceeding with the communications configuration for the ABTCP driver.

The same Ethernet card that is being used for communication can be used for PLC communication

The Resource Manager of the PI can provide access to Data Highway (DH/DH+) links, permitting to communicate through the PI to PLC-5s, PLC-2s, PLC-3s, and SLC-500s (via offlink addressing). An AB-KA module provides similar indirect communication through PLCs.

The maximum request length for the ABTCP protocols is 1888 bits.

Other I/O Devices may also be supported. Contact Technical Support for further information.

## Hot Standby Support

The ABTCP protocol supports hot standby in the case that the PLC is put into **program** mode. See the [\[ABTCP\]Status](#) parameter.

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

**⚠ DANGER**

**DANGER** indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.

**⚠ WARNING**

**WARNING** indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.

**⚠ CAUTION**

**CAUTION** indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.

**NOTICE**

NOTICE used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.

**Please Note**

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

**Before You Begin**

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ WARNING****LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

## Supported Devices via TCP/IP

The following devices are supported by the ABTCP driver via TCP/IP:

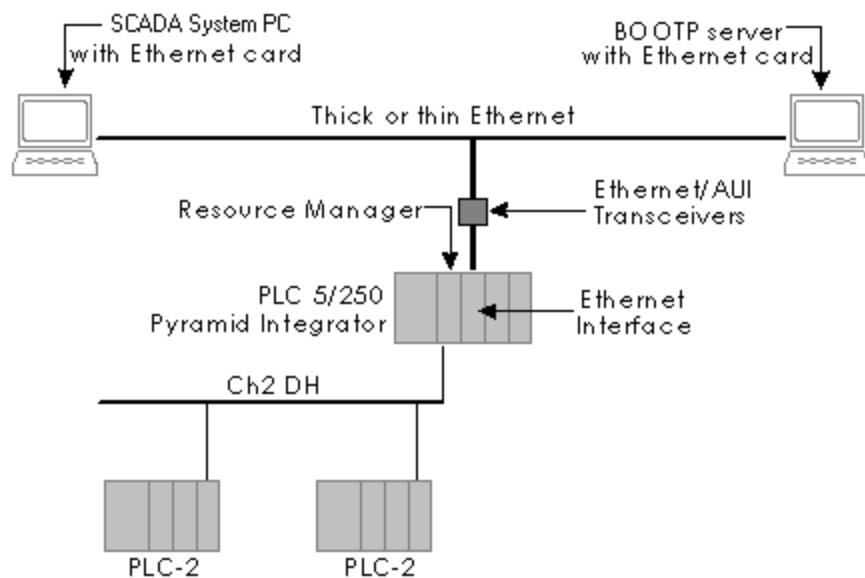
- Allen-Bradley PLC-2
- Allen-Bradley PLC-3
- Allen-Bradley PLC-5
- Allen-Bradley PLC-5/250
- Allen-Bradley SLC-500

**Note:** You can communicate with these devices directly using TCP/IP, or via a [Data Highway Board](#).

### Allen-Bradley PLC-2 via TCP/IP

The information included in the following pages describes how to connect Allen-Bradley PLC-2 devices using the ABTCP protocol to a Plant SCADA system.

Using this method, you can connect to multiple PLCs as in the following diagram:



You cannot make a direct physical Ethernet connection to a PLC-2. An Ethernet Interface (EI) must be provided - typically by using a 5/250 Pyramid Integrator.

The Allen-Bradley Ethernet arrangement requires a BOOTP server. The BOOTP server is required by the EI modules to download configuration information. Refer to your Allen-Bradley documentation for BOOTP information.

The diagram above does not show all of the possible connection options - consult your Allen-Bradley manuals for more information.

#### Allen-Bradley PLC-2 via TCP/IP - Device Address

The address for this device uses the following format:

`-la-Pn -T`

where:

*a* = the destination IP address in standard Internet dot format. (For example, 192.9.2.60)

*n* = the destination Port number. Often one physical port has several virtual ports, used for different purposes. Use this option only if you want to override the default of 2222.

*-T* = forces the driver to use TCP (the default), rather than UDP (-U).

This information is placed in the **Special Options** field of Plant SCADA's **Ports** settings by the Express Communications Wizard.

#### Allen-Bradley PLC-2 via TCP/IP - Hardware Setup

Because this driver does not receive information regarding the state of the physical transport layer, no specific details can be given for setting up communications. Follow these general steps:

1. Set up the PLC 5/250 resource manager or AB-KA module. Assign the IP address, and test that it is valid by using the Ping program.
2. Set up the PLC-2 and 1771-KA/KA2 for communications on the DH network to which it is connected. Be sure to set the Node Address, Baud Rate, and Write Options on each PLC communication module.

3. Check that you can communicate to the PLC using offlink addressing. You can use the utility accompanying the Allen-Bradley Interchange software to verify that you can communicate from a PC over Ethernet.

## Plant SCADA Computer Setup

Setup your specific Ethernet card as required and confirm basic network communications before proceeding with the communications configuration for the ABTCP driver. Refer to the documentation accompanying your hardware for instruction. You must also setup TCP/IP protocol on your Plant SCADA computer.

### Allen-Bradley PLC-2 via TCP/IP - Communication Settings

To establish communication with a device, the associated Board, Port and I/O Device need to be correctly configured in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure them, use the settings outlined below.

## Boards

Field	Value
Board Type	Enter TCPIP.
Address	Enter 0.
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.

## Ports

Field	Value
Port Number	Leave this field blank.
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank.
Special Options	Enter the destination IP address of Ethernet module. Use the following format: <b>-Ia -Pn -T</b>

Field	Value
	<p>where:</p> <p><b>a</b> = the destination IP address in standard Internet dot format. (For example, 192.9.2.60)</p> <p><b>n</b> = the destination Port number. Often one physical port has several virtual ports, used for different purposes. Use this option only if you want to override the default of 2222.</p> <p><b>-T</b> = forces the driver to use TCP (the default), rather than UDP (-U).</p>

## I/O Devices

Field	Value
I/O Device Address	Specify the <a href="#">Offlink Address</a> in the form: <b>PMM:C[options],u</b>
I/O Device Protocol	Enter ABTCP2.

### Allen-Bradley PLC-2 via TCP/IP - Data Types

Data Types	Address Format	Plant SCADA Data Type
Digital Data	x/y	DIGITAL
Integer	x	BCD / INT / LONG /LONGBCD

Where:

**x** = a word in octal (0 - 3777)

**y** = a bit number in octal (0 - 17)

## Examples

Data Type	DIGITAL
Address	200/12
Comment	Digital Data - Word Number 200 / Bit Number 12
Data Type	INT
Address	1700

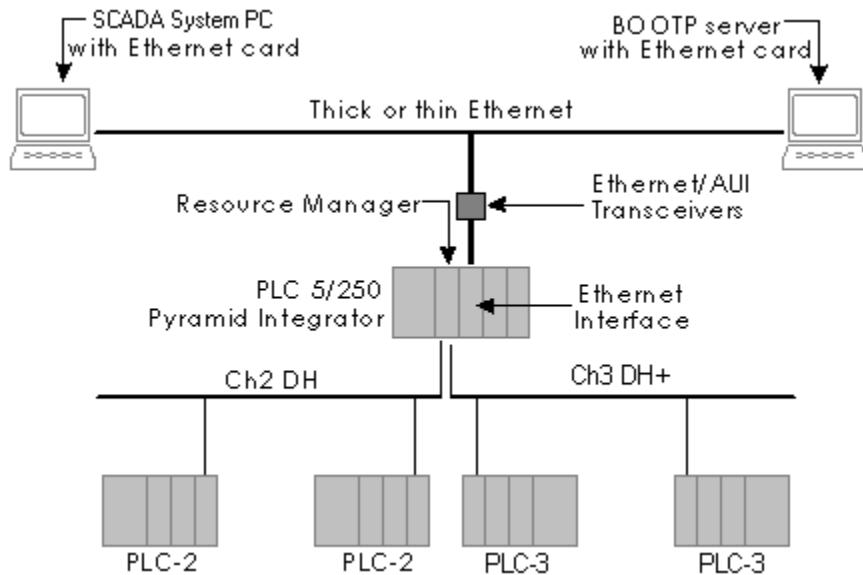
Comment	Integer - Word Number 1700
---------	----------------------------

Allen-Bradley PLC-2s support remapping reads and writes.

## Allen-Bradley PLC-3 via TCP/IP

The information included in the following pages describes how to connect the Allen-Bradley PLC-3 devices using the ABTCP protocol to a Plant SCADA system.

Using this method, you can connect to multiple PLCs as in the following diagram:



You cannot make a direct physical Ethernet connection to a PLC-3. An Ethernet Interface (EI) must be provided - typically by using a 5/250 Pyramid Integrator.

The Allen-Bradley Ethernet arrangement requires a BOOTP server. The BOOTP server is required by the AB EI modules to download configuration information. Refer to your Allen-Bradley documentation for BOOTP information.

The diagram above does not show all the possible connection options. Consult your Allen-Bradley manuals for more information.

## Allen-Bradley PLC-3 via TCP/IP - Device Address

The address for this device uses the following format:

-la -Pn -T

where:

a = the destination IP address in standard Internet dot format. (For example, 192.9.2.60)

n = the destination Port number. Often one physical port has several virtual ports, used for different purposes. The default is 2222.

-T = forces the driver to use TCP (the default), rather than UDP (-U).

This information is placed in the **Special Options** field of Plant SCADA's **Ports** settings by the Express Communications Wizard.

## Allen-Bradley PLC-3 via TCP/IP - Hardware Setup

Because this driver does not receive information regarding the state of the physical transport layer, no specific details can be given for setting up communications. Be sure to follow these general steps:

1. Set up the PLC 5/250 resource manager or AB-KA module. Assign the IP address, and test that it is valid by using the Ping program (explained in the TCP/IP Guide).
2. Set up the PLC-3 and 1775-KA for communications on the DH network to which it is connected. Set the Node Address and Options on each PLC communication module.
3. Check that you can communicate to the PLC using offlink addressing. You can use the utility accompanying the Allen-Bradley Interchange software to verify that you can communicate from a PC over Ethernet.

## Plant SCADA Computer Setup

Setup your specific Ethernet card as required and confirm basic network communications before proceeding with the communications configuration for the ABTCP driver. Refer to the documentation accompanying your hardware for instruction. Setup TCP/IP protocol on your Plant SCADA computer.

## Allen-Bradley PLC-3 via TCP/IP - Communication Settings

To establish communication with a device, the associated Board, Port and I/O Device need to be correctly configured in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure them, use the settings outlined below.

### Boards

Field	Value
Board Type	Enter TCPIP.
Address	Enter 0.
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.

### Ports

Field	Value
Port Number	Leave this field blank.
Baud Rate	Leave this field blank.

Field	Value
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank.
Special Options	<p>Enter the destination IP address of Ethernet module. Use the following format:</p> <p><b>-la -Pn -T</b></p> <p>where:</p> <p><b>a</b> = the destination IP address in standard Internet dot format. (For example, 192.9.2.60)</p> <p><b>n</b> = the destination Port number. Often one physical port has several virtual ports, used for different purposes. Use this option only if you want to override the default of 2222.</p> <p><b>-T</b> = forces the driver to use TCP (the default), rather than UDP (-U).</p>

## I/O Devices

Field	Value
I/O Device Address	Specify the <a href="#">Offlink Address</a> in the form: <b>PMM:C[options],u</b>
I/O Device Protocol	Enter ABTCP3.

### Allen-Bradley PLC-3 via TCP/IP - Data Types

**Note:** The Allen-Bradley PLC-3 has many data types. The driver makes a separate request for each different data type, and each request has a finite overhead. It is more efficient to read fewer data types, so you should move the I/O and other data into a few main blocks to be read by Plant SCADA.

Data Types	Address Format	Plant SCADA Data Type
Output image	<b>Of:O/o</b> (f is optional)	DIGITAL
Input image	<b>If:O/o</b> (f is optional)	DIGITAL
Timer Control bits	<b>TCTL:w/t</b>	DIGITAL
Timer Control bits	<b>TCTL:w/o</b>	DIGITAL

Data Types	Address Format	Plant SCADA Data Type
Timer Preset bits	<b>TPRE:w/o</b>	DIGITAL
Timer Accumulator bits	<b>TACC:w/o</b>	DIGITAL
Counter Control bits	<b>CCTL:w/t</b>	DIGITAL
Counter Control bits	<b>CCTL:w/o</b>	DIGITAL
Counter Preset bits	<b>CPRE:w/o</b>	DIGITAL
Counter Accumulator bits	<b>CACC:w/o</b>	DIGITAL
ASCII	<b>Af:w/o</b>	DIGITAL
BCD	<b>Df:w/o</b>	DIGITAL
REAL	<b>Ef:w/o (o = 0 to 37)</b>	DIGITAL
LONG	<b>Hf:w/o (o = 0 to 37)</b>	DIGITAL
Integer	<b>Nf:w/o</b>	DIGITAL
Binary	<b>Bf:w/o</b>	DIGITAL
Status words	<b>Sf:w/o</b>	DIGITAL
Pointer Section and File	<b>PFIL:w/o</b>	DIGITAL
Pointer word	<b>PWRD:w/o</b>	DIGITAL
Output image	<b>Of:O (f is optional)</b>	BCD / INT
Input image	<b>If:O (f is optional)</b>	BCD / INT
Timer Control	<b>TCTL:w</b>	BCD / INT
Timer Preset	<b>TPRE:w</b>	BCD / INT
Timer Accumulator	<b>TACC:w</b>	BCD / INT
Counter Control	<b>CCTL:w</b>	BCD / INT
Counter Preset	<b>CPRE:w</b>	BCD / INT
Counter Accumulator	<b>CACC:w</b>	BCD / INT
Integer	<b>Nf:w</b>	BCD / INT

Data Types	Address Format	Plant SCADA Data Type
BCD	<b>Df:w</b>	BCD / INT
Binary	<b>Bf:w</b>	BCD / INT
ASCII	<b>Af:w</b>	BCD / INT
Status words	<b>Sf:w</b>	BCD / INT
Pointer Section and File	<b>PFIL:w</b>	BCD / INT
Pointer word	<b>PWRD:w</b>	BCD / INT
Long	<b>Hf:w</b>	LONG / LONGBCD
Floating Point	<b>Ff:w</b>	REAL

Where:

n =	Optional Module Number 0 to 8 decimal
f =	File Number 0 to 999 decimal
:w =	Element Number 0 to 9999 decimal
:O =	Element Number 0 to 7777 octal
/o =	Bit Number 0 to 17 octal PRE = Preset Value ACC = Accumulated Value
.C =	Control Sub Element LEN = Length POS = Position
.P =	PID Sub Element SP = Set Point KP = Proportional Gain KI = Integral Gain KD = Derivative Gain BIAS = Output Bias % MAXS = Maximum Scaled Value MINS = Minimum Scaled Value DB = Deadband SO = Set Output % MAXO = Maximum Output % MINO = Minimum Input % UPD = Update Time PVH = PV Alarm High PVL = PV Alarm Low

	PVDB = PV Alarm Deadband PV = Process Variable ERR = Error OUT = Output DVP = Deviation Alarm + DVN = Deviation Alarm - DVDB = Deviation Alarm Deadband MAXI = Maximum Input MINI = Minimum Input TIE = Tieback %
.t =	Timer bits EN = Enable TT = Timing DN = Done BS LK
.c =	Counter bits CU = Count Up CD = Count Down DN = Done OV = Overflow UN = Underflow LK
.r =	Control bits EN = Enable EU = Enable Unloading DN = Done EM = Empty ER = Error UL = Unload IN = Inhibit Comparisons FD = Found
.p =	PID bits EN = Enable CT = Cascaded Type CL = Cascaded Loop PVT = PV Tracking DO = Derivative Of SWM = Software A/M Mode CA = Control Action MO = Mode PE = PID Equation INI = PID Initialized SPOR = SP Out of Range

	<p>OLL = Output Limit Low OLH = Output Limit High EWD = Error Within Deadband DVPA = Deviation Low Alarm DVNA = Deviation High Alarm PVLA = PV Low Alarm PVHA = PV High Alarm</p>
--	---

### Examples

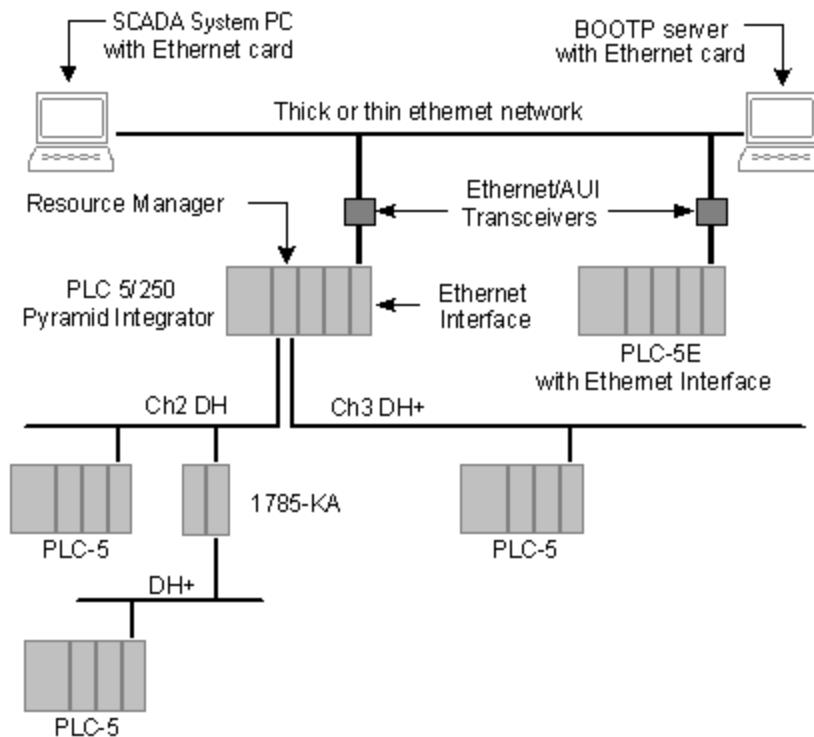
Data Type	DIGITAL
Address	I:1/15
Comment	Input Image - Element Number 1 / Bit Number 15
Data Type	INT
Address	N7:3
Comment	Integer - File Number 7 : Element Number 3

Allen Bradley PLC-3s support remapping reads and writes.

### Allen-Bradley PLC-5 via TCP/IP

The information included in the following pages describes how to connect the Allen-Bradley PLC-5 devices using the ABTCP protocol to a Plant SCADA system.

Using this method, you can connect to multiple PLCs as in the following diagram:



The Allen-Bradley Ethernet arrangement requires a BOOTP server. The BOOTP server is required by the AB EI modules to down-load configuration information. Refer to your Allen-Bradley documentation for BOOTP information.

This diagram does not show all the possible connection options. Consult your Allen-Bradley manuals for more information.

#### Allen-Bradley PLC-5 via TCP/IP - Device Address

The address for this device uses the following format:

`-la -Pn -T`

Where:

*a* = the destination IP address in standard Internet dot format. (For example, 192.9.2.60)

*n* = the destination Port number. Often one physical port has several virtual ports, used for different purposes. The default is 2222.

*-T* = forces the driver to use TCP (the default), rather than UDP (-U).

This information is placed in the **Special Options** field of Plant SCADA's **Ports** settings by the Express Communications Wizard.

#### Allen-Bradley PLC-5 via TCP/IP - Hardware Setup

There are two methods of connecting to a PLC-5 series PLC:

- direct
- offlink.

## Direct Ethernet

PLC-5E model PLCs can support a direct Ethernet connection. In this case all that you need to do is assign the PLC an IP address, by using the BOOTP utility (refer to the Allen-Bradley documentation). Test that the network and IP address is valid by using the Ping program (explained in the TCP/IP Guide).

## Offlink

In the case that you have PLC-5s that are not Ethernet capable, or if they are connected via DH or DH+ through a 5/250 or AB-KA, you can still access them through [Offlink Addressing](#). Follow these general steps:

1. Set up the PLC 5/250 resource manager or AB-KA module. Assign the IP address, and test that it is valid by using the Ping program (explained in the TCP/IP Guide).
2. Set up the PLC-5 for communications on the DH or DH+ network to which it is connected. Set the Node Address, Baud rate etc.
3. Check that you can communicate to the PLC using offlink addressing. You can use the utility accompanying the Allen-Bradley Interchange software to verify that you can communicate from a PC over Ethernet.

### Hints and Tips

- The EI module may be mounted in the rack of the PLC-5/250, or PLC-5 series controllers, or built into the PLC-5 \*E controllers. The module has a DIX connector that requires an AUI cable to attach via a transceiver to the thin or thick Ethernet cable.
- The EI module requires a Signal Quality Error (SQE) test - also known as a heartbeat. Set your Ethernet transceiver with the SQE test enabled.
- If your **machine is very slow** and has a heavy processing load, you may find that this protocol is generating timeout errors (decimal 21). Setting the [Kernel]Idle parameter less than the default may fix this problem.

## Plant SCADA Computer Setup

Setup your specific Ethernet card as required and confirm basic network communications before proceeding with the communications configuration for the ABTCP driver. Refer to the documentation accompanying your hardware for instructions.

[Setup the TCP/IP protocol](#) on your Plant SCADA computer.

### Allen-Bradley PLC-5 via TCP/IP - Communication Settings

To establish communication with a device, the associated Board, Port and I/O Device need to be correctly configured in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure them, use the settings outlined below.

## Boards

Field	Value
Board Type	Enter TCPIP.
Address	Enter 0.
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.

## Ports

Field	Value
Port Number	Leave this field blank.
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank.
Special Options	<p>Enter the destination IP address of Ethernet module. Use the following format:</p> <p><b>-la -Pn -T</b></p> <p>where:</p> <p><b>a</b> = the destination IP address in standard Internet dot format. (For example, 192.9.2.60)</p> <p><b>n</b> = the destination Port number. Often one physical port has several virtual ports, used for different purposes. Use this option only if you want to override the default of 2222.</p> <p><b>-T</b> = forces the driver to use TCP (the default), rather than UDP (-U).</p>

## I/O Devices

Field	Value
I/O Device Address	<p>You can leave this field blank if your PLC has a direct Ethernet connection.</p> <p>If you have your PLC connected to a DH+ or DH</p>

Field	Value
	network, and accessed through another 5/250 or AB-KA, specify the <a href="#">Offlink Address</a> in the form: <b>PMM:C[options],u</b>
I/O Device Protocol	Enter ABTCP5.

**Allen-Bradley PLC-5 via TCP/IP - Data Types**

**Note:** The Allen-Bradley PLC-5 has many data types. The driver makes a separate request for each different data type, and each request has a finite overhead. It is more efficient to read fewer data types, so you should move the I/O and other data into a few main blocks to be read by Plant SCADA.

Data Types	Address Format	Plant SCADA Data Type
Output image	O:O/o	DIGITAL
Input image	I:O/o	DIGITAL
Status words 0 to 127 (to 128 for w parameter)	S:w/b	DIGITAL
Timer	Tf:w.t	DIGITAL
Counter	Cf:w.c	DIGITAL
Control	Rf:w.r	DIGITAL
Binary	Bf:w/b	DIGITAL
Binary	Bf/b*	DIGITAL
Integer	Nf:w/b	DIGITAL
Sequential Function Chart	SCf:w.s	DIGITAL
Block Transfer	BTf:w.b	DIGITAL
Floating-point	Ff:w/b	DIGITAL
PID	PDf:w.p	DIGITAL
Output image	O:O	BCD / INT / LONG / LONGBCD
Input image	I:O	BCD / INT / LONG / LONGBCD
Status words 0 to 127 (to 128 for w parameter)	S:w	BCD / INT / LONG / LONGBCD

Data Types	Address Format	Plant SCADA Data Type
Timer	<b>Tf:w.T</b>	BCD / INT / LONG / LONGBCD
Counter	<b>Cf:w.T</b>	BCD / INT / LONG / LONGBCD
Control	<b>Rf:w.R</b>	BCD / INT / LONG / LONGBCD
Binary	<b>Bf:w</b>	BCD / INT / LONG / LONGBCD
Integer	<b>Nf:w</b>	BCD / INT / LONG / LONGBCD
Sequential Function Chart	<b>SCf:w.S</b>	BCD / INT / LONG / LONGBCD
Block Transfer	<b>BTf:w.B</b>	BCD / INT / LONG / LONGBCD
ASCII	<b>Af:w</b>	BCD / INT / LONG / LONGBCD
BCD	<b>Df:w</b>	BCD / INT
Floating-point	<b>Ff:w</b>	REAL
PID	<b>PDf:w.P</b>	REAL
String 82 bytes long	<b>STf:w</b>	STRING

Where:

f	File Number 0 to 999 decimal
:w	Element number 0 to 999 decimal
:O	Element number 0 to 277 octal
/b	Bit Number 0 to 15 decimal
/o	Bit Number 0 to 17 octal
.T	Timer/Counter Sub Element: PRE = Preset Value ACC = Accumulated Value
.t	Timer bits: EN = Enable TT = Timing DN = Done
.C	Control Sub Element: LEN = Length POS = Position
.c	Counter bits:

	<p>CU = Count Up      CD = Count Down      DN = Done      OV = Overflow      UN = Underflow</p>
.r	<p>Control bits:      EN = Enable      EU = Enable Unloading      DN = Done      EM = Empty      ER = Error      UL = Unload      IN = Inhibit Comparisons      FD = Found</p>
.S	<p>SFC Sub Element:      PRE = Preset Value      TIM = Elapsed step active time</p>
.S	<p>SFC bits:      SA = Scan Active      FS = First Scan      LS = Last Scan      OV = Timer Overflow      ER = Error      DN = Done</p>
.B	<p>Block Transfer Sub Element:      RLEN = Received Length      DLEN = Done Length      FILE = File      ELEM = Element</p>
.b	<p>Block Transfer Bits:      EN = Enable      TT = Start      DN = Done      ER = Error      CO = Continuous      EW = Enable Wait      NR = No Response      TO = Time Out      RW = Read Write</p>
.P	<p>PID Sub Element:      SP = Set Point      KP = Proportional Gain</p>

	KI = Integral Gain KD = Derivative Gain BIAS = Output Bias % MAXS = Maximum Scaled Value MINS = Minimum Scaled Value DB = Deadband SO = Set Output % MAXO = Maximum Output % MINO = Minimum Input % UPD = Update Time PVH = PV Alarm High PVL = PV Alarm Low PVDB = PV Alarm Deadband PV = Process Variable ERR = Error OUT = Output DVP = Deviation Alarm + DVN = Deviation Alarm - DVDB = Deviation Alarm Deadband MAXI = Maximum Input MINI = Minimum Input TIE = Tieback %
.p	PID bits:  EN = Enable CT = Cascaded Type CL = Cascaded Loop PVT = PV Tracking DO = Derivative Of SWM = Software A/M Mode CA = Control Action MO = Mode PE = PID Equation INI = PID Initialized SPOR = SP Out of Range OLL = Output Limit Low OLH = Output Limit High EWD = Error Within Deadband DVPA = Deviation Low Alarm DVNA = Deviation High Alarm PVLA = PV Low Alarm PVHA = PV High Alarm

### Examples

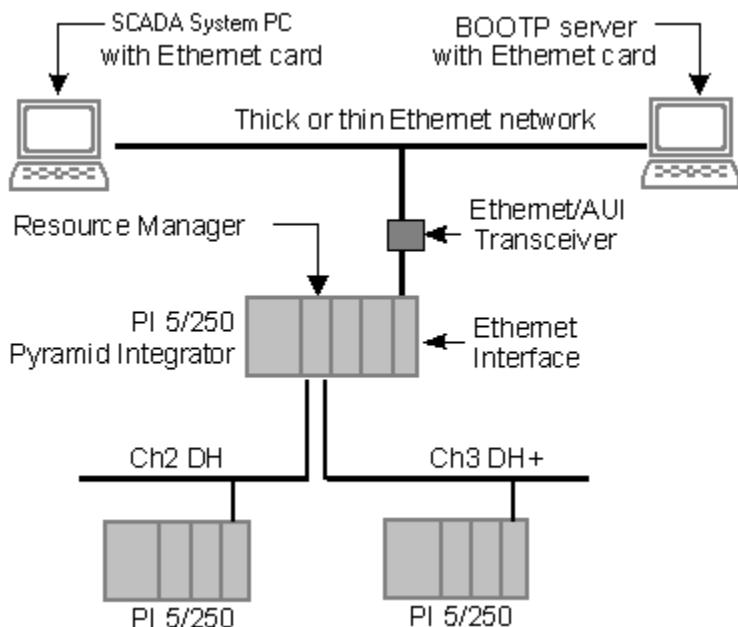
Data Type	DIGITAL
Address	I:1/15

Comment	Input Image - Element Number 1 / Bit Number 15
Data Type	INT
Address	N7:3
Comment	Integer - File Number 7 : Element Number 3

Allen Bradley PLC-5 Series support remapping reads and writes.

### Allen-Bradley PLC-5/250 via TCP/IP

The information included in the following pages describes how to connect Allen-Bradley PLC-5/250 devices using the ABTCP protocol to a Plant SCADA system. Using this method, you can connect to multiple PLCs as in the following diagram:



The Allen-Bradley Ethernet arrangement requires a BOOTP server. The BOOTP server is required by the EI modules to download configuration information. Refer to your Allen-Bradley documentation for BOOTP information.

This diagram does not show all of the possible connection options. Consult your Allen-Bradley manuals for more information.

### Allen-Bradley PLC-5/250 via TCP/IP - Device Address

The address for this device uses the following format:

-la -Pn -T

where:

*a* = the destination IP address in standard Internet dot format. (For example 192.9.2.60)

*n* = the destination Port number. Often one physical port has several virtual ports, used for different purposes. The default is 2222.

-T = forces the driver to use TCP (the default), rather than UDP (-U).

This information is placed in the **Special Options** field of Plant SCADA's **Ports** settings by the Express Communications Wizard.

### Allen-Bradley PLC-5/250 via TCP/IP - Hardware Setup

There are two methods of connecting to a PLC-5 series PLC:

- direct
- offlink.

#### Direct Ethernet

PLC-5E model PLCs can support a direct Ethernet connection. In this case all that you need to do is assign the PLC an IP address, by using the BOOTP utility (refer to the Allen-Bradley documentation). Test that the network and IP address is valid by using the Ping program (explained in the TCP/IP Guide).

#### Offlink

In the case that you have PIs connected via DH or DH+ through another 5/250 or AB-KA, you can still access them through [Offlink Addressing](#). Follow these general steps:

1. Set up the PLC 5/250 resource manager or AB-KA module. Assign the IP address, and test that it is valid by using the Ping program (explained in the TCP/IP Guide).
2. Set up the PLC-5 for communications on the DH or DH+ network to which it is connected. Set the Node Address, Baud rate etc.
3. Check that you can communicate to the PLC using offlink addressing. You can use the utility accompanying the Allen-Bradley Interchange software to verify that you can communicate from a PC over Ethernet.

### Plant SCADA Computer Setup

It is recommended you setup your specific Ethernet card as required and confirm basic network communications before proceeding with the communications configuration for the ABTCP driver. Refer to the documentation accompanying your hardware for instruction.

[Setup the TCP/IP protocol](#) on your Plant SCADA computer.

### Allen-Bradley PLC-5/250 via TCP/IP - Communication Settings

To establish communication with a device, the associated Board, Port and I/O Device need to be correctly configured in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure them, use the settings outlined below.

### Boards

Field	Value
Board Type	Enter TCPIP.
Address	Enter 0.
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.

## Ports

Field	Value
Port Number	Leave this field blank.
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank.
Special Options	<p>Enter the destination IP address of Ethernet module. Use the following format:</p> <p><b>-la -Pn -T</b></p> <p>where:</p> <p><b>a</b> = the destination IP address in standard Internet dot format. (For example, 192.9.2.60)</p> <p><b>n</b> = the destination Port number. Often one physical port has several virtual ports, used for different purposes. Use this option only if you want to override the default of 2222.</p> <p><b>-T</b> = forces the driver to use TCP (the default), rather than UDP (-U).</p>

## I/O Devices

Field	Value
I/O Device Address	<p>You can leave this field blank if your PI has a direct Ethernet connection.</p> <p>If you have your PI connected to a DH+ or DH network,</p>

Field	Value
	and accessed through another 5/250 or AB-KA, specify the <a href="#">Offlink Address</a> in the form: <b>PMM:C[options],u</b>
I/O Device Protocol	Enter ABTCP250.

### Allen-Bradley PLC-5/250 via TCP/IP - Data Types

**Note:** The Allen-Bradley PLC-5/250 has many data types. The driver makes a separate request for each different data type, and each request has a finite overhead. It is more efficient to read fewer data types, so you should move the I/O and other data into a few main blocks to be read by Plant SCADA.

Data Types	Address Format	Plant SCADA Data Type
Output image	O:O/o	DIGITAL
Input image	I:O/o	DIGITAL
Status words 0 to 350	S:w/b	DIGITAL
Internal storage	IS:w/b	DIGITAL
Status words 0 to 31	nS:w/b	DIGITAL
Timer	nTf:w.t	DIGITAL
Timer	nTf:w/b	DIGITAL
Counter	nCf:w.c	DIGITAL
Counter	nCf:w/b	DIGITAL
Control	nRf:w.r	DIGITAL
Control	nRf:w/b	DIGITAL
Binary	nBf:w/b	DIGITAL
Integer	nNf:w/b	DIGITAL
Floating Point	nFf:w/b	DIGITAL
Long	nLf:w/b	DIGITAL
PID	nPDf:w.p	DIGITAL
Block transfer data	nBTdf:w/b	DIGITAL
Output image	O:O	BCD / INT

Data Types	Address Format	Plant SCADA Data Type
Input image	I:O	BCD / INT
Status words 0 to 350	S:w	BCD / INT
Internal storage	IS:w	BCD / INT
Status words 0 to 31	nS:w	BCD / INT
Counter	nCf:w.T	BCD / INT
Control	nRf:w.C	BCD / INT
Binary	nBf:w	BCD / INT
Integer	nNf:w	BCD / INT
Block transfer data	nBTDf:w	BCD / INT
Shared data words 0-1023	nSDf:w	BCD / INT
Timer	nTf:w.T	LONG / LONGBCD
Long	nLf:w	LONG / LONGBCD
Floating Point	nFf:w	REAL
PID	nPDf:w.P	REAL
String 82 bytes long	nSTf:w	STRING

Where:

n	Optional Module Number 0 to 8 decimal
f	File Number 0 to 9999 decimal
:w	Element Number 0 to 9999 decimal
:O	Element Number 0 to 777 octal
/b	Bit Number 0 to 15 decimal
/o	Bit Number 0 to 17 octal
.T	Timer/Counter Sub Element PRE = Preset Value ACC = Accumulated Value
.C	Control Sub Element LEN = Length

	POS = Position
.P	<p>PID Sub Element</p> <p>SP = Set Point</p> <p>KP = Proportional Gain</p> <p>KI = Integral Gain</p> <p>KD = Derivative Gain</p> <p>BIAS = Output Bias %</p> <p>MAXS = Maximum Scaled Value</p> <p>MINS = Minimum Scaled Value</p> <p>DB = Deadband</p> <p>SO = Set Output %</p> <p>MAXO = Maximum Output %</p> <p>MINO = Minimum Input %</p> <p>UPD = Update Time</p> <p>PVH = PV Alarm High</p> <p>PVL = PV Alarm Low</p> <p>PVDB = PV Alarm Deadband</p> <p>PV = Process Variable</p> <p>ERR = Error</p> <p>OUT = Output</p> <p>DVP = Deviation Alarm +</p> <p>DVN = Deviation Alarm -</p> <p>DVDB = Deviation Alarm Deadband</p> <p>MAXI = Maximum Input</p> <p>MINI = Minimum Input</p> <p>TIE = Tieback %</p>
.t	<p>Timer bits</p> <p>EN = Enable</p> <p>TT = Timing</p> <p>DN = Done BS LK</p>
.c	<p>Counter bits</p> <p>CU = Count Up</p> <p>CD = Count Down</p> <p>DN = Done</p> <p>OV = Overflow</p> <p>UN = Underflow LK</p>
.r	<p>Control bits</p> <p>EN = Enable</p> <p>EU = Enable Unloading</p> <p>DN = Done</p> <p>EM = Empty</p> <p>ER = Error</p> <p>UL = Unload</p> <p>IN = Inhibit Comparisons</p>

	FD = Found
.p	PID bits EN = Enable CT = Cascaded Type CL = Cascaded Loop PVT = PV Tracking DO = Derivative Of SWM = Software A/M Mode CA = Control Action MO = Mode PE = PID Equation INI = PID Initialized SPOR = SP Out of Range OLL = Output Limit Low OLH = Output Limit High EWD = Error Within Deadband DVPA = Deviation Low Alarm DVNA = Deviation High Alarm PVLA = PV Low Alarm PVHA = PV High Alarm

## Examples

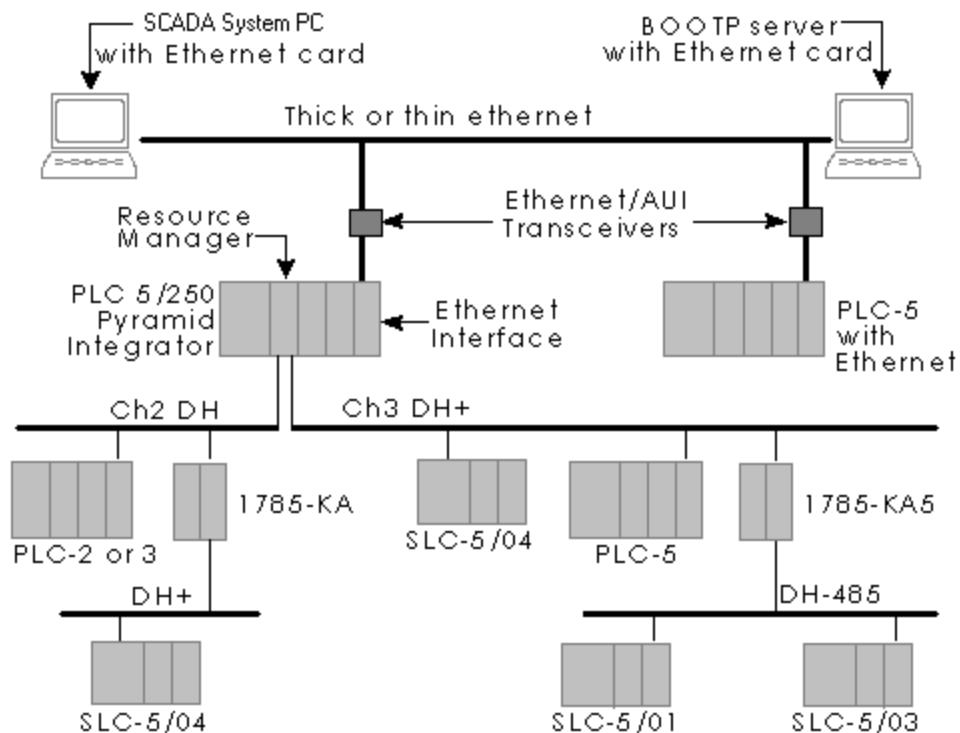
Data Type	DIGITAL
Address	I:1/15
Comment	Input Image - Element Number 1 / Bit Number 15
Data Type	INT
Address	N7:3
Comment	Input Image - Element Number 1 / Bit Number 15

Allen Bradley 5/250 PLCs support remapping reads and writes.

## Allen-Bradley SLC-500 via TCP/IP

The information included in the following pages describes how to connect Allen-Bradley SLC-500 devices using the ABTCP protocol to a Plant SCADA system.

All SLC models (500, 5/01, 5/02, 5/03, and 5/04) are supported. Using this method, you can connect to multiple SLCs as in the following diagram:



You cannot make a direct physical Ethernet connection to an SLC. An Ethernet Interface (EI) must be provided - typically by using a 5/250 Pyramid Integrator.

The Allen-Bradley Ethernet arrangement requires a BOOP server. The BOOP server is required by the AB EI modules to download configuration information. Refer to your Allen-Bradley documentation for BOOP information. Often a 1747-AIC module is used between the SLCs and the DH-485 networks, to give a convenient programming port.

This diagram does not show all the possible connection options - consult your Allen-Bradley manuals for more information.

### Allen-Bradley SLC-500 via TCP/IP - Device Address

The address for this device uses the following format:

-Ia-Pn -T

where:

*a* = the destination IP address in standard Internet dot format. (For example, 192.9.2.60)

*n* = the destination Port number. Often one physical port has several virtual ports, used for different purposes. The default is 2222.

*-T* = forces the driver to use TCP (the default), rather than UDP (-U).

This information is placed in the **Special Options** field of Plant SCADA's **Ports** settings by the Express Communications Wizard.

### Allen-Bradley SLC-500 via TCP/IP - Hardware Setup

Because this driver does not receive information regarding the state of the physical transport layer, no specific details can be given for setting up communications. Follow these general steps:

1. Set up the PLC 5/250 resource manager (or AB-KA module). Assign the IP address, and test that it is valid by using the Ping program.
2. Set up the SLC for communications on the DH, DH+, or DH-485 network to which it is connected. Be sure to set the Node Address, Baud Rate, and Maximum Number of Nodes on each SLC.
3. Check that you can communicate to the SLC using offlink addressing. You can use the utility accompanying the Allen-Bradley Interchange software to verify that you can communicate from a PC over Ethernet.

## Plant SCADA Computer Setup

Setup your specific Ethernet card as required and confirm basic network communications before proceeding with the communications configuration for the ABTCP driver. Refer to the documentation accompanying your hardware for instruction.

Setup TCP/IP protocol on your Plant SCADA computer.

### Allen-Bradley SLC-500 via TCP/IP - Communication Settings

To establish communication with a device, the associated Board, Port and I/O Device need to be correctly configured in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure them, use the settings outlined below.

## Boards

Field	Value
Board Type	Enter TCPIP.
Address	Enter 0.
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.

## Ports

Field	Value
Port Number	Leave this field blank.
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.

Field	Value
Parity	Leave this field blank.
Special Options	<p>Enter the destination IP address of Ethernet module. Use the following format:</p> <p><b>-la -Pn -T</b></p> <p>where:</p> <p><b>a</b> = the destination IP address in standard Internet dot format. (For example 192.9.2.60)</p> <p><b>n</b> = the destination Port number. Often one physical port has several virtual ports, used for different purposes. Use this option only if you want to override the default of 2222.</p> <p><b>-T</b> = forces the driver to use TCP (the default), rather than UDP (-U).</p>

## I/O Devices

Field	Value
I/O Device Address	Specify the <a href="#">Offlink Address</a> in the form: <b>PMM:C[options],u</b>
I/O Device Protocol	Enter ABTCP500.

### Allen-Bradley SLC-500 via TCP/IP - Data Types

**Note:** The Allen Bradley SLC500 has many data types. The protocol makes a separate request for each different data type, and each request has a finite overhead. It is more efficient to read fewer data types, so you should move the I/O and other data into a few main blocks to be read by Plant SCADA.

Data Types	Address Format	Plant SCADA Data Type
Status words	<b>S:w/b</b>	DIGITAL
Timer	<b>Tf:w.t</b>	DIGITAL
Timer	<b>Tf:w/b</b>	DIGITAL
Counter	<b>Cf:w.c</b>	DIGITAL
Counter	<b>Cf:w/b</b>	DIGITAL
Control	<b>Rf:w.r</b>	DIGITAL
Control	<b>Rf:w/b</b>	DIGITAL

Data Types	Address Format	Plant SCADA Data Type
Binary	<b>Bf:w/b</b>	DIGITAL
Integer	<b>Nf:w/b</b>	DIGITAL
Output image	<b>O:O.o</b>	INT
Input image	<b>I:O.o</b>	INT
Status words	<b>S:w</b>	BCD / INT / LONG / LONGBCD/ REAL
Timer	<b>Tf:w.T</b>	BCD / INT / LONG / LONGBCD/ REAL
Counter	<b>Cf:w.T</b>	BCD / INT / LONG / LONGBCD/ REAL
Control	<b>Rf:w.C</b>	BCD / INT / LONG / LONGBCD/ REAL
Binary	<b>Bf:w</b>	BCD / INT / LONG / LONGBCD/ REAL
Integer	<b>Nf:w</b>	BCD / INT / LONG / LONGBCD/ REAL

Where:

f	File Number 0 to 999 decimal
:w	Element number 0 to 999 decimal
:O	Element number 0 to 277 octal
/b	Bit Number 0 to 15 decimal
/o	Bit Number 0 to 17 octal
.T	Timer/Counter Sub Element: PRE - Preset Value ACC - Accumulated Value
.t	Timer bits: EN - Enable TT - Timing DN - Done
.C	Control Sub Element: LEN - Length

	POS – Position
.c	Counter bits: CU - Count Up CD - Count Down DN - Done OV - Overflow UN - Underflow
.r	Control bits: EN - Enable DN - Done ER - Error UL - Unload IN - Inhibit comparisons FD - Found

## Examples

Data Type	DIGITAL
Address	N7:1/15
Comment	Integer - File Number 7 Element Number 1 Bit Number 15
Data Type	INT
Address	N7:3
Comment	Integer - File Number 7 : Element Number 3

Allen-Bradley SLC500 Series PLCs support remapping reads only.

## Supported Devices via Data Highway

The ABTCP driver can be used to communicate with Allen-Bradley I/O devices via Data Highway Plus, using the KT, KTX, KTXD, PKTX cards, or the 5136-SD-PCI board from SS Technologies.

Windows detects the hardware and assigns it an address. Plant SCADA retrieves the configuration details from Windows and, at run-time, downloads appropriate interface firmware to the board, enabling communication. No additional software is required.

You can use this method to communicate with the following devices:

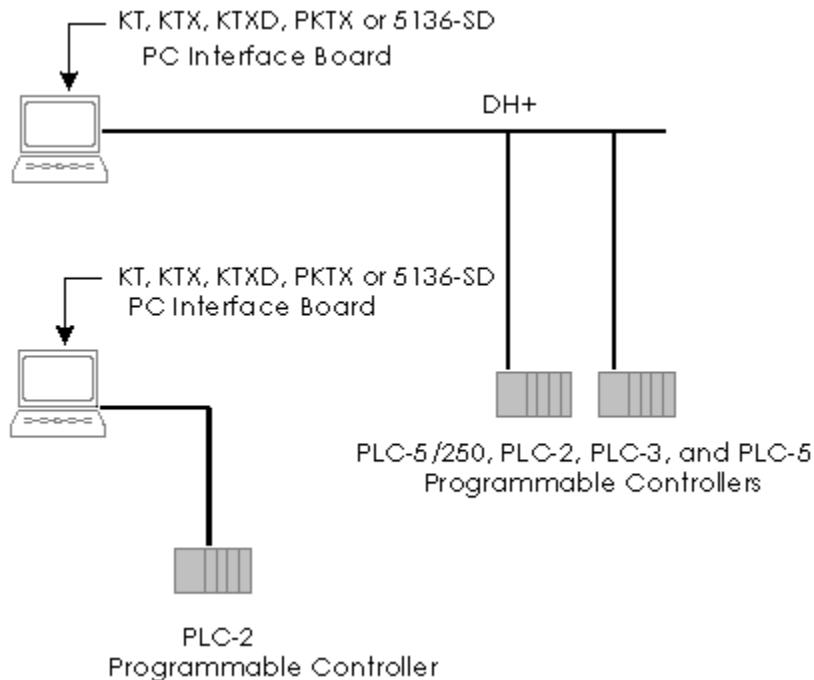
- [Allen-Bradley PLC-2](#)
- [Allen-Bradley PLC-3](#)
- [Allen-Bradley PLC-5](#)

- Allen-Bradley PLC-5/250
- Allen-Bradley SLC-500

When you first start up your computer after installing the 5136-SD-PCI board, a message may display indicating that new hardware has been detected. It is not necessary to install the board using Add New Hardware in the Windows Control Panel, since Plant SCADA detects the Windows configuration. If the card has been previously installed through Control Panel, you will need to uninstall it. Plant SCADA supports the installation and use of only one 5136-SD-PCI board in a computer at a time.

### Allen-Bradley PLC-2 via Data Highway

Plant SCADA can use the ABTCP driver to communicate with Allen-Bradley (AB) PLCs over Data Highway Plus via KT, KTX, KTXD, PKTX, 5136-SD and 5136-SD-PCI cards. Using this method, you can connect to single PLCs or to multiple PLCs as in the following diagram:



This arrangement does not use TCP/IP, even though the ABTCP driver is being used. This method does not support DH-485.

### Allen-Bradley PLC-2 via Data Highway - Device Address

The address for an Allen-Bradley PLC-2 connected to a KT (or KTX, KTXD, or 5136-SD) board is specified as:

**xKT:C[options],u**

Where:

**x** = the board number. If the number is not specified in the ports for then use 1.

**C** = the channel number, for example, KT card = 0, KTX card = 0 or 1.

**u** = the station number of the remote PLC in octal. Station numbers are between 0 and 77 (octal) for DH+ channels, and between 1 and 376 (octal) for DH channels.

**[options]** If required, you can use the [Offlink Address](#) options to allow communication with a station (PLC) that is not on the local link.

#### Example

**1KT:0,1** specifies Port ID **1KT** for access to PLC Station **1**.

### Allen-Bradley PLC-2 via Data Highway - KT Board Setup

Before you install a KT board in your computer, select a starting memory address and an interrupt number. Use an interrupt number and address that has not been used by any of the existing boards in your computer. If you have any other proprietary boards installed, check the documentation accompanying these boards to determine the interrupt numbers and addresses that have been used. (If you are using an EISA bus computer, run the EISA configuration program.)

**Note:** The following information gives specific details on how to set up the KT card - not the KTX or KTXD.

### KT Card Polling vs Interrupts

You can specify the communications to operate via polling or interrupt. This is controlled by the [\[ABTCP\] Polltime](#). If you set this parameter to a value other than 0, then polling will be used instead of interrupts.

### KT Card Switch Settings

Set the configuration options of the 1784-KT board to the following. Refer to your Allen-Bradley documentation for further information.

#### Board Address Switches

Memory Location Range	SW 1	SW 2	SW 3	SW 4	SW 5	SW 6
A000:0000-3F FF	Closed	Closed	Closed	Open	Closed	Open
A400:0000-3F FF	Open	Closed	Closed	Open	Closed	Open
A800:0000-3F FF	Closed	Open	Closed	Open	Closed	Open
AC00:0000-3F FF	Open	Open	Closed	Open	Closed	Open
B400:0000-3F FF	Open	Closed	Open	Open	Closed	Open
B800:0000-3F FF	Closed	Open	Open	Open	Closed	Open
C000:0000-3F	Closed	Closed	Closed	Closed	Open	Open

FF						
C400:0000-3F FF	Open	Closed	Closed	Closed	Open	Open
C800:0000-3F FF	Closed	Open	Closed	Closed	Open	Open
CC00:0000-3F FF *	Open	Open	Closed	Closed	Open	Open
D000:0000-3F FF	Closed	Closed	Open	Closed	Open	Open
D400:0000-3F FF	Open	Closed	Open	Closed	Open	Open
D800:0000-3F FF	Closed	Open	Open	Closed	Open	Open

\* Recommended address **0xCC00**.

This board uses a base memory size of 16k. To prevent conflict with other boards in your system, no memory overlap can exist when setting the memory address. You also require this base memory size when configuring an EISA bus computer.

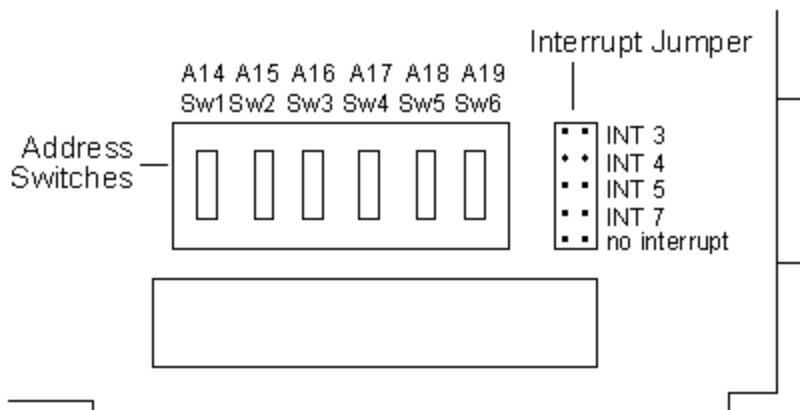
The EISA configuration utility may disable the interrupts of boards that are not set-up correctly. If you cannot get a response from the KT board, check the EISA configuration for the board. If there is still no response, try another interrupt. If you set the KT board to an interrupt that is used by a COM port, the EISA configuration will move the COM port to another interrupt. If you do not get a response from the KT board, disable the COM port that was re-configured.

## Interrupt Jumper Setting

Jumper Selection	Interrupt Selection
INT 3	IRQ3
INT 4	IRQ4
INT 5	IRQ5
INT 7	IRQ7
no interrupt	NONE

\* Recommended interrupt 5 .

The following figure shows the location of the Address Switches and Interrupt Jumper.



Use an AB diagnostics application to confirm your settings.

## Setting up a PKTX Board

This is a plug and play card. No special setup is required - just confirm that the Base Memory Jumper is set to the 32-bit position. Do not attempt to manually install a driver when Windows notifies you it has detected new hardware.

It is recommended that **no** interrupts are set on this board. Interrupt request levels (IRQ) and base memory address values are automatically assigned. Plant SCADA will retrieve these configurations and use them to communicate with a board.

---

**Note:** To properly initialize the Windows component of the Plant SCADA CIKT board driver, you may need to restart your PC after installing the PKTX board, and then again after running your Plant SCADA project for the first time. Do not abort the project start up - allow it to execute right up to the menu page.

---

## Allen-Bradley PLC-2 via Data Highway - 5136-SD-PCI Setup

Communication with Allen-Bradley I/O devices via Data Highway Plus can be established using the 5136-SD-PCI board from SS Technologies. This board was developed with Plug and Play functionality, enabling a simple setup. Plant SCADA supports the installation and use of only one 5136-SD-PCI board in a computer at a time.

Once you have installed the board into the PCI slot in your computer, Windows detects the new hardware and assigns it an address. Plant SCADA retrieves the configuration details from Windows and, at run-time, downloads appropriate interface firmware to the board, enabling communication. No additional software is required.

Once you have installed the board, you will need to set the [Communications Settings](#).

---

**Note:** When you first start up your computer after installing the 5136-SD-PCI board, a message may display indicating that new hardware has been detected. It is not necessary to install the board using Add New Hardware in the Windows Control Panel, since Plant SCADA detects the Windows configuration. If the card has been previously installed through Control Panel, you will need to uninstall it.

---

## Allen-Bradley PLC-2 via Data Highway - 5136-SD-PCI Testing

---

**Note:** For full details of the 5136-SD-PCI board, please refer to its hardware guide.

---

Once the 5136-SD-PCI board is installed, you can tell whether it is functioning correctly by monitoring its LEDs. You will not need to remove your computer's cover to do this, as the board's LEDs are visible at the back of the machine.

The board has two LEDs, the Network LED and the System LED. The first time you start up the computer with the board installed, the System LED will be red. When Plant SCADA successfully downloads the communications firmware to the board, it turns the System LED off. If the loader does not run successfully, the System LED will remain red.

The Network LED displays a flickering green whenever the card is active (transmitting) on the network. The Network LED should never turn off as long as there is a node on the network. If it does not display green, check your hardware or network cable.

### Allen-Bradley PLC-2 via Data Highway - Communication Settings

To establish communication with a device, the associated Board, Port and I/O Device need to be correctly configured in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure them, use the settings outlined below.

## Boards

Field	Value
Board Type	For a <b>KT</b> , <b>KTX</b> , <b>KTXD</b> , or <b>PKTX</b> card, enter CIKT. For a <b>5136-SD</b> or <b>5136-SD-PCI</b> card, enter SSTAB.
Address	For a <b>KT</b> board, enter the address of the card (e.g. 0xCC00). For a <b>KTX</b> or <b>KTXD</b> board, enter the first two digits of the address of the card (e.g. 0xD0). For a <b>PKTX</b> board, enter 0xFFFF. For a <b>PKTXD</b> board channel A, enter 0xEEEE (0xEEEE) For a <b>PKTXD</b> board channel B, enter 0xEEEB. The driver uses this to determine the type of card and then auto-detects the address. Do not append zeros. For a <b>5136-SD</b> board, enter the address of the card, which needs to lie on a 1K boundary (i.e. it can be changed in 0x0400 increments - e.g. 0xCC00). For a <b>Series 2</b> card the board address needs to lie on a 2K boundary (i.e. it can be changed in 0x0800 increments - e.g. 0xD800). For a <b>5136-SD-PCI</b> interface board, enter 0xFFFF. This value indicates that the board is a PCI board.
I/O Port	For a <b>KT</b> , <b>KTX</b> , <b>KTXD</b> , or <b>PKTX</b> board, enter a unique DH+ address in Octal (e.g. 0o77) If using both channels of a <b>PKTXD</b> card, you should give each channel a unique address. For a <b>5136-SD</b> board, enter the I/O Port address of the

Field	Value
	<p>card (e.g. 0x250). For a <b>5136-SD-PCI</b> board, leave this field blank.</p>
Interrupt	<p>For a <b>KT</b>, <b>KTX</b> or <b>KTXD</b> - and polling is not being used - enter the IRQ number set on the card. For a <b>PKTX</b> card, leave this field blank For a <b>5136-SD</b> card, leave this field blank. For a <b>5136-SD-PCI</b> card, enter zero.</p>
Special Options	<p>For <b>KT</b>, <b>KTX</b>, <b>KTXD</b>, or <b>PKTX ISA</b> boards the following special options are available:          -bx specifies a board number. Use this only if more than one card is installed. The default is x=1.          Sn to select the speed on the Data Highway Plus network; n represents the speed required and is one of the following numbers:          n = 1 for 115 kbaud          n = 2 for 230 kbaud          The speed is set, by default, to 57 kbaud.</p> <p>For <b>KT</b>, <b>KTX</b>, <b>KTXD</b>, or <b>PKTX PCI</b> boards the following special option is available:          Sn to select the speed on the Data Highway Plus network. n represents the speed required and is one of the following numbers:          n = 1 for 115 kbaud          n = 2 for 230 kbaud          The speed is set, by default, to 57 kbaud.</p> <p>For the <b>5136-SD</b> board the following special options are available:          -pN to set the Data Highway/Data Highway Plus address of the card to N. By default, N is a decimal value. If you need to specify the address as an octal value then N will take the format OoXX where XX is the octal address.          -Sn to select the speed on the Data Highway Plus network. n represents the speed required and is one of the following numbers:          n = 1 for 115 kbaud          n = 2 for 230 kbaud          The speed is set, by default, to 57 kbaud.</p> <p>-T to use 5136-SD-PCI interface card instead of ISA interface card.</p> <p>For the <b>5136-SD-PCI</b> board the following special options are available:          -pN to set the Data Highway/Data Highway Plus</p>

Field	Value
	<p>address of the card to N. N is, by default, in decimal format. If you wish to specify an octal value, use the format OoXX where XX is the octal address.</p> <p>-Sn to set the speed of the Data Highway Plus network, where n represents the speed, and is one of the following numbers.</p> <ul style="list-style-type: none"> <li>n = 1 for 115 kbaud</li> <li>n = 2 for 230 kbaud</li> </ul> <p>The speed is set, by default, to 57 kbaud.</p>

## Ports

Field	Value
Port Number	A unique number for each port, starting from 1.
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank
Special Options	Leave this field blank.

## I/O Devices

Field	Value
I/O Device Address	<p>Specify the PLC address as:</p> <p><b>xKT:C[options],u</b></p> <p>where:</p> <p>x = the board number. If the number is not specified in the ports for then use 1.</p> <p>C = the channel number, for example, KT card = 0, KTX card = 0 or 1.</p> <p>u = the station number of the remote PLC in octal. Station numbers are between 0 and 77 (octal) for DH+ channels, and between 1 and 376 (octal) for DH channels.</p> <p>[options] If required, you can use the <a href="#">Offlink Address</a> options to allow communication with a station (PLC) that is not on the local link.</p>

Field	Value
	For example: 1KT:0,1 specifies Port ID 1KT for access to PLC Station 1 .
I/O Device Protocol	Enter ABTCP2.

### Allen-Bradley PLC-2 via Data Highway - Data Types

Data Types	Address Format	Plant SCADA Data Type
Digital Data	x/y	DIGITAL
Integer	x	BCD / INT / LONG /LONGBCD

Where:

- **x** = Word in octal (0 - 3777)
- **y** = Bit number in octal (0 - 17)

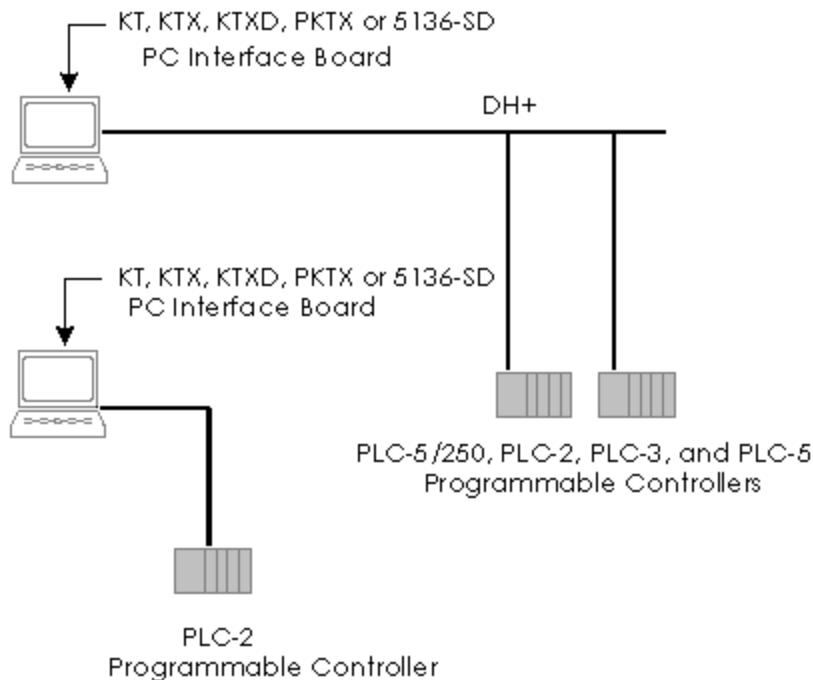
#### Examples

Data Type	DIGITAL
Address	200/12
Comment	Digital Data - Word Number 200 / Bit Number 12
Data Type	INT
Address	1700
Comment	Integer - Word Number 1700

Allen Bradley PLC-2s support remapping reads and writes.

### Allen-Bradley PLC-3 via Data Highway

Plant SCADA can use the ABTCP driver communicate to Allen-Bradley PLC-3s over Data Highway Plus via KT, KTX, KTXD, PKTX, 5136-SD and 5136-SD-PCI cards. Using this method, you can connect to single PLCs or to multiple PLCs as in the following diagram:



This arrangement does not use TCP/IP - even though the ABTCP driver is being used.

A 1775-S5 scanner module provides a DH+ port for PLC3.

This method does not support DH-485.

### Allen-Bradley PLC-3 via Data Highway - Device Address

The address for an Allen-Bradley PLC-3 connected to a KT (or KTX, KTXD, or 5136-SD) board is specified as:

**xKT:C[options],u**

Where:

- **x** = the board number. If the number is not specified in the ports for then use 1.
- **C** = the channel number, for example, KT card = 0, KTX card = 0 or 1.
- **u** = the station number of the remote PLC in octal. Station numbers are between 0 and 77 (octal) for DH+ channels, and between 1 and 376 (octal) for DH channels.
- **[options]** If required, you can use the [Offlink Address](#) options to allow communication with a station (PLC) that is not on the local link.

#### Example

**1KT:0,1** specifies Port ID **1KT** for access to PLC Station **1**.

### Allen-Bradley PLC-3 via Data Highway - KT Board Setup

Before you install a KT board in your computer, select a starting memory address and an interrupt number. Use an interrupt number and address that has not been used by any of the existing boards in your computer. If you have any other proprietary boards installed, check the documentation accompanying these boards to determine the interrupt numbers and addresses that have been used. (If you are using an EISA bus computer, run the EISA

configuration program.)

**Note:** The following information gives specific details on how to set up the KT card - not the KTX or KTXD.

## KT Card Polling vs Interrupts

You can specify the communications to operate via polling or interrupt. This is controlled by the [\[ABTCP\] Poltime](#). If you set this parameter to a value other than 0, then polling will be used instead of interrupts.

## KT Card Switch Settings

Set the configuration options of the 1784-KT board to the following. Refer to your Allen-Bradley documentation for further information.

### Board Address Switches

Memory Location Range	SW 1	SW 2	SW 3	SW 4	SW 5	SW 6
A000:0000-3F FF	Closed	Closed	Closed	Open	Closed	Open
A400:0000-3F FF	Open	Closed	Closed	Open	Closed	Open
A800:0000-3F FF	Closed	Open	Closed	Open	Closed	Open
AC00:0000-3F FF	Open	Open	Closed	Open	Closed	Open
B400:0000-3F FF	Open	Closed	Open	Open	Closed	Open
B800:0000-3F FF	Closed	Open	Open	Open	Closed	Open
C000:0000-3F FF	Closed	Closed	Closed	Closed	Open	Open
C400:0000-3F FF	Open	Closed	Closed	Closed	Open	Open
C800:0000-3F FF	Closed	Open	Closed	Closed	Open	Open
CC00:0000-3F FF *	Open	Open	Closed	Closed	Open	Open
D000:0000-3F FF	Closed	Closed	Open	Closed	Open	Open

D400:0000-3F FF	Open	Closed	Open	Closed	Open	Open
D800:0000-3F FF	Closed	Open	Open	Closed	Open	Open

\* Recommended address **0xCC00**.

This board uses a base memory size of 16k. To prevent conflict with other boards in your system, no memory overlap can exist when setting the memory address. You will also require this base memory size when configuring an EISA bus computer.

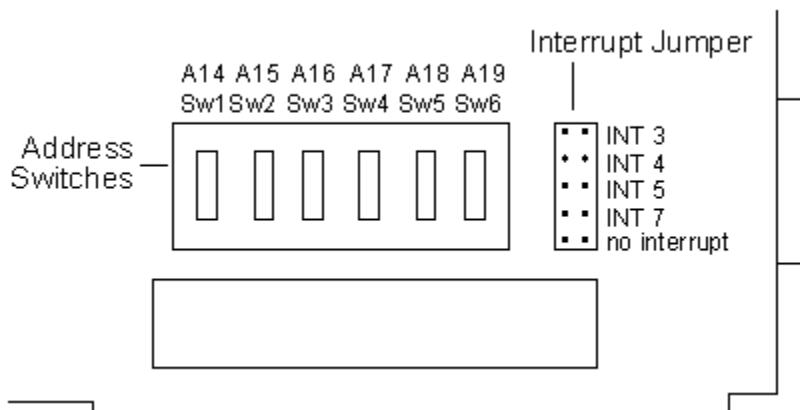
The EISA configuration utility may disable the interrupts of boards that are not set-up correctly. If you can not get a response from the KT board, check the EISA configuration for the board. If there is still no response, try another interrupt. If you set the KT board to an interrupt that is used by a COM port, the EISA configuration will move the COM port to another interrupt. If you do not get a response from the KT board, disable the COM port that was re-configured.

## Interrupt Jumper Setting

Jumper Selection	Interrupt Selection
INT 3	IRQ3
INT 4	IRQ4
INT 5	IRQ5
INT 7	IRQ7
no interrupt	NONE

\* Recommended interrupt **5**.

The following figure shows the location of the Address Switches and Interrupt Jumper.



Use an AB diagnostics application to confirm your settings.

## Setting up a PKTX Board

This is a plug and play card. No special setup is required - just confirm that the Base Memory Jumper is set to the 32-bit position. Do not attempt to manually install a driver when Windows notifies you it has detected new hardware.

Set **no** interrupts on this board. Interrupt request levels (IRQ) and base memory address values are automatically assigned. Plant SCADA will retrieve these configurations and use them to communicate with a board.

---

**Note:** To properly initialize the Windows component of the Plant SCADA CIKT board driver, you may need to restart your PC after installing the PKTX board, and then again after running your Plant SCADA project for the first time. Do not abort the project start up - allow it to execute right up to the menu page.

---

### Allen-Bradley PLC-3 via Data Highway - 5136-SD-PCI Setup

Communication with Allen-Bradley I/O devices via Data Highway Plus can be established using the 5136-SD-PCI board from SS Technologies. This board was developed with Plug and Play functionality, enabling a simple setup. Plant SCADA supports the installation and use of only one 5136-SD-PCI board in a computer at a time.

Once you have installed the board into the PCI slot in your computer, Windows detects the new hardware and assigns it an address. Plant SCADA retrieves the configuration details from Windows and, at run-time, downloads appropriate interface firmware to the board, enabling communication. No additional software is required.

Once you have installed the board, you will need to set the [Communication Settings](#).

---

**Note:** When you first start up your computer after installing the 5136-SD-PCI board, a message may display indicating that new hardware has been detected. It is not necessary to install the board using Add New Hardware in the Windows Control Panel, since Plant SCADA detects the Windows configuration. If the card has been previously installed through Control Panel, you will need to uninstall it.

---

For more information, see:

[Testing the functionality of 5136-SD-PCI boards](#)

### Allen-Bradley PLC-3 via Data Highway - 5136-SD-PCI Test

---

**Note:** For full details of the 5136-SD-PCI board, please refer to its hardware guide.

Once the 5136-SD-PCI board is installed, you can tell whether it is functioning correctly by monitoring its LEDs. You will not need to remove your computer's cover to do this, as the board's LEDs are visible at the back of the machine.

The board has two LEDs, the Network LED and the System LED. The first time you start up the computer with the board installed, the System LED will be red. When Plant SCADA successfully downloads the communications firmware to the board, it turns the System LED off. If the loader does not run successfully, the System LED will remain red.

The Network LED displays a flickering green whenever the card is active (transmitting) on the network. The Network LED should never turn off as long as there is a node on the network. If it does not display green, check your hardware or network cable.

### Allen-Bradley PLC-3 via Data Highway - Communication Settings

To establish communication with a device, the associated Board, Port and I/O Device need to be correctly configured in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure them, use the settings outlined below.

## Boards

Field	Value
Board Type	For a <b>KT</b> , <b>KTX</b> , <b>KTXD</b> , or <b>PKTX</b> card, enter CIKT. For a <b>5136-SD</b> or <b>5136-SD-PCI</b> card, enter SSTAB.
Address	For a <b>KT</b> board, enter the address of the card (e.g. 0xCC00). For a <b>KTX</b> or <b>KTXD</b> board, enter the first two digits of the address of the card (e.g. 0xD0). For a <b>PKTX</b> board, enter 0xFFFF. For a <b>PKTXD</b> board channel A, enter 0xEEEA (0xEEEE) For a <b>PKTXD</b> board channel B, enter 0xEEEB. The driver uses this to determine the type of card and then auto-detects the address. Do not append zeros. For a <b>5136-SD</b> board, enter the address of the card, which needs to be on a 1K boundary (i.e. it can be changed in 0x0400 increments - e.g. 0xCC00). For a <b>Series 2</b> card the board address needs to lie on a 2K boundary (i.e. it can be changed in 0x0800 increments - e.g. 0xD800). For a <b>5136-SD-PCI</b> interface board, enter 0xFFFF. This value indicates that the board is a PCI board.
I/O Port	For a <b>KT</b> , <b>KTX</b> , <b>KTXD</b> , or <b>PKTX</b> board, enter a unique DH+ address in Octal (e.g. 0o77) If using both channels of a <b>PKTXD</b> card, you should give each channel a unique address. For a <b>5136-SD</b> board, enter the I/O Port address of the card (e.g. 0x250). For a <b>5136-SD-PCI</b> board, leave this field blank.
Interrupt	For a <b>KT</b> , <b>KTX</b> or <b>KTXD</b> - and polling is not being used - enter the IRQ number set on the card. For a <b>PKTX</b> card, leave this field blank For a <b>5136-SD</b> card, leave this field blank. For a <b>5136-SD-PCI</b> card, enter zero.
Special Options	For <b>KT</b> , <b>KTX</b> , <b>KTXD</b> , or <b>PKTX ISA</b> boards the following special options are available: -bx specifies a board number. Use this only if more

Field	Value
	<p>than one card is installed. The default is x=1. Sn to select the speed on the Data Highway Plus network; n represents the speed required and is one of the following numbers:</p> <ul style="list-style-type: none"> <li>n = 1 for 115 kbaud</li> <li>n = 2 for 230 kbaud</li> </ul> <p>The speed is set, by default, to 57 kbaud.</p> <p>For <b>KT</b>, <b>KTX</b>, <b>KTXD</b>, or <b>PKTX PCI</b> boards the following special option is available:</p> <p>Sn to select the speed on the Data Highway Plus network. n represents the speed required and is one of the following numbers:</p> <ul style="list-style-type: none"> <li>n = 1 for 115 kbaud</li> <li>n = 2 for 230 kbaud</li> </ul> <p>The speed is set, by default, to 57 kbaud.</p> <p>For the <b>5136-SD</b> board the following special options are available:</p> <ul style="list-style-type: none"> <li>-pN to set the Data Highway/Data Highway Plus address of the card to N. By default, N is a decimal value. If you need to specify the address as an octal value then N will take the format OoXX where XX is the octal address.</li> <li>-Sn to select the speed on the Data Highway Plus network. n represents the speed required and is one of the following numbers:</li> </ul> <ul style="list-style-type: none"> <li>n = 1 for 115 kbaud</li> <li>n = 2 for 230 kbaud</li> </ul> <p>The speed is set, by default, to 57 kbaud.</p> <ul style="list-style-type: none"> <li>-T to use 5136-SD-PCI interface card instead of ISA interface card.</li> </ul> <p>For the <b>5136-SD-PCI</b> board the following special options are available:</p> <ul style="list-style-type: none"> <li>-pN to set the Data Highway/Data Highway Plus address of the card to N. N is, by default, in decimal format. If you wish to specify an octal value, use the format OoXX where XX is the octal address.</li> <li>-Sn to set the speed of the Data Highway Plus network, where n represents the speed, and is one of the following numbers.</li> </ul> <ul style="list-style-type: none"> <li>n = 1 for 115 kbaud</li> <li>n = 2 for 230 kbaud</li> </ul> <p>The speed is set, by default, to 57 kbaud.</p>

## Ports

Field	Value
Port Number	A unique number for each port, starting from 1.
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank
Special Options	Leave this field blank.

## I/O Devices

Field	Value
I/O Device Address	<p>Specify the PLC address as:</p> <p><b>xKT:C[options],u</b></p> <p>where:</p> <p><b>x</b> = the board number. If the number is not specified in the ports for then use 1.</p> <p><b>C</b> = the channel number, for example, KT card = 0, KTX card = 0 or 1.</p> <p><b>u</b> = the station number of the remote PLC in octal.</p> <p>Station numbers are between 0 and 77 (octal) for DH+ channels, and between 1 and 376 (octal) for DH channels.</p> <p><b>[options]</b> If required, you can use the <a href="#">Offlink Address</a> options to allow communication with a station (PLC) that is not on the local link.</p> <p>For example: <b>1KT:0,1</b> specifies Port ID 1KT for access to PLC Station 1 .</p>
I/O Device Protocol	Enter ABTCP3.

### Allen-Bradley PLC-3 via Data Highway - Data Types

---

**Note:** The Allen-Bradley PLC-3 has many data types. The driver makes a separate request for each different data type, and each request has a finite overhead. It is more efficient to read fewer data types, so you should move the I/O and other data into a few main blocks to be read by Plant SCADA.

Data Types	Address Format	Plant SCADA Data Type
Output image	<b>O:f:O/o</b> (f is optional)	DIGITAL
Input image	<b>I:f:O/o</b> (f is optional)	DIGITAL
Timer Control bits	<b>TCTL:w/t</b>	DIGITAL
Timer Control bits	<b>TCTL:w/o</b>	DIGITAL
Timer Preset bits	<b>TPRE:w/o</b>	DIGITAL
Timer Accumulator bits	<b>TACC:w/o</b>	DIGITAL
Counter Control bits	<b>CCTL:w/t</b>	DIGITAL
Counter Control bits	<b>CCTL:w/o</b>	DIGITAL
Counter Preset bits	<b>CPRE:w/o</b>	DIGITAL
Counter Accumulator bits	<b>CACC:w/o</b>	DIGITAL
ASCII	<b>Af:w/o</b>	DIGITAL
BCD	<b>Df:w/o</b>	DIGITAL
REAL	<b>Ef:w/o</b> (o = 0 to 37)	DIGITAL
LONG	<b>Hf:w/o</b> (o = 0 to 37)	DIGITAL
Integer	<b>Nf:w/o</b>	DIGITAL
Binary	<b>Bf:w/o</b>	DIGITAL
Status words	<b>Sf:w/o</b>	DIGITAL
Pointer Section and File	<b>PFIL:w/o</b>	DIGITAL
Pointer word	<b>PWRD:w/o</b>	DIGITAL
Output image	<b>O:f:O</b> (f is optional)	BCD / INT
Input image	<b>I:f:O</b> (f is optional)	BCD / INT
Timer Control	<b>TCTL:w</b>	BCD / INT
Timer Preset	<b>TPRE:w</b>	BCD / INT

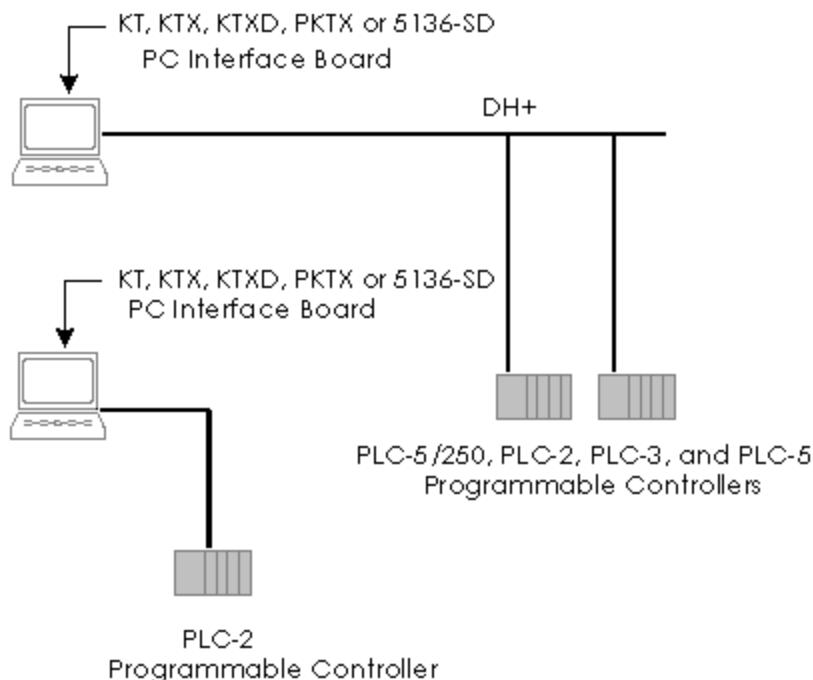
Data Types	Address Format	Plant SCADA Data Type
Timer Accumulator	<b>TACC:w</b>	BCD / INT
Counter Control	<b>CCTL:w</b>	BCD / INT
Counter Preset	<b>CPRE:w</b>	BCD / INT
Counter Accumulator	<b>CACC:w</b>	BCD / INT
Integer	<b>Nf:w</b>	BCD / INT
BCD	<b>Df:w</b>	BCD / INT
Binary	<b>Bf:w</b>	BCD / INT
ASCII	<b>Af:w</b>	BCD / INT
Status words	<b>Sf:w</b>	BCD / INT
Pointer Section and File	<b>PFIL:w</b>	BCD / INT
Pointer word	<b>PWRD:w</b>	BCD / INT
Long	<b>Hf:w</b>	LONG / LONGBCD
Floating Point	<b>Ff:w</b>	REAL

Where:

n =	Optional Module Number 0 to 8 decimal
f =	File Number 0 to 999 decimal
:w =	Element Number 0 to 9999 decimal
:O =	Element Number 0 to 7777 octal
/o =	Bit Number 0 to 17 octal PRE = Preset Value ACC = Accumulated Value
.C =	Control Sub Element LEN = Length POS = Position
.P =	PID Sub Element SP = Set Point KP = Proportional Gain KI = Integral Gain KD = Derivative Gain BIAS = Output Bias %

	MAXS = Maximum Scaled Value MINS = Minimum Scaled Value DB = Deadband SO = Set Output % MAXO = Maximum Output % MINO = Minimum Input % UPD = Update Time PVH = PV Alarm High PVL = PV Alarm Low PVDB = PV Alarm Deadband PV = Process Variable ERR = Error OUT = Output DVP = Deviation Alarm + DVN = Deviation Alarm - DVDB = Deviation Alarm Deadband MAXI = Maximum Input MINI = Minimum Input TIE = Tieback %
.t =	Timer bits EN = Enable TT = Timing DN = Done BS LK
.c =	Counter bits CU = Count Up CD = Count Down DN = Done OV = Overflow UN = Underflow LK
.r =	Control bits EN = Enable EU = Enable Unloading DN = Done EM = Empty ER = Error UL = Unload IN = Inhibit Comparisons FD = Found
.p =	PID bits EN = Enable CT = Cascaded Type

CL = Cascaded Loop
PVT = PV Tracking
DO = Derivative Of
SWM = Software A/M Mode
CA = Control Action
MO = Mode
PE = PID Equation
INI = PID Initialized
SPOR = SP Out of Range
OLL = Output Limit Low
OLH = Output Limit High
EWD = Error Within Deadband
DVPA = Deviation Low Alarm
DVNA = Deviation High Alarm
PVLA = PV Low Alarm
PVHA = PV High Alarm



This arrangement does not use TCP/IP - even though the ABTCP driver is being used.

This method does not support DH-485.

### Allen-Bradley PLC-5 via Data Highway - Device Address

The address for an Allen-Bradley PLC-5 connected to a KT (or KTX, KTXD, or 5136-SD) board is specified as:

**xKT:C[options],u**

where:

- **x** = the board number. If the number is not specified in the ports for then use 1.
- **C** = the channel number, for example, KT card = 0, KTX card = 0 or 1.
- **u** = the station number of the remote PLC in octal. Station numbers are between 0 and 77 (octal) for DH+ channels, and between 1 and 376 (octal) for DH channels.
- **[options]** If required, you can use the [Offlink Address](#) options to allow communication with a station (PLC) that is not on the local link.

### Example

**1KT:0,1** specifies Port ID **1KT** for access to PLC Station **1**.

### Allen-Bradley PLC-5 via Data Highway - KT Board Setup

Before you install a KT board in your computer, select a starting memory address and an interrupt number. Use an interrupt number and address that has not been used by any of the existing boards in your computer. If you have any other proprietary boards installed, check the documentation accompanying these boards to determine the interrupt numbers and addresses that have been used. (If you are using an EISA bus computer, you must run

the EISA configuration program.)

**Note:** The following information gives specific details on how to set up the KT card - not the KTX or KTXD.

## KT Card Polling vs Interrupts

You can specify the communications to operate via polling or interrupt. This is controlled by the [\[ABTCP\] Poltime](#). If you set this parameter to a value other than 0, then polling will be used instead of interrupts.

## KT Card Switch Settings

Set the configuration options of the 1784-KT board to the following. Refer to your Allen-Bradley documentation for further information.

### Board Address Switches

Memory Location Range	SW 1	SW 2	SW 3	SW 4	SW 5	SW 6
A000:0000-3F FF	Closed	Closed	Closed	Open	Closed	Open
A400:0000-3F FF	Open	Closed	Closed	Open	Closed	Open
A800:0000-3F FF	Closed	Open	Closed	Open	Closed	Open
AC00:0000-3F FF	Open	Open	Closed	Open	Closed	Open
B400:0000-3F FF	Open	Closed	Open	Open	Closed	Open
B800:0000-3F FF	Closed	Open	Open	Open	Closed	Open
C000:0000-3F FF	Closed	Closed	Closed	Closed	Open	Open
C400:0000-3F FF	Open	Closed	Closed	Closed	Open	Open
C800:0000-3F FF	Closed	Open	Closed	Closed	Open	Open
CC00:0000-3F FF *	Open	Open	Closed	Closed	Open	Open
D000:0000-3F FF	Closed	Closed	Open	Closed	Open	Open

D400:0000-3F FF	Open	Closed	Open	Closed	Open	Open
D800:0000-3F FF	Closed	Open	Open	Closed	Open	Open

\* Recommended address **0xCC00**.

This board uses a base memory size of 16k. To prevent conflict with other boards in your system, no memory overlap can exist when setting the memory address. You will also require this base memory size when configuring an EISA bus computer.

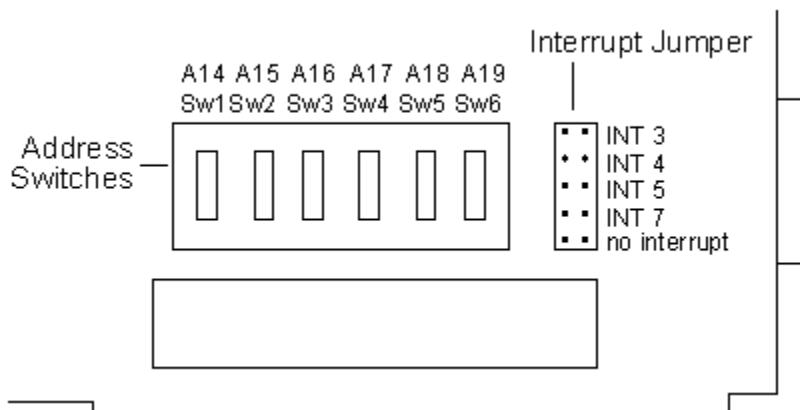
The EISA configuration utility may disable the interrupts of boards that are not set-up correctly. If you cannot get a response from the KT board, check the EISA configuration for the board. If there is still no response, try another interrupt. If you set the KT board to an interrupt that is used by a COM port, the EISA configuration will move the COM port to another interrupt. If you do not get a response from the KT board, disable the COM port that was re-configured.

## Interrupt Jumper Setting

Jumper Selection	Interrupt Selection
INT 3	IRQ3
INT 4	IRQ4
INT 5	IRQ5
INT 7	IRQ7
no interrupt	NONE

\* Recommended interrupt **5**.

The following figure shows the location of the Address Switches and Interrupt Jumper.



Use an AB diagnostics application to confirm your settings.

## Setting up a PKTX Board

This is a plug and play card. No special setup is required - just confirm that the Base Memory Jumper is set to the 32-bit position. Do not attempt to manually install a driver when Windows notifies you it has detected new hardware.

Set **no** interrupts on this board. Interrupt request levels (IRQ) and base memory address values are automatically assigned. Plant SCADA will retrieve these configurations and use them to communicate with a board.

---

**Note:** To properly initialize the Windows component of the Plant SCADA CIKT board driver, you may need to restart your PC after installing the PKTX board, and then again after running your Plant SCADA project for the first time. Do not abort the project start up - allow it to execute right up to the menu page.

---

### Allen-Bradley PLC-5 via Data Highway - 5136-SD-PCI Setup

Communication with Allen-Bradley I/O devices via Data Highway Plus can be established using the 5136-SD-PCI board from SS Technologies. This board was developed with Plug and Play functionality, enabling a simple setup. Plant SCADA supports the installation and use of only one 5136-SD-PCI board in a computer at a time.

Once you have installed the board into the PCI slot in your computer, Windows detects the new hardware and assigns it an address. Plant SCADA retrieves the configuration details from Windows and, at run-time, downloads appropriate interface firmware to the board, enabling communication. No additional software is required.

Once you have installed the board, you will need to set the [Communication Settings](#).

---

**Note:** When you first start up your computer after installing the 5136-SD-PCI board, a message may display indicating that new hardware has been detected. It is not necessary to install the board using Add New Hardware in the Windows Control Panel, since Plant SCADA detects the Windows configuration. If the card has been previously installed through Control Panel, you will need to uninstall it.

---

### Allen-Bradley PLC-5 via Data Highway - 5136-SD-PCI Test

---

**Note:** For full details of the 5136-SD-PCI board, please refer to its hardware guide.

Once the 5136-SD-PCI board is installed, you can tell whether or not it is functioning correctly by monitoring its LEDs. You will not need to remove your computer's cover to do this, as the board's LEDs are visible at the back of the machine.

The board has two LEDs, the Network LED and the System LED. The first time you start up the computer with the board installed, the System LED will be red. When Plant SCADA successfully downloads the communications firmware to the board, it turns the System LED off. If the loader does not run successfully, the System LED will remain red.

The Network LED displays a flickering green whenever the card is active (transmitting) on the network. The Network LED should never turn off as long as there is a node on the network. If it does not display green, check your hardware or network cable.

### Allen-Bradley PLC-5 via Data Highway - Communication Settings

To establish communication with a device, the associated Board, Port and I/O Device need to be correctly configured in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure them, use the settings outlined below.

## Boards

Field	Value
Board Type	For a <b>KT</b> , <b>KTX</b> , <b>KTXD</b> , or <b>PKTX</b> card, enter CIKT. For a <b>5136-SD</b> or <b>5136-SD-PCI</b> card, enter SSTAB.
Address	For a <b>KT</b> board, enter the address of the card (e.g. 0xCC00). For a <b>KTX</b> or <b>KTXD</b> board, enter the first two digits of the address of the card (e.g. 0xD0). For a <b>PKTX</b> board, enter 0xFFFF. For a <b>PKTXD</b> board channel A, enter 0xEEEA (0xEEEE) For a <b>PKTXD</b> board channel B, enter 0xEEEB. The driver uses this to determine the type of card and then auto-detects the address. Do not append zeros. For a <b>5136-SD</b> board, enter the address of the card, which needs to lie on a 1K boundary (i.e. it can be changed in 0x0400 increments - e.g. 0xCC00). For a <b>Series 2</b> card the board address needs to lie on a 2K boundary (i.e. it can be changed in 0x0800 increments - e.g. 0xD800). For a <b>5136-SD-PCI</b> interface board, enter 0xFFFF. This value indicates that the board is a PCI board.
I/O Port	For a <b>KT</b> , <b>KTX</b> , <b>KTXD</b> , or <b>PKTX</b> board, enter a unique DH+ address in Octal (e.g. 0o77) If using both channels of a <b>PKTXD</b> card, you should give each channel a unique address. For a <b>5136-SD</b> board, enter the I/O Port address of the card (e.g. 0x250). For a <b>5136-SD-PCI</b> board, leave this field blank.
Interrupt	For a <b>KT</b> , <b>KTX</b> or <b>KTXD</b> - and polling is not being used - enter the IRQ number set on the card. For a <b>PKTX</b> card, leave this field blank For a <b>5136-SD</b> card, leave this field blank. For a <b>5136-SD-PCI</b> card, enter zero.
Special Options	For <b>KT</b> , <b>KTX</b> , <b>KTXD</b> , or <b>PKTX ISA</b> boards the following special options are available: -bx specifies a board number. Use this only if more than one card is installed. The default is x=1. Sn to select the speed on the Data Highway Plus network; n represents the speed required and is one

Field	Value
	<p>of the following numbers:  n = 1 for 115 kbaud  n = 2 for 230 kbaud  The speed is set, by default, to 57 kbaud.</p> <p>For <b>KT</b>, <b>KTX</b>, <b>KTXD</b>, or <b>PKTX PCI</b> boards the following special option is available:</p> <p>-Sn to select the speed on the Data Highway Plus network. n represents the speed required and is one of the following numbers:</p> <p>n = 1 for 115 kbaud  n = 2 for 230 kbaud  The speed is set, by default, to 57 kbaud.</p> <p>For the <b>5136-SD</b> board the following special options are available:</p> <p>-pN to set the Data Highway/Data Highway Plus address of the card to N. By default, N is a decimal value. If you need to specify the address as an octal value then N will take the format OoXX where XX is the octal address.</p> <p>-Sn to select the speed on the Data Highway Plus network. n represents the speed required and is one of the following numbers:</p> <p>n = 1 for 115 kbaud  n = 2 for 230 kbaud  The speed is set, by default, to 57 kbaud.</p> <p>-T to use 5136-SD-PCI interface card instead of ISA interface card.</p> <p>For the <b>5136-SD-PCI</b> board the following special options are available:</p> <p>-pN to set the Data Highway/Data Highway Plus address of the card to N. N is, by default, in decimal format. If you wish to specify an octal value, use the format OoXX where XX is the octal address.</p> <p>-Sn to set the speed of the Data Highway Plus network, where n represents the speed, and is one of the following numbers.</p> <p>n = 1 for 115 kbaud  n = 2 for 230 kbaud  The speed is set, by default, to 57 kbaud.</p>

## Ports

Field	Value
Port Number	A unique number for each port, starting from 1.
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank
Special Options	Leave this field blank.

## I/O Devices

Field	Value
I/O Device Address	<p>Specify the PLC address as:</p> <p>xKT:C[options],u</p> <p>where:</p> <p>x = the board number. If the number is not specified in the ports for then use 1.</p> <p>C = the channel number, for example, KT card = 0, KTX card = 0 or 1.</p> <p>u = the station number of the remote PLC in octal.</p> <p>Station numbers are between 0 and 77 (octal) for DH+ channels, and between 1 and 376 (octal) for DH channels.</p> <p>[options] If required, you can use the <a href="#">Offlink Address</a> options to allow communication with a station (PLC) that is not on the local link.</p> <p>For example: 1KT:0,1 specifies Port ID 1KT for access to PLC Station 1 .</p>
I/O Device Protocol	Enter ABTCP5.

## Allen-Bradley PLC-5 via Data Highway - Data Types

---

**Note:** The Allen-Bradley PLC-5 has many data types. The driver makes a separate request for each different data type, and each request has a finite overhead. It is more efficient to read fewer data types, so you should move the I/O and other data into a few main blocks to be read by Plant SCADA.

Data Types	Address Format	Plant SCADA Data Type
Output image	<b>O:O/o</b>	DIGITAL
Input image	<b>I:O/o</b>	DIGITAL
Status words 0 to 127 (to 128 for w parameter)	<b>S:w/b</b>	DIGITAL
Timer	<b>Tf:w.t</b>	DIGITAL
Counter	<b>Cf:w.c</b>	DIGITAL
Control	<b>Rf:w.r</b>	DIGITAL
Binary	<b>Bf:w/b</b>	DIGITAL
Integer	<b>Nf:w/b</b>	DIGITAL
Sequential Function Chart	<b>SCf:w.s</b>	DIGITAL
Block Transfer	<b>BTf:w.b</b>	DIGITAL
Floating-point	<b>Ff:w/b</b>	DIGITAL
PID	<b>PDf:w.p</b>	DIGITAL
Output image	<b>O:O</b>	BCD / INT / LONG / LONGBCD
Input image	<b>I:O</b>	BCD / INT / LONG / LONGBCD
Status words 0 to 127 (to 128 for w parameter)	<b>S:w</b>	BCD / INT / LONG / LONGBCD
Timer	<b>Tf:w.T</b>	BCD / INT / LONG / LONGBCD
Counter	<b>Cf:w.T</b>	BCD / INT / LONG / LONGBCD
Control	<b>Rf:w.R</b>	BCD / INT / LONG / LONGBCD
Binary	<b>Bf:w</b>	BCD / INT / LONG / LONGBCD
Integer	<b>Nf:w</b>	BCD / INT / LONG / LONGBCD
Sequential Function Chart	<b>SCf:w.S</b>	BCD / INT / LONG / LONGBCD
Block Transfer	<b>BTf:w.B</b>	BCD / INT / LONG / LONGBCD
ASCII	<b>Af:w</b>	BCD / INT / LONG / LONGBCD
BCD	<b>Df:w</b>	BCD / INT / LONG / LONGBCD

Data Types	Address Format	Plant SCADA Data Type
Floating-point	<b>Ff:w</b>	REAL
PID	<b>PDf:w.P</b>	REAL
String 82 bytes long	<b>STf:w</b>	STRING

Where:

f	File Number 0 to 999 decimal
:w	Element number 0 to 999 decimal
:O	Element number 0 to 277 octal
/b	Bit Number 0 to 15 decimal
/o	Bit Number 0 to 17 octal
.T	Timer/Counter Sub Element: PRE = Preset Value ACC = Accumulated Value
.t	Timer bits: EN = Enable TT = Timing DN = Done
.C	Control Sub Element: LEN = Length POS = Position
.c	Counter bits: CU = Count Up CD = Count Down DN = Done OV = Overflow UN = Underflow
.r	Control bits: EN = Enable EU = Enable Unloading DN = Done EM = Empty ER = Error UL = Unload IN = Inhibit Comparisons FD = Found

.S	SFC Sub Element: PRE = Preset Value TIM = Elapsed step active time
.S	SFC bits: SA = Scan Active FS = First Scan LS = Last Scan OV = Timer Overflow ER = Error DN = Done
.B	Block Transfer Sub Element: RLEN = Received Length DLEN = Done Length FILE = File ELEM = Element
.b	Block Transfer Bits: EN = Enable TT = Start DN = Done ER = Error CO = Continuous EW = Enable Wait NR = No Response TO = Time Out RW = Read Write
.P	PID Sub Element: SP = Set Point KP = Proportional Gain KI = Integral Gain KD = Derivative Gain BIAS = Output Bias % MAXS = Maximum Scaled Value MINS = Minimum Scaled Value DB = Deadband SO = Set Output % MAXO = Maximum Output % MINO = Minimum Input % UPD = Update Time PVH = PV Alarm High PVL = PV Alarm Low PVDB = PV Alarm Deadband PV = Process Variable ERR = Error OUT = Output

	DVP = Deviation Alarm + DVN = Deviation Alarm - DVDB = Deviation Alarm Deadband MAXI = Maximum Input MINI = Minimum Input TIE = Tieback %
.p	PID bits:  EN = Enable CT = Cascaded Type CL = Cascaded Loop PVT = PV Tracking DO = Derivative Of SWM = Software A/M Mode CA = Control Action MO = Mode PE = PID Equation INI = PID Initialized SPOR = SP Out of Range OLL = Output Limit Low OLH = Output Limit High EWD = Error Within Deadband DVPA = Deviation Low Alarm DVNA = Deviation High Alarm PVLA = PV Low Alarm PVHA = PV High Alarm

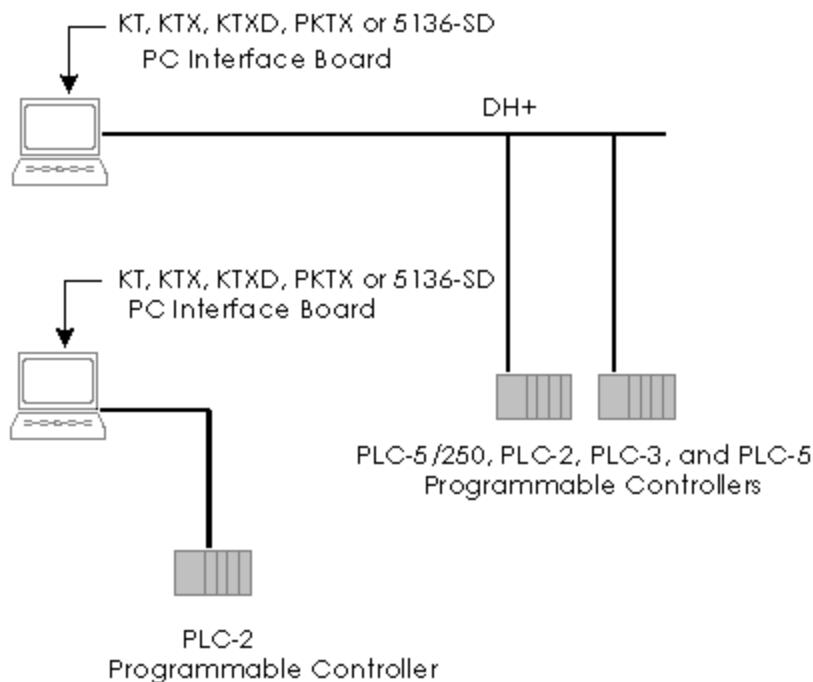
**Examples**

Data Type	DIGITAL
Address	I:1/15
Comment	Input Image - Element Number 1 / Bit Number 15
Data Type	INT
Address	N7:3
Comment	Integer - File Number 7 : Element Number 3

Allen Bradley PLC-5 Series support remapping reads and writes.

**Allen-Bradley PLC-5/250 via Data Highway**

Plant SCADA can use the ABTCP driver communicate to Allen-Bradley PLC-5/250s over Data Highway Plus via KT, KTX, KTXD, PKTX, 5136-SD and 5136-SD-PCI cards. Using this method you can connect to single PLCs or to multiple PLCs as in the following diagram:



This arrangement does not use TCP/IP even though the ABTCP driver is being used.

This method does not support DH-485.

### Allen-Bradley PLC-5/250 via Data Highway - Device Address

The address for an Allen-Bradley PLC-5/250 connected to a KT (or KTX, KTXD, or 5136-SD) board is specified as:

**xKT:C[options],u**

where:

- **x** = the board number. If the number is not specified in the ports for then use 1.
- **C** = the channel number, for example, KT card = 0, KTX card = 0 or 1.
- **u** = the station number of the remote PLC in octal. Station numbers are between 0 and 77 (octal) for DH+ channels, and between 1 and 376 (octal) for DH channels.
- **[options]** If required, you can use the [Offlink Address](#) options to allow communication with a station (PLC) that is not on the local link.

#### Example

**1KT:0,1** specifies Port ID **1KT** for access to PLC Station **1**.

### Allen-Bradley PLC-5/250 via Data Highway - KT Board Setup

Before you install a KT board in your computer, select a starting memory address and an interrupt number. Use an interrupt number and address that has not been used by any of the existing boards in your computer. If you have any other proprietary boards installed, check the documentation accompanying these boards to determine the interrupt numbers and addresses that have been used. (If you are using an EISA bus computer, you must run the EISA configuration program.)

**Note:** The following information gives specific details on how to set up the KT card - not the KTX or KTXD.

## KT Card Polling vs Interrupts

You can specify the communications to operate via polling or interrupt. This is controlled by the [\[ABTCP\] Polltime](#). If you set this parameter to a value other than 0, then polling will be used instead of interrupts.

## KT Card Switch Settings

Set the configuration options of the 1784-KT board to the following. Refer to your Allen-Bradley documentation for further information.

### Board Address Switches

Memory Location Range	SW 1	SW 2	SW 3	SW 4	SW 5	SW 6
A000:0000-3F FF	Closed	Closed	Closed	Open	Closed	Open
A400:0000-3F FF	Open	Closed	Closed	Open	Closed	Open
A800:0000-3F FF	Closed	Open	Closed	Open	Closed	Open
AC00:0000-3F FF	Open	Open	Closed	Open	Closed	Open
B400:0000-3F FF	Open	Closed	Open	Open	Closed	Open
B800:0000-3F FF	Closed	Open	Open	Open	Closed	Open
C000:0000-3F FF	Closed	Closed	Closed	Closed	Open	Open
C400:0000-3F FF	Open	Closed	Closed	Closed	Open	Open
C800:0000-3F FF	Closed	Open	Closed	Closed	Open	Open
CC00:0000-3F FF *	Open	Open	Closed	Closed	Open	Open
D000:0000-3F FF	Closed	Closed	Open	Closed	Open	Open

D400:0000-3F FF	Open	Closed	Open	Closed	Open	Open
D800:0000-3F FF	Closed	Open	Open	Closed	Open	Open

\* Recommended address **0xCC00**.

This board uses a base memory size of 16k. To prevent conflict with other boards in your system, no memory overlap can exist when setting the memory address. You will also require this base memory size when configuring an EISA bus computer.

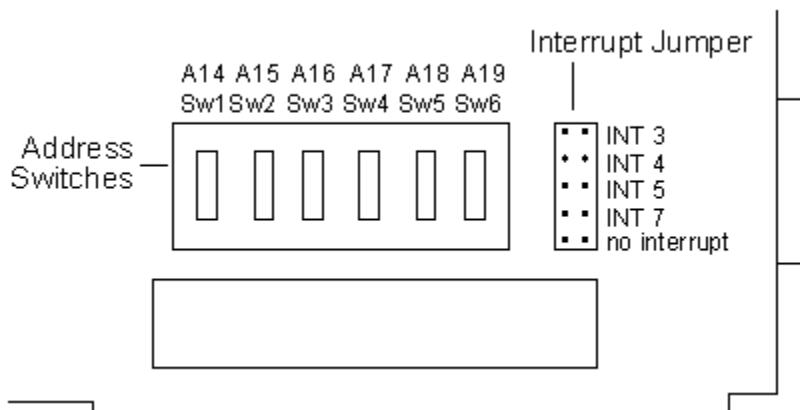
The EISA configuration utility may disable the interrupts of boards that are not set-up correctly. If you cannot get a response from the KT board, check the EISA configuration for the board. If there is still no response, try another interrupt. If you set the KT board to an interrupt that is used by a COM port, the EISA configuration will move the COM port to another interrupt. If you do not get a response from the KT board, disable the COM port that was re-configured.

## Interrupt Jumper Setting

Jumper Selection	Interrupt Selection
INT 3	IRQ3
INT 4	IRQ4
INT 5	IRQ5
INT 7	IRQ7
no interrupt	NONE

\* Recommended interrupt **5**.

The following figure shows the location of the Address Switches and Interrupt Jumper.



Use an AB diagnostics application to confirm your settings.

## Setting up a PKTX Board

This is a plug and play card. No special setup is required - just confirm that the Base Memory Jumper is set to the 32-bit position. Do not attempt to manually install a driver when Windows notifies you it has detected new hardware.

It is recommended that **no** interrupts are set on this board. Interrupt request levels (IRQ) and base memory address values are automatically assigned. Plant SCADA will retrieve these configurations and use them to communicate with a board.

---

**Note:** To properly initialize the Windows component of the Plant SCADA CIKT board driver, you may need to restart your PC after installing the PKTX board, and then again after running your Plant SCADA project for the first time. Do not abort the project start up - allow it to execute right up to the menu page.

---

### Bradley PLC-5/250 via Data Highway - 5136-SD-PCI Setup

Communication with Allen-Bradley I/O devices via Data Highway Plus can be established using the 5136-SD-PCI board from SS Technologies. This board was developed with Plug and Play functionality, enabling a simple setup. Plant SCADA supports the installation and use of only one 5136-SD-PCI board in a computer at a time.

Once you have installed the board into the PCI slot in your computer, Windows detects the new hardware and assigns it an address. Plant SCADA retrieves the configuration details from Windows and, at run-time, downloads appropriate interface firmware to the board, enabling communication. No additional software is required.

Once you have installed the board, you will need to set the [Communication Settings](#).

---

**Note:** When you first start up your computer after installing the 5136-SD-PCI board, a message may display indicating that new hardware has been detected. It is not necessary to install the board using Add New Hardware in the Windows Control Panel, since Plant SCADA detects the Windows configuration. If the card has been previously installed through Control Panel, you will need to uninstall it.

---

### Allen-Bradley PLC-5/250 via Data Highway - 5136-SD-PCI Test

---

**Note:** For full details of the 5136-SD-PCI board, please refer to its hardware guide.

Once the 5136-SD-PCI board is installed, you can tell whether it is functioning correctly by monitoring its LEDs. You will not need to remove your computer's cover to do this, as the board's LEDs are visible at the back of the machine.

The board has two LEDs, the Network LED and the System LED. The first time you start up the computer with the board installed, the System LED will be red. When Plant SCADA successfully downloads the communications firmware to the board, it turns the System LED off. If the loader does not run successfully, the System LED will remain red.

The Network LED displays a flickering green whenever the card is active (transmitting) on the network. The Network LED should never turn off as long as there is a node on the network. If it does not display green, check your hardware or network cable.

### Allen-Bradley PLC-5/250 via Data Highway - Communication Settings

To establish communication with a device, the associated Board, Port and I/O Device need to be correctly configured in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically

configured for you. However, if you need to manually configure them, use the settings outlined below.

## Boards

Field	Value
Board Type	For a <b>KT</b> , <b>KTX</b> , <b>KTXD</b> , or <b>PKTX</b> card, enter CIKT. For a <b>5136-SD</b> or <b>5136-SD-PCI</b> card, enter SSTAB.
Address	For a <b>KT</b> board, enter the address of the card (e.g. 0xCC00). For a <b>KTX</b> or <b>KTXD</b> board, enter the first two digits of the address of the card (e.g. 0xD0). For a <b>PKTX</b> board, enter 0xFFFF. For a <b>PKTXD</b> board channel A, enter 0xEEEA (0xEEEE) For a <b>PKTXD</b> board channel B, enter 0xEEEB. The driver uses this to determine the type of card and then auto-detects the address. Do not append zeros. For a <b>5136-SD</b> board, enter the address of the card, which needs to lie on a 1K boundary (i.e. it can be changed in 0x0400 increments - e.g. 0xCC00). For a <b>Series 2</b> card the board address needs to lie on a 2K boundary (i.e. it can be changed in 0x0800 increments - e.g. 0xD800). For a <b>5136-SD-PCI</b> interface board, enter 0xFFFF. This value indicates that the board is a PCI board.
I/O Port	For a <b>KT</b> , <b>KTX</b> , <b>KTXD</b> , or <b>PKTX</b> board, enter a unique DH+ address in Octal (e.g. 0o77) If using both channels of a <b>PKTXD</b> card, you should give each channel a unique address. For a <b>5136-SD</b> board, enter the I/O Port address of the card (e.g. 0x250). For a <b>5136-SD-PCI</b> board, leave this field blank.
Interrupt	For a <b>KT</b> , <b>KTX</b> or <b>KTXD</b> - and polling is not being used - enter the IRQ number set on the card. For a <b>PKTX</b> card, leave this field blank For a <b>5136-SD</b> card, leave this field blank. For a <b>5136-SD-PCI</b> card, enter zero.
Special Options	For <b>KT</b> , <b>KTX</b> , <b>KTXD</b> , or <b>PKTX</b> ISA boards the following special options are available: -bx specifies a board number. Use this only if more than one card is installed. The default is x=1.

Field	Value
	<p>Sn to select the speed on the Data Highway Plus network; n represents the speed required and is one of the following numbers:</p> <p>n = 1 for 115 kbaud  n = 2 for 230 kbaud  The speed is set, by default, to 57 kbaud.</p> <p>For <b>KT, KTX, KTXD, or PKTX PCI</b> boards the following special option is available:</p> <p>Sn to select the speed on the Data Highway Plus network. n represents the speed required and is one of the following numbers:</p> <p>n = 1 for 115 kbaud  n = 2 for 230 kbaud  The speed is set, by default, to 57 kbaud.</p> <p>For the <b>5136-SD</b> board the following special options are available:</p> <ul style="list-style-type: none"> <li>-pN to set the Data Highway/Data Highway Plus address of the card to N. By default, N is a decimal value. If you need to specify the address as an octal value then N will take the format OoXX where XX is the octal address.</li> <li>-Sn to select the speed on the Data Highway Plus network. n represents the speed required and is one of the following numbers:</li> </ul> <p>n = 1 for 115 kbaud  n = 2 for 230 kbaud  The speed is set, by default, to 57 kbaud.</p> <ul style="list-style-type: none"> <li>-T to use 5136-SD-PCI interface card instead of ISA interface card.</li> </ul> <p>For the <b>5136-SD-PCI</b> board the following special options are available:</p> <ul style="list-style-type: none"> <li>-pN to set the Data Highway/Data Highway Plus address of the card to N. N is, by default, in decimal format. If you wish to specify an octal value, use the format OoXX where XX is the octal address.</li> <li>-Sn to set the speed of the Data Highway Plus network, where n represents the speed, and is one of the following numbers.</li> </ul> <p>n = 1 for 115 kbaud  n = 2 for 230 kbaud  The speed is set, by default, to 57 kbaud.</p>

## Ports

Field	Value
Port Number	A unique number for each port, starting from 1.
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank
Special Options	Leave this field blank.

## I/O Devices

Field	Value
I/O Device Address	<p>Specify the PLC address as:</p> <p>xKT:C[options],u</p> <p>where:</p> <p>x = the board number. If the number is not specified in the ports for then use 1.</p> <p>C = the channel number, for example, KT card = 0, KTX card = 0 or 1.</p> <p>u = the station number of the remote PLC in octal.</p> <p>Station numbers are between 0 and 77 (octal) for DH+ channels, and between 1 and 376 (octal) for DH channels.</p> <p>[options] If required, you can use the <a href="#">Offlink Address</a> options to allow communication with a station (PLC) that is not on the local link.</p> <p>For example: 1KT:0,1 specifies Port ID 1KT for access to PLC Station 1 .</p>
I/O Device Protocol	Enter ABTCP250.

### Allen-Bradley PLC-5/250 via Data Highway - Data Types

---

**Note:** The Allen-Bradley PLC-5/250 has many data types. The driver makes a separate request for each different data type, and each request has a finite overhead. It is more efficient to read fewer data types, so you should move the I/O and other data into a few main blocks to be read by Plant SCADA.

Data Types	Address Format	Plant SCADA Data Type
Output image	<b>O:O/o</b>	DIGITAL
Input image	<b>I:O/o</b>	DIGITAL
Status words 0 to 350	<b>S:w/b</b>	DIGITAL
Internal storage	<b>IS:w/b</b>	DIGITAL
Status words 0 to 31	<b>nS:w/b</b>	DIGITAL
Timer	<b>nTf:w.t</b>	DIGITAL
Timer	<b>nTf:w/b</b>	DIGITAL
Counter	<b>nCf:w.c</b>	DIGITAL
Counter	<b>nCf:w/b</b>	DIGITAL
Control	<b>nRf:w.r</b>	DIGITAL
Control	<b>nRf:w/b</b>	DIGITAL
Binary	<b>nBf:w/b</b>	DIGITAL
Integer	<b>nNf:w/b</b>	DIGITAL
Floating Point	<b>nFf:w/b</b>	DIGITAL
Long	<b>nLf:w/b</b>	DIGITAL
PID	<b>nPDf:w.p</b>	DIGITAL
Block transfer data	<b>nBTdf:w/b</b>	DIGITAL
Output image	<b>O:O</b>	BCD / INT
Input image	<b>I:O</b>	BCD / INT
Status words 0 to 350	<b>S:w</b>	BCD / INT
Internal storage	<b>IS:w</b>	BCD / INT
Status words 0 to 31	<b>nS:w</b>	BCD / INT
Counter	<b>nCf:w.T</b>	BCD / INT
Control	<b>nRf:w.C</b>	BCD / INT
Binary	<b>nBf:w</b>	BCD / INT
Integer	<b>nNf:w</b>	BCD / INT

Data Types	Address Format	Plant SCADA Data Type
Block transfer data	<b>nBTDf:w</b>	BCD / INT
Shared data words 0-1023	<b>nSDf:w</b>	BCD / INT
Timer	<b>nTf:w.T</b>	LONG / LONGBCD
Long	<b>nLf:w</b>	LONG / LONGBCD
Floating Point	<b>nFf:w</b>	REAL
PID	<b>nPDf:w.P</b>	REAL
String 82 bytes long	<b>nSTf:w</b>	STRING

Where:

<b>n</b>	Optional Module Number 0 to 8 decimal
<b>f</b>	File Number 0 to 9999 decimal
<b>:w</b>	Element Number 0 to 9999 decimal
<b>:O</b>	Element Number 0 to 777 octal
<b>/b</b>	Bit Number 0 to 15 decimal
<b>/o</b>	Bit Number 0 to 17 octal
<b>.T</b>	Timer/Counter Sub Element PRE = Preset Value ACC = Accumulated Value
<b>.C</b>	Control Sub Element LEN = Length POS = Position
<b>.P</b>	PID Sub Element SP = Set Point KP = Proportional Gain KI = Integral Gain KD = Derivative Gain BIAS = Output Bias % MAXS = Maximum Scaled Value MINS = Minimum Scaled Value DB = Deadband SO = Set Output % MAXO = Maximum Output % MINO = Minimum Input % UPD = Update Time

	PVH = PV Alarm High PVL = PV Alarm Low PVDB = PV Alarm Deadband PV = Process Variable ERR = Error OUT = Output DVP = Deviation Alarm + DVN = Deviation Alarm - DVDB = Deviation Alarm Deadband MAXI = Maximum Input MINI = Minimum Input TIE = Tieback %
.t	Timer bits EN = Enable TT = Timing DN = Done BS LK
.c	Counter bits CU = Count Up CD = Count Down DN = Done OV = Overflow UN = Underflow LK
.r	Control bits EN = Enable EU = Enable Unloading DN = Done EM = Empty ER = Error UL = Unload IN = Inhibit Comparisons FD = Found
.p	PID bits EN = Enable CT = Cascaded Type CL = Cascaded Loop PVT = PV Tracking DO = Derivative Of SWM = Software A/M Mode CA = Control Action MO = Mode PE = PID Equation INI = PID Initialized SPOR = SP Out of Range OLL = Output Limit Low

OLH = Output Limit High  
 EWD = Error Within Deadband  
 DVPA = Deviation Low Alarm  
 DVNA = Deviation High Alarm  
 PVLA = PV Low Alarm  
 PVHA = PV High Alarm

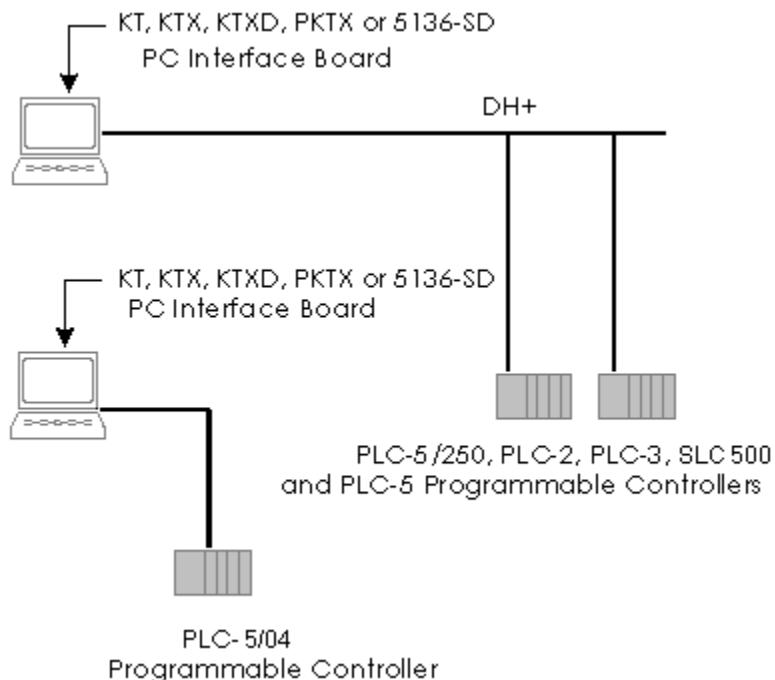
### Examples

Data Type	DIGITAL
Address	I:1/15
Comment	Input Image - Element Number 1 / Bit Number 15
Data Type	INT
Address	N7:3
Comment	Integer - File Number 7 : Element Number 3

Allen Bradley 5/250 PLCs support remapping reads and writes.

### Allen-Bradley SLC-500 via Data Highway

Plant SCADA can use the ABTCP driver communicate to Allen-Bradley PLC-5/250s over Data Highway Plus via KT, KTX, KTXD, PKTX, 5136-SD and 5136-SD-PCI cards. Using this method, you can connect to single PLCs or to multiple PLCs as in the following diagram:



Only the SLC-5/04 can support a direct DH+ connection.

This method does not support DH-485.

### Allen-Bradley SLC-500 via Data Highway - Device Address

The address for an Allen-Bradley SLC-500 connected to a KT (or KTX, KTXD, or 5136-SD) board is specified as:

**xKT:C[options],u**

Where:

- **x** = the board number. If the number is not specified in the ports for then use 1.
- **C** = the channel number, for example, KT card = 0, KTX card = 0 or 1.
- **u** = the station number of the remote PLC in octal. Station numbers are between 0 and 77 (octal) for DH+ channels, and between 1 and 376 (octal) for DH channels.
- **[options]** If required, you can use the [Offlink Address](#) options to allow communication with a station (PLC) that is not on the local link.

#### Example

**1KT:0,1** specifies Port ID **1KT** for access to PLC Station **1**.

### Allen-Bradley SLC-500 via Data Highway - KT Board Setup

Before you install a KT board in your computer, select a starting memory address and an interrupt number. Use an interrupt number and address that has not been used by any of the existing boards in your computer. If you have any other proprietary boards installed, check the documentation accompanying these boards to determine the interrupt numbers and addresses that have been used. (If you are using an EISA bus computer, you must run the EISA configuration program.)

**Note:** The following information gives specific details on how to set up the KT card - not the KTX or KTXD.

### KT Card Polling Vs Interrupts

You can specify the communications to operate via polling or interrupt. This is controlled by the [\[ABTCP\] Polltime](#). If you set this parameter to a value other than 0, then polling will be used instead of interrupts.

### KT Card Switch Settings

Set the configuration options of the 1784-KT board to the following. Refer to your Allen-Bradley documentation for further information.

#### Board Address Switches

Memory Location Range	SW 1	SW 2	SW 3	SW 4	SW 5	SW 6
A000:0000-3F FF	Closed	Closed	Closed	Open	Closed	Open
A400:0000-3F	Open	Closed	Closed	Open	Closed	Open

FF						
A800:0000-3F FF	Closed	Open	Closed	Open	Closed	Open
AC00:0000-3F FF	Open	Open	Closed	Open	Closed	Open
B400:0000-3F FF	Open	Closed	Open	Open	Closed	Open
B800:0000-3F FF	Closed	Open	Open	Open	Closed	Open
C000:0000-3F FF	Closed	Closed	Closed	Closed	Open	Open
C400:0000-3F FF	Open	Closed	Closed	Closed	Open	Open
C800:0000-3F FF	Closed	Open	Closed	Closed	Open	Open
CC00:0000-3F FF *	Open	Open	Closed	Closed	Open	Open
D000:0000-3F FF	Closed	Closed	Open	Closed	Open	Open
D400:0000-3F FF	Open	Closed	Open	Closed	Open	Open
D800:0000-3F FF	Closed	Open	Open	Closed	Open	Open

\* Recommended address **0xCC00**.

This board uses a base memory size of 16k. To prevent conflict with other boards in your system, no memory overlap can exist when setting the memory address. You will also require this base memory size when configuring an EISA bus computer.

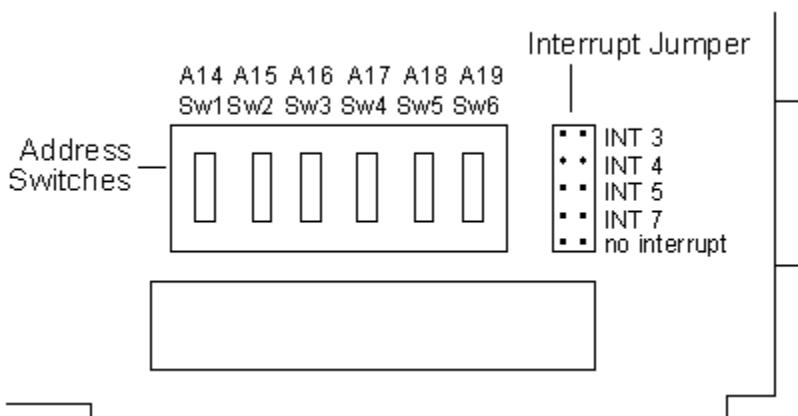
The EISA configuration utility may disable the interrupts of boards that are not set-up correctly. If you cannot get a response from the KT board, check the EISA configuration for the board. If there is still no response, try another interrupt. If you set the KT board to an interrupt that is used by a COM port, the EISA configuration will move the COM port to another interrupt. If you do not get a response from the KT board, disable the COM port that was re-configured.

## Interrupt Jumper Setting

Jumper Selection	Interrupt Selection
INT 3	IRQ3
INT 4	IRQ4
INT 5	IRQ5
INT 7	IRQ7
no interrupt	NONE

\* Recommended interrupt 5.

The following figure shows the location of the Address Switches and Interrupt Jumper.



Use an AB diagnostics application to confirm your settings.

## Setting up a PKTX Board

This is a plug and play card. No special setup is required - just confirm that the Base Memory Jumper is set to the 32-bit position. Do not attempt to manually install a driver when Windows notifies you it has detected new hardware.

Set **no** interrupts on this board. Interrupt request levels (IRQ) and base memory address values are automatically assigned. Plant SCADA will retrieve these configurations and use them to communicate with a board.

---

**Note:** To properly initialize the Windows component of the Plant SCADA CIKT board driver, you may need to restart your PC after installing the PKTX board, and then again after running your Plant SCADA project for the first time. Do not abort the project start up - allow it to execute right up to the menu page.

---

## Allen-Bradley SLC-500 via Data Highway - 5136-SD-PCI Setup

Communication with Allen-Bradley I/O devices via Data Highway Plus can be established using the 5136-SD-PCI board from SS Technologies. This board was developed with Plug and Play functionality, enabling a simple setup. Plant SCADA supports the installation and use of only one 5136-SD-PCI board in a computer at a time.

Once you have installed the board into the PCI slot in your computer, Windows detects the new hardware and assigns it an address. Plant SCADA retrieves the configuration details from Windows and, at run-time, downloads appropriate interface firmware to the board, enabling communication. No additional software is required.

Once you have installed the board, you will need to set the [Communication Settings](#).

**Note:** When you first start up your computer after installing the 5136-SD-PCI board, a message may display indicating that new hardware has been detected. It is not necessary to install the board using Add New Hardware in the Windows Control Panel, since Plant SCADA detects the Windows configuration. If the card has been previously installed through Control Panel, you will need to uninstall it.

### Allen-Bradley SLC-500 via Data Highway - 5136-SD-PCI Test

**Note:** For full details of the 5136-SD-PCI board, please refer to its hardware guide.

Once the 5136-SD-PCI board is installed, you can tell whether or not it is functioning correctly by monitoring its LEDs. You will not need to remove your computer's cover to do this, as the board's LEDs are visible at the back of the machine.

The board has two LEDs, the Network LED and the System LED. The first time you start up the computer with the board installed, the System LED will be red. When Plant SCADA successfully downloads the communications firmware to the board, it turns the System LED off. If the loader does not run successfully, the System LED will remain red.

The Network LED displays a flickering green whenever the card is active (transmitting) on the network. The Network LED should never turn off as long as there is a node on the network. If it does not display green, check your hardware or network cable.

### Allen-Bradley SLC-500 via Data Highway - Communication Settings

To establish communication with a device, the associated Board, Port and I/O Device need to be correctly configured in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure them, use the settings outlined below.

## Boards

Field	Value
Board Type	For a <b>KT</b> , <b>KTX</b> , <b>KTXD</b> , or <b>PKTX</b> card, enter CIKT. For a <b>5136-SD</b> or <b>5136-SD-PCI</b> card, enter SSTAB.
Address	For a <b>KT</b> board, enter the address of the card (e.g. 0xCC00). For a <b>KTX</b> or <b>KTXD</b> board, enter the first two digits of the address of the card (e.g. 0xD0). For a <b>PKTX</b> board, enter 0xFFFF. For a <b>PKTXD</b> board channel A, enter 0xEEEE (0xEEEE) For a <b>PKTXD</b> board channel B, enter 0xEEEB. The driver uses this to determine the type of card and then auto-detects the address. Do not append zeros. For a <b>5136-SD</b> board, enter the address of the card,

Field	Value
	<p>which needs to lie on a 1K boundary (i.e. it can be changed in 0x0400 increments - e.g. 0xCC00).</p> <p>For a <b>Series 2</b> card the board address needs to lie on a 2K boundary (i.e. it can be changed in 0x0800 increments - e.g. 0xD800).</p> <p>For a <b>5136-SD-PCI</b> interface board, enter 0xFFFF. This value indicates that the board is a PCI board.</p>
I/O Port	<p>For a <b>KT, KTX, KTXD, or PKTX</b> board, enter a unique DH+ address in Octal (e.g. 0o77)</p> <p>If using both channels of a <b>PKTXD</b> card, you should give each channel a unique address.</p> <p>For a <b>5136-SD</b> board, enter the I/O Port address of the card (e.g. 0x250).</p> <p>For a <b>5136-SD-PCI</b> board, leave this field blank.</p>
Interrupt	<p>For a <b>KT, KTX or KTXD</b> - and polling is not being used - enter the IRQ number set on the card.</p> <p>For a <b>PKTX</b> card, leave this field blank</p> <p>For a <b>5136-SD</b> card, leave this field blank.</p> <p>For a <b>5136-SD-PCI</b> card, enter zero.</p>
Special Options	<p>For <b>KT, KTX, KTXD, or PKTX ISA</b> boards the following special options are available:</p> <ul style="list-style-type: none"> <li>-bx specifies a board number. Use this only if more than one card is installed. The default is x=1.</li> <li>Sn to select the speed on the Data Highway Plus network; n represents the speed required and is one of the following numbers:</li> <li>n = 1 for 115 kbaud</li> <li>n = 2 for 230 kbaud</li> </ul> <p>The speed is set, by default, to 57 kbaud.</p> <p>For <b>KT, KTX, KTXD, or PKTX PCI</b> boards the following special option is available:</p> <ul style="list-style-type: none"> <li>Sn to select the speed on the Data Highway Plus network. n represents the speed required and is one of the following numbers:</li> <li>n = 1 for 115 kbaud</li> <li>n = 2 for 230 kbaud</li> </ul> <p>The speed is set, by default, to 57 kbaud.</p> <p>For the <b>5136-SD</b> board the following special options are available:</p> <ul style="list-style-type: none"> <li>-PN to set the Data Highway/Data Highway Plus address of the card to N. By default, N is a decimal</li> </ul>

Field	Value
	<p>value. If you need to specify the address as an octal value then N will take the format OoXX where XX is the octal address.</p> <p>-Sn to select the speed on the Data Highway Plus network. n represents the speed required and is one of the following numbers:</p> <ul style="list-style-type: none"> <li>n = 1 for 115 kbaud</li> <li>n = 2 for 230 kbaud</li> </ul> <p>The speed is set, by default, to 57 kbaud.</p> <p>-T to use 5136-SD-PCI interface card instead of ISA interface card.</p> <p>For the <b>5136-SD-PCI</b> board the following special options are available:</p> <ul style="list-style-type: none"> <li>-pN to set the Data Highway/Data Highway Plus address of the card to N. N is, by default, in decimal format. If you wish to specify an octal value, use the format OoXX where XX is the octal address.</li> <li>-Sn to set the speed of the Data Highway Plus network, where n represents the speed, and is one of the following numbers.</li> </ul> <ul style="list-style-type: none"> <li>n = 1 for 115 kbaud</li> <li>n = 2 for 230 kbaud</li> </ul> <p>The speed is set, by default, to 57 kbaud.</p>

## Ports

Field	Value
Port Number	A unique number for each port, starting from 1.
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank.
Special Options	Leave this field blank.

## I/O Devices

Field	Value
I/O Device Address	Specify the PLC address as:

Field	Value
	<p><b>xKT:C[options],u</b></p> <p>where:</p> <p>x = the board number. If the number is not specified in the ports for then use 1.</p> <p>C = the channel number, for example, KT card = 0, KTX card = 0 or 1.</p> <p>u = the station number of the remote PLC in octal. Station numbers are between 0 and 77 (octal) for DH+ channels, and between 1 and 376 (octal) for DH channels.</p> <p>[options] If required, you can use the <a href="#">Offlink Address</a> options to allow communication with a station (PLC) that is not on the local link.</p> <p>For example: 1KT:0,1 specifies Port ID 1KT for access to PLC Station 1 .</p>
I/O Device Protocol	Enter ABTCP500.

### Allen-Bradley SLC-500 via Data Highway - Data Types

**Note:** The Allen Bradley SLC500 has many data types. The protocol makes a separate request for each different data type, and each request has a finite overhead. It is more efficient to read fewer data types, so you should move the I/O and other data into a few main blocks to be read by Plant SCADA.

Data Types	Address Format	Plant SCADA Data Type
Status words	S:w/b	DIGITAL
Timer	Tf:w.t	DIGITAL
Timer	Tf:w/b	DIGITAL
Counter	Cf:w.c	DIGITAL
Counter	Cf:w/b	DIGITAL
Control	Rf:w.r	DIGITAL
Control	Rf:w/b	DIGITAL
Binary	Bf:w/b	DIGITAL
Integer	Nf:w/b	DIGITAL
Output image	O:O.o	INT
Input image	I:O.o	INT

Data Types	Address Format	Plant SCADA Data Type
Status words	<b>S:w</b>	BCD / INT / LONG / LONGBCD/ REAL
Timer	<b>Tf:w.T</b>	BCD / INT / LONG / LONGBCD/ REAL
Counter	<b>Cf:w.T</b>	BCD / INT / LONG / LONGBCD/ REAL
Control	<b>Rf:w.C</b>	BCD / INT / LONG / LONGBCD/ REAL
Binary	<b>Bf:w</b>	BCD / INT / LONG / LONGBCD/ REAL
Integer	<b>Nf:w</b>	BCD / INT / LONG / LONGBCD/ REAL

Where:

<b>f</b>	File Number 0 to 999 decimal
<b>:w</b>	Element number 0 to 999 decimal
<b>:O</b>	Element number 0 to 277 octal
<b>/b</b>	Bit Number 0 to 15 decimal
<b>/o</b>	Bit Number 0 to 17 octal
<b>.T</b>	Timer/Counter Sub Element: PRE - Preset Value ACC - Accumulated Value
<b>.t</b>	Timer bits: EN - Enable TT - Timing DN - Done
<b>.C</b>	Control Sub Element: LEN - Length POS – Position
<b>.c</b>	Counter bits: CU - Count Up CD - Count Down DN - Done OV - Overflow

	UN - Underflow
.r	Control bits: EN - Enable DN - Done ER - Error UL - Unload IN - Inhibit comparisons FD - Found

Allen-Bradley SLC500 Series PLCs support remapping reads only.

## Offlink Addressing

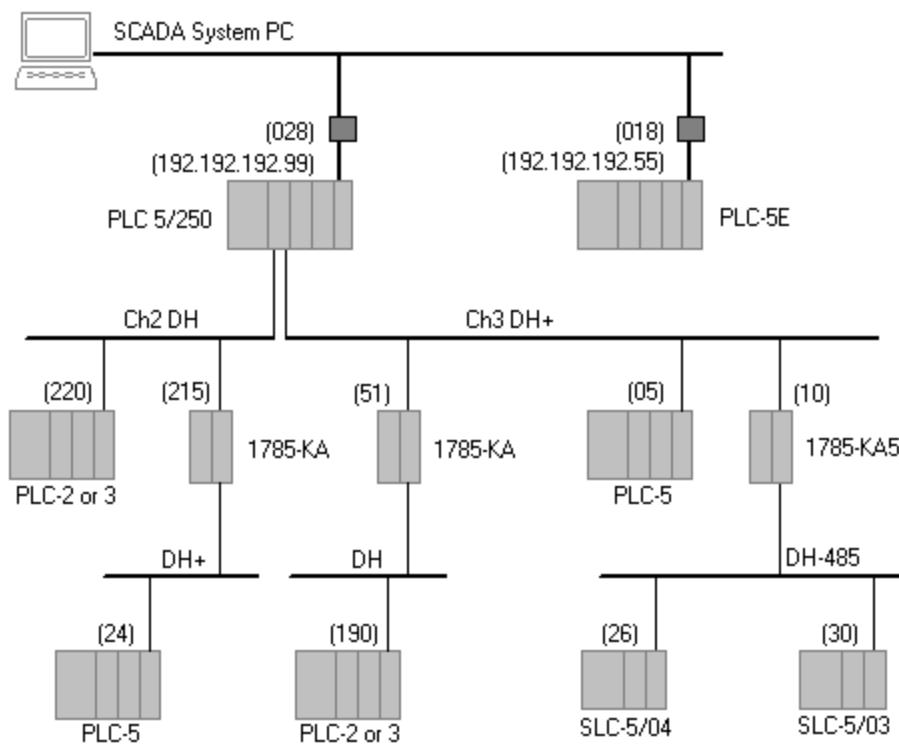
Offlink addressing allows communication with a station (PLC) that is not on the local link and is required for PLCs that cannot have support and Ethernet interface (like SLCs). This is done via 1785-KA, 1785-KA5 or PI routing. The 1785-KA is a communications adaptor card for providing exchange between DH and DH+. The format for offlink addressing is:

**PMM:C[options],u**

Where:

- **P** = the router module pushwheel number (0 to 4) or board number.
- **MM** = the module type (RM, KA or KT)
- **C** = the channel number (0, 2 or 3)
- **[options]** is one or more of the following:
  - **/B:b** where b is the bridge address (1 to 376 octal).
  - **/L:l** where l is the link id (0 to 65535 decimal)
  - **/G:g** where g is the gateway to the final DH-485 link (1 to 376 octal)
  - **/KA** When communicating through a 1785-KA from DH+ to DH
- **u** = the station number of the remote PLC in octal. Station numbers are between 0 and 77 (octal) for DH+ channels, and between 1 and 376 (octal) for DH channels.

Consider the following diagram and examples:



In the picture above station addresses are noted (in parentheses) next to each unit. Each unit is addressed as follows:

Station Number	Address
028	IP address: 192.192.192.99
018	IP address: 192.192.192.55
220	ORM:2,220
05	ORM:3,05
24	ORM:2,0215
190	ORM:3/B:51/KA,190
26	ORM:3/B:10/L:0,26
30	ORM:3/G:10/L1,30

Be aware that the SLCs may be addressed in either Single Hop (link is set to 0) or Gateway (where the incoming DH+ link ID is used) modes.

For detailed information on off-link addressing refer to the Allen Bradley Interchange Software Reference Manual.

## ABTCP Driver Parameters

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

[ABTCP] Parameter	Allowable Values	Default Value	Description
Block	10 - 256	236	A value (bytes) used by the I/O Server to determine if two or more packets can be blocked into one data request before being sent to the I/O Device. For example, if you set the value to 10, and the I/O Server receives two simultaneous data requests - one for byte 3, and another for byte 8 - the two requests will be blocked into a single physical data request packet. This single request packet is then sent to the I/O Device, saving on bandwidth and processing.
Delay	0 to 300	0	The period (in milliseconds) to wait between receiving a response and sending the next command.
MaxBits	0 to 2048	1888	The maximum read size in one request. Decrease the value for non-standard I/O Devices that do not support large reads. MaxBits should be a multiple of 8. If this value is modified from the default, then the protdir.dbf file in the Plant

			SCADA bin directory needs to be manually edited so that the MAX_LENGTH entry for your I/O Device is changed to match. For example, if MaxBits is set to 1920 bits for the ABTCP5 protocol then protdir.dbf should be modified as follows: ABTCP5 AB5 944 1920 0x37f
MaxPending	1 - 32	8	The maximum number of pending commands that the driver holds ready for immediate execution.
PollTime	0 to 300	0	The interrupt or polling service time (in milliseconds). Setting the polling time to 0 puts the driver in interrupt mode. <b>Note:</b> If you are using the PKTX or the 5136-SDI-PCI board, this parameter should be set to 10.
Retry	0 - 8	1	The number of times to retry a command after a timeout.
Status	<p>The value for this parameter is formatted as follows:</p> <p>&lt;RawType&gt;, &lt;BitWidth&gt;, &lt;UnitType&gt;, &lt;UnitAddress&gt;, &lt;UnitCount&gt;</p> <p>where...</p> <p><b>RawType =</b> 0 for Digital 1 for Int 4 for Long 8 for Byte</p> <p><b>BitWidth =</b></p>	<p>No default.</p> <p>For example, to specify the time/date seconds counter <b>S:23</b> as the status variable in a PLC-5/40E, use the following:</p> <p>[ABTCP] Status=1,16,0xd0002,23,1</p> <p>Or, to use the processor status run mode bit <b>S:1/1</b> for a PLC-5/10, use the following:</p> <p>[ABTCP] Status=0,1,0xd0002,17,16</p>	<p>Defines the register that indicates the status of the PLC (for Hot Standby). This register is checked every [ABTCP]WatchTime period.</p> <p>Usually, if the PLC is put into program mode, communications continue as normal, so the operator is unable to tell that control is suspended. However, when this parameter is used, can be made to display #COM to</p>

	<p>1 for Digital 8 for Byte 16 for Int 32 for Long</p> <p><b>UnitType =</b> Protocol specific. See the variable specification .DBF files for the particular PLC.</p> <p><b>UnitAddress =</b> The item number or bit number.</p> <p><b>UnitCount =</b> 1 for Analog 16 for Digital</p>		<p>signify that the PLC is offline (and the driver will return 0x20). If there is a standby unit configured, then it will automatically swap to it.</p> <p>For this to work correctly, you may need to manually configure the status register in your PLC. The register needs to be configured as follows, depending on whether it is a digital or analog variable:</p> <p>Digital - The variable needs to be always on, otherwise the unit is put offline.</p> <p>Analog - The variable needs to be changed for each read. If the variable has not changed since the last read, the unit is put offline.</p>
StringReverse	0 to 1	0	Set to 1 if the bytes in each word of a string needs to be swapped.
Timeout	0 to 32000	1000	Specifies how many milliseconds to wait for a response before displaying an error message.
WatchTime	0 to 128	30	The frequency (in seconds) that the driver uses to check the communications link to the I/O device.

## ABTCP Driver Specific Errors

When a hardware error is detected, Plant SCADA generates an alarm, and displays the alarm on the hardware alarm page (in the alarm description of the hardware alarm). To see the error number, make sure you have Alarm Category 255 defined with a display format that includes {ErrDesc,15} {ErrPage,10}.

There are a number of generic errors that are common to all protocols. In some cases, only the generic error is available, though often both the generic error and a specific error are given.

The following errors, listed in (hexadecimal) sequence, are specific to this protocol. Plant SCADA displays the error number and description for common protocol-specific errors. Uncommon errors are not contained in the error database, in which case will only display the error number.

You may require additional information to enable you to rectify an error. This information should be detailed in the documentation that accompanied the I/O Device (or network). If, after reviewing all documentation, you cannot rectify an error, contact Technical Support.

**0x10100PLC out of buffers**

Check the PLC configuration.

**0x10200 PLC is not online**

Check that the PLC is online.

**0x10300 Duplicated token holder detected**

If this error occurs by itself and is persistent, contact Technical Support for this product.

**0x10400 Local port is disconnected**

If this error occurs by itself and is persistent, contact Technical Support for this product.

**0x11000 Illegal command or format**

If this error occurs by itself and is persistent, contact Technical Support for this product.

**0x12000 Host will not communicate**

Check the project configuration and PLC configuration.

**0x13000 PLC hardware fault**

Address this problem at the PLC.

**0x14000 Host hardware fault**

Your hardware may not be functional. If this is persistent try using duplicate hardware.

**0x15000 Illegal data table address**

If this error occurs by itself and is persistent, contact Technical Support for this product.

**0x16000 PLC is write protected**

Change the configuration of your PLC (if desired).

**0x17000 Processor is in program mode**

Switch your processor to RUN mode (if desired).

**0x18000 Compatibility file is missing**

If this error occurs by itself and is persistent, contact Technical Support for this product.

**0x19000 PLC cannot buffer command**

If this error is persistent, check the PLC configuration and refer to your PLC documentation.

**0x1B000 PLC problem due to download**

If this error is persistent, check the PLC configuration and refer to your PLC documentation.

**0x1F001Error converting block address**

If this error is persistent, check the PLC configuration and refer to your PLC documentation.

**0x1F002 Insufficient levels of address**

If this error occurs by itself and is persistent, contact Technical Support for this product.

**0x1F003 Too many levels in address**

If this error occurs by itself and is persistent, contact Technical Support for this product.

**0x1F004 Symbol not found**

If this error occurs by itself and is persistent, contact Technical Support for this product.

**0x1F005 Symbol is of improper format**

Check the project configuration and PLC configuration.

**0x1F006 Invalid address**

The data requested by is not configured in the PLC. Check the project configuration and PLC configuration.

**0x1F007 File is of wrong size**

The data requested by is out of range or is not defined in the I/O Device. Check the project configuration and PLC configuration.

**0x1F008 Changed situation since start**

If this error is persistent, check the PLC configuration and refer to your PLC documentation.

**0x1F009 File is too large**

The data requested by is out of range or is not defined in the I/O Device. Check the project configuration and PLC configuration.

**0x1F00A Word and count too large**

The data requested by is out of range. Check the project configuration and PLC configuration.

**0x1F00B Access denied**

The PLC is denying access. If this error is persistent, check the PLC configuration and refer to your PLC documentation.

**0x1F00C Resource unavailable**

Check the project configuration and PLC configuration. If this error occurs by itself and is persistent, contact Technical Support for this product.

**0x1F00D File of wrong size**

The data requested by is out of range or is not defined in the I/O Device. Check the project configuration and PLC configuration.

**0x1F00E Shutdown cannot be executed**

If this error is persistent, check the PLC configuration and refer to your PLC documentation.

**0x1F00F File is too large**

The data requested by is not configured in the PLC. Check the project configuration and PLC configuration.

**0x1F010 Histogram overflow**

If this error is persistent, check the PLC configuration and refer to your PLC documentation.

**0x1F011Illegal data type**

The data requested by is not configured in the PLC. Check the project configuration and PLC configuration.

**0x1F012 Bad parameter**

Check the project configuration and PLC configuration.

**0x1F013 Data table deleted**

If this error occurs by itself and is persistent, contact Technical Support for this product.

**0x1FOFF \***

EXT STS error code.

**0x1FF00\***

Local STS error code.

**0x1F000 \***

Remote STS error code.

## BACNET Driver

The Plant SCADA BACNET Driver supports communication with the BACnet Data Communication Protocol, a widely accepted standard for building automation and control systems. This allows Plant SCADA to integrate with building management systems for heating and air conditioning, ventilation, lighting, security and fire detection.

The Plant SCADA BACNET Driver is designed to communicate with BACnet networks and devices using the BACnet/IP option.

### See Also

[About the BACnet Protocol](#)

[Communicating with BACnet](#)

[Configuring Your Project](#)

[BACNET Supported Object Types](#)

[BACNET Parameters](#)

[Logging](#)

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

**⚠ DANGER**

**DANGER** indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.

**⚠ WARNING**

**WARNING** indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.

**⚠ CAUTION**

**CAUTION** indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.

**NOTICE**

NOTICE used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.

**Please Note**

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

**Before You Begin**

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ WARNING****LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

## About the BACnet Protocol

The BACnet protocol is accepted as an ISO standard for building automation and control systems. The protocol defines a number of data link /physical layers, including ARCNET, Ethernet, BACnet/IP, Point-To-Point over RS-232, Master-Slave/Token-Passing over RS-485, and LonTalk. These networks may then be further interconnected by BACnet routers.

---

**Note:** The BACNET driver is designed to communicate to BACnet networks and devices using only BACnet/IP transport option.

---

## BACnet objects

Every BACnet device is represented as a collection of objects with a number of properties. The objects include:

- Analog Input
- Analog Output
- Analog Value
- Binary Input
- Binary Output
- Binary Value
- Multi-State Input
- Multi-State Output
- Calendar
- Event-Enrollment
- File
- Notification-Class
- Group
- Loop
- Program
- Schedule
- Command
- Device

## See Also

[Communicating with BACnet](#)

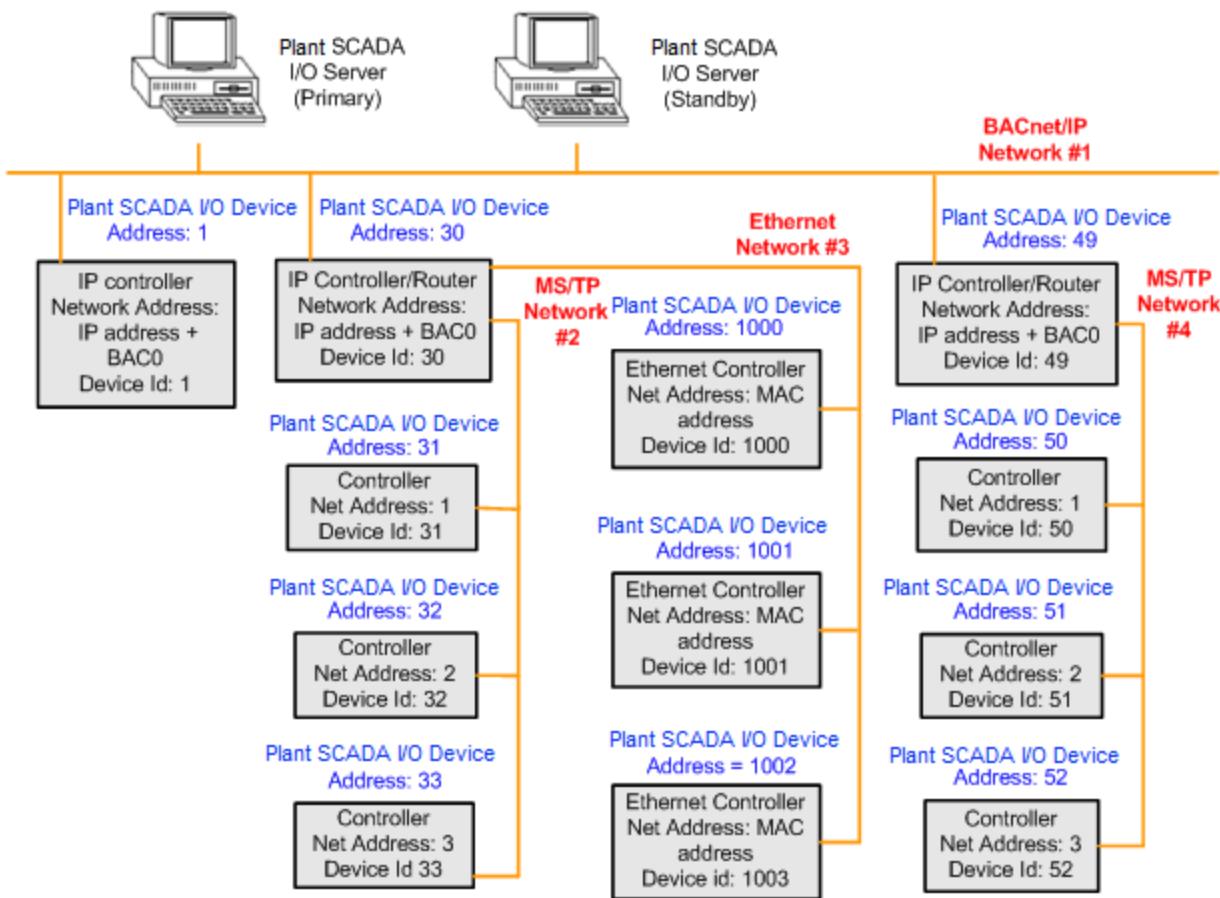
[BACNET Supported Object Types](#)

## Communicating with BACnet

The BACNET driver is designed to communicate with BACnet networks and devices using the BACnet/IP transport option. The following diagram illustrates how Plant SCADA communicates to different types of BACnet networks.

The diagram illustrates the following:

- Each network is uniquely identified by a number.
- Each device has an address uniquely identifying it on a network.
- Each device has a unique device ID in the whole BACnet network.
- The device ID is used to uniquely identify each device in Plant SCADA.
- Each BACnet device corresponds to an I/O Device in Plant SCADA.
- Plant SCADA communicates to non-BACnet/IP networks via BACnet routers.
- Plant SCADA I/O device redundancy is provided.



**Note:** The BACNET driver only needs one port configured in Plant SCADA. This is because the driver will only initialize one instance of Cimetrics BACstac which it will use to communicate with all devices. Therefore, you only need to configure one board and one port in your Plant SCADA project.

## See Also

- [Mapping Tags to BACnet Objects](#)
- [BACnet Data Type Representation](#)
- [Reading BACnet Properties](#)
- [Writing to BACnet Properties](#)
- [Writing to Specific Calendar and Schedule Properties](#)

## Mapping Tags to BACnet Objects

The BACNET driver represents the properties of BACnet objects as tags. A corresponding Plant SCADA data type is chosen for each of the supported properties (see [BACnet Data Type Representation](#)).

BACnet object properties are addressed using the following format:

*<object type>.<object instance>.<property type>*

Where:

- *<object type>* is one of the mnemonics (provided in the [BACNET Supported Object Types](#) table).
- *<object instance>* is the numerical object instance. Object instances may range from 0 to 4194302.
- *<property type>* is the property mnemonics indicated for the particular property type in the object type tables.

## BACnet Data Type Representation

The Plant SCADA data types that correspond to BACnet property data types are fairly straightforward for most BACnet data types. Data types which require further explanation are described below.

## BACnet Date/Time Data Types

BACnet date/time properties are represented as a string in the following formats:

- **BACnetDate:** dd/mm/yyyy  
Examples: "01/09/2008", "01/09/\*\*\*\*" if year is not specified, "\*/\*\*/\*/\*/\*" if none of the date elements are specified.
- **BACnetTime:** hh:mm:ss.mmm.  
Examples: "05:20:30:400", "05:\*\*:30:400" if hour is not specified, "\*\*:\*\*:\*\*:\*\*" if none of the time elements are specified.
- **BACnetDateTime:** dd/mm/yyyy hh:mm:ss.mmm.  
Examples: "01/09/2008 05:20:30:400", "01/09/\*\*\*\* 05:\*\*:30:400" if year and minutes are not specified.
- **BACnetWeekNDay:** dd/ww/mm.  
Examples: "04/02/03", "04/02/\*\*" if month is not specified.

## BACnet Any Data Type

For BACnet properties that can be any primitive data type, the driver supports the following BACnet data types:

- Boolean
- Real
- Unsigned
- Enumerated
- CharacterString
- BACnetDate
- BACnetTime
- BACnetDateTime.

Corresponding Plant SCADA data types can be DIGITAL, LONG, REAL or STRING.

## BACnetPriorityArray

BACnetPriorityArray is represented as a string.

Example: "NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, 78, 89, NULL, NULL, NULL, NULL, NULL".

## Reading BACnet Properties

The BACNET driver can use change-of-value (COV) subscriptions or polling to read values from BACnet properties. In both cases, this behavior can be modified by adjusting parameters in the `citect.ini` file.

## Polling Properties

The driver will only poll BACnet properties data for the tags which have been subscribed by the I/O Server.

The polling rate will be defined by the I/O Server subscription update rate and the value of Cache Time parameter in the I/O Device configuration properties.

The I/O Server subscription update rate will depend on whether the tag is displayed on a page, has an alarm or trend tag associated with it or subscribed in Cicode. For example, the default update rate for the tag displayed on a page is 250 msec, for an alarm tag is 500 msec and for a trend tag it will depend on the trend tag configuration settings.

Also it is possible to set the minimal allowable polling rate using the Cache Time parameter in the I/O Device configuration properties. If that parameter is not set, the default value is 300 msec.

So if the I/O Server subscription update rate is less than the Cache Time value, the driver will use the Cache Time value to define how often the BACnet property will be polled.

If Cache Time value is set to 0, it will be ignored and the polling rate will be defined based on the I/O Server subscription rate.

You need to set the Cache Time parameter according to your system requirements.

## COV Subscriptions

If a BACnet device supports change-of-value reporting, the driver will use the `SubscribeCOV` service to subscribe to value changes of `PresentValue` and `SystemFlags` BACnet properties.

The driver will only send COV subscriptions for the tags which have been subscribed by the I/O Server.

The driver subscribes to COV notifications for the period defined by the `COVLifeTime` parameter.

In the following situations the driver will use polling instead of COV subscriptions:

- A device does not support COV reporting.
- `COVLifeTime` is set to 0.
- The COV subscription failed.

## See Also

[Writing to BACnet Properties](#)

## Writing to BACnet Properties

The driver uses `WriteProperty` and `WritePropertyMultiple` services for writing to BACnet properties.

The priority level when writing to commandable properties (Present Value for Analog Output, Analog Value,

Binary Output, Binary Value, Multi-state Output, Multi-state Value objects) is controlled by the WritePriority setting in the Citect.ini file and the Device object WritePriority custom property.

At start up the write priority level will be set to the value specified in the WritePriority ini setting or set to 16 (lowest priority) if not specified.

If you require a write to be performed with a specified priority, you will need to change the write priority via the value of the tag with the WritePriority address (before performing a write to a commandable property). After the write has completed, the write priority will be set back to the priority specified in the ini setting, or to the default if an ini setting is not specified.

The Device object WriteNULL custom property is used to write NULL property values. When the value of the tag with that address is set to 1, writing property value 0 will be substituted by a NULL value.

It is also possible to change the value which specifies writing a NULL from 0 to any REAL value by using the WriteNULLValue custom device property. In that case when the WriteNULL is set to 1, you can use WriteNULLValue to change the write NULL value. The value of the WriteNULLValue property will be reset back to 0 after the write is completed.

---

**Note:** The Raw and Eng Scale values in the tag configuration should be the same for the WriteNULLValue property to work correctly.

Writing to BACnet properties which have CharacterString data type currently supports only ANSI X3.4 character set.

## See Also

[Reading BACnet Properties](#)

## Writing to Specific Calendar and Schedule Properties

Tag addresses are available that allow you to write to the BACnet properties identified in the following table.

Tag address	Associated BACnet property
DateListEx	Calendar.DateList
WeeklyScheduleExInt[.Monday...Sunday] WeeklyScheduleExDig[.Monday...Sunday] WeeklyScehduleExDbl[.Monday...Sunday] WeeklyScehduleExEnum[.Monday...Sunday] WeeklyScehduleExUInt[.Monday...Sunday] WeeklyScehduleExReal[.Monday...Sunday] See <a href="#">Schedule Objects</a> .	Schedule.WeeklySchedule
ExceptionScheduleExInt[.1...31] ExceptionScheduleExDig[.1...31] ExceptionScheduleExDbl[.1...31] ExceptionScheduleExEnum[.1...31] ExceptionScheduleExUInt[.1...31]	Schedule.ExceptionSchedule

Tag address	Associated BACnet property
ExceptionScheduleExReal[.1...31] See <a href="#">Schedule Objects</a> .	

When configuring these tags, you should consider the following:

- These tags are STRING Plant SCADA data type.
- Using wildcards (\*) when writing to the schedule or calendar is not supported, for example, {{05/\*\*/2011}}. However, any unspecified entries existing in the device will be read as wildcards (\*). For example, the driver will read the Calendar entry {{05/\*\*/2011}} from the device as it is. This is interpreted as the 5th day of any month of the year 2011.
- Currently the only supported date and time format is Australian. Date: dd/mm/yyyy. Time: hh:mm:ss.mmm.
- A write will be unsuccessful if an incorrect string format is used.

The driver is only capable of writing a string no longer than 252 characters because of Plant SCADA limitations. Any attempt to write a string longer than 252 characters will be unsuccessful. The following provides examples of the limitations you need to be aware of:

- **Schedule.WeeklySchedule[0..1].BACnetDailySchedule**, **Schedule.ExceptionSchedule[0..N].BACnetSpecialEvents** and **Calendar.DateList** are BACnet lists with an arbitrary number of elements. A list with a large number of elements may not fit into the string length limitation. If the string requires more than 252 characters, the corresponding tag value will be set to empty string with BAD quality.
- For a **Calendar.DateList** property that contains a list of BACnetDate elements ("{{DT(04/11/2015)} {DT(05/11/2015)}{DT(06/11/2015)}}", the maximum number of elements in the list is 15.
- For a **Calendar.DateList** property that contains a list of BACnetDateRange elements ("{{DR(31/12/2000)(25/10/2020)}{DR(31/12/2000)(25/10/2020)}}"), the maximum number of elements in the list is 8.
- For **Schedule.WeeklySchedule** and **Schedule.ExceptionSchedule** properties, the maximum number of elements depends on the number of time-value pairs in BACnetDailySchedule or BACnetSpecialEvents lists.

For example, a **Schedule.WeeklySchedule.Monday** property that has only digital values can contain 16 time-value pairs. This number will decrease if a weekly or exception schedule contains integer or real values. Also, it will be less for an exception schedules as it will contain date/calendar references.

---

**Note:** Not every BACnet device uses the same object indexing rules. For example, with the Alerton BCM-ETH, both the schedule object index and the calendar object index start independently at 1, whereas Andover™ TAC devices use common indexing for every object type, starting at 7001.

When writing to one of these tags, you need to compose a string in a format described below. The driver will parse the string and update the corresponding property in the device.

## Calendar.DateListEx property

This tag represents a sequence of BACnetCalendarEntry.

### String format for BACnetCalendarEntry:

{DT(dd/mm/yyyy)}

or

{DR(dd/mm/yyyy)(dd/mm/yyyy)}

or

{WD(DayOfWeek/WeekOfMonth/Month)}

where:

- DT represents a 'Date' type BACnetCalendarEntry
- DR represents a 'DateRange' type BACnetCalendarEntry
- WD represents a 'WeekNDay' type BACnetCalendarEntry
- *DayOfWeek* is a numerical representation of a day: 1 for Monday, 2 for Tuesday and so on
- *WeekOfMonth* is 1 for days 1-7; 2 for days 8-14; 3 for days 15-21; 4 for days 22-28; 5 for days 29-31
- *Month* is a numerical representation of a month: 1 for January, 2 for February and so on

For example, when defining a string you could use the following:

1. CAL\_2 = "{DT(04/11/2009)}"
2. CAL\_2 = "{DT(04/11/2009)}{ DT(24/12/2020)}"
3. CAL\_2 = "{DR(31/12/2000)(25/10/2020)}"
4. CAL\_2 = "{WD(1/2/11)}{WD(7/3/10)}"
5. CAL\_2 = "{DT(04/11/2009)}{DR(03/05/2011)(15/08/2017)}{WD(4/5/3)}"

Be aware when specifying a year value (yyyy), that the driver will support 1900 to 2154. However, you will need to consider that the range supported by a device may be more restrictive.

## Schedule.WeeklySchedule Tags

These tags represent a single array of BACnet Schedule.WeeklySchedule property. Each array has seven elements to represent days of the week. You will need to create a separate tag for each day in a week. The driver reads and writes individual array elements of BACnet Schedule.WeeklySchedule property.

There are separate tag addresses for different value data types. The driver support BACnet Boolean, Signed Integer and Double data types.

**The tag address format is:**

SCH.<objectInstance>.WeeklyScheduleExDig[.Monday...Sunday]  
SCH.<objectInstance>.WeeklyScheduleExInt[.Monday...Sunday]  
SCH.<objectInstance>.WeeklyScheduleExDbl[.Monday...Sunday]  
SCH.<objectInstance>.WeeklyScheduleExEnumI[.Monday...Sunday]  
SCH.<objectInstance>.WeeklyScheduleExUInt[.Monday...Sunday]  
SCH.<objectInstance>.WeeklyScheduleExReal[.Monday...Sunday]

For example:

- The following tag represents a daily schedule for Monday, supporting an Integer data type:  
Schedule.3.WeeklyScheduleExInt.Monday
- The following tag represents a daily schedule for Saturday, supporting a Double data type:  
Schedule.1.WeeklyScheduleExDbl.Saturday

- The following tag represents a number of elements in a WeeklySchedule array:  
`Schedule.3.WeeklyScheduleExInt`
- The following tag represents a number of elements in a WeeklySchedule array:  
`Schedule.3.WeeklyScheduleExReal`
- The following tag represents a daily schedule for Monday, supporting an unsigned Integer data type:  
`Schedule.3.WeeklyScheduleExUInt.Monday`
- The following tag represents a daily schedule for Monday, supporting an enumerated data type:  
`Schedule.3.WeeklyScheduleExEnum.Saturday`

The value of these tags is represented by a string in a format.

**The string format is:**

`(time;value)(time;value)(time;value)`

where:

- *time* is in BACnetTime in a format "hh:mm:ss.mmm" or "HH:mm:ss.mmm"
- *value* is a Boolean, Signed Integer or Double, depending on a tag address.

**Example**

`(07:00:00.00;18.5)(09:15:00.00;20.5)`

**Schedule.ExceptionSchedule Tags**

These tags represent a single array element of BACnet Schedule.ExceptionSchedule property. The user will need to create a separate tag for each special event. The driver reads and writes individual array elements of BACnet Schedule.ExceptionSchedule property.

There are separate tag addresses for different value data types. The driver support BACnet Boolean, Signed Integer and Double data types.

**The tag address format is:**

`SCH.<objectInstance>.ExceptionScheduleExDig[.1...31]`  
`SCH.<objectInstance>.ExceptionScheduleExInt[.1...31]`  
`SCH.<objectInstance>.ExceptionScheduleExDbl[.1...31]`  
`SCH.<objectInstance>.ExceptionScheduleExEnum[.1...31]`  
`SCH.<objectInstance>.ExceptionScheduleExUInt[.1...31]`  
`SCH.<objectInstance>.ExceptionScheduleExReal[.1...31]`

For example:

- The following tag represents a special event number 1, supporting an Integer data type:  
`Schedule.3. ExceptionScheduleExInt.1`
- The following tag represents a special event number 4, supporting a Double data type:  
`Schedule.1. ExceptionScheduleExDbl.4`
- The following tag represents a number of elements in an ExceptionSchedule array:  
`Schedule.1. ExceptionScheduleExDbl`
- The following tag represents a special event number 3, supporting a enumerated data type:  
`Schedule.3. ExceptionScheduleExEnum.3`

Schedule.1. ExceptionScheduleExEnum.3

- The following tag represents a special event number 4, supporting unsigned Integer data type:  
Schedule.2.ExceptionScheduleExUInt.4
- The following tag represents a special event number 4, supporting real data type:  
Schedule.2.ExceptionScheduleExReal.4

The value of these tags is represented by a string in a format.

**Note:**

The driver can only read/write the first 31 special events for a configured schedule object, regardless of how many of them exist in the device. For example, for a device supporting a signed integer data type, the tag address for the first special event should be "SCH.<objectInstance>.ExceptionScheduleExInt.1". The tag address of 31st special event should be "SCH.<objectInstance>.ExceptionScheduleExInt.31".

Unlike weekly schedule events, deleting special events or exception schedule events is not supported. However, special events can be overwritten.

The driver cannot create exception schedule or special events in the device. It can only read and write to such events if they have already been created. The driver will return bad quality for non-existent exception schedule or special events, however, the tag address "SCH.<objectInstance>.ExceptionScheduleExInt" will correctly read the total number of special events configured for the schedule object in the device.

## Schedule. EffectivePeriodEx

This tag specifies the range of dates within which the Schedule object is active. This is represented as BACnetDateRange (string).

The tag addressing format is:

SCH.<objectInstance>.EffectivePeriodEx

For example the date range format is:

DR (11/01/2010)(23/06/27)

**String format for a calendar reference:**

CL(*CalendarObjectId*),(*time;value*)(*time;value*),Event Priority

where:

- *CalendarObjectId* represents a configured calendar object ID
- *time* is in BACnetTime in a format "hh:mm:ss.mmm" or "HH:mm:ss.mmm"
- *value* is a Boolean, Signed Integer or Double, depending on a tag address.

For example:

CL(3),(07:07:07.25;1),16

## String format for BACnetCalendarEntry:

DT(dd/mm/yyyy),(*time;value*)(*time;value*),Event Priority

or

DR(dd/mm/yyyy)(dd/mm/yyyy),(time;value)(time;value),Event Priority

or

WD(DayOfWeek/WeekOfMonth/Month),(time;value)(time;value),Event Priority

where:

- DT represents a 'Date' type BACnetCalendarEntry
- DR represents a 'DateRange' type BACnetCalendarEntry
- WD represents a 'WeekNDay' type BACnetCalendarEntry
- *time* is in BACnetTime format (hh:mm:ss.mmm)
- *DayOfWeek* is a numerical representation of a day: 1 for Monday, 2 for Tuesday and so on
- *WeekOfMonth* is 1 for days 1-7; 2 for days 8-14; 3 for days 15-21; 4 for days 22-28; 5 for days 29-31;
- *Month* is a numerical representation of a month: 1 for January, 2 for February and so on.

For example:

DT(05/11/2000),(07:00:00.00;18.5)(09:15:00.00;20.5),12

DR(11/01/2010)(23/06/27),(07:00:00.00;18.5)(09:15:00.00;20.5),7

## Configuring Your Project

This chapter contains the following topics:

- [BACnet Device Address](#)
- [Communication Settings](#)
- [BACstac Configuration](#).

### BACnet Device Address

The address to use for a BACnet device is the Device ID that uniquely identifies each BACnet device within a network. It can be in the range of 0 to 4194302.

See [Communicating with BACnet](#) for examples of device IDs within a BACnet system.

### See Also

[Communication Settings](#)

### Communication Settings

To establish communication with a device, configure the Boards, Ports and I/O Devices settings in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these will be automatically configured for you. However, if you need to manually configure these settings, use the information outlined below.

---

**Note:** The BACNET driver only needs one port configured in Plant SCADA. This is because the driver will only initialize one instance of Cimetrics BACstac which it will use to communicate with all devices (see [BACstac](#))

[Configuration](#)). Therefore, you only need to configure one board and one port in your Plant SCADA project.

## Boards

Field	Description
Board name	This field is user defined.
Board Type	BACNET
Address	0
I/O Port	N/A
Interrupt	N/A
Special Options	N/A
Comments	This field is user defined and not used by the driver.

## Ports

Field	Description
Port Name	This field is user defined.
Port Number	Any valid number.
Board Name	Refers to the board previously defined in the Boards settings.
Baud Rate	N/A
Data Bits	N/A
Stop Bits	N/A
Parity	N/A
Special Options	N/A
Comment	This field is user defined and not used by the driver.

## I/O Devices

Field	Description
Name	This field is user defined.
Number	I/O device number. It needs to be unique for each logical device, but identical for redundant devices.
Address	<p>You can use the unique device ID that identifies each BACnet device. It can be in the range of 0 to 4194302. If only the BACnet device ID is specified, the driver will use the BACnet "Who-Is/I-Am" discovery services to obtain the MAC address and the network number of the BACnet device.</p> <p>When a BACnet device is on a different IP subnet (and cannot be discovered using Who-Is/I-Am services), you can configure additional parameters in the I/O device address field. The following options are available:</p> <p><b>BACnet/IP device:</b></p> <ul style="list-style-type: none"> <li>-i — IP address, for example “-i 127.0.0.1”</li> <li>-u — UDP port number (the default BACnet UDP port 47808 is used by default)</li> <li>-n — BACnet network number.</li> </ul> <p><b>BACnet MSTP device:</b></p> <ul style="list-style-type: none"> <li>-s — station ID, for example “-s 5”</li> <li>-n — BACnet network number.</li> </ul> <p><b>BACnet Ethernet device:</b></p> <ul style="list-style-type: none"> <li>-e — MAC address of the BACnet Ethernet device, for example “-e 00-50-56-9F-20-66”</li> <li>-n — BACnet network number.</li> </ul> <p><b>Examples</b></p> <p>BACnet/IP device address:  &lt;Device ID&gt; -i &lt; device IP Address&gt;</p> <p>&lt;Device ID&gt; -i &lt;device IP Address&gt; -u &lt;device UDP port number&gt;</p> <p>BACnet MSTP device address:  &lt;Device ID&gt; -s &lt;MSTP device station id&gt; -n &lt; network number for MSTP device&gt;</p> <p>BACnet Ethernet device address:  &lt;Device ID&gt; -e &lt;MAC address&gt; -n &lt;network number for the Ethernet device &gt;</p> <p>Note that in order to connect to MSTP device on a different IP subnet, the BBMD IP address for the MSTP</p>

Field	Description
	network needs to be configured in the BACstac ports configuration dialog.
Protocol	BACNET
Port Name	Refers to the port previously defined in the Ports settings.
Comment	This field is user defined and not used by the driver.
Cache Time	Defines the minimum rate in milliseconds at which the driver will poll BACnet properties. The default value is 300 msec.

## BACstac Configuration

The BACNET driver uses Cimetrics™ BACstac protocol stack v7.4e to communicate with BACnet devices.

When you install BACstac with the driver for the first time, the BACnet/IP ports configuration dialog is displayed. If no configuration changes are required, you can just close the dialog by clicking OK.

After BACstac is installed, you can view these configuration settings by running the file "baccfg.bat" in the BACstac application folder, for example:

C:\Program Files (x86)\Cimetrics\BACstac v7.4e\baccfg.bat

## BACNET Supported Object Types

BACnet Object Type	Address Field Mnemonic
Analog Input Objects	AI
Analog Output Objects	AO
Analog Value Objects	AV
Binary Input Objects	BI
Binary Output Objects	BO
Binary Value Objects	BV
Calendar Objects	CAL
Device Objects	DEV
Life Safety Point Objects	LSP
Life Safety Zone Objects	LSZ

BACnet Object Type	Address Field Mnemonic
Loop Objects	LOOP
Multi-state Input Objects	MI
Multi-state Output Objects	MO
Multi-state Value Objects	MV
Notification Class Objects	NC
Positive Integer Value Objects	PIV
Schedule	SCH
Structured View Objects	SV
Timer Objects	TIM

## Analog Input Objects

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
ObjectIdentifier	BACnetObjectIdentifier	LONG	Read Only
ObjectName	CharacterString	STRING	Read Only
ObjectType	BACnetObjectType	LONG	Read Only
PresentValue	REAL	REAL	Read/Write
Description	CharacterString	STRING	Read/Write
StatusFlags	BACnetStatusFlags	STRING	Read Only
EventState	BACnetEventState	LONG	Read Only
OutOfService	BOOLEAN	DIGITAL	Read/Write
Units	BACnetEngineeringUnits	LONG	Read/Write
HighLimit	REAL	REAL	Read/Write
LowLimit	REAL	REAL	Read/Write
Deadband	REAL	REAL	Read/Write
TimeDelay	Unsigned	LONG	Read/Write

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
LimitEnable  BACnetLimitEnable represented as a long integer.  Bits are stored in the integer as follows:  Bit 0 – low limit enable Bit 1 – high limit enable	BACnetLimitEnable	LONG	Read/Write
EventEnable  BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	Read/Write
AckedTransitions  BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	Read Only
EventTimeStamps.ToOffNormal  EventTimeStamps.ToFault  EventTimeStamps.ToNormal  BACnetTimeStamp properties represented as a string in Australian time format	BACnetARRAY[3] of BACnetTimeStamp	STRING	Read Only
Reliability	BACnetReliability	LONG	Read/Write
UpdateInterval	Unsigned	LONG	Read/Write
MaxPresValue	REAL	REAL	Read/Write

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
MinPresValue	REAL	REAL	Read/Write
Resolution	REAL	REAL	Read/Write
COVIncrement	REAL	REAL	Read/Write
NotificationClass	Unsigned	LONG	Read/Write
NotifyType	BACnetNotifyType	LONG	Read/Write
DeviceType	CharacterString	STRING	Read/Write
ProfileName	CharacterString	STRING	Read Only

## Analog Output Objects

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
ObjectIdentifier	BACnetObjectIdentifier	LONG	Read Only
ObjectName	CharacterString	STRING	Read Only
ObjectType	BACnetObjectType	LONG	Read Only
PresentValue	REAL	REAL	Read/Write
Description	CharacterString	STRING	Read/Write
StatusFlags	BACnetStatusFlags	STRING	Read Only
EventState	BACnetEventState	LONG	Read Only
OutOfService	BOOLEAN	DIGITAL	Read/Write
Units	BACnetEngineeringUnits	LONG	Read/Write
RelinquishDefault	REAL	REAL	Read/Write
PriorityArray	BACnetPriorityArray	STRING	Read Only
HighLimit	REAL	REAL	Read/Write
LowLimit	REAL	REAL	Read/Write
Deadband	REAL	REAL	Read/Write
TimeDelay	Unsigned	LONG	Read/Write
LimitEnable	BACnetLimitEnable	LONG	Read/Write

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
BACnetLimitEnable represented as a long integer.  Bits are stored in the integer as follows: Bit 0 – low limit enable Bit 1 – high limit enable			
EventEnable BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	Read/Write
AckedTransitions BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	Read Only
EventTimeStamps.ToOffNormal EventTimeStamps.ToFault EventTimeStamps.ToNormal  BACnetTimeStamp properties represented as a string in Australian time format	BACnetARRAY[3] of BACnetTimeStamp	STRING	Read Only
Reliability	BACnetReliability	LONG	Read/Write
MaxPresValue	REAL	REAL	Read/Write
MinPresValue	REAL	REAL	Read/Write
Resolution	REAL	REAL	Read/Write

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
COVIncrement	REAL	REAL	Read/Write
NotificationClass	Unsigned	LONG	Read/Write
NotifyType	BACnetNotifyType	LONG	Read/Write
DeviceType	CharacterString	STRING	Read/Write
ProfileName	CharacterString	STRING	Read Only

### Analog Value Objects

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
ObjectIdentifier	BACnetObjectIdentifier	LONG	Read Only
ObjectName	CharacterString	STRING	Read Only
ObjectType	BACnetObjectType	LONG	Read Only
PresentValue	REAL	REAL	Read/Write
Description	CharacterString	STRING	Read/Write
StatusFlags	BACnetStatusFlags	STRING	Read Only
EventState	BACnetEventState	LONG	Read Only
OutOfService	BOOLEAN	DIGITAL	Read/Write
Units	BACnetEngineeringUnits	LONG	Read/Write
RelinquishDefault	REAL	REAL	Read/Write
PriorityArray	BACnetPriorityArray	STRING	Read Only
HighLimit	REAL	REAL	Read/Write
LowLimit	REAL	REAL	Read/Write
Deadband	REAL	REAL	Read/Write
TimeDelay	Unsigned	LONG	Read/Write
LimitEnable BACnetLimitEnable represented as a long integer.	BACnetLimitEnable	LONG	Read/Write

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
Bits are stored in the integer as follows: Bit 0 – low limit enable Bit 1 – high limit enable			
EventEnable BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows: Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	Read/Write
AckedTransitions BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows: Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	Read Only
EventTimeStamps.ToOffNormal EventTimeStamps.ToFault EventTimeStamps.ToNormal BACnetTimeStamp properties represented as a string in Australian time format	BACnetARRAY[3] of BACnetTimeStamp	STRING	Read Only
Reliability	BACnetReliability	LONG	Read/Write
COVIncrement	REAL	REAL	Read/Write
NotificationClass	Unsigned	LONG	Read/Write
NotifyType	BACnetNotifyType	LONG	Read/Write
ProfileName	CharacterString	STRING	Read Only

## Binary Input Objects

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
ObjectIdentifier	BACnetObjectIdentifier	LONG	Read Only
ObjectName	CharacterString	STRING	Read Only
ObjectType	BACnetObjectType	LONG	Read Only
PresentValue	BACnetBinaryPV	DIGITAL	Read/Write
Description	CharacterString	STRING	Read/Write
StatusFlags	BACnetStatusFlags	STRING	Read Only
EventState	BACnetEventState	LONG	Read Only
OutOfService	BOOLEAN	DIGITAL	Read/Write
Polarity	BACnetPolarity	DIGITAL	Read/Write
TimeDelay	Unsigned	LONG	Read/Write
EventEnable  BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	Read/Write
AckedTransitions  BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	Read Only
AlarmValue		DIGITAL	Read/Write
EventTimeStamps.ToOffNormal  EventTimeStamps.ToFault  EventTimeStamps.ToNormal  BACnetTimeStamp	BACnetARRAY[3] of BACnetTimeStamp	STRING	Read Only

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
properties represented as a string in Australian time format			
Reliability	BACnetReliability	LONG	Read/Write
ActiveText	CharacterString	STRING	Read/Write
InactiveText	CharacterString	STRING	Read/Write
ChangeOfStateCount	Unsigned	LONG	Read/Write
ElapsedActiveTime	Unsigned32	LONG	Read/Write
ChangeOfStateTime BACnetDateTime property represented as a string in Australian date/time format	BACnetDateTime	STRING	Read Only
TimeOfStateCountReset BACnetDateTime property represented as a string in Australian date/time format	BACnetDateTime	STRING	Read Only
TimeOfActiveTimeReset BACnetDateTime property represented as a string in Australian date/time format	BACnetDateTime	STRING	Read Only
NotificationClass	Unsigned	LONG	Read/Write
NotifyType	BACnetNotifyType	LONG	Read/Write
DeviceType	CharacterString	STRING	Read/Write
ProfileName	CharacterString	STRING	Read Only

## Binary Output Objects

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
ObjectIdentifier	BACnetObjectIdentifier	LONG	Read Only
ObjectName	CharacterString	STRING	Read Only

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
ObjectType	BACnetObjectType	LONG	Read Only
PresentValue	BACnetBinaryPV	DIGITAL	Read/Write
Description	CharacterString	STRING	Read/Write
StatusFlags	BACnetStatusFlags	STRING	Read Only
EventState	BACnetEventState	LONG	Read Only
OutOfService	BOOLEAN	DIGITAL	Read/Write
Polarity	BACnetPolarity	DIGITAL	Read/Write
RelinquishDefault	BACnetBinaryPV	DIGITAL	Read/Write
PriorityArray	BACnetPriorityArray	STRING	Read Only
TimeDelay	Unsigned	LONG	Read/Write
EventEnable  BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	Read/Write
AckedTransitions  BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	Read Only
FeedbackValue		DIGITAL	Read/Write
EventTimeStamps.ToOffNormal  EventTimeStamps.ToFault  EventTimeStamps.ToNormal  BACnetTimeStamp	BACnetARRAY[3] of BACnetTimeStamp	STRING	Read Only

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
properties represented as a string in Australian time format			
Reliability	BACnetReliablity	LONG	Read/Write
ActiveText	CharacterString	STRING	Read/Write
InactiveText	CharacterString	STRING	Read/Write
ChangeOfStateCount	Unsigned	LONG	Read/Write
ElapsedActiveTime	Unsigned32	LONG	Read/Write
ChangeOfStateTime BACnetDateTime property represented as a string in Australian date/time format	BACnetDateTime	STRING	Read Only
TimeOfStateCountReset BACnetDateTime property represented as a string in Australian date/time format	BACnetDateTime	STRING	Read Only
TimeOfActiveTimeReset BACnetDateTime property represented as a string in Australian date/time format	BACnetDateTime	STRING	Read Only
NotificationClass	Unsigned	LONG	Read/Write
NotifyType	BACnetNotifyType	LONG	Read/Write
DeviceType	CharacterString	STRING	Read/Write
MinimumOffTime	Unsigned32	LONG	Read/Write
MaximumOnTime	Unsigned32	LONG	Read/Write
ProfileName	CharacterString	STRING	Read Only

## Binary Value Objects

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
ObjectIdentifier	BACnetObjectIdentifier	LONG	Read Only
ObjectName	CharacterString	STRING	Read Only
ObjectType	BACnetObjectType	LONG	Read Only
PresentValue	BACnetBinaryPV	DIGITAL	Read/Write
Description	CharacterString	STRING	Read/Write
StatusFlags	BACnetStatusFlags	STRING	Read Only
EventState	BACnetEventState	LONG	Read Only
OutOfService	BOOLEAN	DIGITAL	Read/Write
RelinquishDefault	BACnetBinaryPV	DIGITAL	Read/Write
PriorityArray	BACnetPriorityArray	STRING	Read Only
TimeDelay	Unsigned	LONG	Read/Write
EventEnable  BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	Read/Write
AckedTransitions  BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	Read Only
AlarmValue		DIGITAL	Read/Write
EventTimeStamps.ToOffNormal  EventTimeStamps.ToFault  EventTimeStamps.ToNorm	BACnetARRAY[3] of BACnetTimeStamp	STRING	Read Only

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
al BACnetTimeStamp properties represented as a string in Australian time format			
Reliability	BACnetReliability	LONG	Read/Write
ActiveText	CharacterString	STRING	Read/Write
InactiveText	CharacterString	STRING	Read/Write
ChangeOfStateCount	Unsigned	LONG	Read/Write
ElapsedActiveTime	Unsigned32	LONG	Read/Write
ChangeOfStateTime BACnetDateTime property represented as a string in Australian date/time format	BACnetDateTime	STRING	Read Only
TimeOfStateCountReset BACnetDateTime property represented as a string in Australian date/time format	BACnetDateTime	STRING	Read Only
TimeOfActiveTimeReset ACnetDateTime property represented as a string in Australian date/time format	BACnetDateTime	STRING	Read Only
NotificationClass	Unsigned	LONG	Read/Write
NotifyType	BACnetNotifyType	LONG	Read/Write
MinimumOffTime	Unsigned32	LONG	Read/Write
MaximumOnTime	Unsigned32	LONG	Read/Write
ProfileName	CharacterString	STRING	Read Only

## Calendar Objects

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
ObjectIdentifier	BACnetObjectIdentifier	LONG	Read Only
ObjectName	CharacterString	STRING	Read Only
ObjectType	BACnetObjectType	LONG	Read Only
PresentValue	BOOLEAN	DIGITAL	Read Only
Description	CharacterString	STRING	Read/Write
DateList	List of BACnetCalendarEntry	See <a href="#">DateList Sub-Properties</a> .	Read Only
DateListEx (See <a href="#">Writing to Specific Calendar and Schedule Properties</a> .)	List of BACnetCalendarEntry	STRING	Read / Write

### DateList Sub-Properties

DateList (Number of elements in BACnetCalendarEntry list)	List of BACnetCalendarEntry	Plant SCADA Data Type
DateList.1..31 (Type of BACnetCalendarEntry: 0 - BACnetDate, 1 - BACnetDateRange, 2 - BACnetWeekNDay)	BACnetCalendarEntry	LONG
DateList.1..31.Date	BACnetDate	STRING
DateList.1..31.WeekNDay	BACnetWeekNDay	STRING
DateList.1..31.DateRange (Number of elements in BACnetDateRange, const == 2)	BACnetDateRange	LONG
DateList.1..31.DateRange.StartDate	BACnetDate	STRING
DateList.1..31.DateRange.EndDate	BACnetDate	STRING

**Device Objects**

<b>Property Mnemonic</b>	<b>BACnet Data Type</b>	<b>Plant SCADA Data Type</b>	<b>Access</b>
ObjectIdentifier	BACnetObjectIdentifier	LONG	Read Only
ObjectName	CharacterString	STRING	Read Only
ObjectType	BACnetObjectType	LONG	Read Only
SystemStatus	BACnetDeviceStatus	LONG	Read Only
VendorName	CharacterString	STRING	Read Only
VendorId	Unsigned16	LONG	Read Only
ModelName	CharacterString	STRING	Read Only
FirmwareRevision	CharacterString	STRING	Read Only
ApplicationSoftwareVersion	CharacterString	STRING	Read Only
Location	CharacterString	STRING	Read Only
Description	CharacterString	STRING	Read Only
ProtocolVersion	Unsigned	LONG	Read Only
ProtocolRevision	Unsigned	LONG	Read Only
ProtocolServicesSupported	BACnetServicesSupported	STRING	Read Only
ProtocolObjectTypesSupported	BACnetObjectTypesSupported	STRING	Read Only
MaxAPDULengthAccepted	Unsigned	LONG	Read Only
SegmetationSupported	BACnetSegmentation	LONG	Read Only
MaxSegmentsAccepted	Unsigned	LONG	Read Only
APDUSegmentTimeout	Unsigned	LONG	Read Only
APDUTimeout	Unsigned	LONG	Read Only
APDURetries	Unsigned	LONG	Read Only
DatabaseRevision	Unsigned	LONG	Read Only
WriteNULL	N/A	DIGITAL	Read/Write

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
	(Plant SCADA Custom Property)		
WriteNULLValue	N/A (Plant SCADA Custom Property)	REAL	Read/Write
WritePriority	N/A (Plant SCADA Custom Property)	LONG	Read/Write
ProtocolConformanceClasses	Unsigned	LONG	Read/Write
LocalTime	Time	STRING	ReadOnly
LocalDate	Date	STRING	ReadOnly
UtcOffset	Integer	LONG	ReadOnly
DayLightSavingsStatus	Boolean	DIGITAL	Read/Write
MaxMaster	Unsigned(1..127)	LONG	Read/Write
MaxInfoframes	Unsigned	LONG	Read/Write
ConfigurationFiles	BACnetARRAY[N] of BACnetObjectId	LONG	ReadOnly
LastRestoreTime	BACnetTimeStamp	STRING	ReadOnly
BackupFailureTimeout	Unsigned16	LONG	Read/Write
RestorePreparationTime	Unsigned16	LONG	Read/Write
BackupPreparationTime	Unsigned16	LONG	Read/Write
RestorecompletionTime	Unsigned16	LONG	Read/Write
BackupandRestorestate	BACnetBackupState	LONG	Read/Write
LastRestartReason	BACnetRestartReason	LONG	Read/Write
TimeOfDeviceRestart	BACnetTimeStamp	STRING	ReadOnly
TimeSynchronizationInterval	Unsigned	LONG	Read/Write
AlignIntervals	Boolean	DIGITAL	Read/Write

<b>Property Mnemonic</b>	<b>BACnet Data Type</b>	<b>Plant SCADA Data Type</b>	<b>Access</b>
IntervalOffset	Unsigned	LONG	Read/Write
PropertyList.1..63	BACnetARRAY[N] of BACnetPropertyIdentifier	LONG	ReadOnly
SerialNumber	CharacterString	STRING	ReadOnly
StatusFlags	BACnetStatusFlags	STRING	ReadOnly
EventState	BACnetEventState	LONG	ReadOnly
Reliability	BACnetReliability	LONG	Read/Write
EventDetectionEnable	Boolean	DIGITAL	Read/Write
NotificationClass	Unsigned	LONG	Read/Write
EventEnable  BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	Read/Write
AckedTransitions  BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	ReadOnly
NotifyType	BACnetNotifyType	LONG	Read/Write
EventTimestamps.ToOffNormal  EventTimestamps.ToFault  EventTimestamps.ToNormal	BACnetARRAY[3] of BACnetTimeStamp	STRING	ReadOnly

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
EventMessageTexts.ToOffNormal EventMessageTexts.ToFault EventMessageTexts.ToNormal	BACnetARRAY[3] of CharacterString	STRING	Read/Write
EventMessageTextsConfig.ToOffNormal EventMessageTextsConfig.ToFault EventMessageTextsConfig.ToNormal	BACnetARRAY[3] of CharacterString	STRING	Read/Write
ReliabilityEvaluationInhibit	Boolean	DIGITAL	Read/Write
Tags List of Name Values will be represented as a string separated by a comma and Name , Values are separated by ';'	BACnetARRAY[N] of BACnetNameValue	STRING	ReadOnly
ProfileLocation	CharacterString	STRING	ReadOnly
DeployedProfileLocation	CharacterString	STRING	ReadOnly

## Life Safety Point Objects

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
ObjectIdentifier	BACnetObjectIdentifier	LONG	ReadOnly
ObjectName	CharacterString	STRING	ReadOnly
ObjectType	BACnetObjectType	LONG	ReadOnly
PresentValue	BACnetLifeSafetyState	REAL	Read/Write
TrackingValue	BACnetLifeSafetyState	LONG	Read/Write
Description	CharacterString	STRING	Read/Write
DeviceType	CharacterString	STRING	Read/Write

StatusFlags	BACnetStatusFlags	STRING	ReadOnly
EventState	BACnetEventState	LONG	ReadOnly
Reliability	BACnetReliability	LONG	Read/Write
OutOfService	Boolean	DIGITAL	Read/Write
Mode	BACnetLifeSafetyMode	LONG	Read/Write
AcceptedModes	List of BACnetLifeSafetyMode	STRING	Read/Write
TimeDelay	Unsigned	LONG	Read/Write
NotificationClass	Unsigned	LONG	Read/Write
LifeSafetyAlarmValues  List of Unsigned property represented as a string of integers separated by a comma	List of BACnetLifeSafetyState	STRING	Read/Write
AlarmValues  List of Unsigned property represented as a string of integers separated by a comma	List of BACnetLifeSafetyState	STRING	Read/Write
FaultValues  List of Unsigned property represented as a string of integers separated by a comma	List of BACnetLifeSafetyState	STRING	Read/Write
EventEnable  BACnetEventTransitionBit s represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBit s	LONG	Read/Write

AckedTransitions  BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	ReadOnly
NotifyType	BACnetNotifyType	LONG	Read/Write
EventTimeStamps.ToOffNormal  EventTimeStamps.ToFault  EventTimeStamps.ToNormal  BACnetTimeStamp properties represented as a string in Australian time format	BACnetARRAY[3] of BACnetTimeStamp	STRING	ReadOnly
EventMessageTexts.ToOffNormal  EventMessageTexts.ToFault  EventMessageTexts.ToNormal	BACnetARRAY[3] of CharacterString	STRING	ReadOnly
EventMessageTextsConfig.ToOffNormal  EventMessageTextsConfig.ToFault  EventMessageTextsConfig.ToNormal	BACnetARRAY[3] of CharacterString	STRING	ReadOnly
EventDetectionEnable	Boolean	DIGITAL	Read/Write
EventAlgorithmInhibitRef.ObjectIdentifier  EventAlgorithmInhibitRef.PropertyIdentifier  EventAlgorithmInhibitRef.	BACnetObjectPropertyReference	LONG	ReadOnly

PropertyArrayIndex			
EventAlgorithmInhibit	Boolean	DIGITAL	Read/Write
TimeDelayNormal	Unsigned	LONG	Read/Write
ReliabilityEvaluationInhibit	Boolean	DIGITAL	Read/Write
Silenced	BACnetSilensedState	LONG	Read/Write
OperationExpected	BACnetLifeSafetyOperation	LONG	Read/Write
MaintenanceRequired	BACnetMaintenance	LONG	Read/Write
Setting	Unsigned8	LONG	Read/Write
DirectReading	REAL	REAL	Read/Write
Units	BACnetEngineeringUnits	LONG	Read/Write
MemberOf	List of BACnet Device Object Reference	(See <a href="#">Sub-Properties of Life Safety Point</a> )	ReadOnly
PropertyList.1..63	BACnetARRAY[N] of BACnetPropertyIdentifier	LONG	ReadOnly
Tags List of Name Values will be represented as a string separated by a comma and Name , Values are separated by ';'.	BACnetARRAY[N] of BACnetNameValue	STRING	ReadOnly
ProfileLocation	CharacterString	STRING	Read/Write
ProfileName	CharacterString	STRING	Read/Write

**Sub-Properties of Life Safety Point**

List of BACnetDeviceObjectReference

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type
MemberOf (Number of elements in BACnetDeviceObjectReference list)	List of BACnetDeviceObjectReference	LONG
MemberOf.1..31 (Number of elements in BACnetDeviceObjectReference, const == 2)	BACnetDeviceObjectReference	LONG
MemberOf.1..31.DeviceId	BACnetObjectIdentifier	LONG
MemberOf.1..31.ObjectId	BACnetObjectIdentifier	LONG

## Life Safety Zone Objects

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
ObjectIdentifier	BACnetObjectIdentifier	LONG	ReadOnly
ObjectName	CharacterString	STRING	ReadOnly
ObjectType	BACnetObjectType	LONG	ReadOnly
PresentValue	BACnetLifeSafetyState	REAL	Read/Write
TrackingValue	BACnetLifeSafetyState	LONG	Read/Write
Description	CharacterString	STRING	Read/Write
DeviceType	CharacterString	STRING	Read/Write
StatusFlags	BACnetStatusFlags	STRING	ReadOnly
EventState	BACnetEventState	LONG	ReadOnly
Reliability	BACnetReliability	LONG	Read/Write
OutOfService	Boolean	DIGITAL	Read/Write
Mode	BACnetLifeSafetyMode	LONG	Read/Write
AcceptedModes	List of BACnetLifeSafetyMode	STRING	Read/Write

TimeDelay	Unsigned	LONG	Read/Write
NotificationClass	Unsigned	LONG	Read/Write
LifeSafetyAlarmValues	List of BACnetLifeSafetyState	STRING	Read/Write
AlarmValues  List of Unsigned property represented as a string of integers separated by a comma	List of BACnetLifeSafetyState	STRING	Read/Write
FaultValues  List of Unsigned property represented as a string of integers separated by a comma	List of BACnetLifeSafetyState	STRING	Read/Write
EventEnable  BACnetEventTransitionBit s represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBit s	LONG	Read/Write
AckedTransitions  BACnetEventTransitionBit s represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBit s	LONG	ReadOnly
NotifyType	BACnetNotifyType	LONG	Read/Write

EventTimeStamps.ToOffNormal EventTimeStamps.ToFault EventTimeStamps.ToNormal BACnetTimeStamp properties represented as a string in Australian time format	BACnetARRAY[3] of BACnetTimeStamp	STRING	ReadOnly
EventMessageTexts.ToOffNormal EventMessageTexts.ToFault EventMessageTexts.ToNormal	BACnetARRAY[3] of CharacterString	STRING	ReadOnly
EventMessageTextsConfig.ToOffNormal EventMessageTextsConfig.ToFault EventMessageTextsConfig.ToNormal	BACnetARRAY[3] of CharacterString	STRING	ReadOnly
EventDetectionEnable	Boolean	DIGITAL	Read/Write
EventAlgorithmInhibitRef EventAlgorithmInhibitRef.ObjectIdentifier EventAlgorithmInhibitRef.PropertyIdentifier EventAlgorithmInhibitRef.PropertyArrayIndex	BACnetObjectPropertyReference	LONG	ReadOnly
EventAlgorithmInhibit	Boolean	DIGITAL	Read/Write
TimeDelayNormal	Unsigned	LONG	Read/Write
ReliabilityEvaluationInhibit	Boolean	DIGITAL	Read/Write
Silenced	BACnetSilensedState	LONG	Read/Write

OperationExpected	BACnetLifeSafetyOperatio n	LONG	Read/Write
MaintenanceRequired	BACSTAC_BOOLEAN	LONG	Read/Write
ZoneMembers	List of BACnetDeviceObjectRefer ence	See <a href="#">Sub-Properties of Life Safety Zone</a> .	ReadOnly
MemberOf	List of BACnetDeviceObjectRefer ence	See <a href="#">Sub-Properties of Life Safety Zone</a> .	ReadOnly
PropertyList.1..63	BACnetARRAY[N] of BACnetPropertyIdentifier	LONG	ReadOnly
Tags  List of Name Values will be represented as a string separated by a comma and Name , Values are separated by ';'	BACnetARRAY[N] of BACnetNameValue	STRING	
ProfileLocation	CharacterString	STRING	Read/Write
ProfileName	CharacterString	STRING	Read/Write

### Sub-Properties of Life Safety Zone

#### List of BACnetDeviceObjectReference

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type
ZoneMembers  (Number of elements in BACnetDeviceObjectReference list)	List of BACnetDeviceObjectReference	LONG
ZoneMembers.1..31  (Number of elements in BACnetDeviceObjectReference, const == 2)	BACnetDeviceObjectReference	LONG
ZoneMembers.1..31.DeviceId	BACnetObjectIdentifier	LONG
ZoneMembers.1..31.ObjectId	BACnetObjectIdentifier	LONG

MemberOf (Number of elements in BACnetDeviceObjectReference list)	List of BACnetDeviceObjectReference	LONG
MemberOf.1..31 (Number of elements in BACnetDeviceObjectReference, const == 2)	BACnetDeviceObjectReference	LONG
MemberOf.1..31.DeviceId	BACnetObjectIdentifier	LONG
MemberOf.1..31.ObjectId	BACnetObjectIdentifier	LONG

## Loop Objects

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
AckedTransitions  BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBit s	LONG	Read Only
Action	BACnetAction	LONG	Read/Write
OutOfService	Boolean	DIGITAL	Read/Write
ReliabilityEvaluationInhibit	Boolean	DIGITAL	Read/Write
EventAlgorithmInhibit	Boolean	DIGITAL	Read/Write
PresentValue	Real	REAL	Read/Write
ObjectIdentifier	BACnetObjectIdentifier	LONG	Read Only
ObjectType	BACnetObjectType	LONG	Read Only
EventState	BACnetEventState	LONG	Read Only
EventDetectionEnable	Boolean	LONG	Read/Write
UpdateInterval	Unsigned	LONG	Read/Write
PriorityForWriting	Unsigned	LONG	Read/Write

Description	CharacterString	STRING	Read/Write
StatusFlags	BACnetStatusFlags	STRING	Read Only
ObjectName	CharacterString	STRING	Read Only
ErrorLimit	Real	REAL	Read/Write
ControlledVariableUnits	BACnetEngineeringUnits	LONG	Read/Write
ControlledVariableValue	Real	REAL	Read/Write
DerivativeConstantUnits	BACnetEngineeringUnits	LONG	Read/Write
EventMessageTexts.ToOff Normal  EventMessageTexts.ToFault  EventMessageTexts.ToNormal	BACnetARRAY[3] of CharacterString	STRING	Read/Write
EventMessageTextsConfig .ToOffNormal  EventMessageTextsConfig .ToFault  EventMessageTextsConfig .ToNormal	BACnetARRAY[3] of CharacterString	STRING	Read/Write
DerivativeConstant	Real	REAL	Read/Write
Deadband	Real	REAL	Read/Write
TimeDelay	Unsigned	LONG	Read/Write
TimeDelayNormal	Unsigned	LONG	Read/Write
EventEnable  BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBit s	LONG	Read/Write
EventTimeStamps.ToOffNormal  EventTimeStamps.ToFault	BACnetARRAY[3] of BACnetTimeStamp	STRING	Read Only

EventTimeStamps.ToNormal  BACnetTimeStamp properties represented as a string in Australian time format			
Reliability	BACnetReliability	LONG	Read/Write
Setpoint	Real	REAL	Read/Write
MaximumOutput	Real	REAL	Read/Write
MinimumOutput	Real	REAL	Read/Write
ProportionalConstant	Real	REAL	Read/Write
LowDiffLimit	BACnetOptionalReal	REAL	Read/Write
ProportionalConstantUnits	BACnetEngineeringUnits	LONG	Read/Write
IntegralConstant	Real	REAL	Read/Write
IntegralConstantUnits	BACnetEngineeringUnits	LONG	Read/Write
COVIncrement	Real	REAL	Read/Write
NotificationClass	Unsigned	LONG	Read/Write
NotifyType	BACnetNotifyType	LONG	Read/Write
ProfileName	CharacterString	STRING	Read Only
ProfileLocation	CharacterString	STRING	Read/Write
Bias	Real	REAL	Read/Write
SetpointReference.ObjectIdentifier  SetpointReference.PropertyIdentifier  SetpointReference.PropertyArrayIndex	BACnetObjectPropertyReference	LONG	Read Only
SetpointReference.Present	Boolean	DIGITAL	Read Only
ControlledVariableReference.ObjectIdentifier	BACnetObjectPropertyReference	LONG	Read Only

ControlledVariableReference .PropertyIdentifier ControlledVariableReference .PropertyArrayIndex			
ManipulatedVariableReference .ObjectIdentifier ManipulatedVariableReference .PropertyIdentifier ManipulatedVariableReference .PropertyArrayIndex	BACnetObjectPropertyReference	LONG	Read Only
OutputUnits	BACnetEngineeringUnits	LONG	Read/Write
PropertyList.1..63	BACnetARRAY[N] of BACnetPropertyIdentifier	LONG	ReadOnly
Tags List of Name Values will be represented as a string separated by a comma and Name , Values are separated by ';'	BACnetARRAY[N] of BACnetNameValue	STRING	ReadOnly

## Multi-state Input Objects

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
ObjectIdentifier	BACnetObjectIdentifier	LONG	Read Only
ObjectName	CharacterString	STRING	Read Only
ObjectType	BACnetObjectType	LONG	Read Only
PresentValue	Unsigned	LONG	Read/Write
Description	CharacterString	STRING	Read/Write
NumberOfStates	Unsigned	LONG	Read Only
StatusFlags	BACnetStatusFlags	STRING	Read Only
EventState	BACnetEventState	LONG	Read Only

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
OutOfService	BOOLEAN	DIGITAL	Read/Write
TimeDelay	Unsigned	LONG	Read/Write
EventEnable  BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	Read/Write
AckedTransitions  BACnet EventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	Read Only
EventTimeStamps.ToOffNormal  EventTimeStamps.ToFault  EventTimeStamps.ToNormal  BACnetTimeStamp properties represented as a string in Australian time format	BACnetARRAY[3] of BACnetTimeStamp	STRING	Read Only
Reliability	BACnetReliability	LONG	Read/Write
StateText[.1..63]  BACnet Array of Character Strings	BACnetARRAY[N] of CharacterString	STRING	Read/Write
NotificationClass	Unsigned	LONG	Read/Write
NotifyType	BACnetNotifyType	LONG	Read/Write
DeviceType	CharacterString	STRING	Read/Write

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
ProfileName	CharacterString	STRING	Read Only
AlarmValues List of Unsigned property represented as a string of integers separated by a comma	List of Unsigned	STRING	Read/Write
FaultValues List of Unsigned property represented as a string of integers separated by a comma	List of Unsigned	STRING	Read/Write

## Multi-state Output Objects

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
ObjectIdentifier	BACnetObjectIdentifier	LONG	Read Only
ObjectName	CharacterString	STRING	Read Only
ObjectType	BACnetObjectType	LONG	Read Only
PresentValue	Unsigned	LONG	Read/Write
Description	CharacterString	STRING	Read/Write
NumberOfStates	Unsigned	LONG	Read Only
StatusFlags	BACnetStatusFlags	STRING	Read Only
EventState	BACnetEventState	LONG	Read Only
OutOfService	BOOLEAN	DIGITAL	Read/Write
RelinquishDefault	Unsigned	LONG	Read/Write
PriorityArray	BACnetPriorityArray	STRING	Read Only
TimeDelay	Unsigned	LONG	Read/Write
EventEnable BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:	BACnetEventTransitionBits	LONG	Read/Write

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal			
AckedTransitions BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows: Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	Read Only
EventTimeStamps.ToOffNormal EventTimeStamps.ToFault EventTimeStamps.ToNormal BACnetTimeStamp properties represented as a string in Australian time format	BACnetARRAY[3] of BACnetTimeStamp	STRING	Read Only
Reliability	BACnetReliability	LONG	Read/Write
StateText[.1..63] BACnet Array of Character Strings	BACnetARRAY[N] of CharacterString	STRING	Read/Write
NotificationClass	Unsigned	LONG	Read/Write
NotifyType	BACnetNotifyType	LONG	Read/Write
DeviceType	CharacterString	STRING	Read/Write
ProfileName	CharacterString	STRING	Read Only

## Multi-state Value Objects

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
ObjectIdentifier	BACnetObjectIdentifier	LONG	Read Only
ObjectName	CharacterString	STRING	Read Only

<b>Property Mnemonic</b>	<b>BACnet Data Type</b>	<b>Plant SCADA Data Type</b>	<b>Access</b>
ObjectType	BACnetObjectType	LONG	Read Only
PresentValue	Unsigned	LONG	Read/Write
Description	CharacterString	STRING	Read/Write
Number Of States	Unsigned	LONG	Read Only
StatusFlags	BACnetStatusFlags	STRING	Read Only
EventState	BACnetEventState	LONG	Read Only
OutOfService	BOOLEAN	DIGITAL	Read/Write
RelinquishDefault	Unsigned	LONG	Read/Write
PriorityArray	BACnetPriorityArray	STRING	Read Only
TimeDelay	Unsigned	LONG	Read/Write
EventEnable  BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBit s	LONG	Read/Write
AckedTransitions  BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBit s	LONG	Read Only
EventTimeStamps.ToOffNormal  EventTimeStamps.ToFault  EventTimeStamps.ToNormal  BACnetTimeStamp properties represented as a string in Australian time	BACnetARRAY[3] of BACnetTimeStamp	STRING	Read Only

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
format			
Reliability	BACnetReliablity	LONG	Read/Write
StateText[.1..63] BACnet Array of Character Strings	BACnetARRAY[N] of CharacterString	STRING	Read/Write
NotificationClass	Unsigned	LONG	Read/Write
NotifyType	BACnetNotifyType	LONG	Read/Write
ProfileName	CharacterString	STRING	Read Only
AlarmValues List of Unsigned property represented as a string of integers separated by a comma	List of Unsigned	STRING	Read/Write
FaultValues List of Unsigned property represented as a string of integers separated by a comma	List of Unsigned	STRING	Read/Write

## Notification Class Objects

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
ObjectIdentifier	BACnetObjectIdentifier	LONG	Read Only
ObjectName	CharacterString	STRING	Read Only
ObjectType	BACnetObjectType	LONG	Read Only
Description	CharacterString	STRING	Read/Write
NotificationClass	Unsigned	LONG	Read/Write
Priority.1..3	BACnetARRAY[3] of Unsigned	LONG	Read Only

AckRequired  BACnetEventTransitionBits represented as a long integer.  Bits are stored in the integer as follows: Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBit S	LONG	Read/Write
PropertyList.1..63	BACnetARRAY[N] of BACnetPropertyIdentifier	LONG	Read Only
StatusFlags	BACnetStatusFlags	STRING	Read Only
EventState	BACnetEventState	LONG	Read Only
Reliability	BACnetReliability	LONG	Read/Write
EventDetectionEnable	Boolean	DIGITAL	Read/Write
EventEnable  BACnetEventTransitionBits represented as a long integer.  Bits are stored in the integer as follows: Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBit S	LONG	Read/Write
AckedTransitions  BACnetEventTransitionBits represented as a long integer.  Bits are stored in the integer as follows: Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBit S	LONG	Read Only
NotifyType	BACnetNotifyType	LONG	Read/Write

EventTimeStamps.ToOffNormal EventTimeStamps.ToFault EventTimeStamps.ToNormal BACnetTimeStamp properties represented as a string in Australian time format	BACnetARRAY[3] of BACnetTimeStamp	STRING	Read Only
EventMessageTexts.ToOffNormal EventMessageTexts.ToFault EventMessageTexts.ToNormal	BACnetARRAY[3] of CharacterString	STRING	Read/Write
EventMessageTextsConfig.ToOffNormal EventMessageTextsConfig.ToFault EventMessageTextsConfig.ToNormal	BACnetARRAY[3] of CharacterString	STRING	Read/Write
ReliabilityEvaluationInhibit	Boolean	DIGITAL	Read/Write
ProfileLocation	CharacterString	STRING	Read/Write
ProfileName	CharacterString	STRING	Read Only

### Positive Integer Value Objects

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
ObjectIdentifier	BACnetObjectIdentifier	LONG	ReadOnly
ObjectName	CharacterString	STRING	ReadOnly
ObjectType	BACnetObjectType	LONG	ReadOnly
Description	CharacterString	STRING	Read/Write
PresentValue	Unsigned	LONG	Read/Write
StatusFlags	BACnetStatusFlags	STRING	ReadOnly

EventState	BACnetEventState	LONG	ReadOnly
Reliability	BACnetReliability	LONG	Read/Write
OutOfService	Boolean	DIGITAL	Read/Write
Units	BACnetEngineeringUnits	LONG	Read/Write
Priorityarray	BACnetPriorityArray	STRING	ReadOnly
RelinquishDefault	Unsigned	LONG	Read/Write
CovIncrement	Unsigned	LONG	Read/Write
TimeDelay	Unsigned	LONG	Read/Write
NotificationClass	Unsigned	LONG	Read/Write
HighLimit	Unsigned	LONG	Read/Write
LowLimit	Unsigned	LONG	Read/Write
Deadband	Unsigned	LONG	Read/Write
LimitEnable  BACnetLimitEnable represented as a long integer.  Bits are stored in the integer as follows:  Bit 0 – low limit enable Bit 1 – high limit enable	BACnetLimitEnable	LONG	Read/Write
EventEnable  BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	Read/Write

AckedTransitions  BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	ReadOnly
NotifyType	BACnetNotifyType	LONG	Read/Write
EventTimeStamps.ToOffNormal  EventTimeStamps.ToFault  EventTimeStamps.ToNormal  BACnetTimeStamp properties represented as a string in Australian time format	BACnetARRAY[3] of BACnetTimeStamp	STRING	ReadOnly
EventMessageTexts.ToOffNormal  EventMessageTexts.ToFault  EventMessageTexts.ToNormal	BACnetARRAY[3] of CharacterString	STRING	ReadOnly
EventMessageTextsConfig.ToOffNormal  EventMessageTextsConfig.ToFault  EventMessageTextsConfig.ToNormal	BACnetARRAY[3] of CharacterString	STRING	ReadOnly
EventDetectionEnable	Boolean	DIGITAL	Read/Write

EventAlgorithmInhibitRef.ObjectIdentifier	BACnetObjectPropertyReference	LONG	ReadOnly
EventAlgorithmInhibitRef.PropertyIdentifier			
EventAlgorithmInhibitRef.PropertyArrayIndex			
EventAlgorithmInhibit	Boolean	DIGITAL	Read/Write
TimeDelayNormal	Unsigned	LONG	Read/Write
ReliabilityEvaluationInhibitor	Boolean	DIGITAL	Read/Write
MinPresValue	Unsigned	LONG	Read/Write
MaxPresValue	Unsigned	LONG	Read/Write
Resolution	Unsigned	LONG	Read/Write
PropertyList.1..63	BACnetARRAY[N] of BACnetPropertyIdentifier	LONG	ReadOnly
FaultHighLimit	Unsigned	LONG	Read/Write
FaultLowLimit	Unsigned	LONG	Read/Write
CurrentCommandPriority	Unsigned	LONG	Read/Write
LastCommandTime	BACnetTimeStamp	STRING	ReadOnly
CommandTimeArray	BACnetARRAY[16] of BACnetTimeStamp	STRING	ReadOnly
Tags List of Name Values will be represented as a string separated by a comma and Name , Values are separated by ';'	BACnetARRAY[N] of BACnetNameValue	STRING	ReadOnly
ProfileLocation	CharacterString	STRING	Read/Write
ProfileName	CharacterString	STRING	ReadOnly

## Schedule Objects

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
ObjectIdentifier	BACnetObjectIdentifier	LONG	Read Only
ObjectName	CharacterString	STRING	Read Only
ObjectType	BACnetObjectType	LONG	Read Only
PresentValue	Any	DIGITAL, REAL, LONG, STRING	Read Only
Description	CharacterString	STRING	Read/Write
EffectivePeriod	BACnetDateRange	(See <a href="#">Sub-Properties of EffectivePeriod.</a> )	Read Only
WeeklySchedule	BACnetARRAY[7] of BACnetDailySchedule	(See <a href="#">Sub-Properties of WeeklySchedule.</a> )	Read Only
WeeklyScheduleExInt [.Monday... Sunday]  (See <a href="#">Writing to Specific Calendar and Schedule Properties.</a> )	BACnetDailySchedule	STRING	Read / write
WeeklyScheduleExDig [.Monday... Sunday]  (See <a href="#">Writing to Specific Calendar and Schedule Properties.</a> )	BACnetDailySchedule	STRING	Read / write
WeeklyScheduleExDbl [.Monday...Sunday]  (See <a href="#">Writing to Specific Calendar and Schedule Properties.</a> )	BACnetDailySchedule	STRING	Read / write
WeeklyScheduleExEnum [.Monday...Sunday]  (See <a href="#">Writing to Specific Calendar and Schedule Properties.</a> )	BACnetDailySchedule	STRING	Read / write
WeeklyScheduleExUInt [.Monday...Sunday]  (See <a href="#">Writing to Specific Calendar and Schedule Properties.</a> )	BACnetDailySchedule	STRING	Read / write

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
Properties.)			
WeeklyScheduleExReal [.Monday...Sunday] (See <a href="#">Writing to Specific Calendar and Schedule Properties.</a> )	BACnetDailySchedule	STRING	Read / write
ExceptionSchedule	BACnetARRAY[N] of BACnetSpecialEvent	(See <a href="#">Sub-Properties of ExceptionSchedule.</a> )	Read Only
ExceptionScheduleExInt [.1...31] (See <a href="#">Writing to Specific Calendar and Schedule Properties.</a> )	BACnetSpecialEvent	STRING	Read / write
ExceptionScheduleExDig [.1...31] (See <a href="#">Writing to Specific Calendar and Schedule Properties.</a> )	BACnetSpecialEvent	STRING	Read / write
ExceptionScheduleExDbl [.1...31] (See <a href="#">Writing to Specific Calendar and Schedule Properties.</a> )	BACnetSpecialEvent	STRING	Read / write
ExceptionScheduleExEnum [.1...31] (See <a href="#">Writing to Specific Calendar and Schedule Properties.</a> )	BACnetSpecialEvent	STRING	Read / write
ExceptionScheduleExUInt [.1...31] (See <a href="#">Writing to Specific Calendar and Schedule Properties.</a> )	BACnetSpecialEvent	STRING	Read / write
ExceptionScheduleExReal [.1...31] (See <a href="#">Writing to Specific Calendar and Schedule Properties.</a> )	BACnetSpecialEvent	STRING	Read / write

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
Properties.)			
ScheduleDefault	Any	DIGITAL, REAL, LONG, STRING	Read Only
ObjPropRefList	List of BACnetDeviceObject Property Reference	(See ListOf DeviceObjectProperty Reference sub- properties.)	Read Only
PriorityForWriting	Unsigned(1..16)	LONG	Read Only
StatusFlags	BACnetStatusFlags	STRING	Read Only
Reliability	BACnetReliability	LONG	Read Only
OutOfService	BOOLEAN	DIGITAL	Read/Write

#### Sub-Properties of EffectivePeriod

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type
EffectivePeriod (Number of elements in BACnetDateRange, const == 2)	BACnetDateRange	LONG
EffectivePeriod.StartDate	BACnetDate	STRING
EffectivePeriod.EndDate	BACnetDate	STRING

#### Sub-Properties of WeeklySchedule

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type
WeeklySchedule (Number of elements in WeeklySchedule, const == 7)	BACnetARRAY[7] of BACnetDailySchedule	LONG
BACnetDailySchedule WeeklySchedule.Monday WeeklySchedule.Tuesday WeeklySchedule.Wednesday WeeklySchedule.Thursday WeeklySchedule.Friday WeeklySchedule.Saturday WeeklySchedule.Sunday	BACnetDailySchedule	LONG

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type
(Number of BACnetTimeValue elements in BACnetDailySchedule)		
WeeklySchedule.Monday..Sunday.1 ..31 (Number of elements in BACnetTimeValue, const == 2)	BACnetTimeValue	LONG
WeeklySchedule.Monday..Sunday.1 ..31.Time	BACnetTime	STRING
WeeklySchedule.Monday..Sunday.1 ..31.Value	Any	DIGITAL, REAL, LONG, STRING

### Sub-Properties of ExceptionSchedule

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type
ExceptionSchedule (Number of BACnetSpecialEvent elements in ExceptionSchedule)	BACnetARRAY[N]of BACnetSpecialEvent	LONG
ExceptionSchedule.1..31 (Number of elements in BACnetSpecialEvent, const == 3)	BACnetSpecialEvent	LONG
ExceptionSchedule.1..31.TimeValue List (Number of BACnetTimeValue elements in BACnetTimeValue list)	List of BACnetTimeValue	LONG
ExceptionSchedule.1..31.TimeValue List.1..31 (Number of elements in BACnetTimeValue, const == 2)	BACnetTimeValue	LONG
ExceptionSchedule.1..31.TimeValue List.1..31.Time	BACnetTime	STRING
ExceptionSchedule.1..31.TimeValue List.1..31.Value	Any	DIGITAL, REAL, LONG, STRING
ExceptionSchedule.1..31.EventPriority	Unsigned (1..16)	LONG
ExceptionSchedule.1..31.CalendarReference	BACnetObjectIdentifier	LONG

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type
ExceptionSchedule.1..31.CalendarEntry (Type of BACnetCalendarEntry: 0 - BACnetDate, 1 - BACnetDateRange, 2 - BACnetWeekNDay)	BACnetCalendarEntry	LONG
ExceptionSchedule.1..31.CalendarEntry.Date	BACnetDate	STRING
ExceptionSchedule.1..31.CalendarEntry.WeekNDay	BACnetWeekNDay	STRING
ExceptionSchedule.1..31.CalendarEntry.DateRange (Number of elements in BACnetDateRange, const == 2)	BACnetDateRange	LONG
ExceptionSchedule.1..31.CalendarEntry.DateRange.StartDate	BACnetDate	STRING
ExceptionSchedule.1..31.CalendarEntry.DateRange.EndDate	BACnetDate	STRING

### Sub-Properties of Schedule

### List of BACnetDeviceObjectPropertyReference

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type
ObjPropRefList (Number of elements in BACnetDeviceObjectPropertyReference list)	List of BACnetDeviceObjectPropertyReference	LONG
ObjPropRefList.1..31 (Number of elements in BACnetDeviceObjectPropertyReference, const == 4)	BACnetDeviceObjectPropertyReference	LONG
ObjPropRefList.1..31.DeviceId	BACnetObjectIdentifier	LONG
ObjPropRefList.1..31.ObjectId	BACnetObjectIdentifier	LONG
ObjPropRefList.1..31.PropertyId	Unsigned	LONG

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type
ObjPropRefList.1..31.ArrayIndex	Unsigned	LONG

## Structured View Objects

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
ObjectIdentifier	BACnetObjectIdentifier	LONG	ReadOnly
ObjectName	CharacterString	STRING	ReadOnly
ObjectType	BACnetObjectType	LONG	ReadOnly
Description	CharacterString	STRING	Read/Write
NodeType	BACnetNodeType	STRING	ReadOnly
NodeSubtype	CharacterString	STRING	Read/Write
SubordinateList	BACnetARRAY[N] of BACnetDeviceObjectReference	LONG	ReadOnly
SubordinateAnnotations	BACnetARRAY[N] of CharacterString	STRING	ReadOnly
PropertyList.1..63	BACnetARRAY[N] of BACnetPropertyIdentifier	LONG	ReadOnly
Tags List of Name Values will be represented as a string separated by a comma and Name , Values are separated by ';'	BACnetARRAY[N] of BACnetNameValue	STRING	ReadOnly
ProfileLocation	CharacterString	STRING	ReadOnly
ProfileName	CharacterString	STRING	ReadOnly

## Timer Objects

Property Mnemonic	BACnet Data Type	Plant SCADA Data Type	Access
ObjectIdentifier	BACnetObjectIdentifier	LONG	ReadOnly

ObjectName	CharacterString	STRING	ReadOnly
ObjectType	BACnetObjectType	LONG	ReadOnly
Description	CharacterString	STRING	Read/Write
PresentValue	Unsigned	LONG	Read/Write
StatusFlags	BACnetStatusFlags	STRING	ReadOnly
EventState	BACnetEventState	LONG	ReadOnly
Reliability	BACnetReliability	LONG	Read/Write
OutOfService	Boolean	DIGITAL	Read/Write
TimerState	BACnetTimerState	LONG	ReadOnly
TimerRunning	Boolean	DIGITAL	Read/Write
UpdateTime	BACnetDateTime	STRING	ReadOnly
LastStateChange	BACnetTimerTransition	LONG	ReadOnly
ExpirationTime	BACnetDateTime	STRING	ReadOnly
InitialTimeout	Unsigned	LONG	Read/Write
DefaultTimeout	Unsigned	LONG	Read/Write
MaxPresValue	Unsigned	LONG	Read/Write
Resolution	Unsigned	LONG	Read/Write
StateChangeValues.1..7	BACnetARRAY[7] of BACnetTimerStateChange Value	STRING	ReadOnly
ObjPropRefList	BACnetLIST of BACnetDeviceObjectProp ertyReference	(See ListOf DeviceObjectProperty Reference sub-properties at Schedule object )	ReadOnly
PriorityForWriting	Unsigned(1...16)	LONG	Read/Write
EventDetectionEnable	Boolean	DIGITAL	Read/Write
NotificationClass	Unsigned	LONG	Read/Write
TimeDelay	Unsigned	LONG	Read/Write
TimeDelayNormal	Unsigned	LONG	Read/Write

AlarmValues  List of Unsigned property represented as a string of integers separated by a comma	BACnetLIST of BACnetTimerState	STRING	Read/Write
EventEnable  BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	Read/Write
AckedTransitions  BACnetEventTransitionBits represented as a long integer. Bits are stored in the integer as follows:  Bit 0 – to off normal Bit 1 – to fault Bit 2 – to normal	BACnetEventTransitionBits	LONG	ReadOnly
NotifyType	BACnetNotifyType	LONG	Read/Write
EventTimeStamps. ToOffNormal  EventTimeStamps.ToFault  EventTimeStamps.ToNormal  BACnetTimeStamp properties represented as a string in Australian time format	BACnetARRAY[3] of BACnetTimeStamp	STRING	ReadOnly
EventMessageTexts. ToOffNormal  EventMessageTexts.ToFault  EventMessageTexts.	BACnetARRAY[3] of CharacterString	STRING	ReadOnly

ToNormal			
EventMessageTextsConfig. ToOffNormal  EventMessageTextsConfig. ToFault  EventMessageTextsConfig. ToNormal	BACnetARRAY[3] of CharacterString	STRING	ReadOnly
EventAlgorithmInhibitRef. ObjectIdentifier  EventAlgorithmInhibitRef. PropertyIdentifier  EventAlgorithmInhibitRef. PropertyArrayIndex	BACnetObjectPropertyRef erence	LONG	ReadOnly
EventAlgorithmInhibit	Boolean	DIGITAL	Read/Write
ReliabilityEvaluationInhibi t	Boolean	DIGITAL	Read/Write
Tags  List of Name Values will be represented as a string separated by a comma and Name , Values are separated by ';'	BACnetARRAY[N] of BACnetNameValue	STRING	ReadOnly
ProfileLocation	CharacterString	STRING	Read/Write
ProfileName	CharacterString	STRING	ReadOnly

## BACNET Parameters



### UNINTENDED EQUIPMENT OPERATION

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel regarding undocumented features.

The parameters supported by the BACNET Driver are divided into the following categories:

- [Blocking Parameters](#)
- [Discovery Parameters](#)
- [Read Parameters](#)
- [Write Parameters](#)
- [APDU Parameters](#)
- [Logging Parameters.](#)

Custom settings for the BACNET driver need to be included in the citect.ini file.

Any setting under the **[BACNET]** section of the .ini file will globally impact all devices.

Settings can also be made for specific devices using the following format for the .ini file section name:

**[BACNET.<ClusterName>.<DeviceName>]**

where:

- <ClusterName> is the name of the cluster configured within the associated project, and
- <DeviceName> is the name of an I/O device specified under the **I/O Devices** tab of Plant SCADA Studio's **Topology** activity.

## Blocking Parameters

Parameter name	Description	Allowable value	Default value
FailOnBadData	Specifies whether or not to force the display of good data within a block read which contains bad tags.	1: Block read which contains a bad tag would result in the whole block returning #BAD. 0: Ignore any bad tags in the block reads and just show 0 for those tags.	1

## See Also

[BACNET Parameters](#)

## Discovery Parameters

Parameter name	Description	Allowable value	Default value
DiscoveryRate	The discovery rate in milliseconds. Specifies the rate at which the driver sends a 'WhoIs' message for dynamic device	Equal or greater than 0	20000

Parameter name	Description	Allowable value	Default value
	<p>binding.</p> <p><b>Note:</b> This parameter can only be used globally, it can not be set for a specific device.</p> <p>If this parameter is set to 0, the driver will not perform the dynamic device discovery, so the I/O Device address should include the information in order to set up the connection to the device, e.g. device IP address, port number, etc.</p>		
WholsMaxGap	Enables filtered device detection via Whols message. It Blocks adjacent or nearby Device Id numbers within 'WholsMaxGap' range into a single call to Whols message. A non zero value enables filtered device detection according to configured BacNet devices	0 to 65535	0: Use unfiltered device detection

## See Also

[BACNET Parameters](#)

## Read Parameters

Parameter name	Description	Allowable value	Default value
UseReadMultiple	Specifies whether to use the ReadPropertyMultiple Service	1: Allow 0: Not allowed	1
FastPoll	The parameter has been deprecated.		
SlowPoll	The parameter has been deprecated.		

Parameter name	Description	Allowable value	Default value
RetriesOnError	Maximum number of times read transaction will be re-transmitted in case when the message packet received from device is invalid	0 - 10	3
COVLifeTime	Sets the change-of-value (COV) subscription time (in seconds).  The driver attempts to use COV reporting for devices that support it. It subscribes to COV notifications for a period defined by COVLifeTime setting, and periodically updates the existing subscriptions. If COVLifeTime equals 0, then COV reporting is not used.	0 to 86400	3600 (1 hour)
COVExpireLag	Sets the period of time (in milliseconds) to update an existing subscription before it expires. The driver periodically updates existing COV subscriptions.	Equal to or greater than zero (0)	60000 (1 min)
COVCancelAtShutdown	Specifies whether to cancel COV subscriptions at driver shutdown	1: Cancel COV subscriptions 0: Do not cancel COV subscriptions	1

## See Also

[BACNET Parameters](#)

## Write Parameters



UNINTENDED EQUIPMENT OPERATION

Do not change these protocol parameters, except on the advice of Technical Support for this product. All these parameters default to a value tested to work in most cases.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel regarding undocumented features.

Parameter name	Description	Allowable value	Default value
WritePriority	BACnet allows you to specify a priority when writing to commandable properties. This parameter specifies the priority given to writes.	1 - 16	16
WriteQueueSize	Sets the size of the device write queue.	Equal or greater than 1	1000
UseWriteMultiple	Specifies whether to use the WritePropertyMultiple service.	1: Allowed 0: Not allowed	1

## See Also

[BACNET Parameters](#)

## APDU Parameters

APDU stands for Application Protocol Data Unit. The APDU settings allow you specify parameters for the actual data packet that is transmitted to the BACnet device.

Parameter name	Description	Allowable value	Default value
AckTimeOut	Time (in milliseconds) spent waiting for a reply to a message.	500 - 60000	6000
SegmentTimeout	Time (in milliseconds) spent waiting for a segment acknowledgment message.	500 - 60000	5000
Retries	Maximum number of times request message will be resent due to timeout on a request.	0 - 10	3

Parameter name	Description	Allowable value	Default value
FrameSize	Maximum frame size.	50 -1476	1476
MaxSegments	Maximum number of segments which may be transmitted and/or received in one segmented message.	1 - 256	10
WindowSize	Maximum number of segments which can be transmitted before an acknowledgment is sent to the initiator.	1 - 127	5

## See Also

[BACNET Parameters](#)

## Logging Parameters

[BACNET] Parameter	Description	Allowable value	Default value
DebugCategory	This parameter sets which log message categories are written to the log file.	ALL, or any combination of the following (separated by a ' ' pipe character): TAG   PROT   DCB  CACHE   THREAD   MISC	-
DebugLevel	This parameter sets the trace level for the log messages that are written to the log file.	ALL, or any combination of the following (separated by a ' ' pipe character): WARN   ERROR   TRACE   DEBUG	-
DebugUnits	This parameter sets the I/O Device name to limit logging to a specific I/O Device.	I/O device name in the format <ClusterName>.<DeviceName>	-

## See Also

[Logging](#)

## Logging

The BACNET driver can log combinations of trace levels across different categories. This is achieved by setting the following INI parameters.

### [BACNET]DebugLevel

This parameter allows you to define the trace level. The options include:

Option	Description
WARN	Output warning messages.
ERROR	Output error messages.
TRACE	Output trace messages.
DEBUG	Output debug messages.
ALL	All of the above.

### [BACNET]DebugCategory

This parameter allows you to enable logging for a particular category of trace. The options include:

Option	Description
TAG	Tag configuration trace.
PROT	Protocol level debug.
DCB	Front end driver trace.
CACHE	Cache debug.
THREAD	Thread trace.
MISC	Any other debugging.
ALL	All of the above.

In both cases, you can use any combination of the available options separated by a pipe character ( | ).

#### Example

```
[BACNET]
DebugLevel=WARNING|ERROR|TRACE
DebugCategory=ALL
```

In most cases, you can set the DebugLevel and DebugCategory parameters to ALL. If this results in a large, cumbersome log file that causes older data to be overwritten, you can consider refining the debug options.

---

**Note:** These parameters do not have a value specified by default. To enable logging for this driver, you need to

---

manually configure these parameters in your INI file.

The events generated by the BACNET driver are logged in the following Plant SCADA syslog file:

syslog.IOServer.<Cluster name>.<IO Server name>.dat

This file is located in the directory specified by the Citect.ini parameter **[CtEdit]Logs**. By default, this location will be:

%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Logs

The size of the syslog file and its rollover process is configured via the **[Debug]SysLogSize** and **[Debug]SysLogArchive** INI parameters. For more information, see the topic *Debug Parameters* in the Plant SCADA documentation.

---

**Note:** If the number of messages being processed for a driver exceeds the value specified by the Plant SCADA parameter **[Kernel]ErrorBuffers**, the pending messages will not be logged. If this occurs, you can increase the value of this parameter so that all the messages generated by the driver are reported. The number of messages that are not logged can be monitored via the Kernel's General window under the item "Lost Errors". If you adjust **[Kernel]ErrorBuffers** and Lost Errors are still occurring, you may need to adjust the level of system logging.

## See Also

[Logging Parameters](#)

## BACnet Tag Import

You can use Plant SCADA's **Import Tags** tool to automatically create variable tags that represent the object properties on a BACnet device.

**To create variable tags based on BACnet object properties:**

1. In Plant SCADA Studio go to **Topology** activity and select **I/O Devices**.
2. On the Command Bar, select **Import Tags**.

The Import Variable Tags dialog box will display.

3. In the **Destination** panel, select the required **I/O Device**. The drop-down menu includes a list of all the I/O devices currently configured in your Plant SCADA project.

The device you select needs to be configured to use the BACnet protocol, otherwise an error message will display when you attempt to configure the device connection.

4. In the **Source** panel, set the **Database Type** to "BACnet".

At this point, the **External Database** field will display the address configured for the selected device on the I/O Devices Communication Settings. It will use the following format:

<device ID> -i <device IP address>

The "-i <device IP address>" component of the address is optional. However, if the device is on a different IP subnet, you will need to include the IP address.

If an address has not been configured on the I/O Devices form, use the **Browse** button to discover the device ID.

This button will display the BACnet Device Configuration dialog.

- a. To discover the **Device ID** for an I/O device, click the **Browse** button on the BACnet Device Configuration dialog.

The BACnet Device Discovery dialog will appear and display:

- a. A list of all the BACnet devices detected on the local network.

Or:

- b. A list of all devices detected in the EDE file selected in the **Tag Import Options** section of the dialog.

To view additional information about a device, double-click on it.

**Note:** If the BACnet device you are looking for is not included in the list of devices detected on the local network, you will need to return to the Import Variables Tags dialog and manually enter its device ID and IP address into the **External Database** field using the following syntax:

<device ID> -i <device IP address>

Select the required device from this list and click **OK**. The **Device ID** for the selected device will display on the BACnet Device Configuration dialog.

- b. In the **Tag Import Options** panel, select one of the following options:

- a. **Use device for tag import.**
- b. **Use EDE file for tag import.**

If you select the first option, you can view information about the device by clicking the **Device Info...** button.

If you select the EDE file option, you need to identify the required file in the **EDE Path** field. You can use the **Browse** button to locate the file.

- c. In the **Filter Options** panel, select the **Use Filters** check box to apply a filter to the tag import. The **Set Filters** button opens a dialog that lets you select the BACnet object properties you would like to include in the tag import process.
- d. In the **Advanced Options** panel, select any of the following options:

- a. **Use object name as Tag Name** — creates variable tags that are named using the following format:

<IODEVICE>\<OBJECT NAME>\OBJECT TYPE

For example, "AIO\ventilation\PresentValue".

- b. **Import BACnet Units and Descriptions to the present value tag** — imports the units and descriptions for the present value.
  - c. **Import BACnet Schedule Instances as Equipment** — generates equipment definitions based on the device's schedule instances. If you select this option, you need to specify the **Parent Equipment**.

This option is useful if you want to integrate the BACnet schedules on a device into Plant SCADA's Scheduler for runtime interaction. See the topic *Integrate BACnet Schedules into Scheduler* in the main help for more information.

- e. Click **OK** to close the BACnet Device Configuration dialog.

5. On the Import Variable Tags dialog, click **Import**.

Plant SCADA will connect to the device, read the Object\_List property (which contains a list of the BACnet objects on the device), and create the variable tags for the BACnet object properties. Variable tags are only created for the BACnet properties supported by Plant SCADA's BACNET driver.

If the **Import BACnet Schedule Instances as Equipment** option is selected, equipment definitions will be generated for each of the device's schedule instances.

## CTICMP Driver

The CTICMP driver supports TCP/IP communication with any TCP/IP Host on a network that can be pinged.

The maximum request length for the CTICMP protocol is 2048 bits.

This driver supports IP version 4 only. IPV6 is not supported.

### See Also

[TCP/IP Hosts](#)

[TCP/IP Host Data Types](#)

[Customizing a Project using Citect.ini Parameters](#)

[CTICMP Driver Specific Errors](#)

[Logging](#)

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

#### DANGER

**DANGER** indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.

#### WARNING

**WARNING** indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.

**⚠ CAUTION**

**CAUTION** indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.

**NOTICE**

NOTICE used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.

**Please Note**

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

**Before You Begin**

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ WARNING****LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

## TCP/IP Hosts

The CTICMP driver uses the TCP/IP method of communication to talk to any TCP/IP device that can be pinged on a network. The physical network layer below TCP/IP does not matter. The driver can communicate over Ethernet, ATM or FDDI as long as TCP/IP is available.

The CTICMP driver provides the facilities for Plant SCADA to be configured to periodically check the status of hosts on a TCP/IP network. It achieves this by sending Echo messages (pinging) to a host and waiting for an Echo Reply message.

Echo/Echo reply message are part of the Internet Control Message Protocol (CTICMP) which is part of the Internet Protocol (IP) and is available in any TCP/IP installations.

In this documentation, TCP/IP devices on a network are referred to as 'hosts'.

---

**Note:** The same Ethernet card that is being used for Plant SCADA communication can be used for PLC communication, though this may lower performance.

---

## See Also

[Device Address](#)

[Hardware Setup](#)

[Communication Settings](#)

[TCP/IP Host Data Types](#)

## TCP/IP Hosts - Device Address

This setting is used to fill out the **I/O Device Address** field in the **I/O Devices** view in Plant SCADA Studio's **Topology** activity.

For this particular device, you may leave this field blank.

## See Also

[Hardware Setup](#)

## TCP/IP Hosts - Hardware Setup

## Hints and Tips

- The first time a page is entered that has tags animating it for hosts that have not been used elsewhere, the page will take a while to draw. This is because the hosts need to be physically pinged before the driver can return the data that Plant SCADA requires to draw the page.
- The response times on the driver page in the kernel are not valid for this driver. The driver keeps information about each host in an internal cache. The response times in the kernel show how long it takes the driver to read the information out of the cache and send it back to Plant SCADA. It does not show the response time of the hosts being pinged. It will usually be 0.

---

**Note:** The I/O Device will always be online for CTICMP protocol.

---

## Required Components

TCP/IP needs to be installed on the machine on which the driver is operating. It also requires Windows Sockets Version 2.0.

## Plant SCADA Computer Setup

It is recommended that you setup your specific Ethernet card as required and confirm basic network communications before proceeding with the communications configuration for the CTICMP driver. Refer to the documentation accompanying your hardware for instructions.

## See Also

[Communication Settings](#)

## TCP/IP Hosts - Communication Settings

To establish communication with a device, configure the associated Boards, Ports and I/O Device in Plant SCADA Studio's **Topology** activity. If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure these settings, use the information outlined below.

## Board Settings

Field	Value
Board Type	Enter CTICMP.
Address	Enter 0.
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.

## Port Settings

A port definition is required, however some fields can be left blank. Only the port name, port number and board name are required. Only one port maybe specified for the CTICMP driver.

Field	Value
Port Number	Leave this field blank.
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank.
Special Options	Leave this field blank.

## I/O Device Settings

Field	Value
I/O Device Name	Enter a unique name (different to other IO device names with the same port name). This value could be blank if only one IO device exists.
I/O Device Protocol	Enter CTICMP.
I/O Device Address	Leave this field blank.

## TCP/IP Host Data Types

Data Types	Address Format	Plant SCADA Data Type	Description/Special Usage/Limitations/ Valid Ranges
State	<i>i1.i2.i3.i4:STATE</i>	DIGITAL	Device status. 1 if it is responding, 0 if it is not responding. Read only.
Status	<i>i1.i2.i3.i4:STATUS</i>	INTEGER	Device status. 1 if it is responding, 0 if it is not responding. Read only.
Error Number	<i>i1.i2.i3.i4:ERRNUM</i>	INTEGER	A number representing the error condition of the last ping. Read only. 0: No faults 1: Not responding 2: Socket write timed out 3: Socket read timed out
Error Description	<i>i1.i2.i3.i4:ERRDESC</i>	STRING	A description of the status of the last ping. Read only. Blank when no error, or, not responding socket write timed out socket write not successful
Poll Time	<i>i1.i2.i3.i4:POLLTIME</i>	LONG	The duration, in seconds, between each ping to the host. See Driver Parameter HostPollTime for default.
Time Out	<i>i1.i2.i3.i4:TIMEOUT</i>	LONG	The time, in seconds, to wait for a response from the host after it is pinged.
Response Time	<i>i1.i2.i3.i4:RESPTIME</i>	LONG	The time, in milliseconds, it took the host to respond to the last ping. Read only.

Where:

<i>i1</i>	The first octet (most significant) of the IP address of the device to ping
<i>i2</i>	The second octet of the IP address of the device to ping

I3	The third octet of the IP address of the device to ping
I4	The fourth octet of the IP address of the device to ping

## Examples

Data Type	DIGITAL
Address	192.168.1.1:STATE
Comment	FileServer01 state
Data Type	LONG
Address	192.168.1.1:RESPTIME
Comment	FileServer01 response time

## Customizing a Project using Citect.ini Parameters

The CTICMP driver supports the following parameter categories:

- [CTICMP Driver Parameters](#)
- [Logging Parameters.](#)

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

- Do not under any circumstances change or remove any citect.ini parameters without an informed understanding of the possible implications of your actions.
- Do not delete sections of the citect.ini file without an informed understanding of the possible implications of your actions.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** At times during the life cycle of a driver, circumstances may call for the creation of specific citect.ini parameters that remain undocumented. Always seek the advice of Technical Support regarding the possible implementation of undocumented parameters before you make any changes to a citect.ini file.

## CTICMP Driver Parameters

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support for this product regarding undocumented features.

[CTICMP] Parameter	Description	Allowable Values	Default Value
MaxPending	The maximum number of pending commands that the driver holds ready for immediate execution.	1 to 1024	256

## Driver Specific Parameters

[CTICMP] Parameter	Description	Allowable Values	Default Value
HostPollTime	The duration (in seconds) between each ping to hosts. This can be overwritten for individual hosts by writing to the POLLTIME variable.	0001 to 300000 (milliseconds)	500 milliseconds
HostTimeout	Specifies how many milliseconds to wait for a response before displaying an error message.	500 to 32000 (milliseconds)	3000 milliseconds
PacketSize	The size in bytes of the data buffer in the ICMP echo request.	0 to 16 (bytes)	1

## See Also

[Logging Parameters](#)

## Logging Parameters

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

- Do not change or remove any of the undocumented citect.ini parameters under any circumstances.
- Before deleting sections of the citect.ini file, confirm that no undocumented parameters will be deleted.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

[CTICMP] Parameter	Description	Allowable Values	Default Value
DebugLevel	The trace level used for the log file.	Any combination of the following, separated by a pipe character ( ): ERROR WARN TRACE DEBUG ALL For example, ERROR   TRACE. See <a href="#">Logging</a> .	-
DebugCategory	The categories used for the log file.	Any combination of the following, separated by a pipe character ( ): PROT = protocol level debug DCB = front end debug (i.e. I/O server) MISC = any other debugging BEND = Backend CACHE = Cache replication ALL = all of the above For example, PROT   DCB. See <a href="#">Logging</a> .	-

## Logging

The CTICMP driver can log combinations of trace levels across different categories. This is achieved by setting the following INI parameters.

### **[CTICMP]DebugLevel**

This parameter allows you to define the trace level. The options include:

Option	Description
WARN	Output warning messages.
ERROR	Output error messages.
TRACE	Output trace messages.
DEBUG	Output debug messages.
ALL	All of the above.

## [CTICMP]DebugCategory

This parameter allows you to enable logging for a particular category of trace. The options include:

Option	Description
PROT	Protocol level debug.
DCB	Front end driver trace.
CACHE	Cache debug.
BEND	Backend trace.
MISC	Any other debugging.
ALL	All of the above.

In both cases, you can use any combination of the available options separated by a pipe character ( | ).

### Example

```
[CTICMP]
DebugLevel=WARN | ERROR | TRACE
DebugCategory=ALL
```

In most cases, you can set the DebugLevel and DebugCategory parameters to ALL. If this results in a large, cumbersome log file that causes older data to be overwritten, you can consider refining the debug options.

---

**Note:** These parameters do not have a value specified by default. To enable logging for this driver, you need to manually configure these parameters in your INI file.

The events generated by the CTICMP driver are logged in the following Plant SCADA syslog file:

```
syslog.IOServer.<Cluster name>.<IO Server name>.dat
```

This file is located in the directory specified by the Citect.ini parameter **[CtEdit]Logs**. By default, this location will be:

```
%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Logs
```

The size of the syslog file and its rollover process is configured via the **[Debug]SysLogSize** and **[Debug]SysLogArchive** INI parameters. For more information, see the topic *Debug Parameters* in the Plant SCADA documentation.

---

**Note:** If the number of messages being processed for a driver exceeds the value specified by the Plant SCADA parameter [Kernel]ErrorBuffers, the pending messages will not be logged. If this occurs, you can increase the value of this parameter so that all the messages generated by the driver are reported. The number of messages that are not logged can be monitored via the Kernel's General window under the item "Lost Errors". If you adjust [Kernel]ErrorBuffers and Lost Errors are still occurring, you may need to adjust the level of system logging.

---

## See Also

[Logging Parameters](#)

## CTICMP Driver Specific Errors

When a hardware error occurs, Plant SCADA generates an alarm, and displays the alarm on the hardware alarm page (in the alarm description of the hardware alarm). To see the error number, make sure you have Alarm Category 255 defined with a display format that includes {ErrDesc,15} {ErrPage,10}.

There are a number of generic errors that are common to all protocols. In some cases only the generic error is available, though often both the generic error and a specific error are given.

The following errors, listed in (hexadecimal) sequence, are specific to this protocol. Plant SCADA displays the error number and description for common protocol-specific errors. Uncommon errors are not contained in the Plant SCADA error database, in which case Plant SCADA will only display the error number.

You may require additional information to enable you to rectify an error. This information should be detailed in the documentation that accompanied the I/O Device (or network). If, after reviewing all documentation, you cannot rectify an error, contact AVEVA Global Customer Support.

### **0x100 GENERIC\_CHANNEL\_OFFLINE**

Unable to open a socket for the driver to use.

### **0x101 GENERIC\_CHANNEL\_OFFLINE**

Setting the socket option not successful.

### **0x102 GENERIC\_CHANNEL\_OFFLINE**

Unable to create the thread that pings the hosts.

### **0x104 GENERIC\_UNIT\_OFFLINE**

Write to the socket timed out.

### **0x105 GENERIC\_UNIT\_OFFLINE**

Write to the socket not successful.

## DNPR Driver

The DNPR driver allows Plant SCADA to communicate with devices that support DNP3 protocol subsets Level 1 and Level 2. Some level 3 features are also supported.

For an explanation of the protocol subsets, see [The DNP 3.0 Protocol](#).

---

**Note:** To successfully implement the DNPR driver, it is recommended that you are familiar with the terminology and methods involved with configuring devices for DNP 3.0 communication.

---

Communication can take place over serial (RS232) or TCP/IP transport. The type of devices the driver will

communicate with include:

- RTUs
- IEDs
- PLCs.

---

**Note:** With the release of version 7.20, Plant SCADA supports the retrieval of time-stamped data directly from field devices. This capability is enabled by the Driver Runtime Interface (DRI), a component of Plant SCADA that is used by the DNPR driver to directly update time-stamped digital alarms, time-stamped analog alarms and event-based trends. For more information, see the topic *Retrieving Time-stamped Data from Field Devices* in the main Plant SCADA help.

---

## See Also

- [Device Setup](#)  
[Configure Your Project](#)  
[Advanced Configuration and Maintenance](#)  
[Troubleshooting](#)

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

### DANGER

**DANGER** indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.

**⚠ WARNING**

**WARNING** indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.

**⚠ CAUTION**

**CAUTION** indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.

**NOTICE**

**NOTICE** used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.

**Please Note**

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

**Before You Begin**

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ WARNING****LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

## The DNP 3.0 Protocol

DNP 3.0 is an open protocol that has been adopted as a standard by the utilities industries. As a result, there are numerous DNP 3.0 compatible devices available that support this protocol.

### Protocol Subsets

The DNP 3.0 protocol encompasses a broad spectrum of functionality. For a lot of supported devices, it is not necessary to implement the entire protocol, as only a subset is required.

In order to maintain compatibility between subset implementations, the DNP 3.0 User Group has defined three official subsets of the DNP 3.0 protocol. These are:

- Level 1
- Level 2
- Level 3

The higher the level, the more functionality that is implemented by the subset. A lot of devices have a Level 2 implementation as their base, with selected additional functionality as required.

To support this, Plant SCADA is based on a Level 2 implementation and will fully support level 2 functionality. It will also support some of the level 3 functionality.

As the DNP3 protocol expands, further functionality will be introduced as a result of feedback regarding user requirements and needs.

## Device Setup

If a device supports multiple protocols, it should be set for the DNP 3.0 protocol.

### Communications setup

The DNPR driver can use the RS-232, TCP, or UDP communication methods, using a standard COM port, digiboard, or Ethernet card. RS-232 communications can also be expanded through an adaptor. The type of adaptor used is determined by the DNP3 network Plant SCADA is interfacing with.

Plant SCADA operates on the presumption that a DNP 3 device will use RS-232 for communication. If you use the Communications Express Wizard to connect to a device, the following settings are used by default in your project.

Setting	Value
Baud Rate	9600
Data Bits	8
Stop Bits	1
Parity	0 (None)

These settings are recommendations only; your hardware may support other values. If you do set the baud rate, data bits, stop bits, or parity to another value, manually set the new value(s) in your project.

**Note:** DNP3 is a binary protocol, which means hardware flow control (not software flow control) should be used for serial connections.

A device should also be configured so that static data required by Plant SCADA is allocated to class 0. (For more information on classing data, see [Data Acquisition within DNP3](#).)

Configuring Class 0 with points not required by Plant SCADA places an extra communication burden on the system during the regular integrity polling. Event data corresponding to the above-mentioned static data should be assigned to classes 1, 2, or 3.

The maximum request length for the DNPR protocol is 256 bytes.

### See Also

[The DNP 3.0 Protocol](#)

## Configure Your Project

This section contains reference information related to the DNPR Driver that you'll require to configure your Plant SCADA project.

**Note:** You need to set a DNP address for your Plant SCADA system using the parameter [Basic Parameters](#), otherwise a popup message will appear at startup.

## See Also

- [DNPR - Device Address](#)
- [Communication Settings](#)
- [Configure Variables](#)
- [Projects Using Super Genies](#)

## DNPR - Device Address

The Device Address for a DNP3 compatible device is:

*DNP3\_address Citect.INI\_section\_name*

For example: *128 SITERTU*

where:

- *128* - is the DNP3 address of this device (valid addresses are in the range 0...65533)
- *SITERTU* - is the name of the section heading in Citect.ini under which parameters related to SITERTU will be held.

---

**Note:** If no *Citect.INI\_section\_name* is specified, it defaults to *Unit\_x*, where *x* is the value in the Number field on the I/O Device settings (for example, *Unit\_10*).

---

## See Also

- [Communications Settings](#)

## Communication Settings

To establish communication with a device, the associated Board, Port and I/O Device require configuration in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. If you need to manually configure them, you should use the values outlined below.

## Boards

### Serial Communication

The DNPR driver uses standard serial communications via a standard COM port or a dedicated serial board.

Field	Value
Board name	This field is user defined. (e.g. "Board1")
Board type	COMX
Address	0

Field	Value
I/O port	
Interrupt	
Special opt.	
Comments	This field is user defined and is not used by the driver.

**TCP/IP Communication**

Field	Value
Board name	This field is user defined. (e.g. "Board1")
Board type	TCP/IP
Address	0
I/O port	
Interrupt	
Special opt.	
Comments	This field is user defined and is not used by the driver.

**Ports****Serial communication**

Field	Value
Port Name	This field is user defined.
Port number	ComPort
Board name	Refers to the board previously defined in the 'boards' settings.
Baud rate	9600 (allowable values: 2400 – 38400)
Data bits	8
Stop bits	1
Parity	None
Special Opt	Leave this field blank
Comment	This field is user defined and is not used by the driver.

**TCP/IP communication**

Field	Value
Port Name	This field is user defined.
Port number	
Board name	Refers to the board previously defined in the 'boards' settings.
Baud rate	Leave this field blank
Data bits	Leave this field blank
Stop bits	Leave this field blank
Parity	Leave this field blank
Special Opt	<p>-laaa.bbb.ccc.ddd -Pe (Allowable values: &lt;-LP&gt; &lt;-U -T&gt;)</p> <p>where:</p> <ul style="list-style-type: none"> <li>• <i>aaa.bbb.ccc.ddd</i> is the IP address of the device</li> <li>• <i>e</i> is the port number of the device</li> <li>• <i>-U</i> for use when UDP being used (TCP is the default)</li> </ul>
Comment	This field is user defined and is not used by the driver.

**Note:** The maximum number of ports you can use is 512.

**I/O Devices**

Field	Value
Name	This field is user defined.
Number	The I/O device number. A unique number is required for each logical device, however primary and standby redundant devices require the same number.
Address	<p>DNP3_address Citect.INI_section_name</p> <p>Example: 128 SITERTU</p> <p>See <a href="#">DNPR - Device Address</a>.</p>
Protocol	DNPR
Port name	Refers to the port previously defined in the 'ports'

Field	Value
	settings.
Comment	This field is user defined and is not used by the driver.
Minimum Update Time	<p>This field is used to determine the frequency of the integrity polls.</p> <p>If this field is none zero and is smaller than the integrity poll frequency from the Citect.ini parameters, then this field will be used and the poll period will be recalculated.</p> <p>INI integrity poll frequency:</p> $\text{Integrity poll} = \text{INI poll period} * (\text{INI event poll ratio} + 1),$ <p>For more details, see the following INI parameters:</p> <ul style="list-style-type: none"> <li>[DNPR]EventPollPeriodDefault</li> <li>[&lt;unit&gt;]EventPollPeriod</li> <li>[DNPR]EventPollRatioDefault</li> <li>[&lt;unit&gt;]EventPollRatio</li> </ul> <p>New poll period recalculation:</p> $\text{Poll period} = \text{Minimum Update Time} / (\text{INI event poll ratio} + 1).$ <p><b>Note:</b> If the poll period is 0 then it will be defaulted to 10 seconds</p>

## Configure Variables

The data types used by the DNP3 protocol are represented across three main groups:

- Monitoring data types
- Control data types
- Other data types.

The tables in this section of the help describe the variable address required for each data type.

---

**Note:** Some of the data types listed in the following tables are identified as "advanced". They are recommended for experienced users who are familiar with the terminology and methods associated with the DNP 3.0 protocol.

## Monitoring Data Types

- [Binary Inputs](#)
- [Binary Outputs](#)
- [Counters](#)
- [Analog Inputs](#)

- Analog Outputs
- Time and Date
- IIN Flags

## Control Data Types

The driver can support multiple writes per Plant SCADA request for Binary Output and Analog output control.

- Binary Outputs
- Counters
- Analog Outputs
- Time and Date
- Utilities

## Other Types

- Polling Regime Data Types
- Global Data Types
- Miscellaneous Data Types

### Monitoring Data Types - Binary Inputs

This table defines the Plant SCADA variable address required for the specified DNP3 data types.

Address	Plant SCADA Type	DNP3 Type	Description
BI<n>.Val	DIGITAL	Binary Input Value Object 01 Variations 1,2 Object 02 Variations 1,2,3	R/O. Current state of Binary Input (This is Bit 7 of Status byte)
BI<n>.Val.TS	LONG	Binary Input Time Stamp (seconds since 1970)	UTC time when value last changed or when the value was acquired from the initial integrity poll
BI<n>.Val.TMS	LONG	Binary Input Time Stamp (milliseconds since midnight)	Milliseconds since midnight when value last changed or when the value was acquired from the initial integrity poll
BI<n>.Val.QUAL	LONG	Binary Input Quality	OPC quality of point converted from value of flags.

where:

- n = Point index number (range from 0 to 2048)
- R/O = read only
- W/O = write only
- R/W = read or write

## Advanced monitoring data types - binary inputs

**Note:** The following data types are only recommended for experienced users who are familiar with the terminology and methods associated with the DNP 3.0 protocol.

Address	Plant SCADA Type	DNP3 Type	Description
BI< n >.Stat	BYTE	Binary Input Status Status byte (excluding state) Object 01 Variation 2 Object 02 Variations 1,2,3	R/O. Status data related to Value. Bit 7 (State bit) not included – always zero.
BI< n >.Stat.Online	DIGITAL	Binary Input Status Online bit Object 01 Variation 2 Object 02 Variations 1,2,3	R/O. Bit 0 of Status byte. If Offline, command will not be successful. 0=Offline,1=Online.
BI< n >.Stat.Restart	DIGITAL	Binary Input Status Restart bit Object 01 Variation 2 Object 02 Variations 1,2,3	R/O. Bit 1 of Status byte Originating field device has been restarted. 0=normal, 1=Restart.
BI< n >.Stat.CommLost	DIGITAL	Binary Input Status Communication Lost bit Object 01 Variation 2 Object 02 Variations 1,2,3	R/O. Bit 2 of Status byte. If comms lost, command will not be successful. 0=normal, 1=Lost.
BI< n >.Stat.RemForced	DIGITAL	Binary Input Status Remote Forced Data bit Object 01 Variation 2 Object 02 Variations 1,2,3	R/O. Bit 3 of Status byte. Originating field device is in control 0=normal, 1=Forced.
BI< n >.Stat.LocalForced	DIGITAL	Binary Input Status Local Forced Data bit	R/O. Bit 4 of Status byte.

Address	Plant SCADA Type	DNP3 Type	Description
		Object 01 Variation 2 Object 02 Variations 1,2,3	This device is in control. 0=normal, 1=Forced.
BI< n >.Stat.ChatFilter	DIGITAL	Binary Input Status ChatterFilter bit Object 01 Variation 2 Object 02 Variations 1,2,3	R/O. Bit 5 of Status byte Low Pass Filter for filtering unwanted transitions. 0=normal, 1=Filter on.

where:

- n = Point index number (range from 0 to 2048)
- R/O = read only
- W/O = write only
- R/W = read or write

### Monitoring Data Types - Binary Outputs

This table defines the Plant SCADA variable address required for the specified DNP3 data types.

Address	Plant SCADA Type	DNP3 Type	Description
BO< n >.Val	DIGITAL	Binary Output Value Object 10 Variation 2	R/O. Current state of Binary Output.
BO< n >.Val.QUAL	LONG	Binary Output Quality	R/O. OPC quality of point converted from value of flags.

where:

- n = Point index number (range from 0 to 2048)
- R/O = read only
- W/O = write only
- R/W = read or write

### Advanced monitoring data types - binary outputs

---

**Note:** The following data types are only recommended for experienced users who are familiar with the terminology and methods associated with the DNP 3.0 protocol.

---

Address	Plant SCADA Type	DNP3 Type	Description
BO<n>.Stat	BYTE	Binary Output Status Status byte (excluding state) Object 10 Variation 2	R/O. Status data related to Value. Bit 7 (State bit) not included – always zero.
BO<n>.Stat.Online	DIGITAL	Binary Output Status Online bit Object 10 Variation 2	R/O. Bit 0 of Status byte If Offline, command will not be successful. 0=Offline,1=Online
BO<n>.Stat.Restart	DIGITAL	Binary Output Status Restart bit Object 10 Variation 2	R/O. Bit 1 of Status byte Originating field device has been restarted. 0=normal,1=Restart
BO<n>.Stat.CommLost	DIGITAL	Binary Output Status Communication Lost bit Object 10 Variation 2	R/O. Bit 2 of Status byte If comms lost, command will not be successful 0=normal,1=Lost
BO<n>.Stat.RemForced	DIGITAL	Binary Output Status Remote Forced Data bit Object 10 Variation 2	R/O. Bit 3 of Status byte Originating field device is in control 0=normal,1=Forced
BO<n>.Stat.LocalForced	DIGITAL	Binary Output Status Local Forced Data bit Object 10 Variation 2	R/O. Bit 4 of Status byte This device is in control 0=normal,1=Forced
BO<n>.Stat.SelPend	DIGITAL	Binary Output Status Plant SCADA – Object Selected in RTU	R/O. Bit 5 of Status byte This device is in control 0=not selected, 1=Selected, waiting on OPERATE

Address	Plant SCADA Type	DNP3 Type	Description
BO<n>.Stat.WrtPend	DIGITAL	Binary Output Status Plant SCADA – Select Initiated in RTU	R/O. Bit 6 of Status byte This device is in control 0=not initiated, 1=SELECT started
BO<n>.Stat. WrtFail	DIGITAL	Binary Output Status Plant SCADA – Problem in SELECT/OPERATE sequence	R/O. Bit 7 of Status byte This device is in control 0=normal, 1=Write unsuccessful
BO<n>.CtrlStat	BYTE	Binary Output Control Status Status byte Object 12 Variation(s) 1 <b>Note:</b> DNP (Subset level 2) supports an 'Immediate Freeze without acknowledgment', but as this does not verify the success of the command, only 'Immediate Freeze' is currently implemented.	R/O. Status of Control Operation. 0=Request accepted, initiated or queued 1= Request rejected – OPERATE received after SELECT had already timed-out 2= Request rejected – OPERATE received without receiving matching SELECT 3= Request rejected – Formatting errors in control request (Select, Operate or Doperate) 4= Request rejected – Control Operation not supported for this bit 5= Request rejected – Queue full or point already active 6= Request rejected – Control hardware problems 7-127=Undefined.

where:

- n = Point index number (range from 0 to 2048)
- R/O = read only

- W/O = write only
- R/W = read or write

### Monitoring Data Types - Counters

This table defines the Plant SCADA variable address required for the specified DNP3 data types.

Address	Plant SCADA Type	DNP3 Type	Description
C<n>.Val	LONG	Counter Value (16/32 bit) Object 20 Variations 1,2,5,6 Object 22 Variations 1,2	R/O. Accumulated transitions of a SW or HW point (16bit or 32bit value)
C<n>.Val.Qual	LONG	Counter Value Quality	R/O. OPC quality of point converted from value of flags.
FC<n>.Val	LONG	Frozen Counter Value (16/32 bit) Object 21 Variations 1,2,9,10	R/O. Counter value when last freeze performed. (16bit or 32bit value)
FC<n>.Val.Qual>	LONG	Frozen Counter Value Quality	R/O. OPC quality of point converted from value of flags.

where:

- n = Point index number (range from 0 to 2048)
- R/O = read only
- W/O = write only
- R/W = read or write

### Advanced monitoring data types - counters

**Note:** The following data types are only recommended for experienced users who are familiar with the terminology and methods associated with the DNP 3.0 protocol.

Address	Plant SCADA Type	DNP3 Type	Description
C<n>.Stat	BYTE	Counter Status Status byte Object 20 Variations 1,2	R/O. Status data related to Value.

Address	Plant SCADA Type	DNP3 Type	Description
		Object 22 Variations 1,2	
C<n>.Stat.Online	DIGITAL	Counter Status Online bit Object 20 Variations 1,2 Object 22 Variations 1,2	R/O. Bit 0 of Status byte If Offline, command will not be successful 0=Offline, 1=Online
C<n>.Stat.Restart	DIGITAL	Counter Status Restart bit Object 20 Variations 1,2 Object 22 Variations 1,2	R/O. Bit 1 of Status byte Originating field device has been restarted 0=normal, 1=Restart
C<n>.Stat.CommLost	DIGITAL	Counter Status Communication Lost bit Object 20 Variations 1,2 Object 22 Variations 1,2	R/O. Bit 2 of Status byte If comms lost, command will not be successful 0=normal, 1=Lost
C<n>.Stat.RemForced	DIGITAL	Counter Status Remote Forced Data bit Object 20 Variations 1,2 Object 22 Variations 1,2	R/O. Bit 3 of Status byte Originating field device is in control 0=normal, 1=Forced
C<n>.Stat.LocalForced	DIGITAL	Counter Status Local Forced Data bit Object 20 Variations 1,2 Object 22 Variations 1,2	R/O. Bit 4 of Status byte This device is in control 0=normal, 1=Forced
C<n>.Stat.RollOver	DIGITAL	Counter Status Roll Over bit Object 20 Variations 1,2 Object 22 Variations 1,2	R/O. Bit 5 of Status byte Counter has reached limit and reset to 0 0=normal, 1=Rollover
C<n>.DeltaVal	LONG	Delta Counter Value (16/32 bit) Object 20 Variations 3,4,7,8	R/O. Accumulated transitions of a SW or HW point This value is reset to 0 after each read.

Address	Plant SCADA Type	DNP3 Type	Description	
		(16bit or 32bit value)		
C<n.>.DeltaStat	BYTE	Delta Counter Status Status byte Object 20 Variations 3,4	R/O. Status data related to Value	
C<n.>.DeltaStat.Online	DIGITAL	Delta Counter Status Online bit Object 20 Variations 3,4	R/O. Bit 0 of Status byte If Offline, command will not be successful 0=Offline, 1=Online	
C<n.>.DeltaStat.Restart	DIGITAL	Delta Counter Status Restart bit Object 20 Variations 3,4	R/O. Bit 1 of Status byte Originating field device has been restarted 0=normal, 1=Restart	
C<n.>.DeltaStat.CommLos t	DIGITAL	Delta Counter Status Communication Lost bit Object 20 Variations 3,4	R/O. Bit 2 of Status byte If comms lost, command will not be successful 0=normal, 1=Lost	
C<n.>.DeltaStat.RemForce d	DIGITAL	Delta Counter Status Remote Forced Data bit Object 20 Variations 3,4	R/O. Bit 3 of Status byte Originating field device is in control 0=normal, 1=Forced	
C<n>.DeltaStat.LocalForce d	DIGITAL	Delta Counter Status Local Forced Data bit Object 20 Variations 3,4	R/O. Bit 4 of Status byte This device is in control 0=normal, 1=Forced	
C<n>.DeltaStat.RollOver	DIGITAL	Delta Counter Status Roll Over bit Object 20 Variations 3,4	R/O. Bit 5 of Status byte Counter has reached limit and reset to 0 0=normal, 1=Rollover	
FC<n>.Stat	BYTE	Frozen Counter Status	R/O.	

Address	Plant SCADA Type	DNP3 Type	Description
		Status byte Object 21 Variations 1,2	Status data related to Value
FC< n >.Stat.Online	DIGITAL	Frozen Counter Status Online bit Object 21 Variations 1,2	R/O. Bit 0 of Status byte If Offline, command will not be successful 0=Offline, 1=Online
FC< n >.Stat.Restart	DIGITAL	Frozen Counter Status Restart bit Object 21 Variations 1,2	R/O. Bit 1 of Status byte Originating field device has been restarted 0=normal, 1=Restart
FC< n >.Stat.CommLost	DIGITAL	Frozen Counter Status Communication Lost bit Object 21 Variations 1,2	R/O. Bit 2 of Status byte If comms lost, command will not be successful 0=normal, 1=Lost
FC< n >.Stat.RemForced	DIGITAL	Frozen Counter Status Remote Forced Data bit Object 21 Variations 1,2	R/O. Bit 3 of Status byte Originating field device is in control 0=normal, 1=Forced
FC< n >.Stat.LocalForced	DIGITAL	Frozen Counter Status Local Forced Data bit Object 21 Variations 1,2	R/O. Bit 4 of Status byte This device is in control 0=normal, 1=Forced
FC< n >.Stat.RollOver	DIGITAL	Frozen Counter Status Roll Over bit Object 21 Variations 1,2	R/O. Bit 5 of Status byte Counter has reached limit and reset to 0 0=normal, 1=Rollover

where:

- n = Point index number (range from 0 to 2048)
- R/O = read only

- W/O = write only
- R/W = read or write

### Monitoring Data Types - Analog Inputs

This table defines the Plant SCADA variable address required for the specified DNP3 data types.

Address	Plant SCADA Type	DNP3 Type	Description
AI<n>.Val	LONG	Analog Input Value (16/32 bit) Object 30 Var(s) 1,2,3,4 Object 32 Var(s) 1,2,3,4	R/O. Current value of Analog Input(16bit or 32bit value)
AI<n>.Val.Float	REAL	Analog Input Value (float) Object 30 Var(s) 5 Object 32 Var(s) 5,7	R/O. Current value of Analog Input (float value)
AI<n>.Val.TS	LONG	Analog Input Time Stamp (seconds since 1970)	UTC time when value last changed or when the value was acquired from the initial integrity poll.
AI<n>.Val.TMS	LONG	Analog Input Time Stamp (milliseconds since midnight)	Milliseconds since midnight when value last changed or when the value was acquired from the initial integrity poll.
AI<n>.Val.Qual	LONG	Analog Input Quality	OPC quality of point converted from value of flags.

where:

- n = Point index number (range from 0 to 2048)
- R/O = read only
- W/O = write only
- R/W = read or write

### Advanced monitoring data types - analog inputs

---

**Note:** The following data types are only recommended for experienced users who are familiar with the terminology and methods associated with the DNP 3.0 protocol.

---

<b>Address</b>	<b>Plant SCADA Type</b>	<b>DNP3 Type</b>	<b>Description</b>
AI<n>.Stat	BYTE	Analog Input Status Status byte Object 30 Variations 1,2,5 Object 32 Var(s) 1,2,3,4,5,7	R/O. Status data related to Value
AI<n>.Stat.Online	DIGITAL	Analog Input Status Online bit Object 30 Variations 1,2,5 Object 32 Var(s) 1,2,3,4,5,7	R/O. Bit 0 of Status byte If Offline, command will not be successful 0=Offline, 1=Online
AI<n>.Stat.Restart	DIGITAL	Analog Input Status Restart bit Object 30 Variations 1,2,5 Object 32 Var(s) 1,2,3,4,5,7	R/O. Bit 1 of Status byte Originating field device has been restarted. 0=normal, 1=Restart
AI<n>.Stat.CommLost	DIGITAL	Analog Input Status Communication Lost bit Object 30 Variations 1,2,5 Object 32 Var(s) 1,2,3,4,5,7	R/O. Bit 2 of Status byte If comms lost, command will not be successful 0=normal, 1=Lost
AI<n>.Stat.RemForced	DIGITAL	Analog Input Status Remote Forced Data bit Object 30 Variations 1,2,5 Object 32 Var(s) 1,2,3,4,5,7	R/O. Bit 3 of Status byte Originating field device is in control 0=normal, 1=Forced
AI<n>.Stat.LocalForced	DIGITAL	Analog Input Status Local Forced Data bit Object 30 Variations 1,2,5 Object 32 Var(s) 1,2,3,4,5,7	R/O. Bit 4 of Status byte This device is in control 0=normal, 1=Forced
AI<n>.Stat.OverRange	DIGITAL	Analog Input Status Over Range bit Object 30 Variations 1,2,5 Object 32 Var(s) 1,2,3,4,5,7	R/O. Bit 5 of Status byte Analog is over/under range and has been clipped 0=normal, 1=Over Range

Address	Plant SCADA Type	DNP3 Type	Description
AI<n>.Stat.RefCheck	DIGITAL	Analog Input Status Reference Check bit Object 30 Variations 1,2,5 Object 32 Var(s) 1,2,3,4,5,7	R/O. Bit 6 of Status byte A/D reference signal unreliable. Value is unreliable 0=normal, 1=Error

**where:**

- n = Point index number (range from 0 to 2048)
- R/O = read only
- W/O = write only
- R/W = read or write

#### Monitoring Data Types - Analog Outputs

This table defines the Plant SCADA variable address required for the specified DNP3 data types.

Address	Plant SCADA Type	DNP3 Type	Description
AO<n>.Val - or - AO<n>.Val.Long	LONG	Analog Output Value (32 bit) Object 40 Variation 1	R/O Current value of Analog Output. (32bit value)
AO<n>.Val	LONG	Analog Output Value (16 bit) Object 40 Variation 2	R/O Current value of Analog Output. (16bit value)
AO<n>.Val.Float	REAL	Analog Output Value (float) Object 40 Variation 3	R/O Current value of Analog Output. (16bit value)
AO<n>.Val.QUAL	LONG	Analog Output Quality	OPC quality of point converted from value of flags.

**where:**

- n = Point index number (range from 0 to 2048)
- R/O = read only
- W/O = write only

- R/W = read or write

## Advanced monitoring data types - analog outputs

**Note:** The following data types are only recommended for experienced users who are familiar with the terminology and methods associated with the DNP 3.0 protocol.

Address	Plant SCADA Type	DNP3 Type	Description
AO<n>.Stat	BYTE	Analog Output Status Status byte Object 40 Variations 1,2,3	R/O. Status data related to Value
AO<n>.Stat.Online	DIGITAL	Analog Output Status Online bit Object 40 Variations 1,2,3	R/O. Bit 0 of Status byte If Offline, command will not be successful 0=Offline, 1=Online
AO<n>.Stat.Restart	DIGITAL	Analog Output Status Restart bit Object 40 Variations 1,2,3	R/O. Bit 1 of Status byte Originating field device has been restarted. 0=normal, 1=Restart
AO<n>.Stat.CommLost	DIGITAL	Analog Output Status Communication Lost bit Object 40 Variations 1,2,3	R/O. Bit 2 of Status byte If comms lost, command will not be successful 0=normal, 1=Lost
AO<n>.Stat.RemForced	DIGITAL	Analog Output Status Remote Forced Data bit Object 40 Variations 1,2,3	R/O. Bit 3 of Status byte Originating field device is in control 0=normal, 1=Forced
AO<n>.Stat.SelPend	DIGITAL	Analog Output Status Plant SCADA – Object Selected in RTU	R/O. Bit 5 of Status byte This device is in control 0=not selected, 1=Selected, waiting on OPERATE
AO<n>.Stat.WrtPend	DIGITAL	Analog Output Status	R/O.

Address	Plant SCADA Type	DNP3 Type	Description
		Plant SCADA – Select Initiated in RTU	Bit 6 of Status byte This device is in control 0=not initiated, 1=SELECT started
AO<n>.Stat.WrtFail	DIGITAL	Analog Output Status Plant SCADA – Problem in SELECT/OPERATE sequence	R/O. Bit 7 of Status byte This device is in control 0=normal, 1=Write unsuccessful
AO<n>.CtrlStat	BYTE	Analog Output Control Status Status byte Object 41 Variations 1,2,3	R/O. Status of Control Operation. 0=Request accepted, initiated or queued 1= Request rejected – OPERATE received after SELECT had already timed-out 2= Request rejected – OPERATE received without receiving matching SELECT 3= Request rejected – Formatting errors in control request (Select, Operate or Doperate) 4= Request rejected – Control Operation not supported for this bit 5= Request rejected – Queue full or point already active 6= Request rejected – Control hardware problems 7-127=Undefined.

where:

- n = Point index number (range from 0 to 2048)
- R/O = read only

- W/O = write only
- R/W = read or write

### Monitoring Data Types - Time and Date

**Note:** The following data types are only recommended for experienced users who are familiar with the terminology and methods associated with the DNP 3.0 protocol.

Address	Plant SCADA Type	DNP3 Type	Description
ReadTime	LONG	RTU Time & Date Object 50 Variation 1	R/W  Read RTU time from driver cache.  Write RTU time to driver cache.  (Forcing a time read request to device and then store the time from the reply to the cache).

where:

- n = Point index number (range from 0 to 2048)
- R/O = read only
- W/O = write only
- R/W = read or write

### Monitoring Data Types - IIN Flags

**Note:** The following data types are only recommended for experienced users who are familiar with the terminology and methods associated with the DNP 3.0 protocol.

Address	Plant SCADA Type	DNP3 Type	Description
IIN	INTEGER	The value of the Internal Indication bits retrieved from the last response (see note 2 below).  No Object	R/O  (see Note 1 below)
IIN.BadFunction	DIGITAL	Bit 0 of byte 2 of the Internal Indication bits (see Note 2 below).  Function code not implemented.  No Object	R/O  (see Note 1 below)

Address	Plant SCADA Type	DNP3 Type	Description
IIN.ObjectUnknown	DIGITAL	Bit 1 of byte 2 of the Internal Indication bits (see Note 2 below). Request object unknown. No Object	R/O (see Note 1 below)
IIN.OutOfRange	DIGITAL	Bit 2 of byte 2 of the Internal Indication bits (see Note 2 below). Parameters in qualifier, range or data fields are not valid or out of range. No Object	R/O (see Note 1 below)
IIN.BufferOvfl	DIGITAL	Bit 3 of byte 2 of the Internal Indication bits (see Note 2 below). Event or other buffers overflowed. No Object	R/O (see Note 1 below)
IIN.ExecutingIIN	DIGITAL	Bit 4 of byte 2 of the Internal Indication bits (see Note 2 below) Request understood but requested operation already executing. No Object	R/O (see Note 1 below)
IIN.ConfigIIN	DIGITAL	Bit 5 of byte 2 of the Internal Indication bits (see Note 2 below) Configuration in outstation corrupt. No Object	R/O (see Note 1 below)
IIN.AllStations	DIGITAL	Bit 0 of byte 1 of the Internal Indication bits (see Note 2 below). All stations message received. No Object	R/O (see Note 1 below)

<b>Address</b>	<b>Plant SCADA Type</b>	<b>DNP3 Type</b>	<b>Description</b>
IIN.Class1	DIGITAL	Bit 1 of byte 1 of the Internal Indication bits (see Note 2 below). Class 1 data available. No Object	R/O (see Note 1 below)
IIN.Class2	DIGITAL	Bit 2 of byte 1 of the Internal Indication bits (see Note 2 below). Class 2 data available. No Object	R/O (see Note 1 below)
IIN.Class3	DIGITAL	Bit 3 of byte 1 of the Internal Indication bits (see Note 2 below). Class 3 data available. No Object	R/O (see Note 1 below)
IIN.TimeIIN	DIGITAL	Bit 4 of byte 1 of the Internal Indication bits (see Note 2 below). Time synchronization required from the master. No Object	R/O (see Note 1 below)
IIN.LocalIIN	DIGITAL	Bit 5 of byte 1 of the Internal Indication bits (see Note 2 below). Some or all of the outstations digital output points are in local state. No Object	R/O (see Note 1 below)
IIN.TroubleIIN	DIGITAL	Bit 6 of byte 1 of the Internal Indication bits (see Note 2 below) Abnormal condition exists in the outstation. No Object	R/O (see Note 1 below)
IIN.Restart	DIGITAL	Bit 7 of byte 1 of the Internal Indication bits	R/O (see Note 1 below)

Address	Plant SCADA Type	DNP3 Type	Description
		(see Note 2 below). Device restart. No Object	

**Where:**

- n = Point index number (range from 0 to 2048)
- R/O = read only
- W/O = write only
- R/W = read or write

**Note:**

1. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
2. Not every transition of the IIN will be visible using this tag since the driver reacts to the IIN bits. Also, IIN error consistent reflect the response to a single request therefore may only be briefly stored.  
For instance, if the IIN bits indicate time action is required, and the driver is configured to do so, then the action will occur, and the time action bit become zero before it can be displayed.  
The IIN tag is useful for viewing IIN conditions that are global to the device (not specific to a request) and are not being responded to by the driver (for instance, the Local bit).

**Control Data Types - Binary Outputs**

This table defines the Plant SCADA variable address required for the specified DNP3 data types.

Address	Plant SCADA Type	DNP3 Type	Description
BO<n>.Select.Latch	DIGITAL	Binary Output Control Select (Latch Mode) Object 12 Variation 1	W/O. Selects point and mode of operation (the driver will automatically send out the corresponding operate command). Maintains output in specified state.
BO<n>.Select.Pulse	DIGITAL	Binary Output Control Select (Pulse Mode) Object 12 Variation 1	W/O. Selects point and mode of operation (the driver will automatically send out the corresponding operate command). Only Pulse ON supported.
BO<n>.Select.TC	DIGITAL	Binary Output Control	W/O.

Address	Plant SCADA Type	DNP3 Type	Description
		Select (TC Mode) Object 12 Variation 1	Selects point and mode of operation (the driver will automatically send out the corresponding operate command). Trip and Close both use the ONTIME parameter. 0=Trip breaker, 1=Close breaker.
BO<n>.Operate.Latch	DIGITAL	Binary Output Control Operate (Latch Mode) Object 12 Variation 1	W/O. Activates previously selected operation. Maintains output in specified state.
BO<n>.Operate.Pulse	DIGITAL	Binary Output Control Operate (Pulse Mode) Object 12 Variation 1	W/O. Activates previously selected operation. Only Pulse ON supported
BO<n>.Operate.TC	DIGITAL	Binary Output Control Operate (TC Mode) Object 12 Variation 1	W/O. Activates previously selected operation. Trip and Close both use the ONTIME parameter. 1=Close breaker, 0=Trip breaker.
BO<n>.SelectOnly.Latch	DIGITAL	Binary Output Control Select (Latch Mode) Object 12 Variation 1	W/O. Selects point and mode of operation but does not operate. Maintains output in specified state.
BO<n>.SelectOnly.Pulse	DIGITAL	Binary Output Control Select (Pulse Mode) Object 12 Variation 1	W/O. Selects point and mode of operation but does not operate. Only Pulse ON supported. Pulse OFF may be supported in future release.

Address	Plant SCADA Type	DNP3 Type	Description
BO<n>.SelectOnly.TC	DIGITAL	Binary Output Control Select (TC Mode) Object 12 Variation 1	W/O. Selects point and mode of operation but does not operate. Trip and Close both use the ONTIME parameter. 0=Trip breaker, 1=Close breaker.
BO<n>.DOperate.Latch (See Note below)	DIGITAL	Binary Output Control Direct Operate (Latch Mode) Object 12 Variation 1	W/O. Selects and activates point operation. Maintains output in specified state.
BO<n>.DOperate.Pulse (See Note below)	DIGITAL	Binary Output Control Direct Operate (Pulse Mode) Object 12 Variation 1	W/O. Selects and activates point operation. Only Pulse ON supported.
BO<n>.DOperate.TC (See Note below)	DIGITAL	Binary Output Control Direct Operate (TC Mode) Object 12 Variation 1	W/O. Selects and activates point operation. Trip and Close both use the ONTIME parameter. 1=Close breaker, 0=Trip breaker.
BO<n>.Configure.Ontime	LONG	Binary Output Control Configure Ontime Object 12 Variation 1	W/O. Time in milliseconds of momentary contact closure. (Used in Pulse ON, Trip and Close modes).
BO<n>.CancelSel	DIGITAL	Binary Output Control Cancel Current SelectOnly	W/O. Clear OPERATE status bits and cancel the SELECT operation to the RTU.

**where:**

- n = Point index number (range from 0 to 2048)
- R/O = read only
- W/O = write only

- R/W = read or write

---

**Note:** DNP (Subset level 1) supports a ‘Direct Operate without acknowledgment’, but as this does not verify the success of the command, only ‘Direct Operate’ is currently implemented.

---

### Control Data Types - Counters

This table defines the Plant SCADA variable address required for the specified DNP3 data types.

Address	Plant SCADA Type	DNP3 Type	Description
C.Freeze (See Note 1 below)	DIGITAL	Counter Freeze Command Object 20 Variations 1..8	W/O. Copy all counters to a freeze buffer.
C.FreezeClear (See Note 2 below)	DIGITAL	Counter Freeze Command Object 20 Variations 1..8	W/O. Copy all counter to a freeze buffer, and then clear all counters to 0.

**where:**

- n = Point index number (range from 0 to 2048)
- R/O = read only
- W/O = write only
- R/W = read or write

---

**Note:**

1. DNP (Subset level 2) supports an ‘Immediate Freeze without acknowledgment’, but as this does not verify the success of the command, only ‘Immediate Freeze’ is currently implemented.
  2. DNP (Subset level 2) supports a ‘Freeze and Clear without acknowledgment’, but as this does not verify the success of the command, only ‘Freeze and Clear’ is currently implemented.
- 

### Control Data Types - Analog Outputs

This table defines the Plant SCADA variable address required for the specified DNP3 data types.

Address	Plant SCADA Type	DNP3 Type	Description
AO<n>.Select.Long	LONG	Analog Output Control (32bit) Select Object 41 Variation 1	W/O. Selects point to be controlled (the driver will automatically send out the corresponding operate command). (32bit value).
AO<n>.Select	INTEGER	Analog Output Control	W/O.

Address	Plant SCADA Type	DNP3 Type	Description
		(16bit) Select Object 41 Variation 2	Selects point to be controlled (the driver will automatically send out the corresponding operate command). (16bit value).
AO<n>.Select.Float	REAL	Analog Output Control (Float) Select Object 41 Variation 3	W/O. Selects point to be controlled (the driver will automatically send out the corresponding operate command). (float value).
AO<n>.SelectOnly.Long	LONG	Analog Output Control (32bit) Select Object 41 Variation 1	W/O. Selects point to be controlled but does not operate. (32bit value).
AO<n>.SelectOnly	INTEGER	Analog Output Control (16bit) Select Object 41 Variation 2	W/O. Selects point to be controlled but does not operate. (16bit value).
AO<n>.SelectOnly.Float	REAL	Analog Output Control (Float) Select Object 41 Variation 3	W/O. Selects point to be controlled but does not operate. (float value).
AO<n>.Operate.Long	LONG	Analog Output Control (32bit) Operate Object 41 Variation 1	W/O. Activates control operation on previously selected point. (32bit value).
AO<n>.Operate	INTEGER	Analog Output Control (16bit) Operate Object 41 Variation 2	W/O. Activates control operation on previously selected point. (16bit value).

Address	Plant SCADA Type	DNP3 Type	Description
AO<n>.Operate.Float	REAL	Analog Output Control (Float) Operate Object 41 Variation 3	W/O. Activates control operation on previously selected point. (float value).
AO<n>.DOperate.Long (See Note below)	LONG	Analog Output Control (32bit) Direct Operate Object 41 Variation 1	W/O. Selects and activates control operation on point. (32bit value).
AO<n>.DOperate (See Note below)	INTEGER	Analog Output Control (16bit) Direct Operate Object 41 Variation 2	W/O. Selects and activates control operation on point. (16bit value).
AO<n>.DOperate.Float (See Note below)	REAL	Analog Output Control (Float) Direct Operate Object 41 Variation 3	W/O. Selects and activates control operation on point. (float value).
AO<n>.CancelSel	DIGITAL	Analog Output Control Cancel Current . SelectOnly	W/O. Clear OPERATE status bits and cancel the SELECT operation to the RTU.

where:

- n = Point index number (range from 0 to 2048)
- R/O = read only
- W/O = write only
- R/W = read or write

---

**Note:** DNP (Subset level 1) supports a ‘Direct Operate without acknowledgment’, but as this does not verify the success of the command, only ‘Direct Operate’ is currently implemented.

---

#### Control Data Types - Time and Date

---

**Note:** The following data types are only recommended for experienced users who are familiar with the terminology and methods associated with the DNP 3.0 protocol.

---

Address	Plant SCADA Type	DNP3 Type	Description
WriteTime	DIGITAL	RTU Time & Date Object 50 Variation 1	W/O. Write '1' to synchronise RTU time with Plant SCADA time.

where:

- n = Point index number (range from 0 to 2048)
- R/O = read only
- W/O = write only
- R/W = read or write

## Control Data Types - Utilities

---

**Note:** The following data types are only recommended for experienced users who are familiar with the terminology and methods associated with the DNP 3.0 protocol.

Address	Plant SCADA Type	DNP3 Type	Description
ColdRestart	DIGITAL	Cold Restart Command No Object	W/O. Instruct the Outstation to perform a complete restart of the user application. The only valid value to write is 1. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.

where:

- n = Point index number (range from 0 to 2048)
- R/O = read only
- W/O = write only
- R/W = read or write

## Global Data Types

---

**Note:** The following data types are only recommended for experienced users who are familiar with the terminology and methods associated with the DNP 3.0 protocol.

Address	Plant SCADA Type	DNP3 Type	Description
EventQueSize	LONG	N/A. No Object	R/O Current size of the event queue.

where:

- n = Point index number (range from 0 to 2048)
- R/O = read only
- W/O = write only
- R/W = read or write

### Polling Regime Data Types

**Note:** The following data types are only recommended for experienced users who are familiar with the terminology and methods associated with the DNP 3.0 protocol.

Address	Plant SCADA Type	DNP3 Type	Description
PollPeriod	LONG	Adjust the Polling Period  No Object	R/W  If the Citect.ini parameter 'EnablePollPeriodWrites' is set to TRUE (1), then this tag can be used to adjust the Event Polling Period.  This tag can be used to display the current Event Polling Period regardless of the state of 'EnablePollPeriodWrites'.  This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
EnablePoll	INTEGER	Returns if Polling enabled for reads  No Object	R/W  Returns if the unit has polling enabled. A write of '0' will turn polling off, '1' will re-enable polling.  This address does not relate to I/O data internal to the device, but relates

<b>Address</b>	<b>Plant SCADA Type</b>	<b>DNP3 Type</b>	<b>Description</b>
			to data internal to the driver, or an action to be performed by the driver.
ReinitPollSequence	DIGITAL	Force the re-initialization of the Polls sequence.  No Object	W/O  Writing a '1' to this tag will force the driver to re-initialize the poll sequence of all devices. The poll sequence will restart from the Integrity poll when this command is received.  This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
ForceIntegrityPoll	DIGITAL	Force an integrity poll  Object 60 Variation 1 Object 60 Variation 2 Object 60 Variation 3 Object 60 Variation 4	W/O.  Writing a '1' to this tag will force the driver to execute an Integrity poll of the device (in addition to the routine polling that it automatically undertakes).  This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
ForceEventPoll	DIGITAL	Force an integrity poll  Object 60 Variation 2 Object 60 Variation 3 Object 60 Variation 4	W/O.  Writing a '1' to this tag will force the driver to execute an Event poll of the device (in addition to the routine polling that it automatically undertakes).  This address does not relate to I/O data internal

Address	Plant SCADA Type	DNP3 Type	Description
			to the device, but relates to data internal to the driver, or an action to be performed by the driver.
ForceClass1Poll	DIGITAL	Force an integrity poll Object 60 Variation 2	W/O. Writing a '1' to this tag will force the driver to execute a poll for class 1 data from the device (in addition to the routine polling that it automatically undertakes).
ForceClass2Poll	DIGITAL	Force an integrity poll Object 60 Variation 3	W/O. Writing a '1' to this tag will force the driver to execute a poll for class 2 data from the device (in addition to the routine polling that it automatically undertakes).
ForceClass3Poll	DIGITAL	Force an integrity poll Object 60 Variation 4	W/O. Writing a '1' to this tag will force the driver to execute a poll for class 3 data from the device (in addition to the routine polling that it automatically undertakes).
DisableUnsolicited	DIGITAL	Disable all Unsoliciteds Object 60 Variation 2 Object 60 Variation 3 Object 60 Variation 4	W/O. Writing a '1' to this tag will instruct the unit not to broadcast any more unsolicited responses. It is W/O, as there is no way of determining if unsoliciteds were disabled before Plant SCADA startup. This address does not

Address	Plant SCADA Type	DNP3 Type	Description
			<p>relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.</p> <p><b>Note:</b> This a Level 3 feature.</p>
EnableUnsolicited	DIGITAL	Enable all Unsoliciteds Object 60 Variation 2 Object 60 Variation 3 Object 60 Variation 4	<p>W/O.</p> <p>Writing a '1' to this tag will allow the unit to broadcast unsolicited responses. The Classes enabled are determined by the INI parameter [DNPR]AutoUnsolicitedEnableDefault or [&lt;unit&gt;] AutoUnsolicitedEnable. It is W/O, as there is no way of determining if unsoliciteds were enabled before Plant SCADA startup.</p> <p>This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.</p> <p><b>Note:</b> This a Level 3 feature.</p>
TimeToNextPoll	LONG	Time to go to next Integrity poll No Object	<p>R/O Seconds.</p> <p>Will count down from the set Integrity poll time, and may go -ve if the Integrity poll has been delayed.</p> <p>This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.</p>
TimeToNextClass1Poll	LONG	Time to go to next class 1	R/O Seconds.

Address	Plant SCADA Type	DNP3 Type	Description
		poll No Object	Will count down from the set class 1 poll time and may go –ve if the class 1 poll has been delayed. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
TimeToNextClass2Poll	LONG	Time to go to next class 1 poll No Object	R/O Seconds. Will count down from the set class 2 poll time and may go –ve if the class 2 poll has been delayed. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
TimeToNextClass3Poll	LONG	Time to go to next class 1 poll No Object	R/O Seconds. Will count down from the set class 3 poll time and may go –ve if the class 3 poll has been delayed. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
Polls	INTEGER	Count of all successful polls (Forced and periodic). No Object	R/W Count of successful Integrity, Event, class 1, class 2 and class 3 polls of the device. Write ‘1’ to reset counter. This address does not relate to I/O data internal to the device, but relates to data internal to the

<b>Address</b>	<b>Plant SCADA Type</b>	<b>DNP3 Type</b>	<b>Description</b>
			driver, or an action to be performed by the driver.
FailedPolls	INTEGER	Count of all unsuccessful polls (Forced and periodic). No Object	R/W Count of unsuccessful Integrity, Event, class 1, class 2 and class 3 polls of the device (ie failure of full timeout sequence). Write '1' to reset counter. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
Ipolls	INTEGER	Count of successful Integrity polls (Forced and periodic). No Object	R/W Count of successful Integrity polls of the device. Write '1' to reset counter. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
FailedIPolls	INTEGER	Count of unsuccessful Integrity polls (Forced and periodic). No Object	R/W Count of unsuccessful Integrity polls of the device (i.e. failure of full timeout sequence). Write '1' to reset counter. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
Epolls	INTEGER	Count of successful Event polls (Forced and periodic).	R/W Count of successful Event polls of the device. Write

Address	Plant SCADA Type	DNP3 Type	Description
		No Object	'1' to reset counter. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
FailedEpolls	INTEGER	Count of unsuccessful Event polls (Forced and periodic). No Object	R/W Count of unsuccessful Event polls of the device (i.e. failure of full timeout sequence). Write '1' to reset counter. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
UnsolicitedRxd	INTEGER	Count of Unsoliciteds received. No Object	R/W Count of valid Unsolicited Responses received by the device. Write '1' to reset counter. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
LateRefresh	DIGITAL	Late cache refresh for device No Object	R/O Indicates when a device poll has been delayed, resulting in a late cache refresh. Refer to Citect.ini parameter DataLag for further details. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be

Address	Plant SCADA Type	DNP3 Type	Description
			performed by the driver.
RequestEventRefresh	INTEGER	Force Event Refresh for device No Object	R/W  Writing 1 will cause the unit to refresh all event data the next integrity poll. A read reads this state.  <b>Note:</b> This value will be set if the [dnpr]RefreshEventsOnStartup=1 was set.  This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
RTUEvents	LONG	RTU Events received No Object	R/O  RTU events PER PORT received from the field device.  This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
RTUStatus	DIGITAL	RTU Online status No Object	R/O  Internal online status of a unit. Of limited use as a tag on a unit as if the unit was offline, you could not read the tag anyway.  This tag has been superceded by the UnitActiveState.X tags.  This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.

**where:**

- n = Point index number (range from 0 to 2048)
- R/O = read only
- W/O = write only
- R/W = read or write

## Miscellaneous Data Types

**Note:** The following data types are only recommended for experienced users who are familiar with the terminology and methods associated with the DNP 3.0 protocol.

Address	Plant SCADA Type	DNP3 Type	Description
DebugReport	INTEGER	Generate a debug report for the device. No Object	W/O The debug info are written in the driver log file. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.

**where:**

- n = Point index number (range from 0 to 2048)
- R/O = read only
- W/O = write only
- R/W = read or write

## Projects Using Super Genies

To use Super Genies with digital tags, you will need to modify your project settings to allow blocking of requests up to 16 bits (default is 8 bits). To do this, go to the topic [Data Point and Cache Parameters](#) and make the modifications outlined in the description of the [DNPR]IsItemisedRequest parameter.

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

- Do not under any circumstances change or remove any of the undocumented citect.ini parameters.
- Before deleting sections of the citect.ini file, confirm that no undocumented parameters will be deleted.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

You will also need to:

1. Add the following to your Citect.ini file (in the [DNPR] section):
  - Block = 2
  - MaxBits = 16
  - IsItemisedRequest = 0
2. Modify the protodir.db file from BIT\_BLOCK=8, MAX\_LENGTH=8 to BIT\_BLOCK=16, MAX\_LENGTH=16 and copy to the Include, System and your project directory (exists under the User directory of your Plant SCADA installation).
3. Recompile your project (with the incremental compile option disabled).

---

**Note:** The MAX\_LENGTH value above is adjustable by the user to 2048 bits

---

## Hints and Tips

### Startup considerations

In a site with many RTUs, it is useful to consider staggering port startup using the parameter [DNPR]StartupStaggerDelay to spread out the load.

Also, it is a good idea to use the [<unit>]PollPeriodOffset parameter to stagger when periodic polls are issued, especially where many units are coming through one port. For example:

- Unit 1: 5
- Unit 2: 10
- Unit 3: 15

This would send polls five seconds after the default time for Unit 1, 10 seconds after for Unit 2, and 15 seconds after for Unit 3.

### Resource release at shutdown

If StopUnit() and Stop/CloseChannel() code is not executed when a unit or channel is offline, the system may become inoperable.

To avoid this, set the following Citect.ini parameters:

```
[IoServer]
AlwaysCallStopChannel=1
AlwaysCallStopUnit=1
```

### Unit online/offline

The following Citect.ini parameter settings are recommended to address unit online/offline issues:

```
[DNPR] DebugCategory = PROT|STATE|DCB
```

[DNPR] DebugLevel =ALL

Also use drivertraces with a mask of c0f, (e.g. "drivertrace mask=c0f").

## Data integrity

- If repeatable, then make the page refresh time artificially long (using page properties), for example, every 5 seconds (5000). This will minimize the trace storm if a page has many tags.
- Disable alarms and trends to isolate a trace to just one source of logic, i.e. a page refresh or alarm scan.
- A kernel "probe" window will show tag requests to and from the client, but not necessarily to the driver. Probe results may come from the I/O server cache or an I/O server on another PC. This is only for Plant SCADA v6.10 and before.
- A kernel "probe" window will show tag requests to and from the client, but not necessarily to the driver. Probe results may come from the I/O server cache or an I/O server on another PC. This is only for Plant SCADA v6.10 and before.
- Kernel "drivertrace xxxx" will show requests to/from the driver on the local I/O server. These traces will also show the returned RAW value from the driver. Remember that the client will scale this value if engineering units are defined.
- Useful INIs for data issues are [DNPR] DebugCategory = PROT|CACHE|DCB|MISC, [DNPR] DebugLevel =ALL or DriverTraces.

## Maintenance Page

With more complex systems, it is recommended you have a maintenance page that allows you to enable or disable units and force integrity polls.

This page could display poll counts and have a couple of sample data points per unit. Thus during commissioning or later, allowing you to test the data path to various units. This is particularly true in a multidrop situation. Verifying field communications is very helpful when serial links exist.

## Advanced Configuration and Maintenance

This section of the DNPR Driver Help covers areas that are recommended for experienced users who are familiar with the terminology and methods associated with the DNP 3.0 protocol.

- [Customize a Project using Citect.ini Parameters](#)
- [DNPR Redundancy](#)
- [Data Acquisition within DNP3](#)
- [Device Polling Regime](#)
- [Device Recovery](#)
- [SELECT and OPERATE](#)
- [Virtual Units](#)

### Customize a Project using Citect.ini Parameters

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

- Do not under any circumstances change or remove any citect.ini parameters without an informed understanding of the possible implications of your actions.
- Do not delete sections of the citect.ini file without an informed understanding of the possible implications of your actions.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** At times during the life cycle of a driver, circumstances may call for the creation of specific citect.ini parameters that remain undocumented. Always seek the advice of Technical Support regarding the possible implementation of undocumented parameters before you make any changes to a citect.ini file.

The DNPr parameters are grouped into the following categories:

- [Basic Parameters](#)  
The standard set of protocol parameters.
- [Logging/Diagnostic Parameters](#)  
Parameters that can assist with troubleshooting.
- [Driver-specific Parameters](#)  
Parameters specific to the DNPr driver.
- [Peer Redundancy Parameters](#)  
These parameters need only be set if you are using a redundant DNPr system.
- [Port-based Parameters](#)  
These parameters are used to manipulate port control.
- [Event Processing Parameters](#)  
These parameters control how events are handled.
- [Polling Regime Parameters](#)  
These parameters are used to manipulate polling.
- [Communication Parameters](#)  
These parameters configure the data link to devices.
- [Data Point and Cache Parameters](#)  
These parameters control variations of behavior to some data types, as well as data cache behavior.
- [Miscellaneous Parameters](#)

**Note:** Some parameters can be used to configure behavior that will only apply to units within the same device group. These are represented in the following lists using [<unit>] as the defined Citect.ini section. See [Device Group Parameters](#) for more information.

**Basic Parameters**

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

The DNPR driver supports the following basic parameters:

Parameter	Allowable values	Default value	Description
[DNPR]Block	1-256	1	Dynamic I/O server blocking is adjustable by the user up to 256 bytes. e.g. [DNPR]Block=256 This allows DCBs to be blocked together in the I/O server and make fewer requests down to the driver.
[DNPR]ConnectionTimeout	1000 to 65534 (ms)	8000	The maximum amount of time to allow the low level transport for the driver to attempt a connection.
[DNPR]Delay	0-300 (ms)	0	The period (in milliseconds) to wait between receiving a response and sending the next command.
[DNPR]EventPollPeriodDefault	0 to 4,000,000 (seconds)	60 Sec	Default period to poll devices for event class data. Integrity poll will poll for all classes (event and static data), and occurs after every EventPollRatioDefault event polls.  If the value is set to zero then the polling mode in the Citect.ini parameter [DNPR]ResponseModeDefault will be disabled. The

Parameter	Allowable values	Default value	Description
			<p>device will stay offline until an integrity poll is forced out or an unsolicited message from the device is coming in.</p> <p>See "Device Polling Regime" for an example.</p> <p>This parameter can be over-written by the parameter [&lt;unit&gt;]EventPollPeriod.</p> <p><b>Note:</b> This INI parameter can be overwritten if the "Minimum Update Time" field for the I/O Device is greater than 0. (This option is only available with Plant SCADA from version 7.20.)</p>
[DNPR]MaxPending	0-32	16	The maximum number of pending commands that the driver holds ready for immediate execution.
[DNPR]Poltime	50-1000 (ms)	200	<p>The frequency with which messages are processed to/from the driver to the RTUs.</p> <p>It is recommended this value not be changed.</p>
[DNPR]Retry	0-8	1	<p>Default number of application layer retries for all I/O devices as well as physical transport writes allowed before a channel is placed offline.</p> <p>This parameter can be over-written by the parameter [&lt;unit&gt;]Retry.</p>

Parameter	Allowable values	Default value	Description
[DNPR]SCADAAddress	0-65534	3	Defines the DNP Address for Plant SCADA. If a value for this parameter is not provided, a popup box will appear at startup. This parameter can be over-written by the parameter [<unit>]SCADAAddress.
[DNPR]Timeout	500-32767 (ms)	5000	Default application layer timeout (in milliseconds) for all I/O devices. This parameter can be over-written by the parameter [<unit>]Timeout.
[DNPR]WatchTime	5-128 Sec	10	The frequency (in seconds) with which the I/O server checks the communications link to an I/O device if the device is offline.

## Logging/Diagnostic Parameters

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

These parameters are read every few seconds by the driver. This means the values can be changed in real time, for example, you can turn logging on or off.

Parameter	Allowable values	Default value	Description
[DNPR]DebugLevel	Any combination of the following, separated by a pipe character ( ):  WARN	Empty string (no logging)	The trace level used for the log file.

Parameter	Allowable values	Default value	Description
	ERROR TRACE DEBUG ALL See <a href="#">Logging</a> for examples.		
[DNPR]DebugCategory	Any combination of the following, separated by a pipe character ( ): PROT = protocol level debug CACHE = tag cache data STATE = unit state transitions EVENT = events cache data DCB = front end debug (that is, the I/O server) BEND = peer communication MISC = any other debugging ALL = all of the above See <a href="#">Logging</a> for examples.	Empty string (no logging)	The categories used for the log file:

## See Also

[Debugging Messages](#)

## Driver-specific Parameters



### UNINTENDED EQUIPMENT OPERATION

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Parameter	Allowable values	Default value	Description
[<unit>]SCADAAddress	0-65534	[DNPR]SCADAAddress	This parameter has the same functionality as the

Parameter	Allowable values	Default value	Description
			driver-level DNPR parameter [DNPR]SCADAAddress, but only applies to units in this device group. It allows you to specify multiple master devices, instead of only one (e.g. Plant SCADA).
[<unit>]Timeout	500-32767 ms	5000	Default application layer timeout (in milliseconds) for the specified I/O device group. This parameter has the same functionality as the driver-level DNPR parameter [DNPR]Timeout.
[<unit>]Retry	0 - 8	1	Default number of application layer retries for the specified device group, as well as physical transport writes allowed before a channel is placed offline. This parameter has the same functionality as the driver-level DNPR parameter [DNPR]Retry.
[DNPR] TimeoutExtensionDefault [<unit>] TimeoutExtension	0-2,000,000,000 ms	0	Default extension to be added to the standard 'timeout' parameter. This extension is only required if a timeout greater than 32000 milliseconds is needed.
[DNPR] OnTimeBODNPDefault [<unit>] OnTimeBODDeviceDefault	0-2,000,000,000 ms	1000	Default value to be used by theOnTimeBODDeviceDefault device level .ini parameters. The

Parameter	Allowable values	Default value	Description
			OnTimeBODDeviceDefault in turn defines the default value for momentary Binary Output closure times for that device (unit). Each Binary Output can have its closure time individually configured.

## Peer Redundancy Parameters

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

The following [DNPR] parameters need only be set if you are using a redundant system:

Parameter	Allowable values	Default value	Description
[DNPR] ActivateIfPeerIsActiveOffline <unit> ActivateIfPeerIsActiveOffline	0 – Disable 1 – Enable	1	This parameter is used in a scenario where one of the redundant I/O devices is active and can open the TCPIP connection to the RTU, but stays offline as the driver does not get any response for the DNP3 messages it sends to the RTU.  If this is the case and the parameter is set to 1, the driver performs the switchover to the redundant I/O device peer and makes it active giving that I/O device a chance to communicate to the RTU.  If this behavior is not required, this parameter should be set to 0.

[DNPR] EnablePeerRedundancy	0 – Disable peer driver redundancy. 1 – Enable peer driver redundancy.	0	Enable the peer redundancy process. Only 1 peer driver is supported.
[DNPR]LocalAddress	Valid IP address.	127.0.0.1	The IP address of yourself for listening on.
[DNPR]LocalPort	Port.	5000	Valid Port you want to listen on.
[DNPR]MaxPeerOutQueueSize	1 to 100,000	10,000	If DNPR redundancy is configured, the driver replicates the value updates received from the device to the redundant peer. This provides the peer driver with the latest values in case a switchover occurs.  This parameter controls the maximum number of value updates which the driver stores in the queue before sending them to the peer. Limiting the queue size may be required if the driver receives more values updates than it can replicate.  <b>Note:</b> If the parameter value is greater than 100,000, it will be reset to 100,000.
[DNPR]PeerAddress	Valid IP address.	None	The IP address of your peer driver.
[DNPR]PeerPort	Port	5200	Any Valid Peer Destination Port.
[IOSERVER] WatchdogPrimary	0 – Disable 1 – Enable	0	Should be set when peer redundancy is in use. This helps ensure that the check that your unit is still online occurs for every unit.

			Not only does this enable the unit's offline to be noticed if NO reads are happening to that unit, but enables the driver to minimize #COMs on changeover by reporting offline in this check, and not on actual read requests.
[DNPR] MaxMsgPerPollTime	1 to 4294967295	500	The maximum number of peer cache updates to be processed per call to Cpu() which is controlled by the "PollTime" parameter.
[DNPR] PeerCommRetryCount	1 to 10	3	Retries allowed on peer messages before the peer communications are considered down.
[DNPR] PeerKeepAliveTime	1 to 65535	30000	Ping time between pairs to check "they are there".
[DNPR]PeerTimeout	1 to 65535 ms	4000 ms	Timeout time in getting a response back from your peer.
[DNPR] UnitUsrpModeDefault [<unit>]UnitUsrpMode	0 means that an integrity poll will be issued  1 means that an RTU time read will be issued.  2 means nothing will be done.  3 means that a static (Class0) Poll will be issued.	3	Defines the default behavior on usurp of a unit.  The online status of the RTU is replicated between servers. So regardless of what the value is set to, the unit will initially have the same status as it had on the other server.  The polling sequence is also replicated, so the next periodic poll on the unit will occur at the same time it would have if the usurp had not occurred (unless a value of 0 is used, which results in the poll period being

			reset).
			<b>Note:</b> No event will be generated if the usurp mode is 3 (Static Poll) and the value in the device is different than the value in the driver cache.

## Port-based Parameters

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

This parameters are used to manipulate port control.

Parameter	Allowable Values	Default Values	Description
[DNPR_<Port>] Exclusive Access	0 – Non-exclusive access 1 – Exclusive access	0	When set, the port and units adopt exclusive access mode. In this mode, only one server is physically connected to the port.  When a unit is usurped, the other port connects after the originating port disconnects.

## Event Processing Parameters

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

These parameters are used to manipulate the handling of events.

Parameter	Allowable Values	Default Values	Description
[DNPR]	0 – Disable the digital/	1	Enables/disables event

EventProcessingEnable	<p>analog events to be pushed to the alarm/trend server.</p> <p>1 – Enable the digital/analog events to be pushed to the alarm/trend server.</p>		<p>processing.</p> <p>May be useful during testing.</p>
[DNPR] EventAtStartupDisable	<p>0 – Events are generated on the initial integrity poll.</p> <p>1 – Stop the initial integrity poll from generating events.</p>	0	<p>Help prevent an event that is not initialized from cache coming through summary system.</p>
[DNPR] PushIntegrityStaticsAsEvents	<p>0 - Do not enable generation of alarm/trend events from static data.</p> <p>1 - Enable generation of alarm/trend events from static data.</p>	1	<p>Enables generation of alarm/trend events from static data (with the timestamp set to the current time on the SCADA machine).</p> <p>It is recommended that this parameter is turned off (set to 0) when the DNP slave should be solely responsible for event generation. This may be desired when there is a time disparity between when the RTU acquires data and when it is received by SCADA, in which case the event timestamps would be incorrect.</p>
[DNPR] UTCTimeSyncDefault  [<unit>] UTCTimeSync	<p>0 – Local time used</p> <p>1 – UTC time used</p>	0	<p>Indicate if the stamp time of the events sent by the device is whether UTC or local time.</p> <p>The default is local time.</p>
[DNPR] RefreshEventsOnStartup	<p>0 – No events refresh</p> <p>1 – Event refresh</p>	0	<p>This Citect.ini parameter should only be set on one server - not on both peer sides. When this server is restarted, it will set the refresh tag. If this server</p>

			<p>is the active server for a unit, it will regenerate events.</p> <p>If another server is the active server for a unit, it asks the redundant server to regenerate events. So it is consistent that only active unit will generate events.</p> <p>This may be needed in some systems to force fresh alarms or trends on all data points to help ensure that the system starts in the correct state. The events are generated when receiving a response to an integrity poll or static poll.</p> <p>This parameter should not be set to 1 if the parameters [DNPR]IssueIntegrityPollOnUnitRecovery or [&lt;unit&gt;]IssueIntegrityPollOnUnitRecovery are set to 0. Otherwise, the events that were time-stamped before the timestamp of the events generated by the static poll will be ignored.</p>
--	--	--	--

## Communication Parameters



### UNINTENDED EQUIPMENT OPERATION

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

The DNPR driver supports the following communication parameters.

Parameter	Allowable Values	Default Values	Description
[DNPR]StartupStaggerDelay	0 to 30000 ms	0	Time between allowing the first unit on each port to come online at startup. This is to avoid large amounts of startup traffic.
[DNPR]MaxDNPTimeOuts [<unit>]MaxTimeOuts	0 to 65535	1	Number of unit timeouts before returning a ‘unit offline’ reply.  Unit timeouts can result from both the I/O server requests and driver polls (timeouts may result from the I/O server request or driver poll). Refer also to parameter [DNPR]RapidOfflineDetectionDefault.
[DNPR] RapidOfflineDetectionDefault [<unit>]RapidOfflineDetection	0 – Rapid detection is disable 1 – Rapid detection is enable	1	If RapidOfflineDetectionDefault is set to 0 (false), then a unit will only be placed offline due to a detected communication breakdown during regular polling requests.  Unsuccessful I/O server requests (e.g. Tag Writes), will not be taken into account when determining a units online status.  This parameter allows the driver to be configured to either provide rapid detection of offline status, or to provide a consistent interval of time before a unit is marked as offline.
[DNPR]MinorIINSelectErrorFilter	Refer to description	0x0000	This parameter allows the user to turn on the ability of the driver to generate a driver error (CDNP_ERR_IIN_ERROR)

			<p>to notify the user of several IIN (Internal indications bits) errors. These IIN errors are normally not considered serious, however in some circumstances, the IIN errors may need to be treated as if the unit is not operating correctly. Any combination of the following IIN errors can be detected by OR-ing the selected IIN error bits together to create the MinorIINSelectErrorFilter value.</p> <p>It should be noted that the errors are only notified under certain conditions. The conditions are noted in the Driver Specific errors section.</p> <p>The IIN bit errors that can be OR-ed together for selection purposes are shown below. The meanings of the terms are also described in the Driver-specific errors section.</p> <p>BAD_FUNCTION_IIN 0x0001 OBJECT_UNKNOWN_IIN 0x0002 OUT_OF_RANGE_IIN 0x0004 BUFFER_OVFL 0x0008 EXECUTING_IIN 0x0010 CONFIG_IIN 0x0020 LOCAL_IIN 0x2000 TROUBLE_IIN 0x4000 0x0000 means the error filter is off.</p> <p>Be reminded that the user can define tags to determine the status of</p>
--	--	--	--

			<p>the IIN flags returned from the device. Be reminded also that CDNP_ERR_IIN_ERROR is a severe level driver error and will force the unit offline and generate #COMs on the screen.</p>
[DNPR]MinorIINSelectFilter	Refer to description	0xFFFF	<p>This parameter allows the user to turn on the ability of the driver to generate a driver error (CDNP_WARN_IIN_ERROR) to notify the user of several IIN (Internal indications bits) errors. These IIN errors are normally not considered serious. Any combination of the following IIN errors can be detected by OR-ing the selected IIN error bits together to create the MinorIINSelectFilter value.</p> <p>It should be noted that the errors are only notified under certain conditions. The conditions are noted in the Driver Specific errors section. The IIN bit errors that can be OR-ed together for selection purposes are shown below. The meanings of the terms are also described in the Driver-specific Errors section.</p> <p>BAD_FUNCTION_IIN 0x0001 OBJECT_UNKNOWN_IIN 0x0002 OUT_OF_RANGE_IIN 0x0004 BUFFER_OVFL 0x0008 EXECUTING_IIN 0x0010</p>

			<p>CONFIG_IIN 0x0020 LOCAL_IIN 0x2000 TROUBLE_IIN 0x4000</p> <p>Note that the user can define tags to determine the status of the IIN flags returned from the device. Note also that CDNP_WARN_IIN_ERROR will not generate any #COMs on the screen.</p>
[DNPR]DataLinkConfirmDefault  [<unit>]DataLinkConfirm	0 - Off 1 - Sometimes (Multiframe only) 2 - On	0	Default Data Link Confirm mode.
[DNPR]DataLinkNoRetriesOnControlsDefault  [<unit>]DataLinkNoRetriesOnControls	1 - There are no data link layer retries when sending control messages such as direct operate, or select/operate.  0 - Normal retries apply.	0	Data Link retries on controls. Note: The data link layer will only retry if data link layer confirmations are used, see DataLinkConfirmDefault.
[DNPR]DataLinkRetriesDefault  [<unit>]DataLinkRetries	0 to 255	3	<p>Default number of attempts to retransmit after a data link timeout. This parameter is only relevant if Data Link confirmation is enabled (see parameter [DNPR] DataLinkConfirmDefault).</p> <p>Note: This parameter is ignored if the application function code is from 2 to 6 (Control requests) and data link retries are disabled for these function codes (see parameter [DNPR]DataLinkNoRetriesOnControlsDefault).</p>
[DNPR]DataLinkTimeOutDefault	0 to 65,535 (ms)	2000 ms	Default waiting period since last frame for a data link confirm –only if

[<unit>]DataLinkTimeOut			confirm requested.
[DNPR]DataLinkFrameSize	28 to 292	292	<p>For DNP devices that have small data link layer frames, then for the same size application layer buffer, more data link layer frames will be produced.</p> <p>The library that the DNPR driver utilizes was previously assuming that the DNP device would have a maximum data link layer frame size of 292 bytes, and an application layer fragment maximum size of 2048 bytes. This combination would result in a maximum of 9 frames per fragment.</p> <p>The library has been modified such that it will allow up to 64 frames per fragment, however the main practical restriction introduced is that the resulting application layer data from the combined frames will not overflow the application layer buffer.</p> <p>Now DNP devices with maximum data link layer frame size of less than 292 bytes are also supported.</p> <p>Similarly, the INI parameter [DNPR]DataLinkFrameSize allows the specification of the Max data link layer transmit frame size.</p> <p>The default value is 292 bytes. The parameter value is upper limited to 292 bytes, and lower</p>

			limited to 28 bytes.
[DNPR]TransDelay	0 to 65,535 (ms)	0 ms	Transmission Delay for use in RS485 networks.
<unit>]AppLayerTimeOut	0 to 2,000,000,000 (ms)	[<unit>]Timeout + [<unit>]TimeoutExtension (ms)	<p>Waiting period for application layer response.</p> <p>It is recommended that this should be set to approximately 10 times the DataLinkTimeOut to allow for confirms and link layer resets that may happen as part of a successful application layer transaction.</p> <p>There is no need to allow for the number of frames or fragments that are sent as part of an integrity poll, as the timeout gets reset when each frame of each response fragment is received.</p> <p>The timer is also reset when a data link layer transaction times out.</p>

## Polling Regime Parameters

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

These parameters are used to manipulate polling.

Parameter	Allowable values	Default value	Description
[<unit>]EventPollPeriod	0 to 4,000,000 (seconds)	The current value for [DNPR] EventPollPeriodDefault	<p>Period to poll device for event class data.</p> <p>Integrity poll will poll for all classes (event and static data) after every</p>

Parameter	Allowable values	Default value	Description
			<p>EventPollRatio event polls.</p> <p>If the value is set to zero then the polling mode in the Citect.ini parameter [DNPR]ResponseModeDefault will be disabled. The device will stay offline until an integrity poll is forced out or an unsolicited message from the device is coming in.</p> <p>See <a href="#">Device Polling Regime</a> for an example.</p> <p><b>Note:</b> This Citect.ini parameter can be overwritten if the "Minimum Update Time" field for the I/O Device is greater than 0. (This option is only available with Plant SCADA from version 7.20.)</p>
[<unit>]PollPeriodOffset	0 to 2,000,000,000 (seconds)	0	<p>The amount of time in seconds before the first periodic poll occurs for this unit.</p> <p>A common start time is used for all units for this calculation. Subsequent polls occur every EventPollPeriod after the first poll, thus the offset is maintained between units for all subsequent polls.</p>
[DNPR] EventPollRatioClass1Default  [<unit>] EventPollRatioClass1	0 to 65,535	1	<p>How many class 1 event polls to be issued.</p> <p>A value of 2 would mean that this event class happens every n second poll period.</p> <p>0 will turn class 1 polls off (Note: The integrity poll</p>

Parameter	Allowable values	Default value	Description
			will still ask for class 1 static data). See <a href="#">Device Polling Regime</a> for an example.
[DNPR] EventPollRatioClass2Default [<unit>] EventPollRatioClass2	0 to 65,535	1	How many class 2 event polls to be issued. A value of 2 would mean that this event class happens every n second poll period. 0 will turn class 2 polls off. (NB: The integrity poll will still ask for class 2 static data). See <a href="#">Device Polling Regime</a> for an example.
[DNPR] EventPollRatioClass3Default [<unit>] EventPollRatioClass3	0 to 65,535	1	How many Class3 event polls to be issued. A value of 2 would mean that this event class happens every n second poll period. 0 will turn Class3 polls off. (NB: The integrity poll will still ask for class 3 static data). See <a href="#">Device Polling Regime</a> for an example.
[DNPR]EventPollRatioDefault [<unit>]EventPollRatio	0 to 65,535	10	Default number of event class polls for every integrity poll. Integrity poll will poll for all classes (event and static data). Be reminded that a value of zero implies that only integrity polls are conducted. See <a href="#">Device Polling Regime</a> for an example.
[DNPR] IssueIntegrityPollOnUnitR	0 – Unit recovery carried out with a static poll	0	Enable this setting if the driver should recover a

Parameter	Allowable values	Default value	Description
eccovery [<unit>] IssueIntegrityPollOnUnitR ecoverystatic	1 – unit recovered using integrity poll		device by issuing integrity (static class and event class) poll. static (static class) polls are issued to recover a device if this setting is disabled (default)
[DNPR] AutoUnsolicitedEnableDef ault [<unit>]AutoUnsolicitedE nable	1 for class 1 2 for class 2 4 for class 3 (or any combination thereof)  A value of zero (0) will result in NO classes being enabled for unsolicited responses.  The default value of 7 enables all classes.	7	This provides the ability to explicitly specify the classes to be enabled for unsolicited responses at start-up.  Currently when a device is brought on-line, the driver will send a command to enable unsolicited responses for all classes. This command is only sent if the "Restart" bit is set in the IIN flags.  This parameter will only take effect if the parameter "[DNPR]ResponseModeDefault" has "unsolicited msg" enabled.  Be reminded that the driver also has a 'virtual' data type for enabling/disabling unsolicited responses in a unit. The classes enabled will be as specified by this parameter.
[DNPR] IntegrityPollOnUnsolicited Default [<unit>] IntegrityPollOnUnsolicited	0 – No integrity polls are issued when receiving unsolicited data from an offline device  1 – Unsolicited data from the device causes an integrity poll when the device is offline, in order	0	Send an integrity poll when receiving an unsolicited message for an offline device.

Parameter	Allowable values	Default value	Description
	to bring the device online if possible.		
[DNPR] IPonECPModeDefault [<unit>]IPonECPMode	0 – For any response, immediately send an Event Poll  1 – For any response, immediately send an integrity poll  2 – For an integrity poll response, immediately send another integrity poll, for any other response immediately issue an Event Poll  3 – Disables any polls being sent on IIN bits set except the Overflow bit  4 – Disables any polls being sent on IIN bits including the overflow bit.	2	Defines the follow-up action to be taken when a response is received with any of the class 1, 2, or 3 IIN bits set.
[DNPR] EnableRestartIINDefault [<unit>]EnableRestartIIN	0 – Unit restart sequence is not carried out when restart IIN flag is detected  1 – Unit restart sequence is carried out when restart IIN flag is detected	1	Only disable if all devices are not able to clear their restart IIN flag (as required for DNP compliance). When disabled, the driver will not perform a unit restart sequence (clear restart flag, integrity poll and Enable Unsolicited messages) when a restart IIN flag is detected in a response.
[DNPR] EnablePollPeriodWritesDefault [<unit>] EnablePollPeriodWrites	0 – Disable 1 – Enable	0	Enable the adjustment of the device's poll period via writing to a tag of address 'PollPeriod'.  This is a security feature to help ensure that an operator is not inadvertently given access to adjust the poll period.

Parameter	Allowable values	Default value	Description
[DNPR] IssueClassPollOnUnitRecover [<unit>] IssueClassPollOnUnitRecover	0 – Unit will recover based on the INI parameter "IssueIntegrity PollOnUnitRecovery"  1 – Unit recovered using Class 1 poll  2 – Unit recovered using Class 2 poll  3 – Unit recovered using Class 3 poll	0	Enable this setting if the driver should recover a device by issuing Class 1, Class 2, or Class 3 poll.  Note: The driver will not issue any polls if we configure this parameter along with the INI parameter "EventPollPeriodDefault" with the value Zero.
[DNPR] MaxOperateDelayDefault [<unit>]MaxOperateDelay	1000 to 60000 ms	5000 ms	The RTU configured max delay between a SELECT to OPERATE command in ms.  This is used in the driver to freeze any polling for up to this period to allow the OPERATE command to work.
[DNPR] ChannelTransmissionsPerCpuCount	0 to 20	0	During each poltime, the waiting messages are sent to each RTU. The default behavior is to send these messages out in the order they exist in the driver (when set to 0).  However, some RTUs prefer the messages to go out on one COMs channel, then the next etc.  Setting this value allows this behavior to happen, e.g. if set to 2 (and data was waiting for Ports 0,1,2,3), then the first poltime scan would send data out on Port 0, then Port 1, then wait until the next poltime scan, and send data out to Port 2, then Port 3.

## Data Point and Cache Parameters

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

The DNPR driver supports the following data point and cache parameters. These parameters control variations of behavior for some data types, as well as data cache behavior.

Parameter	Allowable values	Default value	Description
[DNPR]UseFeedBackPoll [<unit>]UseFeedBackPoll	0 - 3 (See description.)	1	Sets the FeedBack mode. 0=Use Integrity poll instead of Feedback Poll to update cache after control action or counter freeze action. 1=Use Feedback poll to update cache after counter freeze or analogue or binary writes (Feedback Poll requires devices to support reads to Binary Output Status, Analogue Output Status, Binary Inputs, Analogue Inputs, Counters and Frozen Counters). The following reads are carried out if the operation is a counter freeze write: class 1, class 2, class 3, Counters and Frozen Counters reads. The following reads are carried out if the operation is an analogue or binary write: Binary Output Status, Analogue Output Status, class 1, class 2, class 3, Binary Inputs and Analogue Inputs reads. This mode requires Level 3 DNP support in the RTU.

Parameter	Allowable values	Default value	Description
			2= Do not perform any type of immediate poll after a control write. This mode will be implemented for devices that can provide an unsolicited update of any feedback changes (caused by control writes). 3= Use an individual point read to update the cache after analogue or binary writesSee "Allowable Values" for more details.
[DNPR]UseSelectClear	0 – Time read request is used to cancel a SELECT request. 1 – An OPERATE with the CLEAR bit set is used to cancel a SELECT request.	0	Default way to handle a cancel of select operation. This is to send an OPERATE with the CLEAR bit set. If your RTU hardware does not support that then setting this parameter to 0 will send a dummy time read instruction to cancel the SELECT operation (that is because the OPERATE command has to be the next sequence number and a dummy command will force the cancel the SELECT request.)
[DNPR]FailOnTagFlagSet	0 - Check quality disabled 1 - Check quality enabled	0	Enable quality check in tag read when the tag flag is set. Fail tag on tag read if the quality is not GOOD (#com will be shown on client display). Note:This parameter is only valid when the driver is running on SCADA 7.10 or early.
[DNPR]DataLagDefault	0 to 2,000,000,000	0	Data in cache is only

Parameter	Allowable values	Default value	Description
[<unit>]DataLag	(seconds)		<p>considered fresh if it is no older than the combined periods of:</p> <ul style="list-style-type: none"> <li>Period between integrity polls</li> <li>AND Period of complete timeout sequence (ie initial and retried timeouts)</li> <li>AND DataLagDefault period (tolerated poll delay).</li> </ul> <p>The first time data is retrieved from cache for a unit and the data is not fresh, error 0x61 is generated and a tag read with virtual address LateRefresh will return TRUE.</p> <p>This condition is reset when the late poll for the device is eventually received.</p>
[DNPR] DataInvalidationFactorDefault  [<unit>] DataInvalidationFactor	0 to 65535.	2	<p>As explained in DataLagDefault, data in cache is only considered to be fresh for a certain amount of time. After this time has been exceeded, the data is considered to be valid, but not fresh.</p> <p>If the data in cache is still not refreshed after a further 'n' Event periods of time (where 'n' = DataInvalidationFactorDefault), then the data in cache is no longer considered valid, and data retrieved from cache in this state will be marked with error 0x54.</p>
[DNPR]	0 – An error is returned if	0	Indicate if errors of read

Parameter	Allowable values	Default value	Description
IgnoreReadErrorsOnWrite	trying to read "write only" tags. 1 – 0000 data with no error is returned when reading "write only" tags.		requests on "write only" tags are to be ignored. This is needed when using dialup devices as SCADA reads ALL tags at connect time.
[DNPR] IgnoreWarnIfOneOK	0 – A bad Quality is returned if there is missing or stale data in a read block. 1 – Bad Quality is not returned when there is missing or stale data in a read block provided at least one item in the block is OK. This is useful when points are not configured in the RTU and the read block is requesting these, but SCADA does not use them.	0	Bad Quality should be indicated or not if at least one item is valid in a block read request. This parameter applies to digital points as well as other points.
[DNPR] RetZeroOnCROBInvalid	0=Test validity of Binary Output Control Status data. 1= Do not test validity of Binary Output Control Status data.	0	Flag indicating whether Binary Output Control Status data is to be tested for validity (before a Binary Output Control Status point can be valid, the corresponding binary point requires a written value). If not tested for validity, then if an invalid Binary Output Control Status point is read, then a zero will be displayed.  If the type IS to be tested for validity, then if any point within a I/O server multiple point request is invalid, then the response to the request indicate an error condition. A solution in this case might be to write to all Binary points at startup.

Parameter	Allowable values	Default value	Description
[DNPR] RetZeroOnAOOBInvalid	0=Test validity of Analog Output Control Status data  1= Do not test validity of Analog Output Control Status data.	0	Flag indicating whether Analog Output Control Status data is to be tested for validity in the same manner as for RetZeroOnCROBInvalid above.
[DNPR]Commissioning	0= Treat non-polled points as irrelevant  1= Force non-polled points to #COM (default applies to all devices)	0	A device may have digital types as a non-contiguous list, and may start or end on points not on an 8 bit boundary. As Plant SCADA will block a minimum of 8 bits for digitals, and enforce an 8 bit boundary, requests may include points that are not received by the device or that are invalid for the device. For this reason, the driver assumes that bits that are not updated by the polling sequence are irrelevant and not configured as tags by the user.  These points will return 0 and NO_ERROR to avoid forcing the valid points in the same DCB into error. The driver waits until the first successful data acquisition poll, before determining if a point as irrelevant.  Commission all points of the project to ensure that every tag does indeed refer to a point that is refreshed by the polling routine. The flag commissioning tests for invalid BI points, by not forcing the point to 0 and NO_ERROR. This will then

Parameter	Allowable values	Default value	Description
			#COM all points in the DCB containing the invalid digital point.  Also when this INI is set, the event system will log to "dnpr_logger.txt" the name of tags which are considered irrelevant, e.g. "#TRACE,Irrelevant_point, BI023, in,Unit_035".
[DNPR] IsItemisedRequest	1=Individual I/O server-to-Driver Non-digital requests (RECOMMENDED)  0= I/O server-to-Driver requests are blocked (default applies to all devices).	1	Requests for a basic DNP poll do not use a blocking (grouping) algorithm as applies to a lot of protocols. Instead the required points are predefined in the device (as Class 0), and are requested during an integrity poll.  The DNPR driver can support requests of up to 2048 bits in one request from The I/O server, but this is for contiguous data. The data available in the driver cache will not be contiguous if the predefined points in Class 0 are not contiguous.  The I/O server request will be flagged as bad quality (#COM) if any one of the points in the requested block are not defined in Class 0.  To prevent this situation from occurring, non-digital data is not blocked by default. Requests from The I/O server to the driver, for non-digital data, are for individual points. Be reminded that

Parameter	Allowable values	Default value	Description
			<p>this does not affect the data acquisition algorithm between the driver and the device, only between the I/O server and the driver. Digital types still will be optimized together. A device may have digital types as a non-contiguous list, and may start or end on points not on an 8 bit boundary. As the I/O server will block a minimum of 8 bits for digitals, and enforce an 8 bit boundary, the driver assumes that bits that are not updated by the polling sequence are irrelevant and not configured as tags by the user. These points will return 0 and NO_ERROR to avoid forcing the valid points in the same DCB into error.</p> <p>The flag commissioning tests for invalid BI points. The default should provide the solution for a DNP project, however, if it is decided by a Plant SCADA representative that blocking is required between the driver and the I/O server, set IsItemisedRequest to zero.</p> <p>Change MAX_LENGTH and BIT_BLOCK fields for the DNP protocol (in the \BIN\PROTDIR.DBF file) to the required blocking length in bits, and copy the PROTDIR file to your Plant SCADA project</p>

Parameter	Allowable values	Default value	Description
			<p>directory.</p> <p>Modify the Citect.ini file to configure the 'Block' parameter to the required value in bytes, and also the 'MaxBits' parameter to the required value in bits. Disable the incremental compile option and then recompile your project.</p> <p>Note: If you reinstall Plant SCADA, the PROTDIR.DBF file is overwritten, so repeat the change.</p>
[DNPR]MaxBits	1 to 2048 (bits)	2048	<p>Max Block Size constant (only applies when IsItemisedRequest is set to zero).</p> <p>This value should be left at default unless modifying IsItemisedRequest parameter to block read requests to a specific value. Refer to description of IsItemisedRequest.</p> <p>Note: This field only applies to read requests; writes will be allowed to the maximum that will result in one application layer fragment, but cannot be greater than 2048 bits.</p> <p>Write blocking can be disabled by setting the INI parameter [&lt;IOServer&gt;] 'Blockwrites' to 0.</p>
[DNPR]ReqDelay	0 to 65535 (ms)	20	Delay used when servicing a read request by reading from cache.

## Miscellaneous Parameters

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Parameter	Allowable values	Default value	Description
[DNPR] ResponseModeDefault [<unit>]ResponseMode	0 to 15	11 (Polled, Unsolicited enabled, Perform time synch when requested)	Default response mode, composed from a combination of:  1 = Polled (Must be set to enable device polling for event class data)  2 = Unsolicited enabled (Must be set to enable unsolicited value updates)  4 = Use Delay Measurement (By default, the driver will use the Record Current Time command (Function Code 24) when performing time synchronization. When this bit is set, it will use Delay Measurement command (Function Code 23))  8 = perform time synch when requested (When set, the driver performs time synchronization after receiving IIN1.4 bit from the device)  In the case where "Use Delay Measurement" is selected, then the parameter [DNPR]ReqDelay should also be set.
[DNPR]StartupDelay	0 to 'WatchTime' (seconds)	0	Allows the unit initialization (and subsequent online check) to be delayed.  This Citect.ini parameter can be used, in conjunction with a Cicode IODeviceControl() call at start-up to disable units

Parameter	Allowable values	Default value	Description
			<p>before they transmit an online check for unit initialization.</p> <p>A delay of a couple of seconds should suffice. Do not delay more than the setting for WatchTime.</p>

## Device Group Parameters

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

For the following parameters, [<unit>] represents the Citect.ini section name that is defined with the following precedence:

- *Device group*: as defined in the Address field of the I/O Device. (This section is not recommended. You should choose the next section instead.)
- *Protocol.DeviceGroup*: Protocol is DNPR, DeviceGroup defined in the ‘Address’ field of the I/O Device.
- *Protocol.PortName*: Protocol is DNPR, PortName is defined in the ‘Port Name’ field of the Port.
- *Unit\_x* where x is the value in the Number field on the I/O Device. (This section is not recommended. You should choose the next section instead.)
- *Protocol.PortName.UnitName*: Protocol is DNPR, PortName is defined in the ‘Port Name’ field of the Port, UnitName is defined in the ‘Unit Name’ field of the IODevice.

## Example:

- ‘Address’ field of the IODevice: 128 SITERTU
- ‘Number’ field of the IODevice: 10
- ‘Unit Name’ field of the IODevice: PrimaryIODev128
- ‘Port Name’ field of the Port: PrimaryPort1

## INI section names:

- [SITERTU] (Not recommended anymore)
- [DNPR.SITERTU]
- [DNPR.PrimaryPort1]
- [Unit\_10] (Not recommended anymore)

- [DNPR. PrimaryPort1. PrimaryIODev128]

These parameters are used to configure behavior that will only apply to units with the same device group, in the Citect.ini section called <unit>.

They are as follows:

## Driver-specific Parameters

- [<unit>]SCADAAddress
- [<unit>]Timeout
- [<unit>]Retry
- [<unit>]TimeoutExtension
- [<unit>]OnTimeBODeviceDefault

## Peer Redundancy Parameters

- [<unit>]ActivateIfPeerIsActiveOffline
- [<unit>]UnitUsurpMode
- [<unit>]SyncReplication

## Event Processing Parameters

- [<unit>]UTCTimeSync

## Polling Regime Parameters

- [<unit>]EventPollPeriod
- [<unit>]PollPeriodOffset
- [<unit>]EventPollRatioClass1
- [<unit>]EventPollRatioClass2
- [<unit>]EventPollRatioClass3
- [<unit>]EventPollRatio
- [<unit>]IssueIntegrityPollOnUnitRecovery
- [<unit>]AutoUnsolicitedEnable
- [<unit>]IntegrityPollOnUnsolicited
- [<unit>]IPonECPMode
- [<unit>]EnableRestartIIN
- [<unit>]EnablePollPeriodWrites
- [<unit>]MaxOperateDelay

## Communication Parameters

- [<unit>]MaxTimeOuts
- [<unit>]RapidOfflineDetection
- [<unit>]DataLinkConfirm
- [<unit>]DataLinkNoRetriesOnControls
- [<unit>]DataLinkRetries
- [<unit>]DataLinkTimeOut
- [<unit>]AppLayerTimeOut

## Data Point and Cache Parameters

- [<unit>]UseFeedBackPoll
- [<unit>]DataLag
- [<unit>]DataInvalidationFactor

## Miscellaneous Parameters

- [<unit>]ResponseMode

## DNPR Redundancy

In other Plant SCADA drivers, the I/O Server controls the redundancy switchover based on the redundant I/O Devices state (online/offline) and configured I/O Device priorities. The DNPR driver redundancy support is different. It performs the redundancy switchover (makes either the primary or the standby I/O device active and usurps the connection) by itself, or you can do it manually by using the virtual unit control tags.

In normal driver redundancy, if both primary and standby units (the representation of the I/O device in the driver software) are connected to the device, communication happens via the primary unit. If the connection to the primary unit is lost, the I/O Server switches communication to the standby unit. If the connection to the primary unit is restored, the I/O Server switches communication back to the primary unit.

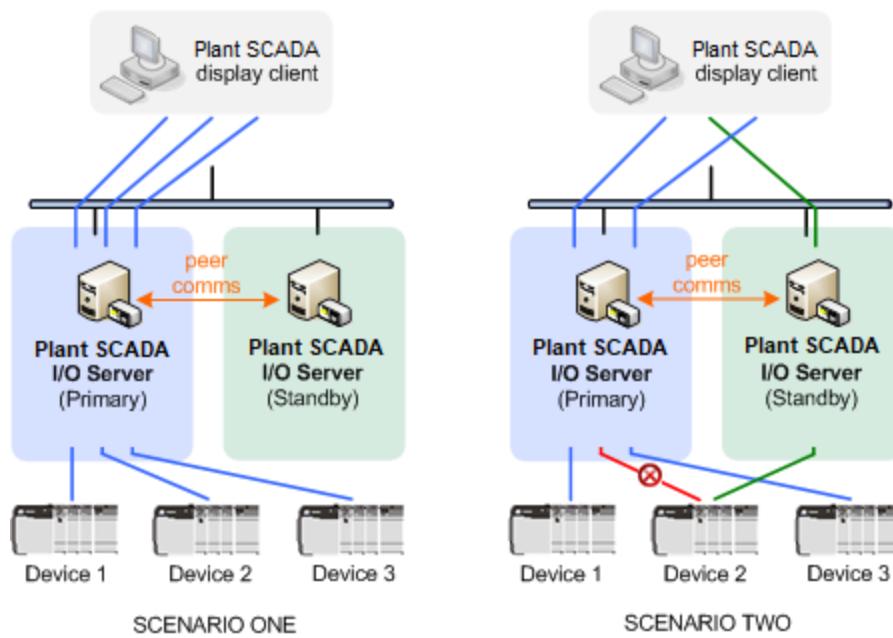
In DNPR driver, the priority for unit use is not based on the primary unit when a driver is started. The unit that is in use continues in service. If both drivers become active (that is, peer connection lost) then the priority for unit use is based on the primary unit when the peer connection is reestablished.

In the redundant configuration, the primary and standby I/O Device pair is configured on different I/O Servers. The DNPR drivers on these servers communicate to each other to synchronize the redundant unit states, the tag and the event/trend cache data.

The driver only communicates with a device if the unit is in the active state. The unit is active only on one peer driver at one time, except for the case when the communication between the redundant I/O Servers is lost and both units go into the active state.

When a unit is active, the driver reports the current online/offline state to Plant SCADA. If the unit is not active, it appears as offline to Plant SCADA. The driver does not communicate to the device if the unit is not active. It does not send any DNP3 frames to the device, or process any frames received from the device.

The following diagram illustrates a typical redundancy configuration showing a pair of redundant I/O servers and a network of DNP3 devices:

**SCENARIO ONE:**

Normal operation, with devices communicating via the primary server.

**SCENARIO TWO:**

Redundancy switchover due to an interruption to the communication channel between the primary I/O server and device two. Even after the communication channel to the primary I/O Server is restored, communication to device two continues via the standby I/O Server.

## Redundancy Configuration

The driver supports single level redundancy only, so a unit can only have one peer (redundant) unit. Configure this unit on a separate I/O server. A unit cannot have a redundant unit within the same I/O server.

The ini parameters required for setting up the peer redundancy are described in [Peer Redundancy Parameters](#).

The following example shows the minimum number of parameters required to set on primary and standby I/O Servers:

**Primary I/O Server:**

```
[DNPR]
EnablePeerRedundancy = 1
LocalAddress = 10.176.234.11
LocalPort = 5200
PeerAddress = 10.176.232.61
PeerPort = 5200
```

```
[IOMServer]
WatchDogPrimary = 1
```

**Standby I/O Server:**

```
[DNPR]
EnablePeerRedundancy = 1
LocalAddress = 10.176.232.61
LocalPort = 5200
```

```
PeerAddress = 10.176.234.11
PeerPort = 5200
[IOServer]
WatchDogPrimary = 1
```

### Manual Redundancy Switchover (Usurping)

To choose the peer driver (I/O server) on which a unit will be active, use the following tags:

#### Primary request tags:

- Unit<n>.RequestUnitActive.Primary

Writing 1 to a primary request tag causes the unit to be active on the driver on the primary I/O server. Writing 0 causes the unit to be active on the driver on the standby I/O server.

#### Standby request tags:

- Unit<n>.RequestUnitActive.Standby

Writing 1 to a standby request tag causes the unit to be active on the driver on the standby I/O server. Writing 0 causes the unit to be active on the driver on the primary I/O server.

The request tags can be read at any time, and return the value last written to the tag.

The user can monitor which peer driver (I/O server) has an active unit, using the following tags:

- Unit<n>.UnitActiveState.Primary
- Unit<n>.UnitActiveState.Standby

If the unit is active on the primary I/O server driver, then the primary tags return 1 and the standby tags return 0. If the unit is active on the standby I/O server driver, then the standby tags return 1 and the primary tags return 0.

You can also use the following addresses to determine a port's online status for diagnostic purposes.

- Unit<n>.UnitPortState.Primary
- Unit<n>.UnitPortState.Standby

This online status is the TRUE online status. Be aware of this when using ExclusiveAccess mode for ports.

Remember that the port that is reported on is the one which matches your tag. If required, you can have units on one port that are PRIMARY, STANDBY, PRIMARY, and so on.

### Automatic Redundancy Switchover (Usurping)

When either primary or standby I/O Server unit starts up, it establishes a connection with a configured peer unit if available. If the peer unit is active, the driver puts the local unit to inactive state and leaves the peer driver to communicate with the device.

Both units become active when communication is inoperable between the I/O Servers. The DNPR drivers exchange states when the communication is reestablished. As both units are active, the driver sets the standby

I/O unit inactive and communication to the device continues via the primary unit.

Only the active unit sends DNP3 frames to the device and processes the frames received from the device.

A unit becomes active when:

- There is no redundant unit. (It is not configured or disabled.)
- The driver finds that its peer is deactivating or is in an inactive state.
- The communication between the I/O Servers is lost. After the connection re-establishes, the standby unit sets to inactive.
- Forcing it to be active by writing to one of the virtual unit control tags.

The automatic switchover (usurping) of units can also happen when the unit port is set offline. The driver reports "CHANNEL\_OFFLINE": driver error 20 (0x14) and generic error 21 (0x15).

The port is set offline when:

- The TCP/IP connection is closed or cannot be opened.
- There is an error report from one of the low-level communication functions.
- There is a request to disconnect the internal port. For example, an I/O Server shutdown, or scheduled use of I/O Server.

The lack of an RTU reply does not mean that a TCP/IP connection is dropped. For example, the removal of a cable at a router is unlikely to close a connection. The specific situation and redundancy control of a site is unique to the site.

That is why usurp control is available via driver tags. Users need to engineer the control (via Cicode) if the default driver handling does not match their needs. It is recommended to make the override usurp control available via an engineering screen to cover any unusual circumstances. An active server has its DNPr units usurped across to the other server before the server is shutdown.

Also, configure the field communication hardware to drop the connections if regular updates does not occur. It helps to ensure that the communications can be reinstated. When gateways are used, and availability of a RTU unit is hidden, then some active update counter should be used and Cicode added to check on count changes. That is, if counter is not incrementing then do X.

## Port-level Redundancy

Normally, Plant SCADA port control involves opening a port at startup and closing it at shutdown (except when used in a scheduled device or dialup situation). However, if an RTU supports multiple IP connections, it will normally get a connection from the primary AND secondary port (normal Plant SCADA redundancy). This may not be acceptable for an RTU.

To help resolve this, ExclusiveAccess mode was introduced in to the DNPR driver (see [Port-based Parameters](#)).

## Exclusive Port Access

When you connect to devices like terminal servers, which only support one connection at a time, port control is required.

An Citect.ini parameter has been provided to set a port's mode to "ExclusiveAccess". In this mode, when any unit on a port is usurped (moved from one state to another) from being active to inactive, the port connection is

dropped. This will allow the port on the redundant server to have access to the device(s), bringing any associated units into action.

When using ExclusiveAccess mode, ports on both sides need to have the Citect.ini parameter set. Additionally, the virtual unit on the port needs to follow the active port, as the physical port may not be online (e.g. the RTU is offline).

Hardware which only allows one connection should have an "inactivity timer" set, so the connection is dropped if it is not used. For example, a terminal server needs to drop its connection if no communication has occurred after "x" seconds. This allows access to the unit from another server if the current connection is broken.

If the hardware doesn't support this, you may want to consider using the -K option on the TCP/IP driver combined with registry "Keep Alive" changes.

## Driver Cache Replication

The driver maintains a database of values received from devices known as the cache.

The cache is updated by both event and static data. When updating this cache with values received from the device, the active unit replicates these values to the non-active unit on a peer driver over the TCP link between the peer drivers.

To send the DNP3 acknowledge to the device after all received values have been replicated to the peer, use the Citect.ini parameters SyncReplicationDefault and SyncReplication.

This behavior is preferable (but not default) as it helps ensure that no data will be lost in case the active server dies in the instant straight after it acknowledges the transaction to the RTU but before the values are replicated to the peer. Also, the DNP3 acknowledge will be sent straight away if the peer TCP link is lost.

On cold start-up, if there is a running peer driver, the cache is synchronized from the running peer's cache prior to the unit, on the starting driver, being allowed to become active. If no peer driver is found, or there is no redundant unit, the cache is cleared.

The cache value replication throughput is faster than 100 values per second provided that the underlying Windows operating system and TCP network does not introduce delays. This rate can be expected on a 10Mbit network that is not congested.

There is no disk file image of the driver cache. On cold start after a total system shutdown (shutdown of both peer drivers) the cache is empty.

## Trend and Alarm Backfilling

The trend and alarm backfilling use event queues that hold events until they are back filled into the Plant SCADA trend and alarm systems. The queue of events waiting to be processed for backfilling will be replicated between the drivers using the same TCP communications link that the cache replication uses.

The event queues in peer drivers are kept synchronized in the same manner as the driver cache is kept synchronized. That is, the image is copied on unit start-up, if there is a peer, and is kept synchronized by replication on queue state change.

There is no disk file image of the event queue. On cold start after a total system shutdown (shutdown of both peer drivers) the queue is empty.

---

**Note:** If some events have not been pushed to the alarm/trend servers and both I/O servers shutdown then the events and trend data will be lost.

---

## Example

A device generates an event and sends it to the active I/O server. The active I/O server will then send the event to the peer I/O server. When it receives the event, the peer I/O server will add the event into its internal queue. Then the active I/O server then adds the event to its internal queue before sending it to the alarm/trend server. The alarm/trend server sends an acknowledgement to the active server on receiving the event. The active I/O server sends a message to the peer to remove the event from its internal queue. The event is then removed from the internal queue of the active I/O server.

### Value Writes

Tag writes (such as controls) which result in a DNP3 transaction, are not replicated between peer drivers.

The 'Write Fail' and 'Write Pending' flags are zero in an inactive peer driver's database. Therefore, if a usurp occurs during a control write, then the WritePending and WriteFail flags will both be zero regardless of whether or not the write was completed.

### Action on Becoming Active

The driver is configurable to perform the following action when a unit becomes active:

- Send DNP3 link reset and then an integrity poll to the device. The polling sequence is reset to start from the time the unit became active.
- Send DNP3 link reset and then read the RTU time. The polling sequence is not changed and continues as if nothing happened.

---

**Note:** The driver may perform additional DNP3 transactions depending on the states of the IIN bits. In particular if the IIN restart bit is set, the driver will perform an integrity poll. If any of the class 1, 2, or 3 event flags are set the driver will perform an event poll if the reset bit is not set.

---

- Send DNP3 link reset and then a Class0 (Static) poll to the device. The polling sequence is reset to start from the time the unit became active.

### Data Acquisition within DNP3

DNP3 offers an array of methods for data acquisition from DNP3 devices, but not all devices will support all methods of data acquisition. The simplest form of data acquisition, and the one that all DNP3 devices support, is the acquisition of a group of points that have been pre-defined in the device. This is known as a class request.

DNP3 devices have four definable groups (classes) of I/O points:

- Class 0 lists the I/O points that are to have their current value sent back to Plant SCADA, when Plant SCADA sends a Class 0 request.
- Class 1 lists the I/O points that are to have their events sent back to Plant SCADA, when Plant SCADA sends a class 1 request.
- Class 2 and class 3 are conceptually the same as class 1 but are separate groups to allow prioritizing of event data.

To summarize, class 0 requests will result in a reply containing the current value of all of the predefined I/O points, whereas class 1, class 2, and class 3 requests will result in a reply containing all of the buffered events that have occurred for all of the predefined I/O points in class 1, class 2, or class 3.

The DNP3 protocol ensures that if a request is made to a device for Class 0, class 1, class 2, and class 3 data, then the device will reply with all of the buffered events (with the most recent event received last), and then reply with the current data. The order in which the data arrives guarantees the most recent value is held in the driver cache, and the inclusion of Class 0 static data assures that data acquisition is not overly dependent on the report-by-exception mechanism. This form of all-classes poll is termed an integrity poll.

The device may automatically send event data to Plant SCADA (this is termed an Unsolicited Response), or it may wait until Plant SCADA requests class 1, class 2, or class 3 data. In either case the driver will update its cache with the value received in the event data, and also check if the event is related to an alarm tag or a trend tag that needs processing.

Citect.ini parameters exist that allow event polls to be isolated into classes. The default behavior is that all event classes are polled on each event poll. The following Citect.ini parameters control the ratio for each class to the event poll (see [Polling Regime Parameters](#)):

- [DNPR]EventPollRatioClass1Default
- [DNPR]EventPollRatioClass2Default
- [DNPR]EventPollRatioClass3Default
- [<unit>]EventPollRatioClass1
- [<unit>]EventPollRatioClass2
- [<unit>]EventPollRatioClass3

A value of zero will turn event polls off for that class. A value of 2 would mean that this event class happens every n second poll period. Be reminded that the integrity poll always asks for the static data for all event classes.

The default DNP3 master address for Plant SCADA is defined by the parameter [DNPR]SCADAAddress. Define this master address in the slave device, if the slave is to send unsolicited responses to Plant SCADA. A different master address can be specified in each unit, using the device-level parameter [<unit>]SCADAAddress.

If the device does support unsolicited responses, then having all I/O points report through exception (send events on change of data) will allow Plant SCADA to poll (with an integrity poll) the device infrequently. This data acquisition algorithm requires very low traffic throughput on the network.

If the device does not support unsolicited responses, then more frequent Plant SCADA polling (with event polls for all classes) will keep its cache up-to-date. After every *n* event polls, Plant SCADA sends an integrity poll to ensure the data integrity of the driver cache. The period between polls for a device is defined by [DNPR]EventPollPeriodDefault and [<unit>]EventPollPeriod, and the ratio of Event Polls for every integrity poll for a device, is defined by [DNPR]EventPollRatioDefault and [<unit>]EventPollRatio.

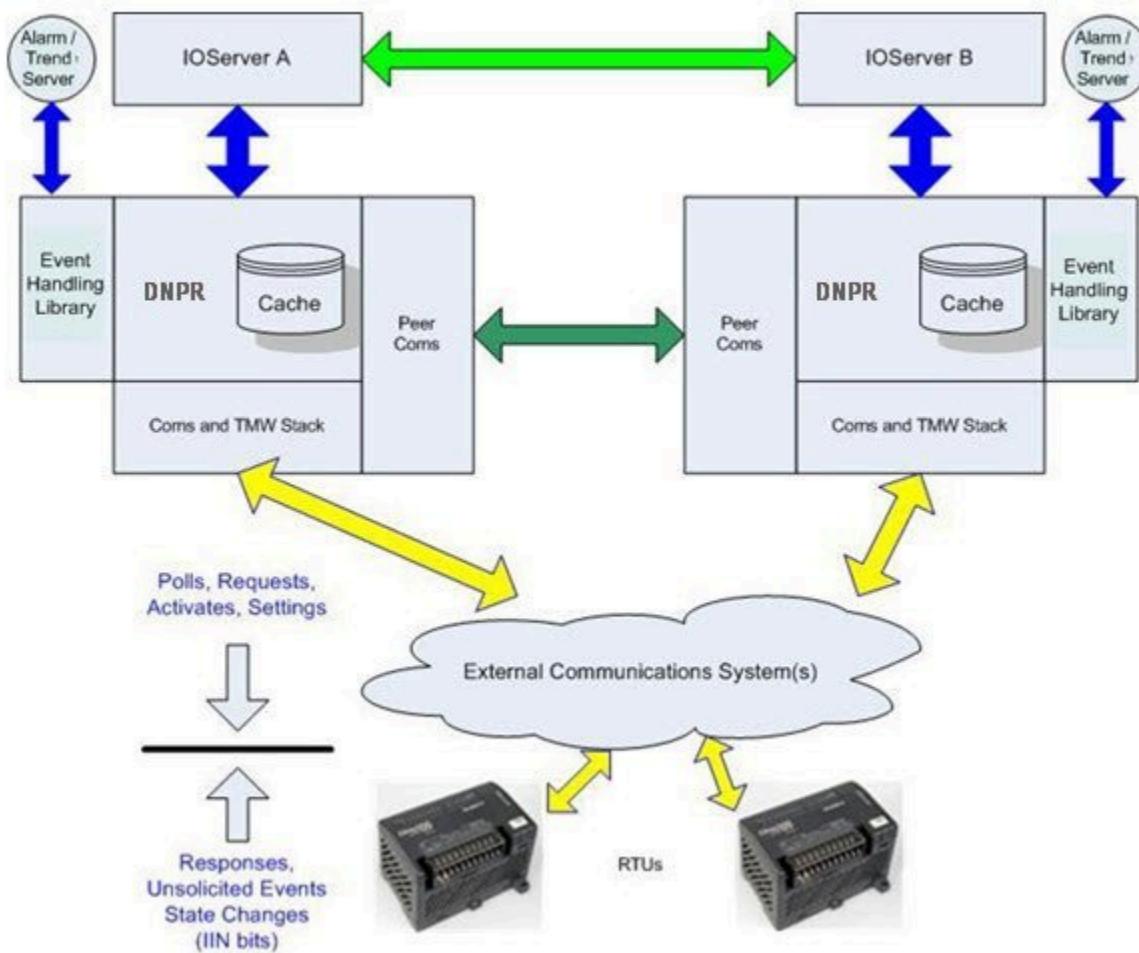
If the device does not support report-by-exception (i.e. it does not buffer event data but will only reply with current data) then the EventPollRatio should be set to zero so that only integrity polls are issued, and the EventPollPeriod should be set to a low value in order to keep the driver cache up-to-date.

The DNP3 protocol also offers other data acquisition algorithms in which the device master (Plant SCADA) can request points that are defined by the master, not by the slave. There are numerous methods, each offering varying levels of control the master has in determining which points it can request. As the level of support for these data acquisition algorithms differs from device to device, and some devices do not support these data acquisition algorithms, Plant SCADA will only implement the Class read data acquisition algorithm described above.

In summary, all of the I/O points in a device that are required by Plant SCADA should be allocated to Class 0 in the device. If the device supports report-by-exception (which is strongly recommended), then also allocate each point to one of the event classes.

## Example

The following diagram gives an overview of the DNPR functional blocks. Debug options are also along these functional block lines.



## Blocking

Requests for a basic DNP3 poll do not use a blocking (grouping) algorithm as applies to a lot of protocols. Instead, the required points are predefined in the device (as Class 0), and are requested during an integrity poll. The DNPR driver can support requests of up to 2048 bits in one request from Plant SCADA, but this is for contiguous data. The data available in the driver cache will not be contiguous if the predefined points in Class 0 are not contiguous. If any of the points in the requested block is not defined in Class 0, all points in a Plant SCADA request will be flagged as bad quality (#COM).

To avoid this situation, non-digital data is not blocked by default. All requests from Plant SCADA to the driver for non-digital data are for individual points. This does not affect the data acquisition algorithm between the driver and the device, only between Plant SCADA and the driver. Digital types will still be optimized together. A device may have digital types as a non-contiguous list, and may start or end on points not on an 8-bit boundary.

As Plant SCADA will block a minimum of 8 bits for digitals, and enforce an 8-bit boundary, the driver assumes that bits not updated by the polling sequence are irrelevant and not configured as tags by the user. These points

will return 0 and NO\_ERROR to avoid forcing the valid points in the same DCB into error. The [DNPR]Commissioning Parameter tests for invalid BI points.

Digital Reads are blocked by Plant SCADA to a minimum of 8 bits. A DNP device can configure points in a non continuous fashion, for example, points 0 to 4, then 10 to 14, and so on. Therefore, non-digital Plant SCADA requests are not blocked so as to avoid the situation where a blocked request may also include a point in cache that is not part of the pre-defined DNP3 group. This is why the default protdir.dbf entries are 8 & 8 bits.

If you know your RTU devices have continuous point allocations, then the compile time value can be increased. For example, if 32 digitals are configured then the value could be set to 32 & 32 in protdir.dbf.

Runtime blocking is controlled by the [DNPR]Block=x, the [DNPR]IsItemisedRequest=y parameter, and the [DNPR]MaxBits=z parameters. This allows you to block together several read requests in the IOServer.

Block = MaxBits / 8, IsItemisedRequest=0

Set the runtime values at least equal the compile time settings. They can however be larger than compile time settings.

If you wish to remove any residual block errors, then you can use the [DNPR]IgnoreWarnIfOneOK parameter. This should be done only after you are happy that all points are configured correctly.

## Device Polling Regime

The DNPR driver has Citect.ini parameters that allow event polls to be isolated into classes.

The default behavior is that all event classes are polled each event poll. The EventPollRatioClassx parameters control the ratio for each class to the event poll. A value of zero will turn event polls off for the specified class. A value of 2 would mean an event class happens every 2nd poll period.

---

**Note:** The integrity poll asks for the static data for all event classes.

The diagram below shows the relationship between the following [Polling Regime Parameters](#):

EventPollPeriod=60 (seconds)

EventPollRatio=10

Integrity polls are issued every 660 seconds (11 mins)

((EventPollRatio+1)\* EventPollPeriod) ----> (10+1)\*60 == 660.

EventPollRatioClass1=2

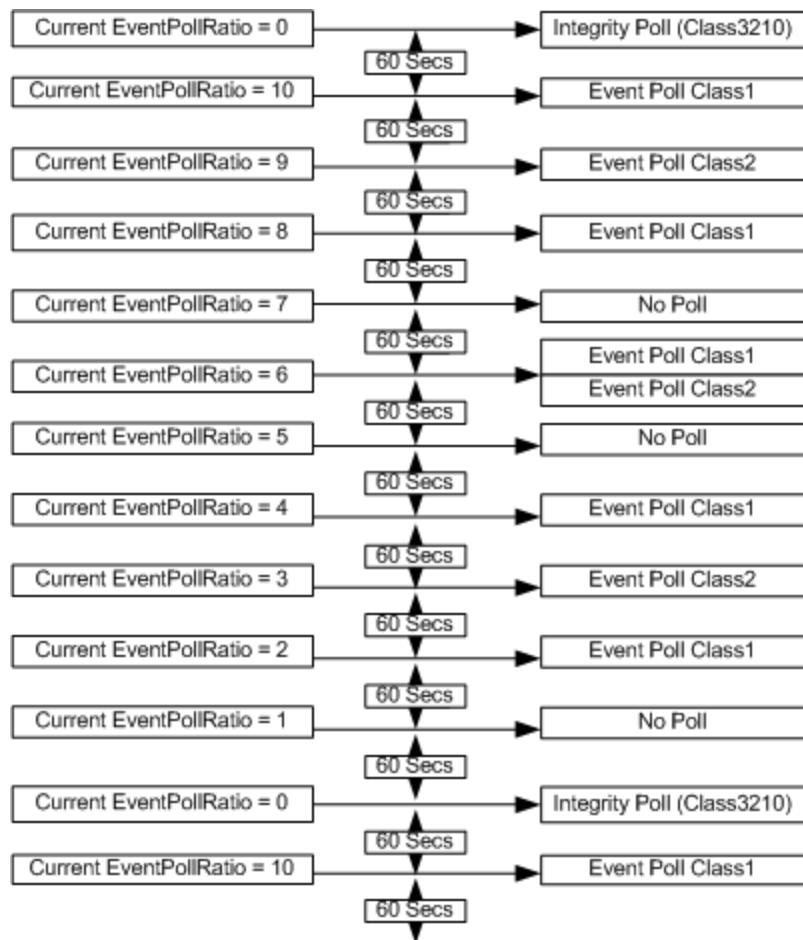
Class 1 polls are issued every second EventPollPeriod from the first module after the integrity poll (poll every EventPollPeriod\* EventPollRatioClass1: 60\*2 == 120 seconds).

EventPollRatioClass2=3

Class 2 polls are issued every third EventPollPeriod from the first module after the integrity poll (poll every EventPollPeriod\* EventPollRatioClass2: 60\*3 == 180 seconds).

EventPollRatioClass3=0

No class 3 polls will be issued.



## Device Recovery

Device recovery refers to the process that occurs when a device goes offline (due to interrupted communication) or the system is starting.

There are two modes to recover a device:

- [Device Recovery - Normal](#)
- [Device Recovery - Fast](#).

The first two attempts to recover a device will place the integrity/static polls with high priority in the DNP3 transmission queue. This means that these polls will be transmitted before any other requests in the exception of write time, delay measurement and control requests. If the device is still not recovered after the first two attempts, then the priority of the subsequent integrity/static polls is the same as the periodic polls of online device. This is to give the chance to online device to be polled.

Once a device is recovered then the periodic polls will be staggered if the Citect.ini parameter "[<unit>]  
PollPeriodOffset" is set.

The Citect.ini parameter to enable one or the other mode (integrity/static polls) is called  
"[DNPR]IssueIntegrityPollOnUnitRecovery" or "[<unit> IssueIntegrityPollOnUnitRecovery". If set to 0 (default) then static polls are issued otherwise integrity polls are issued.

It is also possible to issue Class 1, Class 2, or Class 3 poll on the device recovery. You can enable this behaviour by

the Citect.ini parameter "[DNPR] IssueClassPollOnUnitRecovery" or "[<unit>] IssueClassPollOnUnitRecovery". By default, these parameters remain in disable mode. So the unit will recover based on the "IssueIntegrityPollOnUnitRecovery" ini parameter.

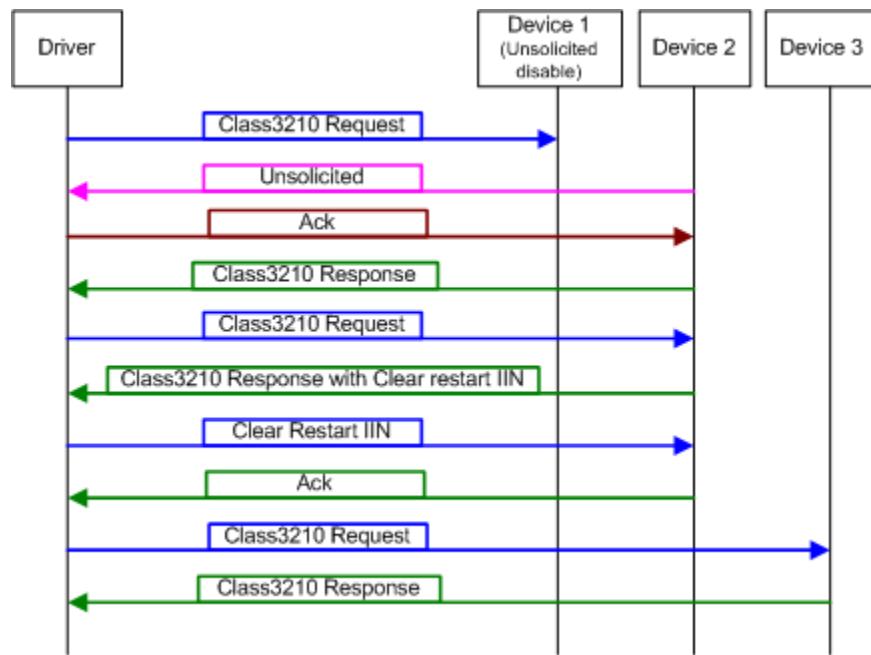
## See Also

[Device Recovery - Startup Considerations](#)

### Device Recovery - Normal

An integrity (Class 0123) poll is issued. This mode will take a long time to bring the device online in the case the device has a lot of events stored. The integrity polls are staggered if the Citect.ini parameter "[<unit>] PollPeriodOffset" is set.

This mode might have less interaction between the driver and the devices than the fast recovery mode but the responses to integrity poll request will take a much longer time if the devices have a lot of events. Also unsolicited messages will slow the responses to the driver's requests.



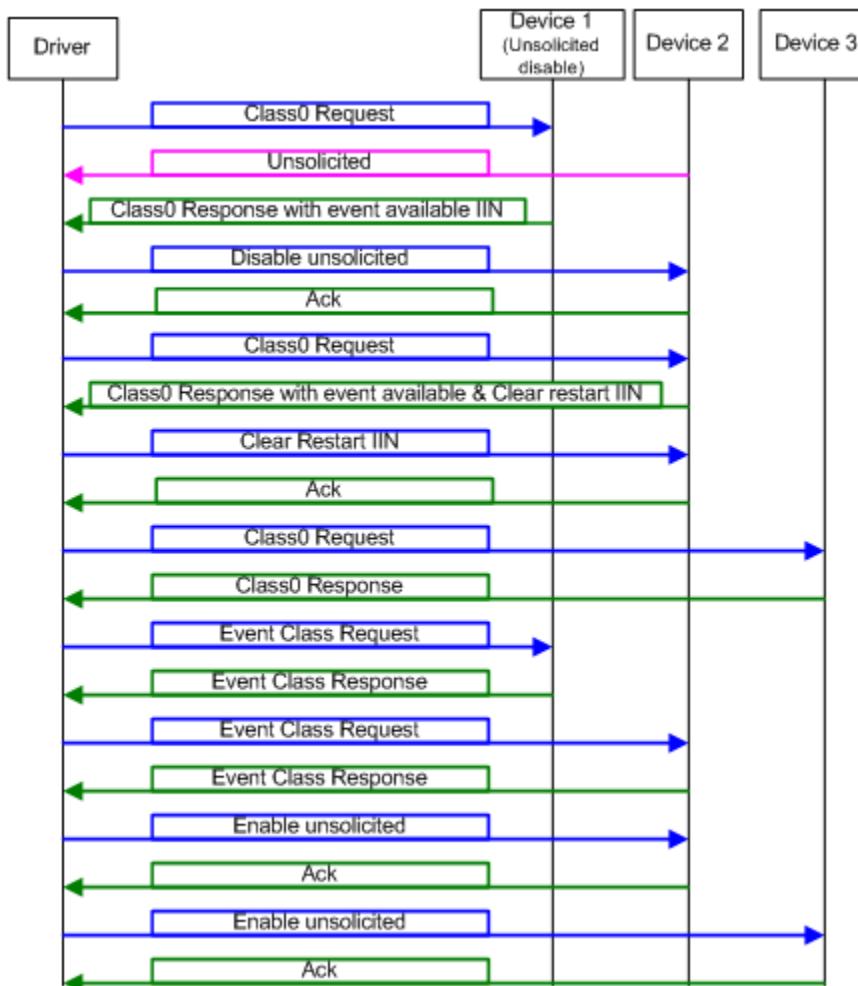
## See Also

[Device Recovery - Fast](#)

### Device Recovery - Fast

A static (Class 0) poll is issued. The IIN flags from the response to the static poll will determine if event needs to be retrieved. Also unsolicited message will not be processed, and a disable unsolicited request will be sent to the device upon receiving unsolicited messages during the device recovery. Unsolicited messages will be re-enabled when the device is recovered if unsolicited message were enabled in the first place. This mode allows to bring the device online quickly in the case the devices have a lot of events stored.

This mode might require more interaction than the normal recovery mode but the response to the static poll requests will be very quick. Therefore, the devices will become online faster than the normal recovery mode.



## See Also

[Device Recovery - Normal](#)

### Device Recovery - Startup Considerations

In a site with many RTUs, it may be useful to consider port startup staggering to spread the load out. This is configured using the Citect.ini parameter [DNPR]StartupStaggerDelay.

It may also be a good idea to use the Citect.ini parameter [<unit>]PollPeriodOffset to stagger when polls are issued, especially where many units are coming through one port.

For example, unit 1 – 5, unit 2 – 10, unit 3 – 15, would send polls 5 seconds after the default time for unit 1, 10 seconds after unit 2, and 15 seconds after unit 3.

---

**Note:** The static polls, which are used to recover a device (after interrupted communications or system startup), will NOT be staggered. This is because the driver will try to bring the devices online as soon as possible and static (class 0) polls will not load the system too much. Subsequent periodic polls will be staggered using the parameter [<unit>] PollPeriodOffset.

## See Also

[Device Recovery](#)

### SELECT and OPERATE

It is possible to have separate SELECT and OPERATE operations for binary outputs (BO) and analog outputs (AO).

The timeout value for how soon an OPERATE is needed after a SELECT is set in the RTU. Therefore, the default Citect.ini parameters MaxOperateDelayDefault and MaxOperateDelay (per unit) need to be changed to match this time.

Because this time is used to stall any polls for the unit while a SELECT[\_ONLY] is pending. This is because the DNP protocol requires that the next request after a SELECT[\_ONLY] is an OPERATE.

Be reminded that only one SELECT/OPERATE sequence can be done at a time on a single RTU. Multiple SELECTs cannot be done at once. This is a DNP3 limitation.

Users should display the WRITE PENDING bit and the WRITE FAIL bit to verify correct operation. Typically, if you send an OPERATE after the max. delay time, you will see the WRITE FAIL tag light up.

A SELECT can be manually canceled by writing to the BO/AOx.CancelSel tag. This will clear all other status bits for SELECT/OPERATE.

Forced Integrity or Event polls should not occur while a SELECT[\_ONLY] command is pending. The software will automatically delay regular scheduled polling help ensure no poll occurs while a SELECT[\_ONLY] is pending.

Status bits BO/AOx.<WrtPend,WrtFail,SelPend> are available to provide user feedback on the SELECT/OPERATE sequence.

---

**Note:** In an earlier version of the DNPR driver, the SELECT command was effectively a SELECT and OPERATE (automatic). This remains unchanged. Where you see an address with ".Select.", this will be a SELECT and OPERATE. For a pure select only, the address needs to be ".SelectOnly".

---

### Virtual Units

Virtual units allow to monitor and control the physical units which are connected to the real device. As Plant SCADA does not allow you to read or write data for an offline unit, it needs virtual unit support to overcome this limitation.

There are two types of virtual unit supported by Plant SCADA:

- Global virtual units
- Local virtual units.

The global virtual units are configured on a separate port and remain online. The local virtual units are configured on the same port as the physical units and are online only if the port is online.

---

**Note:** It is recommended that only global virtual devices are used, as it can be difficult to determine on which I/O Server a virtual device is active. Also, global virtual devices are easier to use.

---

The virtual unit has an address of 65534. It does not send any DNP3 frames, and the driver does not process any frames received from a device with an address of 65534.

A global virtual unit has an address of 65535. A global virtual until allows you to access globally over all channels. It allows you to define a dummy port which does not go offline and add a virtual unit with the number 65535.

---

**Note:** Each virtual unit must be configured as a primary I/O Device and not included into a redundant pair as the driver can only access the unit information on the I/O Server local to the virtual unit.

---

## See Also

[Virtual Units Example](#)

[Virtual Units Configuration](#)

[Configuring Variables for Virtual Units](#)

### Virtual Units Example

The following example shows the implementation of virtual devices on a primary and standby server.

The primary server, which hosts the virtual device "Virtual DevP", has the following Citect.ini parameter settings:

```
[DNPR]
EnablePeerRedundancy = 1
LocalAddress = 10.176.234.11
LocalPort = 5200
PeerAddress = 10.176.232.61
PeerPort = 5200
```

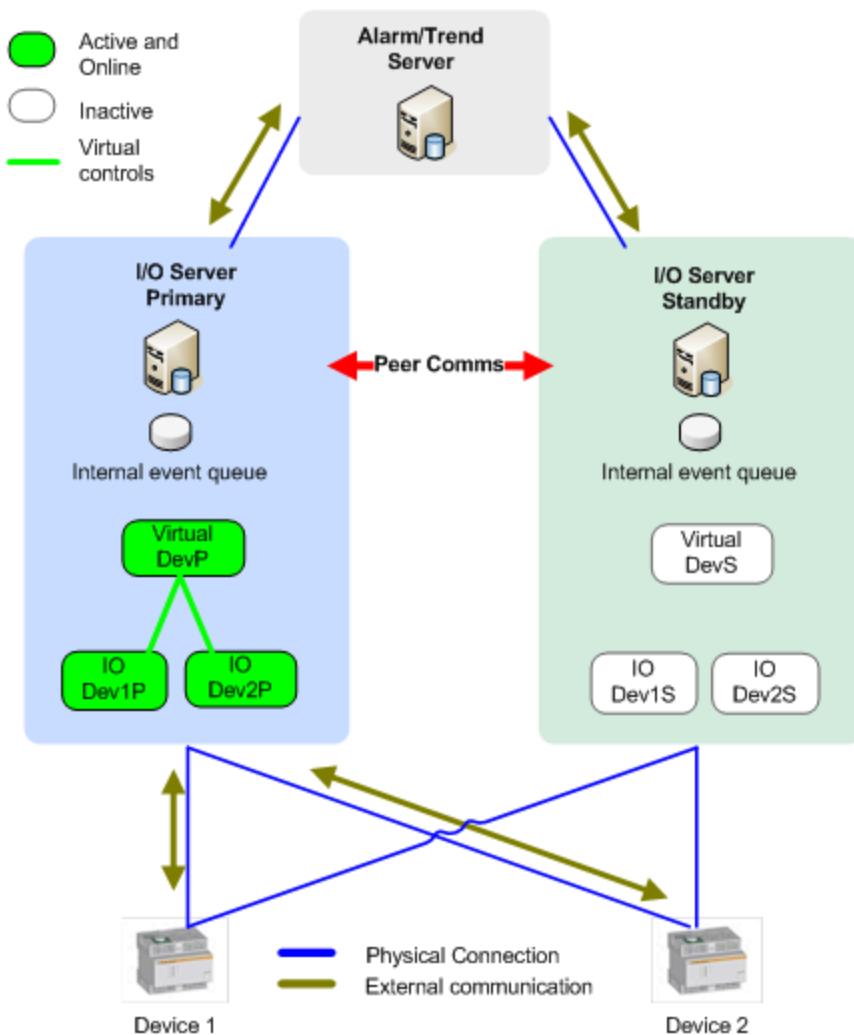
```
[IOServer]
WatchDogPrimary = 1
```

The standby server, which hosts the virtual device "Virtual DevS", has the following Citect.ini parameter settings:

```
[DNPR]
EnablePeerRedundancy = 1
LocalAddress = 10.176.232.61
LocalPort = 5200
PeerAddress = 10.176.234.11
PeerPort = 5200
```

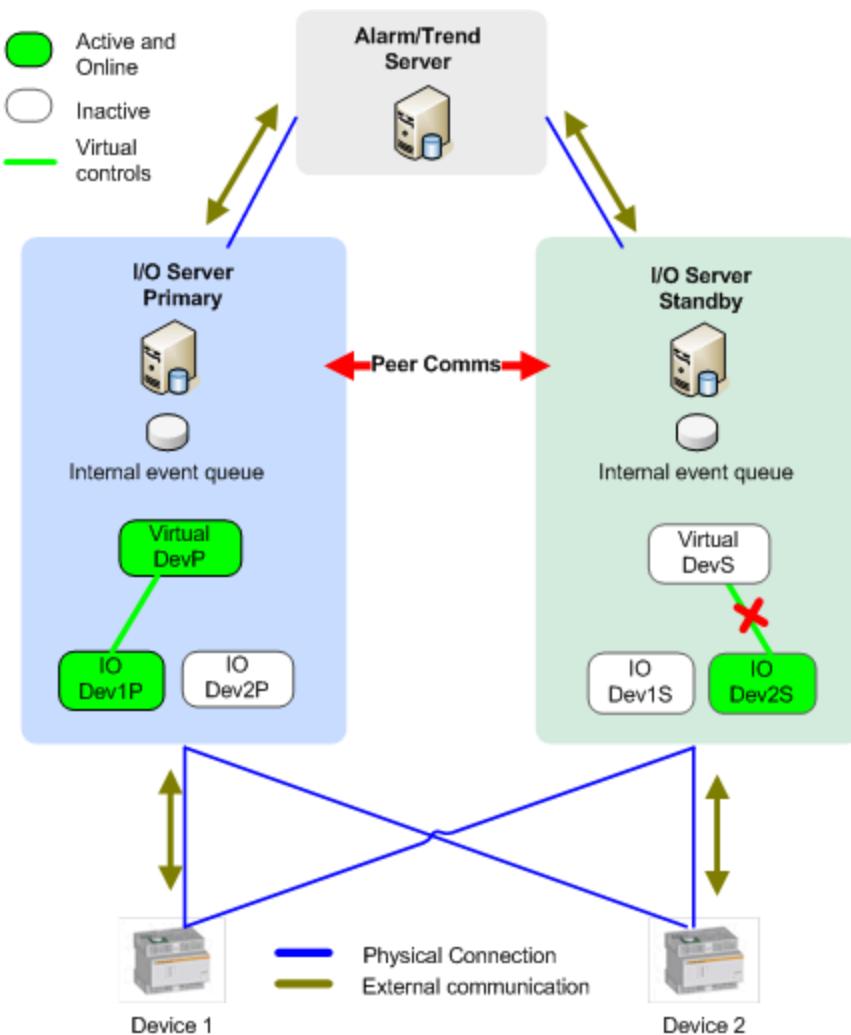
```
[IOServer]
WatchDogPrimary = 1
```

In the following diagram, the virtual device and the I/O devices are communicating on the same port. The three devices are active and online on the primary I/O server. The 3 devices are inactive on the standby I/O server. The I/O devices can be controlled by the virtual device.

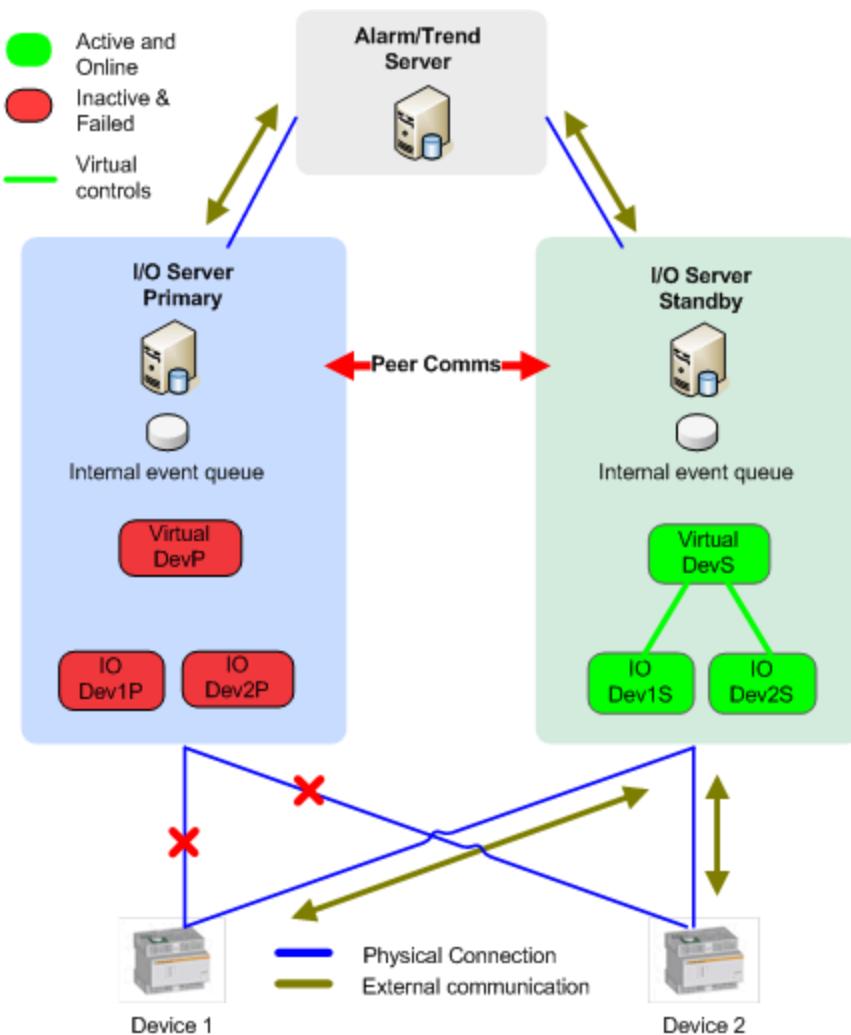


In the diagram below, the virtual device and the I/O devices are communicating on the same port. The I/O device was usurped onto the standby I/O server. The virtual device and I/O device 1 are active and online on the primary I/O server. I/O device 1 can be controlled using the virtual device.

I/O device 2 is active and online on the standby I/O server. The I/O device 2 cannot be controlled with the virtual device because they are not active on the same I/O server.

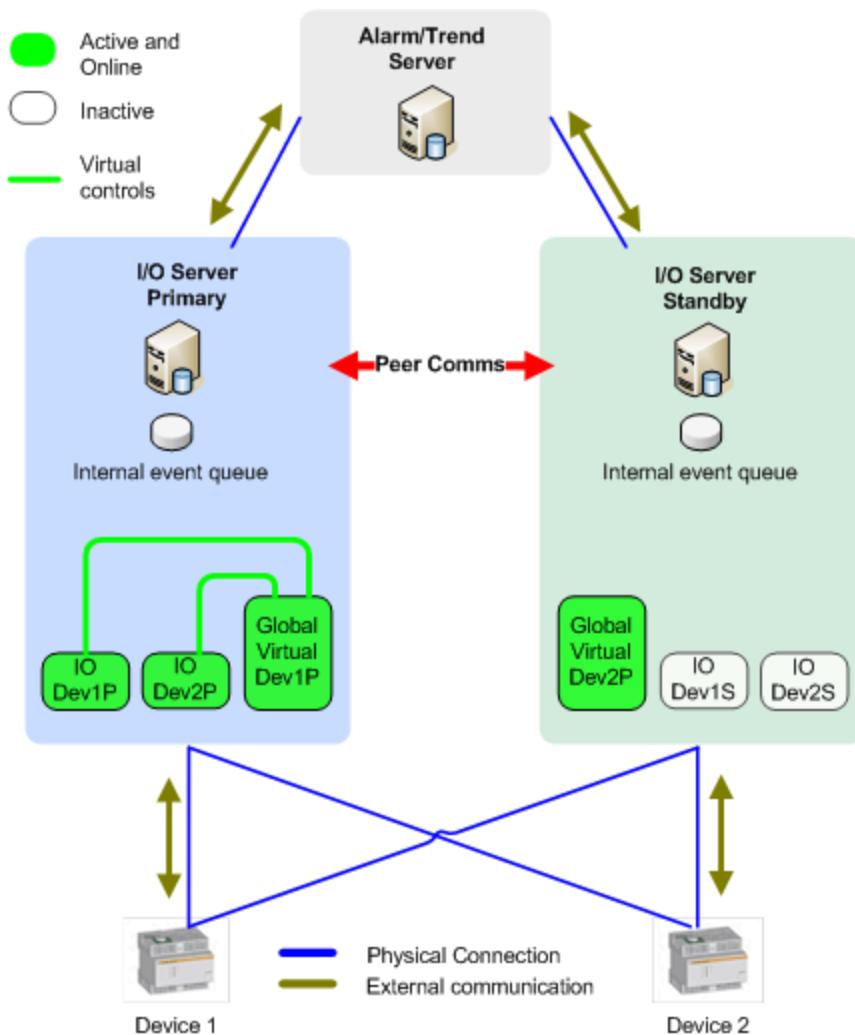


In the next diagram, the virtual device and the I/O devices are communicating on the same port. The connection of the port has been interrupted and the devices are automatically usurped onto the standby I/O server. The three devices are active and online on the standby I/O server. Both I/O devices can be controlled by the virtual device.



In the diagram below, both I/O devices are communicating on the same port. The global virtual devices are on a different port than the I/O devices, and both global devices are primary devices.

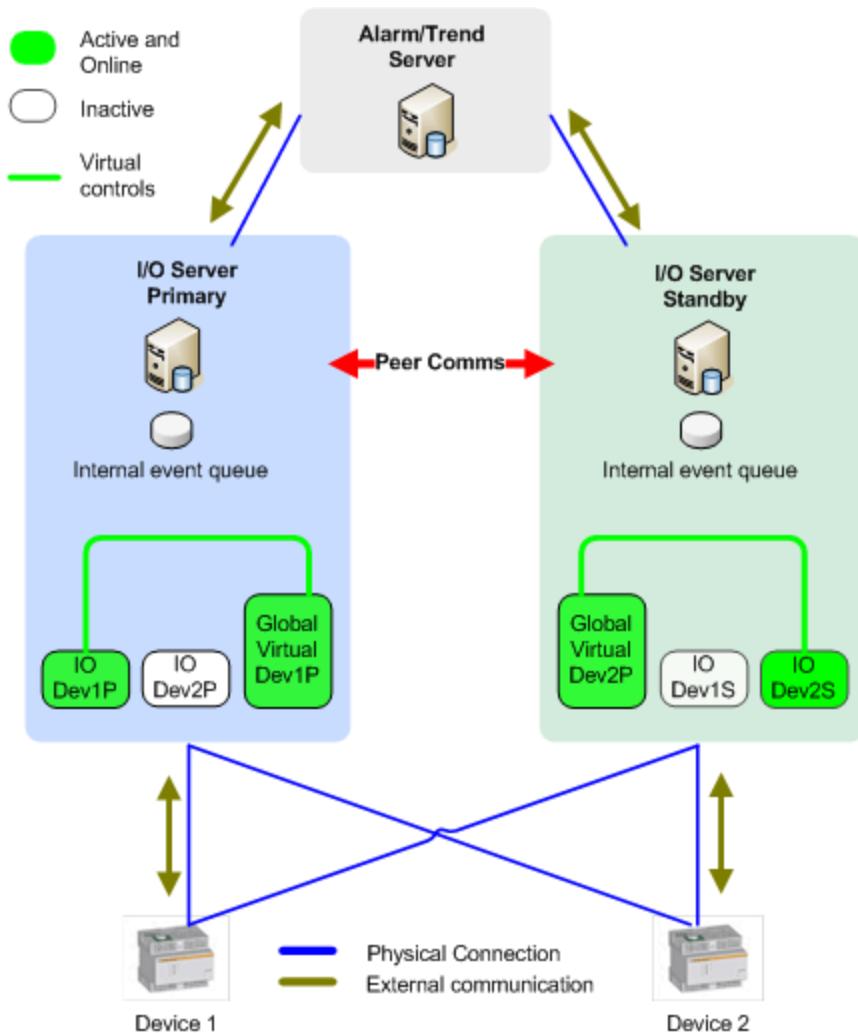
The primary global virtual device 1 is active and online on the primary I/O server. The primary global virtual device 2 is active and online on the standby I/O server. Both I/O devices are active and online on the primary I/O server and can be controlled by the global virtual device 1 on the primary I/O server.



In the last example, both I/O devices are communicating on the same port. The global virtual devices are on different ports to the I/O devices and both global devices are primary devices.

The primary global virtual device 1 is active and online on the primary I/O server. The primary global virtual device 2 is active and online on the standby I/O server. I/O device 2 was usurped onto the standby I/O server. I/O device 1 is active and online on the primary I/O server and can be controlled by the global virtual device 1 on the primary I/O server.

I/O device 2 is active and online on the standby I/O server and can be controlled by the global virtual device 2 on the standby I/O server.



## See Also

[Virtual Units Configuration](#)  
[Configuring Variables for Virtual Units](#)

### Virtual Units Configuration

#### Port configuration for the Global Virtual Unit

Field	Value
Port Name	This field is user defined.
Port number	
Board name	Refers to the board previously defined in the Boards view.
Baud rate	Leave this field blank

Field	Value
Data bits	Leave this field blank
Stop bits	Leave this field blank
Parity	Leave this field blank
Special Opt	-I127.0.0.1 -P12345 -U
Comment	This field is user defined and is not used by the driver.

**Device configuration for the virtual unit**

Field	Value
Name	This field is user defined.
Number	This is a unique I/O device number. <b>Note:</b> Use a different number for each virtual unit as they should not be configured as a redundant pair.
Address	<b>65534</b>
Protocol	DNPR
Port name	Refers to the port defined in the Ports view.
Comment	This field is user defined and is not used by the driver.

**Device configuration for the global virtual unit**

Field	Value
Name	This field is user defined.
Number	This is a unique I/O device number. <b>Note:</b> Use a different number for each virtual unit as they should not be configured as a redundant pair.
Address	<b>65535</b>
Protocol	DNPR
Port name	Refers to the port defined in the Ports view.
Comment	This field is user defined and is not used by the driver.

**See Also**

[Virtual Units Example](#)

## Configuring Variables for Virtual Units

### Configuring Variables for Virtual Units

The Plant SCADA I/O server does not write to any tags if a unit is offline. Virtual units are useful to implement virtual tags that write to other units on the same port. They provide a way to write to some tags on units regardless of whether the unit is online or offline.

For example, consider the following project units and tags:

#### Unit 42

Tag1 address: *ForceIntegrityPoll*

#### Virtual unit (65534)

Tag2 address: *UNIT42. ForceIntegrityPoll*

If unit 42 is offline, it is not possible to write to Tag1. If the virtual unit stays online, then it is possible to write to the virtual tag Tag2. This causes the driver to do an integrity poll to unit 42 even if unit 42 is offline.

The virtual unit can only access the physical unit information on the I/O Server local to the virtual unit. To monitor or control the primary and standby units via virtual tags, configure a virtual unit on the primary unit I/O Server, a virtual unit on the standby I/O Server and then define two virtual tags on these virtual units.

For example, to monitor the time to the next poll, define a tag with address "*Unit<x>.TimeToNextPoll*" for one virtual unit and a tag with address "*Unit<x>.TimeToNextPoll*" for another virtual unit. At runtime only one of these tags shows data. The tag configured on a virtual unit is located on the same I/O Server as the active physical unit. The other tag sets to 0.

If a switchover happens, the physical unit on this I/O Server becomes active, and the tag on the second virtual unit starts showing data. Use *UnitActiveState.Primary* and *UnitActiveState.Standby* virtual tags to find out the active redundant unit and the virtual tag that is currently showing the data.

For the usurp virtual tags, configure the following only on one virtual unit, as the driver replicates this information between the redundant units.

- *RequestUnitActive.Primary*
- *RequestUnitActive.Standby*
- *UnitActiveState.Primary*
- *UnitActiveState.Standby*
- *UnitPortState.Primary*
- *UnitPortState.Standby*.

### Virtual Unit Data Types

- [Virtual Units - Polling Regime Data Types](#)
- [Virtual Units - Usurp Data Types](#)
- [Virtual Units - Control Data Types](#)
- [Virtual Units - Monitoring Data Types](#)

# Virtual Units - Polling Regime Data Types

**Note:** The following data types are only recommended for experienced users who are familiar with the terminology and methods associated with the DNP 3.0 protocol.

Address	Plant SCADA Type	DNP3 Type	Description
Unit<x>.PollPeriod	LONG	Adjust the Polling Period  No Object	R/W. If the Citect.ini parameter 'EnablePollPeriodWrites' is set to TRUE (1), then this tag can be used to adjust the Event Polling Period. This tag can be used to display the current Event Polling Period regardless of the state of 'EnablePollPeriodWrites'. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
Unit<x>.EnablePoll	INTEGER	Returns if Polling enabled for reads  No Object	R/W. Returns if the unit has enabled polling. A write of '0' will turn polling off, '1' will re-enable polling. <b>Note:</b> The driver schedules regular integrity/event poll intervals from the time the unit comes online. When the polling is re-enabled, the next integrity/event poll executes at the next regular poll time. For example, if the unit EventPollPeriod is set to 60 sec and the unit comes online at 1.00.00 AM, the polls schedule at 1.01.00,

Address	Plant SCADA Type	DNP3 Type	Description
			<p>1.02.00, etc. If the polling gets disabled at 1.00.30 and re-enabled at 1.00.45, the next poll happens at 1.01.00, but if polling gets disabled at 1.01.20, the next polling happens at 1.02.00.</p> <p>This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.</p>
Unit<x>.ForceIntegrityPoll	DIGITAL	Force an integrity poll Object 60 Variation 1 Object 60 Variation 2 Object 60 Variation 3 Object 60 Variation 4	<p>W/O.</p> <p>Writing a '1' to this tag forces the driver to execute an Integrity poll of the device (in addition to the routine polling that it automatically undertakes).</p> <p><b>Note:</b> When the forced integrity poll is completed, the driver checks the next scheduled regular integrity/event poll time. If the time until the next scheduled poll time is less than 33% of the unit EventPollPeriod, the driver increases the next scheduled integrity/ event poll time by the EventPollPeriod value.</p> <p>This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.</p>
Unit<x>.ForceEventPoll	DIGITAL	Force an integrity poll Object 60 Variation 2	<p>W/O.</p> <p>Writing a '1' to this tag</p>

Address	Plant SCADA Type	DNP3 Type	Description
		Object 60 Variation 3 Object 60 Variation 4	will force the driver to execute an Event poll of the device (in addition to the routine polling that it automatically undertakes).  This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
Unit< x >.TimeToNextPoll	LONG	Time to go to next Integrity poll No Object	R/O Seconds. Will count down from the set Integrity poll time, and may become negative if the Integrity poll has been delayed.  This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
Unit< x >.TimeToNextClass1 Poll	LONG	Time to go to next class 1 poll No Object	R/O Seconds. Will count down from the set class 1 poll time and may become negative if the class 1 poll has been delayed. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
Unit< x >.TimeToNextClass2 Poll	LONG	Time to go to next class 1 poll No Object	R/O Seconds. Will count down from the set class 2 poll time and may become negative if the class 2 poll has been delayed.

Address	Plant SCADA Type	DNP3 Type	Description
			This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
Unit<x>.TimeToNextClass3Poll	LONG	Time to go to next class 1 poll No Object	R/O Seconds. Will count down from the set class 3 poll time and may become negative if the class 3 poll has been delayed. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
Unit<x>.Polls	INTEGER	Count of successful polls (Forced and periodic). No Object	R/W. Count of successful Integrity, Event, class 1, class 2 and class 3 polls of the device. Write '1' to reset counter. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
Unit<x>.FailedPolls	INTEGER	Count of unsuccessful polls (Forced and periodic). No Object	R/W. Count of unsuccessful Integrity, Event, class 1, class 2 and class 3 polls of the device (i.e. incomplete full timeout sequence). Write '1' to reset counter. This address does not relate to I/O data internal to the device, but relates to data internal to the driver.

Address	Plant SCADA Type	DNP3 Type	Description
			driver, or an action to be performed by the driver. Note that this counter is incremented only after the poll message and the following retry requests have timed out. The number of retry requests is configured using the "Retry" parameter.
Unit<x>.Ipolls	INTEGER	Count of successful Integrity polls (Forced and periodic). No Object	R/W. Count of successful Integrity polls of the device. Write '1' to reset counter. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
Unit<x>.FailedIPolls	INTEGER	Count of unsuccessful integrity polls (Forced and periodic). No Object	R/W. Count of unsuccessful Integrity polls of the device (i.e. failure of full timeout sequence). Write '1' to reset counter. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver. Note that this counter is incremented only after the poll message and the following retry requests have timed out. The number of retry requests is configured using the "Retry" parameter.
Unit<x>.Epolls	INTEGER	Count of successful Event	R/W.

Address	Plant SCADA Type	DNP3 Type	Description
		polls (Forced and periodic). No Object	Count of successful Event polls of the device. Write '1' to reset counter.  This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
Unit<x>.FailedEpolls	INTEGER	Count of unsuccessful Event polls (Forced and periodic). No Object	R/W.  Count of unsuccessful Event polls of the device (i.e. failure of full timeout sequence). Write '1' to reset counter.  This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.  Note that this counter is incremented only after the poll message and the following retry requests have timed out. The number of retry requests is configured using the "Retry" parameter.
Unit<x>.UnsolicitedRxd	INTEGER	Count of Unsoliciteds received. No Object	R/W  Count of valid Unsolicited Responses received by the device. Write '1' to reset counter.  This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
Unit<x>.RequestEventRefresh	INTEGER	Force Event Refresh for device	R/W.  Writing 1 will cause the

Address	Plant SCADA Type	DNP3 Type	Description
		No Object	<p>unit to refresh all event data the next integrity poll. A read reads this state.</p> <p><b>Note:</b> This value will be set if [DNPR]RefreshEventsOnStartup=1 was set.</p> <p>This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.</p>
Unit<x>.RTUEvents	LONG	RTU Events received No Object	R/O. RTU events PER PORT received from the field device. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.

**Where:**

- x = device address of the real device as specified in the I/O Device's **Address** field
- R/O = read only
- W/O = write only
- R/W = read or write

**See Also**

[Configuring Variables for Virtual Units](#)

## Virtual Units - Usurp Data Types

---

**Note:** The following data types are only recommended for experienced users who are familiar with the terminology and methods associated with the DNP 3.0 protocol.

---

Address	Plant SCADA Type	DNP3 Type	Description
Unit< x >.RequestUnitActive.Primary	INTEGER	Request unit active on primary. No Object	R/W Writing a '1' to this tag will force the unit be active on the primary I/O server. Writing '0' will force the unit be active on the standby I/O server. A read returns the last value written to the tag. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
Unit< x >.RequestUnitActive.Standby	INTEGER	Request unit active on standby. No Object	R/W Writing a '1' to this tag will force the unit be active on the standby I/O server. Writing '0' will force the unit be active on the primary I/O server. A read returns the last value written to the tag. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
Unit< x >.UnitActiveState.Primary	INTEGER	Unit active state on primary. No Object	R/O Returns 1 if the unit is active on the primary I/O server and 0 if active on the standby I/O server. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
Unit< x >.	INTEGER	Unit active state on	R/O

Address	Plant SCADA Type	DNP3 Type	Description
UnitActiveState.Standby		standby. No Object	Returns 1 if the unit is active on the standby I/O server and 0 if active on the primary I/O server.  This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
Unit< x >.UnitPortState.Primary	INTEGER	True port connection state of primary unit. No Object	R/O Useful to check if the port, a unit is on, is connected. -1 No peer update 0 Port not connected 1 Port connected.  This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
Unit< x >.UnitPortState.Standby	INTEGER	True port connection state of standby unit. No Object	R/O Useful to check if the port, a unit is on, is connected. -1 No peer update 0 Port not connected 1 Port connected.  This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.

**Where:**

- x = device address of the real device as specified in the I/O Device's **Address** field
- R/O = read only
- W/O = write only
- R/W = read or write

## See Also

[Configuring Variables for Virtual Units](#)

# Virtual Units - Control Data Types

**Note:** The following data types are only recommended for experienced users who are familiar with the terminology and methods associated with the DNP 3.0 protocol.

Address	Plant SCADA Type	DNP3 Type	Description
Unit<x>.DisableUnsolicited	DIGITAL	Disable all Unsoliciteds Object 60 Variation 2 Object 60 Variation 3 Object 60 Variation 4	W/O. Writing a '1' to this tag will instruct the unit not to broadcast any more unsolicited responses. Need to be W/O, as there is no way of determining if unsoliciteds were disabled before Plant SCADA startup. This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver. <b>Note:</b> This a Level 3 feature.
Unit<x>.EnableUnsolicited	DIGITAL	Enable all Unsoliciteds Object 60 Variation 2 Object 60 Variation 3 Object 60 Variation 4	W/O. Writing a '1' to this tag will allow the unit to broadcast unsolicited responses. The Classes enabled are determined by the Citect.ini parameter [DNPR]AutoUnsolicitedEnableDefault or [<unit>] AutoUnsolicitedEnable. Need to be W/O, as there is no way of determining if unsoliciteds were enabled before Plant

Address	Plant SCADA Type	DNP3 Type	Description
			<p>SCADA startup.</p> <p>This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.</p> <p><b>Note:</b> This a Level 3 feature.</p>

**Where:**

- x = device address of the real device as specified in the I/O Device's **Address** field
- R/O = read only
- W/O = write only
- R/W = read or write

**See Also**

[Configuring Variables for Virtual Units](#)

## Virtual Units - Monitoring Data Types

**Note:** The following data types are only recommended for experienced users who are familiar with the terminology and methods associated with the DNP 3.0 protocol.

Address	Plant SCADA Type	DNP3 Type	Description
Unit<x>.RTUStatus	DIGITAL	RTU Online status No Object	<p>R/O</p> <p>Internal online status of a unit. Of limited use as a tag on a unit as if the unit was offline, you could not read the tag anyway. This tag has been superseded by the UnitActiveState.X tags.</p> <p>This address does not</p>

Address	Plant SCADA Type	DNP3 Type	Description
			relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.
Unit<x>.LateRefresh	DIGITAL	Late cache refresh for device No Object	R/O Indicates when a device poll has been delayed, resulting in a late cache refresh. Refer to Citect.ini parameter DataLag for further details.  This address does not relate to I/O data internal to the device, but relates to data internal to the driver, or an action to be performed by the driver.

**Where:**

- x = device address of the real device as specified in the I/O Device's **Address** field
- R/O = read only
- W/O = write only
- R/W = read or write

**See Also**

[Configuring Variables for Virtual Units](#)

**Troubleshooting**

The section of the help contains troubleshooting information for the DNPR driver.

- [DNPR Driver-specific Errors](#)
- [Driver Statistics](#)
- [Logging](#)
- [Debugging Messages](#)

**DNPR Driver-specific Errors**

When a hardware error occurs, Plant SCADA generates an alarm, and displays the alarm on the hardware alarm page (in the alarm description of the hardware alarm). To see the error number, make sure you have Alarm

Category 255 defined with a display format that includes {ErrDesc,15} {ErrPage,10}.

There are a number of generic errors that are common to protocols. In some cases only the generic error is available, though often both the generic error and a specific error are given.

The following errors, listed in (hexadecimal) sequence, are specific to this protocol. Plant SCADA displays the error number and description for common protocol-specific errors. Uncommon errors are not contained in the Plant SCADA error database, in which case only the error number will be displayed.

You may require additional information to enable you to rectify an error. This information should be detailed in the documentation that accompanied the I/O Device (or network). If, after reviewing the documentation, you cannot rectify an error, contact Technical Support for this product. Due to the complex nature of this protocol, technical support is only available at hourly rates. Please contact Technical Support for further details.

**0x00050GENERIC\_SOFTWARE\_ERROR**

Software driver error

**0x00051GENERIC\_SOFTWARE\_ERROR**

Memory could not be allocated for cache

**0x00052GENERIC\_TOO\_MANY\_COMMANDS**

Could not insert request to lower API layer because of existing duplicate request

**0x00053GENERIC\_ADDRESS\_RANGE\_ERROR**

A point used in a write request is invalid

**0x00054GENERIC\_INVALID\_DATA**

Point(s) in a read request are not yet valid because they are older than poltime

**0x00055GENERIC\_BAD\_PARAMETER**

User has set poltime to zero

**0x00056GENERIC\_INVALID\_COMMAND**

This type is write only

**0x00057GENERIC\_ADDRESS\_RANGE\_ERROR**

More points were requested than defined by the user

**0x00058GENERIC\_SOFTWARE\_ERROR**

Invalid Object read tag object sub field. Check for the correct DNPR.dbf file

**0x00059GENERIC\_SOFTWARE\_ERROR**

No matching entry in tag object table can be found. Check for the correct DNPR.dbf file

**0x0005AGENERIC\_INVALID\_DATA**

A point or points in read request not valid

**0x0005BGENERIC\_SOFTWARE\_ERROR**

Cannot find a matching dcb for a write response

**0X0005CGENERIC\_ADDRESS\_RANGE\_ERROR**

Data does not exist in cache

**0x0005DGENERIC\_GENERAL\_ERROR**

Unable to complete write command

**0x0005EGERERIC\_UNIT\_WARNING**

IIN Error detected that will generate an alarm. The ability of the driver to detect any individual member of this

group of alarm conditions can be turned on and off using the Citect.ini parameter MinorIINSelectFilter. The default for the driver is to turn each of these alarms ON.

This Plant SCADA error will only be returned under the following circumstances:

On Plant SCADA write requests for Analog Outputs, CROBs, Counter Freeze and Restart requests. The error will only be returned when there are no other errors considered more serious for that request (e.g. Analog Output/CROB status errors described above).

On a front-end read request (e.g. a tag is displayed) following a back-end periodic poll where the error was detected. This error will only be returned when there are no other errors considered more serious resulting from the back-end poll (low level API application errors have precedence).

If this occurs, it indicates one or more of the selected minor IIN errors have occurred under the circumstances described above. The error code cannot differentiate which of the enabled IIN flag(s) has caused the IIN error. To accomplish this, a "Debug \* ALL" command should be issued in the kernel, and the individual IIN flags will be reported in the syslog through the protocol diagnostic reporting.

IIN flags that can be detected are:

**LOCAL\_IIN**: Bit 5 in byte 1 of IIN - Outstation Digital Output(s) are in Local mode (driver cannot control)

**TROUBLE\_IIN**: Bit 6 in byte 1 of IIN - Abnormal condition flagged in Outstation (refer to device profile).

**BUFFER\_OVFL**: Bit 3 in byte 2 of IIN - Outstation lost data from overflow of event buffers

**EXECUTING\_IIN**: Bit 4 in byte 2 of IIN - Request understood but operation already executing

**CONFIG\_IIN**: Bit 5 in byte 2 of IIN - Current configuration in Outstation is corrupt.

#### **0x0005F GENERIC\_GENERAL\_ERROR**

Cannot open database - Plant SCADA form may be open

#### **0x00060 GENERIC\_GENERAL\_ERROR**

Error detected in queue

#### **0x00061 GENERIC\_GENERAL\_ERROR**

Cache refresh late due to poll delays

#### **0x00062 GENERIC\_INVALID\_DATA\_TYPE**

Requested raw type invalid for native type

#### **0x00063 GENERIC\_UNIT\_WARNING**

An error has occurred in a device using a virtual unit tag

#### **0x00064 GENERIC\_UNIT\_WARNING**

General **null pointer** issue, e.g. a request has occurred internally for a device no longer installed

#### **0x00065 GENERIC\_UNIT\_OFFLINE**

A unit's Startup mode has not been configured (for example, "Primary" or "Standby") for the I/O Device, and peer redundancy is in use.

#### **0x00068 GENERIC\_UNIT\_OFFLINE**

IIN error detected that will generate an error alarm. The ability of the driver to detect any individual member of this group of alarm conditions can be turned on and off using the Citect INI parameter MinorIINSelectErrorFilter. Note: the default for the driver is to turn each of these alarms OFF. See error 0.x0005E for the conditions that trigger this error.

#### **0x00069 GENERIC\_UNIT\_REMOTE**

We have done another SELECT while one is in progress.

**0x0006AGENERIC\_UNIT\_REMOTE**

Waiting for an RTU response before coming online

**0x0006B GENERIC\_GENERAL\_ERROR**

Port level indication that transport is unavailable

**0x0006C GENERIC\_UNIT\_WARNING**

Unit is offline for valid reasons, for example, its pair unit is active

**0x0006D GENERIC\_CHANNEL\_OFFLINE**

Still trying to connect to a port and have been prompted to do it again

**0x10002GENERIC\_GENERAL\_ERROR**

Low Level API Application Layer error code 2 - Function code in response not supported

**0x10003GENERIC\_GENERAL\_ERROR**

Low Level API Application Layer error code 3 - Response sequence number does not match the latest request

**0x10004GENERIC\_GENERAL\_ERROR**

Low Level API Application Layer error code 4 - Response fragment received with no active request

**0x10005GENERIC\_GENERAL\_ERROR**

Low Level API Application Layer error code 5 - Object/Variation not supported

**0x10006GENERIC\_GENERAL\_ERROR**

Low Level API Application Layer error code 6 - Object/Qualifier combination not supported

**0x10007GENERIC\_GENERAL\_ERROR**

Low Level API Application Layer error code 7 - Response qualifier code not supported

**0x10008GENERIC\_GENERAL\_ERROR**

Low Level API Application Layer error code 8 - Timeout waiting for application layer response fragment or restart

**0x10009GENERIC\_GENERAL\_ERROR**

Low Level API Application Layer error code 9 - Object/Variation not supported in read response

**0x1000AGENERIC\_GENERAL\_ERROR**

Low Level API Application Layer error code 10 - Object/Variation not supported in operate response

**0x1000BGENERIC\_GENERAL\_ERROR**

Low Level API Application Layer error code 11 - Error considered serious by low level API. Internal indications bits relating to remote device parsing of last request indicate either:

Device does not recognize requested objects (Bit 1 in byte 2 of IIN).

Outstation rejected application request as invalid or out of range due to formatting errors (Bit 2 in byte 2 of IIN).

Function code not implemented (Bit 0 in byte 2 of IIN).

**0x1000CGENERIC\_GENERAL\_ERROR**

Low Level API Application Layer error code 12 - Response does not match request for 'Select before Operate'

**0x1000DGENERIC\_GENERAL\_ERROR**

Low Level API Application Layer error code 13 - Response to Restart message does not include time object

**0x1000EGERIC\_GENERAL\_ERROR**

Low Level API Application Layer error code 14 - Object/Variation not supported in write response

**0x1000FGENERIC\_GENERAL\_ERROR**

Low Level API Application Layer error code 15 - Invalid device number (index) in application layer transmit queue

**0x10010GENERIC\_GENERAL\_ERROR**

Low Level API Application Layer error code 16 - Response to Delay measurement does not include time object

**0x10011GENERIC\_GENERAL\_ERROR**

Low Level API Application Layer error code 17 - Message not processed as unsolicited responses disabled

**0x10012 GENERIC\_GENERAL\_ERROR**

Low Level API Application Layer error code 18 - Invalid application layer frame received

**0x20001GENERIC\_GENERAL\_ERROR**

Analog Output/CROB Status Value 1 - Bad control request - OPERATE received after SELECT timeout (This Plant SCADA error will only be returned on Plant SCADA Write requests)

**0x20002GENERIC\_GENERAL\_ERROR**

Analog Output/CROB Status Value 2 - Bad control request - OPERATE received without prior SELECT (This Plant SCADA error will only be returned on Plant SCADA Write requests)

**0x20003GENERIC\_GENERAL\_ERROR**

Analog Output/CROB Status Value 3 - Bad control request - formatting errors in control request (This Plant SCADA error will only be returned on Plant SCADA Write requests)

**0x20004GENERIC\_GENERAL\_ERROR**

Analog Output/CROB Status Value 4 - Bad control request - control request not supported for point (This Plant SCADA error will only be returned on Plant SCADA Write requests)

**0x20005GENERIC\_GENERAL\_ERROR**

Analog Output/CROB Status Value 5 - Bad control request - control queue full or point already active (This Plant SCADA error will only be returned on Plant SCADA Write requests)

**0x20006GENERIC\_HARDWARE\_ERROR**

Analog Output/CROB Status Value 6 - Bad control request - not accepted due to hardware problems (This Plant SCADA error will only be returned on Plant SCADA Write requests)

## Driver Statistics

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

- Do not use the kernel for normal Plant SCADA operation. The kernel is only for diagnostics and debugging purposes.
- Configure your security so that only approved personnel can view or use the kernel.
- Do not view or use the kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:**

Items will update on a per Port basis. Non " \* " items (14 on) are global to the driver.

---

Items B14 to B19 appear on the second port.

---

Items 14 & 15 will NOT be reset when driver stats are cleared. These stats are for actual counts which cannot be cleared.

---

## Logging

The DNPR driver can log combinations of trace levels across different categories. This is achieved by setting the following INI parameters.

### [DNPR]DebugLevel

This parameter allows you to define the trace level. The options include:

Option	Description
WARN	Output warning messages.
ERROR	Output error messages.
TRACE	Output trace messages.
DEBUG	Output debug messages.
ALL	All of the above.

### [DNPR]DebugCategory

This parameter allows you to enable logging for a particular category of trace. The options include:

Option	Description
PROT	Protocol level debug.
CACHE	Cache debug.
STATE	Unit state transitions.
EVENT	Events cache data.
DCB	Front end driver trace.
BEND	Backend trace.
MISC	Any other debugging.
ALL	All of the above.

In both cases, you can use any combination of the available options separated by a pipe character ( | ).

## Example

```
[DNPR]
DebugLevel=WARN | ERROR | TRACE
DebugCategory=ALL
```

In most cases, you can set the DebugLevel and DebugCategory parameters to ALL. If this results in a large, cumbersome log file that causes older data to be overwritten, you can consider refining the debug options.

**Note:** These parameters do not have a value specified by default. To enable logging for this driver, you need to manually configure these parameters in your INI file.

The events generated by the DNPR driver are logged in the following Plant SCADA syslog file:

```
syslog.IOServer.<Cluster name>.<IO Server name>.dat
```

This file is located in the directory specified by the Citect.ini parameter **[CtEdit]Logs**. By default, this location will be:

```
%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Logs
```

The size of the syslog file and its rollover process is configured via the **[Debug]SysLogSize** and **[Debug]SysLogArchive** INI parameters. For more information, see the topic *Debug Parameters* in the Plant SCADA documentation.

**Note:** If the number of messages being processed for a driver exceeds the value specified by the Plant SCADA parameter **[Kernel]ErrorBuffers**, the pending messages will not be logged. If this occurs, you can increase the value of this parameter so that all the messages generated by the driver are reported. The number of messages that are not logged can be monitored via the Kernel's General window under the item "Lost Errors". If you adjust **[Kernel]ErrorBuffers** and Lost Errors are still occurring, you may need to adjust the level of system logging.

## See Also

[Logging/Diagnostic Parameters](#)

[Debugging Messages](#)

## Debugging Messages

### DriverTrace command

The DriverTrace command is still useful to check the communication (DCB requests and replies) between the IOServer and the driver.

Generally speaking, have this turned on with a [debug]drivertracemask of 21c0f. This setting traces things down to the unit level (not data) and allows to understand what state the I/O server thinks a driver is in.

When the issue is data integrity, then the higher volume traces may be needed to confirm the actual error or data value being returned to the I/O server. This mask will be something like fffe (this excludes the CPU calls which you normally don't need).

Drivertrace allows either all ports or a single port to be traced. If the issue is isolated to a single port, then just trace this port. [debug]drivertraceport=PortABC

## Syslog Size

To minimize the chance of missing something, [debug]syslogsize=32000 should be set and the syslog.dat's

deleted or renamed prior to capturing fresh traces.

## DCB Debugging

When the INI parameter DebugCategory is set to DCB, then trace statements will be produced in the driver log file. The trace statements detail the actions undertaken by the driver in order to process requests received by I/O server.

The following legend describes the symbol abbreviations used in the trace statements:

Symbol	Description
ad	I/O point address
ua	Unit address
ut	Unit type
uc	Unit count

## GENERIC Driver

The GENERIC driver is a special pseudo-protocol supported by Memory I/O Devices and Disk I/O Devices. GENERIC is not a real protocol, and does not communicate with any physical I/O device.

The GENERIC driver supports the basic data types, providing a convenient and consistent way to represent memory and disk data.

---

**Note:** There are no communication specific errors associated with the GENERIC driver. Although you cannot use the GENERIC protocol to communicate with a physical I/O device, memory and disk devices can use other protocols.

---

### See Also

[GENERIC Driver Communications Settings](#)

[GENERIC Driver Data Types](#)

## GENERIC Driver Communications Settings

To establish communication with a device, configure the associated **Board**, **Port** and **I/O Device** in Plant SCADA Studio's **Topology** activity.

### Boards

You do not need to complete any board settings.

## Ports

You do not need to complete any port settings.

## I/O Devices

Complete the I/O Devices settings as follows.

- **I/O Device Name**

A name for your Disk I/O Device, for example: DISK\_PLC or MEMORY\_PLC.

Each I/O Device must have a unique name in the Plant SCADA system.

- **I/O Device Number**

A unique number for the Disk I/O Device (1-4095)

Each I/O Device must have a unique number in the Plant SCADA system.

- **I/O Device Address**

Leave this property blank if you are using a memory device. If you are using a disk device, enter the path and file name of the disk file, for example: [RUN]:DSKDRV.DSK

If you are using redundant Disk I/O Devices, you must specify the path to both the primary file and the Standby file in the configuration of both disk I/O Devices.

For example, if this is the Primary Disk I/O Device enter:

**Primary\_File, Standby\_File**

If this is the Standby Disk I/O Device enter:

**Standby\_File, Primary\_File**

Where:

**Primary\_File** is the name (and path) of the Primary Disk I/O Device file. You may use path substitution in this part of the field.

**Standby\_File** is the name (and path) of the Standby Disk I/O Device file. You may use path substitution in this part of the field.

---

**Note:**

1. If the specified Disk I/O Device file is not found, Plant SCADA will create a new (empty) file with the specified file name.

2. To use redundant Disk I/O Devices, you must be using a network that supports peer-to-peer communication.

3. Disk files must have write permission (on both Primary and Standby servers).

4. If a corrupt file is detected during a read or write, Plant SCADA will change the mode to read only, and display an error message.

5. The frequency with which Plant SCADA writes data to the Disk I/O Device(s) is determined by the **[DiskDrv]UpDateTime** parameter.

---

- **I/O Device Protocol**

Enter GENERIC.

- **I/O Device Port Name**

If you are creating a disk device, you must enter DISK.

If you are creating a memory device, you must enter MEMORY.

- **I/O Device Startup Mode**

If you are not using redundant Disk I/O Devices, leave this property blank.

If you are using redundant Disk I/O Devices, use either:

**PrimaryEnable** immediate use of this Disk I/O Device

**StandbyWrite** This Disk I/O Device will remain unused until the computer with the primary Disk I/O Device configured becomes inoperable. All write requests sent to the primary Disk I/O Device are also sent to this Disk I/O Device.

---

**Note:** To use StandbyWrite mode, you must also configure an I/O Disk Device in the primary server. Both I/O Devices must have the same I/O Device name and number.

- **I/O Device Log Write, Log Read, Cache and Cache Time**

Leave these properties blank.

## See Also

[GENERIC Driver Data Types](#)

## GENERIC Driver Data Types

The following is a list of data types for the Generic Driver:

Data Type	Address Format	Short Format	where # is	Range
DIGITAL	Digital#	D#	0 to 64000	0 or 1
INTEGER	Int#	I#	0 to 32752	-32,768 to 32,767
LONG	Long#	L#	0 to 16376	-2,147,483,648 to 2,147,483,647
REAL	Real#	R#	0 to 16376	-3.4E38 to 3.4E38
STRING	String#	S#	0 to 510	128 bytes

**Note:** Cicode INT types are 32 bit (4 bytes), whereas PLC INT types are only 16 bit (2 bytes).

**STRING Handling** - Plant SCADA by default supports 8 bit characters (a byte). Though the GENERIC driver supports the read and write of 8 bit characters, the PLC based devices support depends on the format of the data in the end device, it might be a 7 bit character.

On multi byte operating systems, then 2 byte character use means that only 64 characters could be displayed per string for the GENERIC driver.

## GETCP Driver

The GETCP protocols supports communication with the GE Fanuc series [90-30 ,90-70](#) and RX3i PLCs.

The network communication layer is Ethernet and requires an Ethernet board installed in your computer.

---

**Note:** Plant SCADA is not restricted to using Ethernet. Other network standards, such as Token Ring or FDDI, may

---

be adopted if supported by the PLC.

The maximum request length for the GETCP protocols is 2048 bits.

## Setting Up the Ethernet Board

It is recommended you setup your specific Ethernet card as required and confirm basic network communications before proceeding with the communications configuration for the GETCP driver. Refer to the documentation accompanying your hardware for instruction.

---

**Note:** The same Ethernet card that is being used for Plant SCADA communication can be used for PLC communication

The GE Ethernet interface manual should be consulted for details of installation of the PLC Ethernet unit and options. We recommend using Thin Ethernet (10base2) and using a transceiver at the PLC.

## Setting Up the Software

TCP/IP software must be properly installed before configuring this driver. In particular, the correct values for IP address, subnet mask and default gateway must be set before proceeding.

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

### DANGER

DANGER indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.

**⚠ WARNING**

**WARNING** indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.

**⚠ CAUTION**

**CAUTION** indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.

**NOTICE**

**NOTICE** used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.

**Please Note**

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

**Before You Begin**

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ WARNING****LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

## Supported Devices

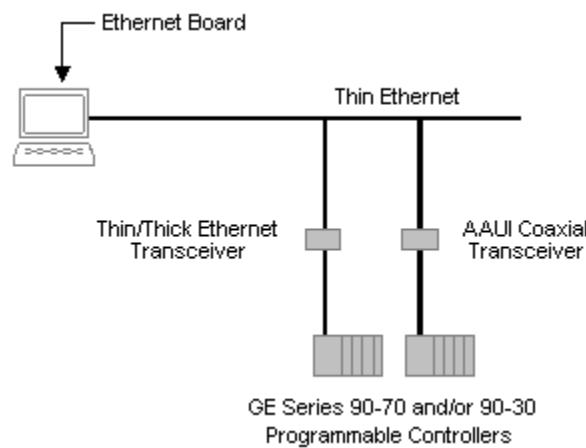
The GETCP driver supports communication with the following devices via Ethernet:

- GE Fanuc 9030 PLCs
- GE Fanuc 9070 PLCs
- GE Fanuc GERx3i via TCP/IP

### GE Fanuc GE9030 via TCP/IP

The GETCP Driver enables TCP/IP communication with GE Fanuc 90-30 PLCs over Ethernet.

Using this method, you can connect to single PLCs or to multiple PLCs as in the following diagram:



## GE Fanuc GE9030 via TCP/IP - Device Address

The information you enter here will be placed in the Special Options field of Plant SCADA's Port settings, with the following format:

-la -Pn -T

Where:

- *a* = the destination IP address in standard Internet dot format. (For example 192.9.2.60)
- *n* = the destination Port number. Often one physical port has several virtual ports, used for different purposes. The default is 18245.
- -T = forces the driver to use TCP (the default), rather than UDP (-U).

See the [GE9030 TCP/IP Overview](#) for more information about this setup.

## GE Fanuc GE9030 via TCP/IP - Hardware Setup

Set up the IP address on the 9030 Ethernet module.

## Hints and Tips

- The 9030 PLCs have an AAUI (Apple Attachment Unit Interface). A transceiver will be needed to connect to other standards (such as Thin Ethernet).
- The transceiver module requires a Signal Quality Error (SQE) test - also known as a heartbeat. Make sure your Ethernet transceiver is set with the SQE test enabled.
- The 9030 PLC CPU can support only one incoming network reference through the Ethernet Interface at a time. Also, the 9030 PLC CPU can support only one programmer connection at a time through either the Ethernet interface or the serial ports. These restrictions will be removed in future upgrades of the PLC and/or Ethernet Interface.

## Plant SCADA Computer Setup

It is recommended that you setup your specific Ethernet card as required and confirm basic network communications before proceeding with the communications configuration for the TCPIP driver. Refer to the documentation accompanying your hardware for instructions.

**Note:** You must also setup the TCP/IP protocol on your Plant SCADA computer.

## GE Fanuc GE9030 via TCP/IP - Communication Settings

To establish communication with a device, the associated Board, Port and I/O Device need to be correctly configured in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure them, use the settings outlined below.

## Boards

Typically, you would use a normal Ethernet card for this communication. Complete the Board settings with the following specific information.

Field	Value
Board Name	Enter TCPIP.
Address	Enter zero (0).
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.

## Ports

Field	Value
Port Name	Any unique number
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank.
Special Options	<p>Enter the destination IP address of Ethernet module. Use the following format:</p> <p><b>-la -Pn -T</b></p> <p>where:</p> <p><b>a</b> = the destination IP address in standard Internet dot format. (For example 192.9.2.60)</p> <p><b>n</b> = the destination Port number. Often one physical port has several virtual ports, used for different purposes. Use this option only if you want to override the default of 18245.</p> <p><b>-T</b> = forces the driver to use TCP (the default), rather than UDP (-U).</p>

## I/O Devices

Field	Value
Address	Leave this field blank.
Protocol	Enter GETCP30.

**GE Fanuc GE9030 via TCP/IP - Data Types**

Data Types	Address Format	Plant SCADA Data Type
Discrete Input	Ia	DIGITAL
Discrete Output	Qa	DIGITAL
System Register	Rf	INT, LONG, REAL, BCD, LONGBCD, STRING
Global	GAh	DIGITAL
Global	GBh	DIGITAL
Global	GCh	DIGITAL
Global	GDh	DIGITAL
Global	GEh	DIGITAL
Temporary Coil	Tg	DIGITAL
Momentary Coil	Ma	DIGITAL
System Status	Sb	DIGITAL (RO)
System Memory A	SAb	DIGITAL
System Memory B	SBb	DIGITAL
System Memory C	SCb	DIGITAL
Analog Input	AIC	INT, LONG, REAL, BCD, LONGBCD, STRING (RO)
Analog Output	AQc	INT, LONG, REAL, BCD, LONGBCD, STRING
PLC Fault Table	PLCFd	BYTE (RO)
IO Fault Table	IOFe	BYTE (RO)

Where

<i>a</i>	1 to 12288
<i>b</i>	1 to 128
<i>c</i>	1 to 8192
<i>d</i>	0 to 683
<i>e</i>	0 to 1355
<i>f</i>	1 to 16384
<i>g</i>	1 to 256
<i>h</i>	1 to 1280

**Note:** Currently the maximum STRING length is 8 bytes. Only the first eight characters are written to the PLC and set the rest to zero. For example: assume the input value is "abcdefghijklmn" , the value in the PLC will be "abcdefghijklmn\0\0\0\0\0\0\0\0".

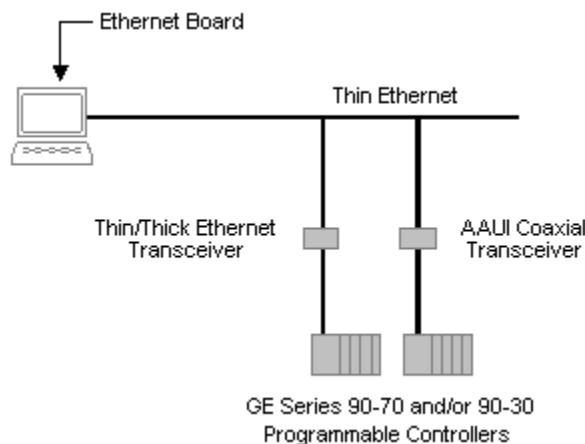
#### Examples:

Data Type	DIGITAL
Address	I0025
Comment	Discrete Digital Input 25
Data Type	INT
Address	R0100
Comment	System Register 100

#### GE Fanuc GE9070 via TCP/IP

The GETCP Driver enables TCP/IP communication with GE Fanuc 90-70 PLCs over Ethernet.

Using this method, you can connect to single PLCs or to multiple PLCs as in the following diagram:



## GE Fanuc GE9070 via TCP/IP - Device Address

The information you enter here will be placed in the Special Options field of Plant SCADA's Port settings, with the following format:

-la -Pn -T

where:

- *a* = the destination IP address in standard Internet dot format. (For example, 192.9.2.60)
- *n* = the destination Port number. Often one physical port has several virtual ports, used for different purposes. The default is 18245.
- -T = forces the driver to use TCP (the default), rather than UDP (-U).

See the [GE9070 TCP/IP Overview](#) for more information about this setup.

## GE Fanuc GE9070 via TCP/IP - Hardware Setup

Set up the IP address on the 9070 Ethernet module.

## Hints and Tips

The 9070 PLC has a Thick Ethernet port. A transceiver will be needed to connect to other standards (such as Thin Ethernet).

- The transceiver module requires a Signal Quality Error (SQE) test - also known as a heartbeat. Make sure your Ethernet transceiver is set with the SQE test enabled.

## Plant SCADA Computer Setup

It is recommended that you setup your specific Ethernet card as required and confirm basic network communications before proceeding with the communications configuration for the TCPIP driver. Refer to the documentation accompanying your hardware for instruction.

---

**Note:** You must also setup the TCP/IP protocol on your Plant SCADA computer.

---

## GE Fanuc GE9070 via TCP/IP - Communication Settings

To establish communication with a device, the associated Board, Port and I/O Device need to be correctly configured in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure them, use the settings outlined below.

## Boards

Typically, you would use a normal Ethernet card for this communication. Complete the Board settings with the following information.

Field	Value
Board Name	Enter TCPIP.
Address	Enter zero (0).
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.

## Ports

Field	Value
Port Name	This field is user defined.
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank.
Special Options	<p>Enter the destination IP address of Ethernet module.          Use the following format:  <code>-la -Pn -T</code>          where:  <math>a</math> = the destination IP address in standard Internet dot format. (For example 192.9.2.60)  <math>n</math> = the destination Port number. Often one physical port has several virtual ports, used for different purposes. Use this option only if you want to override the default of 18245.  <math>-T</math> = forces the driver to use TCP (the default), rather than UDP (-U).</p>

## I/O Devices

Field	Value
Address	Leave this field blank.
Protocol	Enter GETCP70.

## GE Fanuc GE9070 via TCP/IP - Data Types

Data Types	Address Format	Plant SCADA Data Type
Discrete Input	Ia	DIGITAL
Discrete Output	Qa	DIGITAL
System Register	Rf	INT, LONG, REAL, BCD, LONGBCD, STRING
Global	GAh	DIGITAL
Global	GBh	DIGITAL
Global	GCh	DIGITAL
Global	GDh	DIGITAL
Global	GEh	DIGITAL
Temporary Coil	Tg	DIGITAL
Momentary Coil	Ma	DIGITAL
System Status	Sb	DIGITAL (RO)
System Memory A	SAb	DIGITAL
System Memory B	SBb	DIGITAL
System Memory C	SCb	DIGITAL
Analog Input	AIC	INT, LONG, REAL, BCD, LONGBCD, STRING (RO)
Analog Output	AQc	INT, LONG, REAL, BCD, LONGBCD, STRING
PLC Fault Table	PLCFd	BYTE (RO)
IO Fault Table	IOFe	BYTE (RO)

Where

a	1 to 12288
b	1 to 128
c	1 to 8192
d	0 to 683

<i>e</i>	0 to 1355
<i>f</i>	1 to 16384
<i>g</i>	1 to 256
<i>h</i>	1 to 1280

**Note:** Currently the maximum STRING length is 8 bytes. Only the first eight characters are written to the PLC and set the rest to zero. For example: assume the input value is "abcdefghijklmn" , the value in the PLC will be "abcdefghijkl\0\0\0\0\0\0\0".

### Examples

Data Type	DIGITAL
Address	I0025
Comment	Discrete Digital Input 25
Data Type	INT
Address	R0100
Comment	System Register 100

### GE Fanuc GERx3i via TCP/IP

The GETCP Driver enables TCP/IP communication with GE Fanuc RX3i PLCs over Ethernet.

Using this method, you can connect to single PLCs or to multiple PLCs.

#### GE Fanuc GERx3i via TCP/IP - Device Address

The information you enter here will be placed in the Special Options field of Plant SCADA's Port settings, with the following format:

-la -Pn -T

where:

- *a* = the destination IP address in standard Internet dot format. (For example 192.9.2.60)
- *n* = the destination Port number. Often one physical port has several virtual ports, used for different purposes. The default is 18245.
- -T = forces the driver to use TCP (the default), rather than UDP (-U).

#### GE Fanuc GERx3i via TCP/IP - Hardware Setup

Set up the IP address on the Rx3i Ethernet module.

## Hints and Tips

- The Rx3i PLC has a Thick Ethernet port. A transceiver will be needed to connect to other standards (such as Thin Ethernet).
- The transceiver module requires a Signal Quality Error (SQE) test - also known as a heartbeat. Make sure your Ethernet transceiver is set with the SQE test enabled.

## Plant SCADA Computer Setup

It is recommended that you setup your specific Ethernet card as required and confirm basic network communications before proceeding with the communications configuration for the TCPIP driver. Refer to the documentation accompanying your hardware for instruction.

**Note:** You must also setup the TCP/IP protocol on your Plant SCADA computer.

### GE Fanuc GERx3i via TCP/IP - Communication Settings

To establish communication with a device, the associated Board, Port and I/O Device need to be correctly configured in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure them, use the settings outlined below.

## Boards

Typically, you would use a normal Ethernet card for this communication. Complete the Board settings with the following information.

Field	Value
Board Name	Enter TCPIP.
Address	Enter zero (0).
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.

## Ports

Field	Value
Port Name	Any unique number
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.

Field	Value
Stop Bits	Leave this field blank.
Parity	Leave this field blank.
Special Options	<p>Enter the destination IP address of Ethernet module. Use the following format:</p> <p><b>-Ia -Pn -T</b></p> <p>where:</p> <p><b>a</b> = the destination IP address in standard Internet dot format. (For example 192.9.2.60)</p> <p><b>n</b> = the destination Port number. Often one physical port has several virtual ports, used for different purposes. Use this option only if you want to override the default of 18245.</p> <p><b>-T</b> = forces the driver to use TCP (the default), rather than UDP (-U).</p>

## I/O Devices

Field	Value
Address	Leave this field blank.
Protocol	Enter GETCP3.

### GE Fanuc GERx3i via TCP/IP - Data Types

**Note:** GETCP driver will not allow writing to individual bits in W register. User can write 16 bit value to W register.

Data Types	Address Format	Plant SCADA Data Type
Discrete Input	Ia	DIGITAL
Discrete Output	Qa	DIGITAL
System Register	Rf	INT, LONG, REAL, BCD, LONGBCD, STRING
Retentive Bulk Memory Register	Wi	INT, LONG, REAL, BCD, LONGBCD, STRING
Global	GAh	DIGITAL
Global	GBh	DIGITAL

Data Types	Address Format	Plant SCADA Data Type
Global	GCh	DIGITAL
Global	GDh	DIGITAL
Global	GEh	DIGITAL
Temporary Coil	Tg	DIGITAL
Momentary Coil	Ma	DIGITAL
System Status	Sb	DIGITAL (RO)
System Memory A	SAb	DIGITAL
System Memory B	SBb	DIGITAL
System Memory C	SCb	DIGITAL
Analog Input	AIC	INT, LONG, REAL, BCD, LONGBCD, STRING (RO)
Analog Output	AQc	INT, LONG, REAL, BCD, LONGBCD, STRING
PLC Fault Table	PLCFd	BYTE (RO)
IO Fault Table	IOFe	BYTE (RO)

Where

a	1 to 32768
b	1 to 128
c	1 to 32640
d	0 to 683
e	0 to 1355
f	1 to 32768
g	1 to 1024
h	1 to 1280
i	1 to 2621441

**Note:** Currently the maximum STRING length is 8 bytes. Only the first eight characters are written to the PLC and set the rest to zero. For example: assume the input value is "abcdefghijklmn" , the value in the PLC will be "abcdefghijkl\0\0\0\0\0\0\0".

### Examples

Data Type	DIGITAL
Address	I0025
Comment	Discrete Digital Input 25
Data Type	INT
Address	R0100
Comment	System Register 100

## GETCP Driver Parameters

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support for this product regarding undocumented features.

### [GETCP]Block

A value (bytes) used by the I/O Server to determine if two or more packets can be blocked into one data request before being sent to the I/O Device. For example, if you set the value to 10, and the I/O Server receives two simultaneous data requests - one for byte 3, and another for byte 8 - the two requests will be blocked into a single physical data request packet. This single request packet is then sent to the I/O Device, saving on bandwidth and processing.

**Allowable Values** 5 to 256

**Default Value** 256

### [GETCP]Delay

The period (in milliseconds) to wait between receiving a response and sending the next command.

**Allowable Values** 0 to 300 (milliseconds)

**Default Value** 0

### [GETCP]MaxPending

The maximum number of pending commands that the driver holds ready for immediate execution.

**Allowable Values** 1 to 32

**Default Value** 15

### [GETCP]PollTime

The interrupt or polling service time (in milliseconds). Setting the polling time to 0 puts the driver in interrupt mode.

**Allowable Values** 0 to 300 (milliseconds)

**Default Value** 0

### [GETCP]Retry

The number of times to retry a command after a timeout.

**Allowable Values** 0 to 8

**Default Value** 1

### [GETCP]Timeout

Specifies how many milliseconds to wait for a response before displaying an error message.

**Allowable Values** 0 to 32000 (milliseconds)

**Default Value** 1000

### [GETCP]WatchTime

The frequency (in seconds) that the driver uses to check the communications link to the I/O Device.

**Allowable Values** 0 to 128 (seconds)

**Default Value** 30

## GETCP Driver Specific Errors

When a hardware error occurs, Plant SCADA generates an alarm, and displays the alarm on the hardware alarm page (in the alarm description of the hardware alarm). To see the error number, make sure you have Alarm Category 255 defined with a display format that includes {ErrDesc,15} {ErrPage,10}.

There are a number of generic errors that are common to all protocols. In some cases only the generic error is available, though often both the generic error and a specific error are given.

The following errors, listed in (hexadecimal) sequence, are specific to this protocol. Plant SCADA displays the error number and description for common protocol-specific errors. Uncommon errors are not contained in the Plant SCADA error database, in which case Plant SCADA will only display the error number.

You may require additional information to enable you to rectify an error. This information should be detailed in the documentation that accompanied the I/O Device (or network). If, after reviewing all documentation, you cannot rectify an error, contact Technical Support.

#### 0x05F4

Invalid input parameters in request.

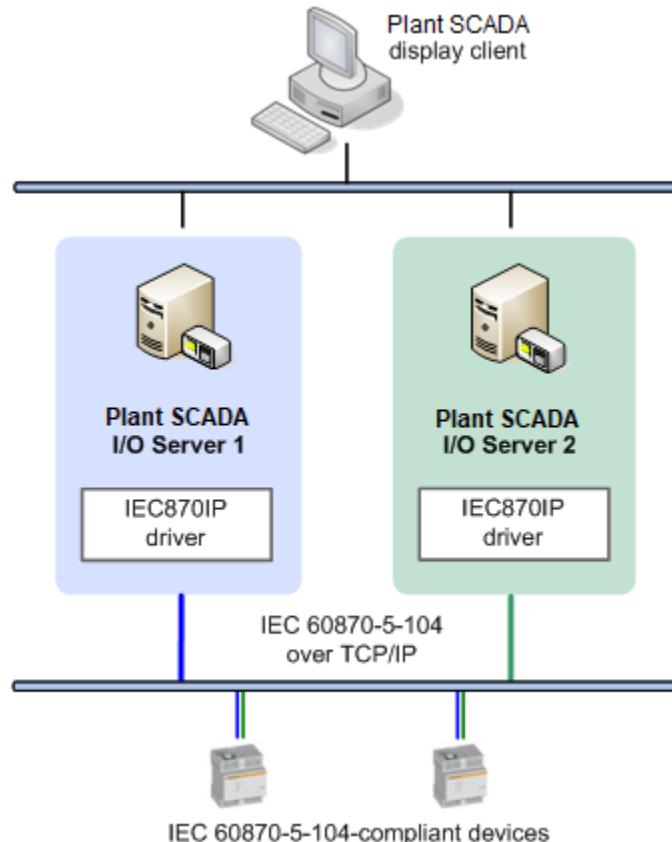
#### 0x05C3

Text length does not match traffic type.

## IEC870IP Driver

The IEC870IP driver enables Plant SCADA to communicate via TCP/IP with devices that use the IEC 60870-5-104 communication protocol.

The following diagram demonstrates a simple system configuration, with Plant SCADA connected to IEC 60870-5-104 devices via the IEC870IP driver hosted on a single pair of redundant I/O servers.



### The Driver Runtime Interface (DRI)

With the release of version 7.20, Citect SCADA supported the retrieval of time-stamped data directly from field devices. This capability is enabled by the Driver Runtime Interface (DRI), a component of Plant SCADA that is used by the IEC870IP driver to directly update time-stamped digital alarms, time-stamped analog alarms and event-based trends. For more information, see the topic *Retrieving time-stamped data from field devices* in the main Plant SCADA help.

---

**Note:**

- As version 4.05.01.0 (or later) of the IEC870IP driver relies on the DRI, it requires Citect SCADA version 7.20 (or later) to operate successfully.
  - This documentation was created to support version 4.05.01.0 (or later) of the IEC870IP driver. Due to significant changes to the driver with this release, this documentation should not be used with an earlier version of the driver.
-

## See Also

- [Supported Devices](#)
- [Configuring Variable Tags](#)
- [Value Quality Support](#)
- [Value Timestamping](#)
- [Advanced Configuration and Maintenance](#)
- [Troubleshooting](#)

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

#### DANGER

**DANGER** indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.

#### WARNING

**WARNING** indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.

#### CAUTION

**CAUTION** indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.

**NOTICE**

NOTICE used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.

**Please Note**

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

**Before You Begin**

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

** WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

** WARNING****LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and

Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

## Supported Devices

The IEC870IP driver allows Plant SCADA to communicate with devices that use the IEC 60870-5-104 communication protocol. The driver is limited to the 104 extension of the protocol, as it uses TCP/IP as its network interface.

---

**Note:** The driver can only support up to 256 devices per I/O server. If you configure more than this amount, the 257th device will not come online, and a hardware alarm will be raised.

---

### Interoperability

The IEC 60870-5-104 standard defines a set of parameters and alternatives it can support, compared against those used by the 101 standard. This is outlined in the standard in the interoperability list.

Any telecontrol equipment or systems that are designed to adhere to the standard can implement a subset of the alternatives defined in this list, based on specific requirements. To view the subset of options used by the IEC870IP driver, see the appendix [IEC 870-5-104 Interoperability List](#).

## See Also

[Setting up Device Communications](#)

## Setting up Device Communications

The information included in the following topics is required to establish communication between Plant SCADA and an IEC60870-5-104 device connected via the IEC870IP driver.

- [Using the Express Communications Wizard](#) - the information you will need if you are using the Express Communications Wizard to connect to a device.
- [Communications Settings](#) - the recommended board, port and devices settings for manual configuration.

## Using the Express Communications Wizard

To set up communication using the Plant SCADA Express Communications Wizard, you will need to provide an **Address**, an **IP address** and **Port** number.

- **Address**

The address to use for an IEC 870-5-104 device is the common ADSU address. This value is configured on the device and will be a number between 1 and 65535.

- **IP address**

The IP address of the device.

- **Port**

The default port number for the IEC 870-5-104 protocol is 2404.

---

**Note:** It is recommended you use TCP protocol when setting up an IEC 870-5-104 device.

---

This information is placed in the **Special Options** field of the **Ports** settings by the Express Communications

Wizard using the following format:

—I<device IP address> —P<port number> —<T (TCP)>

For example:

—I192.1.1.182—P2404 —T

## Communications Settings

To establish communication with a device, the associated Board, Port and I/O Device need to be correctly configured in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to configure the settings manually, use the values outlined below.

### Board settings

The Boards view lists boards used in the Plant SCADA project. Each board record defines a separate board within the project.

The IEC870IP driver is a board-level driver, which means it requires a single board configured for each I/O server.

Field	Value
Board Name	Any user-defined name (e.g. "Board1")
Board Type	TCP/IP
Address	0 (zero)
I/O Port	Leave this field blank
Interrupt	Leave this field blank
Special Options	Leave this field blank
Comment	Any useful comment (not required by the driver)

### Port settings

The Ports view lists the ports used in the Plant SCADA project. Each port record defines a separate port within the project.

A separate port should be created for each IEC 870-5-104 device.

Field	Value
Port Name	Any user-defined name (e.g. "Port1")
Port Number	The default port number for the IEC 870-5-104 protocol is 2404.
Board Name	Refers to the board defined in the Boards form

Field	Value
Baud Rate	Leave this field blank
Data Bits	Leave this field blank
Stop Bits	Leave this field blank
Parity	Leave this field blank
Special Options	<p>Use the address for the IEC 870-5-104 device using the following format:</p> <p>—I&lt;device IP address&gt; —P&lt;port number&gt; —&lt;T (TCP)&gt;</p> <p>For example:</p> <p>—I192.1.1.182—P2404 —T</p> <p>The default port number for the IEC 870-5-104 protocol is 2404</p>
Comment	Any useful comment (not required by the driver)

## I/O Device settings

The I/O Devices view lists I/O devices used in the Plant SCADA project. Each device record defines a device within the project.

A Plant SCADA I/O device represents a single IEC 870-5-104 device.

Field	Value
Name	Any user-defined name (e.g. "IODevice1")
Number	I/O device number needs to be unique for each I/O device, but identical for redundant devices.
Address	The address to use for an IEC 870-5-104 device is the common ADSU address. This value is configured on the device and will be a number between 1 and 65535.
Protocol	IEC870IP
Port Name	Refers to the port defined in the Ports form
Comment	Any useful comment. This field is user defined and not required by the driver

## Configuring Variable Tags

The IEC870IP driver supports the following variable tag types:

- [Input Tags](#) - tags that allow process information to be passed to Plant SCADA in a monitoring direction
- [Read Tags](#) - tag that send a read command to request data
- [Output Tags](#) - tag that allow process information to be sent in a control direction
- [Control Tags](#) -tags that can be used to send commands in a control direction
- [Array Tags](#) - tags that represent array elements
- [Virtual Tags](#) - tags that retrieve the status of specific commands.

---

**Note:** The driver supports unstructured tag addresses (e.g. "<IOA>"), as well as structured tag addresses (e.g. "<IOA1>.<IOA2>.<IOA3>"). For more information, see [Structured Tag Addressing](#).

---

### Read Tags

Read tags can be used to retrieve data from IO points that require a read command. See [Using Read Commands to Retrieve Data](#).

<IOA> is used to represent an information object address, defined as a decimal number in the range 0 — 16777215.

**Note:** The tags with the following addresses are only updated when the driver receives a timestamped value directly from a device.

- RSD<IOA>
- RSI<IOA>
- RSL<IOA>
- RSF<IOA>
- RSQD<IOA>

See [Timestamp Handling](#) for more information on the correct syntax to use for your input tags.

Address format	Plant SCADA type	IEC 60870 type	Description
RD<IOA> RSD<IOA>	DIGITAL	Single point Double point	<p>Read command for digital value: Example: RD100</p> <p><b>Note:</b> By default, digital tags report double point states as:</p> <ul style="list-style-type: none"><li>• 1 "On"</li><li>• 0 "Off"</li><li>• 0 "Indeterminate"</li><li>• 0 "Intermediate".</li></ul> <p>This behaviour can be changed to report just the "On" and "Off" states using the INI parameter <a href="#">Data Interrogation</a></p>

Address format	Plant SCADA type	IEC 60870 type	Description	
		<a href="#">Parameters.</a>		
RI<IOA> RSI<IOA>	INTEGER	Measured value: normalized and scaled value Step Position Double Point	Read command for integer value: Example: RI200	
RL<IOA> RSL<IOA>	LONG	Integrated totals	Read command for long value: Example: RL300	
RF<IOA> RSF<IOA>	FLOAT	Short Floating Point	Read command for float value: Example: RF350	
RQD<IOA> RSQD<IOA>	INTEGER	Quality descriptor	Read command for IEC 60870 quality flags and the IEC870IP driver specific flags Example: RQD350	

The address format used in the table above demonstrates unstructured tag addresses (i.e. "<IOA>"). The IEC870IP driver also supports structured tag addresses (e.g. "<IOA1>.<IOA2>.<IOA3>"). For more information, see [Structured Tag Addressing](#).

---

**Note:** Care should be taken when configuring read tags in a system because the read process is less efficient than the usual spontaneous update process. Due to the polling mechanism, read tags will consume more system and network resources than other tags. For example, if you have more than 500 read command tags configured, it may impact the tag value update rate on a page as the driver will have to issue a read command for each tag on a page.

## Output Tags

The driver defines a number of tag addresses which can be used to send process information in a control direction. The output tags are write only.

The following tags are used to perform direct write operations.

<IOA> is used to represent an information object address, defined as a decimal number in the range 0 — 16777215.

---

**Note:** To send the command with the time tag *CP56Time2a*, use the address with a suffix "T".

Address format	Plant SCADA type	IEC 60870 command	Description
C<IOA> CT<IOA>	DIGITAL	Single command Double command	Digital output: Double commands are only ever used for trip/

Address format	Plant SCADA type	IEC 60870 command	Description
		Regulating step command	<p>close operations, so only 2 states are valid. Hence they will be treated as single digital output points by Plant SCADA, and translated in the driver.</p> <ul style="list-style-type: none"> <li>• 0 becomes (0,1) (trip)</li> <li>• 1 becomes (1,0) (close)</li> </ul> <p>Regulating step commands are similar to single commands, except that writing a 0 means "next step lower", and 1 means "next step higher".</p>
CI<IOA> CIT<IOA>	INTEGER	Set point command: normalized and scaled value	Integer output
CF<IOA> CFT<IOA>	FLOAT	Set point command: short floating point number	Float point output

The following tags are used for digital select/execute operations.

Address format	Plant SCADA type	Description
C<IOA>.Sel CT<IOA>.Sel	DIGITAL	Transmits select request
C<IOA>.Desel CT<IOA>.Desel	DIGITAL	Transmits break off command (deselect request)
C<IOA>.Ex CT<IOA>.Ex	DIGITAL	Transmits execute request
CI<IOA>.Sel CIT<IOA>.Sel	INTEGER	Transmits set point command(select request) for normalized or scaled value.
CI<IOA>.Desel CIT<IOA>.Desel	INTEGER	Transmits set point command(deselect request) for normalized or scaled value.
CI<IOA>.Ex CIT<IOA>.Ex	INTEGER	Transmits set point command(execute request) for

Address format	Plant SCADA type	Description
		normalized or scaled value.
CF<IOA>.Sel CFT<IOA>.Sel	FLOAT	Transmits set point command(select request) for short floating point.
CF<IOA>.Desel CFT<IOA>.Desel	FLOAT	Transmits set point command(deselect request) for short floating point.
CF<IOA>.Ex CFT<IOA>.Ex	FLOAT	Transmits set point command(execute request) for short floating point.

**Note:** Double commands, regulating step commands and set-point commands for scaled value are only supported if a tag is configured in the IOA configuration file with the appropriate information. See [Configuring Commands for Individual IOAs](#).

The following tag is used to get a cause of transmission code passed in monitoring direction after completion of single and double commands, regulating step command and set point commands. This tag has read/write access.

Address format	Plant SCADA type	Description
C<IOA>.Conf CT<IOA>.Conf	INTEGER	<p>Confirmation code:</p> <ul style="list-style-type: none"> <li>• 0x7 - activation confirmation</li> <li>• 0x9 - deactivation confirmation</li> <li>• 0x10 - activation termination</li> <li>• 0x44 - unknown type identification</li> <li>• 0x45 - unknown cause of transmission</li> <li>• 0x46 - unknown common address of ASDU</li> <li>• 0x47 - unknown information object address</li> </ul>

**Note:** The address format used in the tables above demonstrates unstructured tag addresses (i.e. "<IOA>"). The IEC870IP driver also supports structured tag addresses (e.g. "<IOA1>.<IOA2>.<IOA3>"). For more information, see [Structured Tag Addressing](#).

## Control Tags

The driver defines a number of tag addresses which can be used to send system information in a control direction. These tags are write-only.

Address format	Plant SCADA type	IEC 60870 command	Description
FORCEGI	DIGITAL	Interrogation command	Writing "1" to this digital

Address format	Plant SCADA type	IEC 60870 command	Description
			register will force a general interrogation of the device.
FORCECS	DIGITAL	Clock synchronization command	Writing "1" to this digital register will force a clock synchronization of the device.
FORCECI	DIGITAL	Counter interrogation command	Writing "1" to this digital register will force a counter interrogation of the device (without freeze or reset of the counters).
COUNTFZ	DIGITAL	Counter interrogation command	Writing "1" to this digital register will perform a counter freeze on the device.
COUNTRS	DIGITAL	Counter interrogation command	Writing "1" to this digital register will perform a counter reset on the device.
COUNTFZRS	DIGITAL	Counter interrogation command	Writing "1" to this digital register will perform a counter freeze and reset on the device.

## Array Tags

You can configure array tags in Plant SCADA for an IEC 60870-5-104 device using the following syntax:

**D<IOA>[array size]**

Be aware, however, that each element of an array is delivered independently. This means Plant SCADA will receive separate array value updates after each array element value changes.

### Example

Assume a user configures an array I100[5], and its initial value is {0,5,0,0,10}.

If the values for the first two elements change to 4 and 10, Plant SCADA will first display {4,5,0,0,10} and then {4,10,0,0,10}.

The quality of the array tag will be equal to the worst quality of the array elements, and the timestamp will be equal to the latest timestamp of the array elements.

---

**Note:** The address format used above demonstrates an unstructured tag address (i.e. "<IOA>"). The IEC870IP driver also supports structured tag addresses (e.g. "<IOA1>.<IOA2>.<IOA3>"). For more information, see [Structured Tag Addressing](#).

## Virtual Tags

The driver defines tag addresses which can be used to retrieve the status of the commands listed in the table below. These tags are read-only.

The driver will update the value of these tags after a command has completed successfully, or failed to complete successfully. If the command completes successfully, the tag value will be set to False otherwise it will be set to True.

Address format	Plant SCADA type	Status for IEC 60870 command
IsGI	DIGITAL	Interrogation command
IsTimeSync	DIGITAL	Clock synchronization command
IsDelayAcq	DIGITAL	Delay acquisition command
IsProtoReset	DIGITAL	Reset command
IsLinkStatus	DIGITAL	Link status command
IsProcessControl	DIGITAL	Control command

## Structured Tag Addressing

The IEC870IP driver supports structured tag addresses using the following format:

<IOA1>.<IOA2>.<IOA3>

When constructing a structured address, each element represents a decimal number in the range 0 — 255. For example:

0.1.126

This can be used instead of an unstructured address (i.e. <IOA>), which represents the information object address as a decimal number in the range 0 — 16777215.

## See Also

[Configuring Variable Tags](#)

## Value Quality Support

The IEC870IP driver can confirm the quality of values received from a device through support for the quality flags implemented by the IEC870-5-101 protocol.

The quality flags are defined by a quality descriptor that is delivered with each value received from a device. The IEC870IP driver then performs a conversion of the quality flags it receives, providing a description of value quality based upon OPC quality.

## See Also

[Quality Descriptors](#)  
[Quality Mapping](#)  
[Value Timestamping](#)

## Quality Descriptors

The IEC870IP driver uses the quality flags implemented by the IEC870-5-101 protocol to determine the quality of a value.

The quality descriptor word that defines these flags has bits defined as follows:

Bit number	Description
0 (least significant)	For Single Points: If the StoreDigitalInQual parameter is 1, this bit contains the current point value  For Double Points: If the StoreDigitalInQual parameter is set this bit contains the lower bit of the point value (DPI)  For Counters: Overflow or counter carry indication
1	Transient for integer inputs  For Double Points (if the StoreDigitalInQual parameter is 0 ): Indeterminate / intermediate state for double digital inputs (DDI)  For Double Points (if the StoreDigitalInQual parameter is 1 ): The upper bit of the point value (DPI)
2	For Double Points (if the StoreDigitalInQual parameter is 0 ): Invalid state for double digital inputs (DDI)
3	Blocked
4	Substituted
5	Not topical
6	Invalid
7	Counter adjusted since last reading
8... 12	Counter Sequence number (number from 0 to 31)

Bit number	Description
13	Quality Descriptor out of date
14	Timestamp was provided by PC, not RTU
15 (most significant)	Timestamp from RTU has "invalid" set. (This bit is undefined if timestamp is provided by PC, i.e. bit 14 is set.)

**Note:** The StoreDigitalInQual parameter can be used to allow the state of a point to be stored within a quality descriptor. However, this will impact the way double point digital values are read by the IEC870IP driver. See [Data Interrogation Parameters](#) for more information.

Once a quality flag is received by the driver, it is mapped to an OPC-based quality value that is presented in the Plant SCADA project.

## See Also

[Quality Mapping](#)

## Quality Mapping

The driver performs a conversion of IEC60870 quality flags into a Plant SCADA quality representation. This conversion is based on OPC quality.

The following table describes the mapping of the IEC 60870 quality flags into Plant SCADA quality. The flags are listed in the order of priority so that if multiple quality flags are set, the driver will push the OPC quality flags for the quality bit with the highest priority:

IEC6870 quality bit	OPC quality flags
Invalid	General: Bad Substatus: Device Failure
Blocked	General: Uncertain Substatus: Last Usable Value
Not topical	General: Uncertain Substatus: Last Usable Value
Substituted	General: Good Substatus: Local Override
Overflow	General: Uncertain Substatus: Engineering Units Exceeded

**Note:** When using TagRead or TagReadEx Cicode functions, the driver is only able to return an error code for conversion to OPC quality in Plant SCADA. This means that the tag quality will only represent the general component of a quality flag.

## Value Timestamping

The IEC780IP driver pushes values to Plant SCADA as they are received via the DRI.

Depending on the device, the driver may receive timestamped or non-timestamped value updates. When a timestamp value is not provided by a device, it can be manufactured by the driver based on the time the update is received. This behavior can be manipulated by the driver parameter [Data Interrogation Parameters](#).

The way Plant SCADA processes timestamp values can also be impacted by the type of input tags you use. See [Timestamp Handling](#) for more information.

The timestamp value the driver pushes to Plant SCADA is a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601 (UTC).

By default when a timestamped value is received, the values for timestamped and non-timestamped tags will be updated. That behavior can be changed by the [Data Interrogation Parameters](#) ini parameter.

## Timestamp Handling

The IEC60870-5 protocol supports data updates through three types of transmission:

non-timestamped

timestamped

double transmission

Plant SCADA requires a timestamp to be pushed into the I/O server and the rest of the SCADA system for every data update. Where a protocol has not provided a timestamp, a Plant SCADA driver will typically create a timestamp based on the time the data was received.

A special case exists in the IEC870IP driver, where two input tag address options are available:

- "S" prefix addresses are only updated by protocol messages with a timestamp:
  - SD<IOA>
  - SI<IOA>
  - SL<IOA>
  - SF<IOA>
  - SQD<IOA>
- "Non-S" type tags use a driver manufactured timestamp by default:
  - D<IOA>
  - I<IOA>
  - L<IOA>
  - F<IOA>
  - QD<IOA>

For data archiving and event analysis, Plant SCADA offers event trend and timestamped alarm recording options with millisecond accuracy. To avoid superfluous or missing updates due to out of sequence timestamps during startup, active device transitions, or double transmissions, the IEC870IP driver exposes "S" prefix addresses which are only ever updated by protocol messages with a timestamp.

These tags remain #BAD quality at startup until a message with a timestamp arrives. Messages with a value

change but no timestamp will not be pushed to Plant SCADA for these tags.

Double transmission is a IEC60870-5 feature where an RTU will place a high priority value update in its transmit queue as soon as possible after an event. The device will also place a second lower priority message with the same value and also a timestamp.

To capture and action the first high priority data change, a "non-S" tag can be displayed on a page or non-timestamped alarm. Using the "S" prefix tags allows Plant SCADA to archive only the timestamped messages for analysis.

In summary:

- When configuring event trends, timestamped digital or timestamped analog alarms, an "S" prefix tag address should be used.
- When configuring display pages and non-timestamped alarms, the "non-S" tags should be used.

---

**Note:** You need to carefully consider the tag address you use for timestamped alarms and event trends, as the use of a "non-S" tag may cause a delay in the delivery of information to an operator.

---

## The ManufactureTimestamp parameter

The Citect.ini parameter [Data Interrogation Parameters](#) allows one of two options to be selected when configuring "non-S" type tags. The default is to push a driver manufactured timestamp to the I/O server each time for consistency. The alternative option is for the driver to update a tag with a protocol timestamp if one is available. Be aware that this may cause a value displayed on a page to flicker or step backward at times.

## Advanced Configuration and Maintenance

This section provides instructions for advanced configuration and maintenance tasks.

- [Using Read Commands to Retrieve Data](#)
- [Events Handling](#)
- [Configuring Redundancy](#)
- [Customizing a Project using Citect.ini Parameters](#)
- [Configuring Commands for Individual IOAs](#)

### Using Read Commands to Retrieve Data

In some cases, the value for an I/O point can only be retrieved using a read command. For example, a particular I/O point may not support spontaneous events or general interrogation.

Under these circumstances, there are two ways you can read from an I/O point:

- the TagRead function
- the IEC870IP driver's polling engine

In both cases, you will need to configure [Read Tags](#) to successfully retrieve data from field devices. Where a read tag has been used, the driver will send a read command to request data.

#### The TagRead function

The TagRead function will send a read command to the specified IO point and process the response before responding to replying to the function call.

### The polling engine

The IEC870IP driver has an integrated polling engine that allows you to retrieve values for any IO point using regular read command requests sent directly to the device.

The rate at which requests are executed is defined by the subscription rate of the tag. If more than one tag is using the same item, the polling rate will be determined by the highest subscription rate.

The polling engine consists of two main components:

- poll item buckets - which contain the variables that need to be scanned, grouped according to their scan rate.
- the polling manager - which creates the poll item buckets as required, and notifies them if read requests need to be executed.

At start up, all variables are stored in the "not polled" bucket (scan rate is -1). When a subscription is received, the polling manager assigns variables to the poll item buckets according to their scan rate, creating the buckets as required.

If the subscription changes for a tag, the polling rate for any affected item is changed as well and the variables are moved to an appropriate bucket. If the subscription for a variable ends, it is placed back in the "not polled" bucket.

---

**Note:** The polling interval used by the polling manager can be configured using the parameter [Polling Parameters](#). The default value is 100ms.

---

### Events Handling

Spontaneous events are pushed to the relevant servers through the DRIClosed. The IEC870IP driver will confirm with the DRI that events are delivered to the appropriate servers before responding back to the device with acknowledgment. It is possible to change that behavior so that the driver will acknowledge spontaneous events immediately after receiving it from the device by using the parameter DRIGuaranteedEventsDelivery.

If the DRI does not indicate that an event was successfully delivered within the time specified by the parameter DRIResponseTimeout, the driver will still send acknowledgment to the device, though it will also update the driver log with an error message.

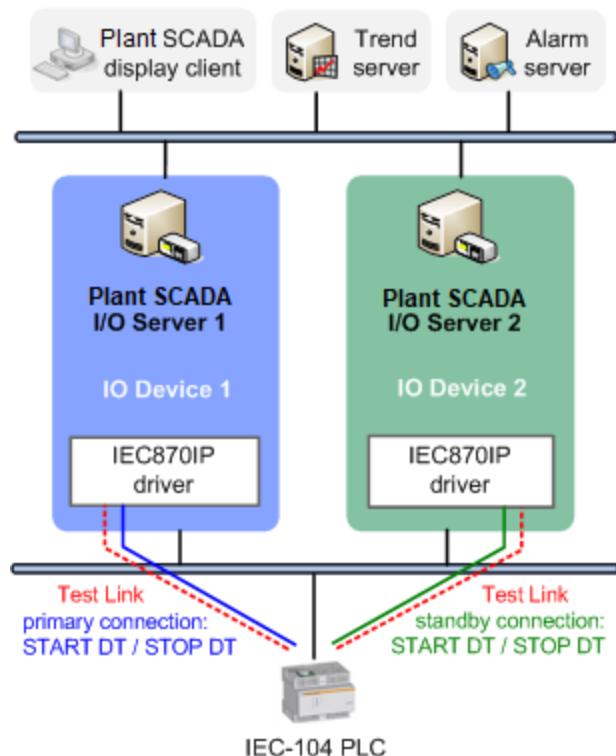
By default, DRIResponseTimeout is set to 1000 milliseconds. This means a response can be sent back to the device to allow acknowledgment to take place in a situation where a destination for an event is not configured in the SCADA system.

See [Communication Parameters](#).

### Configuring Redundancy

The IEC870IP driver supports redundant communication with devices at the I/O server level.

The following diagrams shows the configuration of a simple redundant system with a pair of I/O servers connected to an IEC-60870-5-104 device. The physical device is represented on both I/O servers.



Both I/O servers establish a TCP/IP connection to a device and periodically send test link messages to confirm that it is still online.

As the IEC 60870-5-104 standard specifies that only one master station can perform data interrogation on a device at a time, redundancy is managed through the implementation of START DT/STOP DT requests.

The I/O server with the highest priority connection to the I/O device sends a START DT request to the device to commence data interrogation. If the test link determines the connection to the device has stopped working, the second I/O server assumes highest priority and sends a START DT request.

If the initial server then comes back online, it once again assumes highest priority and issues a START DT request. This will prompt the other server to send a STOP DT request.

---

**Note:** You can adjust the rate of transmission used for test link message via the driver parameter [Communication Parameters](#).

---

## Customizing a Project using Citect.ini Parameters

The IEC870IP driver supports the following parameter categories:

- General Parameters
- Communication Parameters
- Data Interrogation Parameters
- Write Parameters
- Counter Parameters
- Time Synchronization Parameters
- Polling Parameters
- Logging Parameters

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

- Do not under any circumstances change or remove any citect.ini parameters without an informed understanding of the possible implications of your actions.
- Do not delete sections of the citect.ini file without an informed understanding of the possible implications of your actions.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** At times during the life cycle of a driver, circumstances may call for the creation of specific citect.ini parameters that remain undocumented. Always seek the advice of Technical Support regarding the possible implementation of undocumented parameters before you make any changes to a citect.ini file.

Citect.ini parameters are used to tune the performance of the Plant SCADA IEC870IP driver, and to perform runtime maintenance diagnostics.

You can customize the way Plant SCADA communicates with the IEC870IP system (and even individual PLCs) by creating or editing the [IEC870IP] section of the citect.ini file for your project.

When Plant SCADA starts at runtime, it reads configuration values from the citect.ini file that is stored locally. Therefore, any IEC870IP configuration settings need to be included in the citect.ini file located on the computer acting as the I/O server to the IEC870IP system.

By default, Plant SCADA looks for the citect.ini file in the bin directory of the Plant SCADA project. If it can't find the file there, it searches the default Windows directory.

**Note:** The IEC870IP driver can be supported by an IOA configuration file that allows output points to be operated using output commands on a per-IOA basis. See [Configuring Commands for Individual IOAs](#).

You can also set some parameters for every device connected to a specific port. See [Port-specific Parameters](#).

**General Parameters****⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** At times during the life cycle of a driver, circumstances may call for the creation of specific citect.ini parameters that remain undocumented. Always seek the advice of Technical Support regarding the possible implementation of undocumented parameters before you make any changes to a citect.ini file.

[IEC870IP] Parameter	Description	Allowable Values	Default Value
Block	A value (bytes) used by the I/O server to determine if two or more	512	512

	<p>packets can be blocked into one data request before being sent to the I/O device.</p> <p><b>Note:</b> This is a common I/O server parameter that does not impact this driver as blocking is not supported. There is no benefit to changing the value of this parameter.</p>		
Delay	<p>The period (in milliseconds) to wait between receiving a response and sending the next command.</p> <p><b>Note:</b> Do not change this parameter from the default value of 0 (zero), as applying a delay will impact the performance of this driver.</p>	0 (milliseconds)	0
MaxPending	This parameter limits the number of outstanding asynchronous request issued from the I/O server to the driver.	1 - 32	1
Poltime	<p>This parameter is required to clear the receive queue.</p> <p>Each poll time, the driver will check for any entries in the send and receive queue. If any entries are found, the driver will process the data from the queues into the driver's data image. From there, the driver will push into the DRI and SCADA system.</p> <p>The size of the receive queue is governed by the parameter [IEC870IP]ReceiveQueueSi</p>	0 - 300 (milliseconds)	100

	ze (default = 1000)		
Retry	If a DCB command to the driver times out, this parameter determines how many times the command will be tried again before reporting the driver offline.	0 - 8	1
Timeout	This timeout relates to any DCB request issued to the driver.  If a response is not received, this parameter determines how long to wait before the request is issued again.  The driver will be considered offline if no response is received following the number of request attempts specified by the Retry parameter.	0 - 32000 (milliseconds)	2000
Watchtime	The frequency with which the I/O server checks the communications link to an I/O device if the device is determined to be offline.	0 - 128 (seconds)	1

## Communication Parameters

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** At times during the life cycle of a driver, circumstances may call for the creation of specific citect.ini parameters that remain undocumented. Always seek the advice of Technical Support regarding the possible implementation of undocumented parameters before you make any changes to a citect.ini file.

[IEC870IP] Parameter	Description	Allowable Values	Default Value
CommandTimeout	Specifies how long the	0 - 32767	15000

	<p>driver will wait for acknowledgment of a command sent to a device.</p> <p><b>Note:</b> You can set this parameter for every device connected to a specific port. See <a href="#">Port-specific Parameters</a>.</p>	(milliseconds)	
ConnectionTimeout	<p>Specifies how long the driver will wait for the TCP/IP connection to be established with a device before setting the device offline.</p> <p><b>Note:</b> You can set this parameter for every device connected to a specific port. See <a href="#">Port-specific Parameters</a>.</p>	0 - 32767 (milliseconds)	30000
DelayChannelOffline	<p>Specifies how long driver should wait before reporting channel offline if TCP/IP connection is closed from device.</p> <p><b>Note:</b> This parameter shouldn't be adjusted without advise from support engineer.</p> <p>You can set this parameter for every device connected to a specific port. See <a href="#">Port-specific Parameters</a>.</p>	0 - 32767 (milliseconds)	5000
DRIGuaranteedEventsDelivery	<p>Specifies how the driver acknowledges spontaneous events received from the device. If the parameter is set to 1, the driver will acknowledge a spontaneous event only after it has been successfully delivered to the appropriate Plant</p>	0 or 1	1

	SCADA servers. Otherwise that event will be acknowledged immediately after receiving it from the device.		
DRIResponseTimeout	<p>Specifies the amount of time the driver will wait for a response from the DRI following delivery of a spontaneous event. Once a response is received, acknowledgment is sent to the device.</p> <p>If a response is not received within the time specified, acknowledgment is still sent to the device, though an error message is logged.</p> <p>See <a href="#">Events Handling</a>.</p> <p><b>Note:</b> You can set this parameter for every device connected to a specific port. See <a href="#">Port-specific Parameters</a>.</p>	1000 - 150000 (milliseconds)	1000
GIPollTimeSec	<p>Specifies how often the driver performs a general interrogation (GI).</p> <p>If the parameter is set to 0, the GI is performed only once when the driver commences data interrogation.</p>	0 - 32767 (seconds)	3600
ReceiveQueueSize	<p>Sets the maximum number of messages supported by the channel receive queue.</p> <p>If the receive queue is full, the driver will replace the oldest message with a new one. If this occurs, an error message will be</p>	100 - 32767	1000

	recorded in the driver log file.		
ResetSendReceiveSequence NumbersOnSwitchover	Specifies whether the driver resets the message send and receive sequence numbers during redundancy switchover.  By default, the I/O device which becomes active resets the send and receive sequence numbers after a STARTDT message.	0 or 1	1
TxTestLink	Specifies how often the driver sends Test Link messages.  Setting this parameter to 0 will disable the periodic sending of Test Link messages.  (See <a href="#">Configuring Redundancy</a> .)	0 - 3600000 (milliseconds)	20000

## Data Interrogation Parameters

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** At times during the life cycle of a driver, circumstances may call for the creation of specific citect.ini parameters that remain undocumented. Always seek the advice of Technical Support regarding the possible implementation of undocumented parameters before you make any changes to a citect.ini file.

[IEC870IP] Parameter	Description	Allowable Values	Default Value
ManufacturedIsUTC	Determines if the 56-bit timestamp value received from the device is considered to be UTC time or not.  <b>Note:</b> You can set this parameter for every	0 = timestamp is not UTC 1 = timestamp is UTC	0

	device connected to a specific port. See <a href="#">Port-specific Parameters</a> .		
ManufactureTimestamp	<p>This parameter controls the value timestamp for tags with following addresses:</p> <ul style="list-style-type: none"> <li>• D&lt;IOA&gt;</li> <li>• I&lt;IOA&gt;</li> <li>• L&lt;IOA&gt;</li> <li>• F&lt;IOA&gt;</li> <li>• QD&lt;IOA&gt;</li> </ul> <p>If the parameter is set to 0 (the default), the value timestamp will be equal to the device timestamp, or to the driver manufactured timestamp when the driver receives a value update.</p> <p>If the parameter is set to 1, every value timestamp will be manufactured by the driver.</p> <p><b>Note:</b> You can set this parameter for every device connected to a specific port. See <a href="#">Port-specific Parameters</a>.</p>	0 or 1	0
ReportDoubleDigitalPointOnOffStates	<p>When a double digital point is configured using a digital tag, by default it reports double point states as:</p> <ul style="list-style-type: none"> <li>• 1 "On"</li> <li>• 0 "Off"</li> <li>• 0 "Indeterminate"</li> <li>• 0 "Intermediate".</li> </ul> <p>If this parameter is enabled, the driver will only report a value update when the double point value is set to "On" or "Off" states.</p>	0 or 1	0

StoreDigitalInQual	<p>Determines if the value of a single or double digital point is stored inside the quality descriptor. This allows a single tag to be used for both value and quality information.</p> <p><b>Note:</b> The following information only applies when StoreDigitalInQual is set to 1.</p> <p>The IEC standards specifies the following for a double-point digital DPI field:</p> <ul style="list-style-type: none"> <li>• If DPI=0, the field is "indeterminate or intermediate"</li> <li>• If DPI=1, the field is "valid"</li> <li>• If DPI=2, the field is "valid"</li> <li>• If DPI=3, the field is "indeterminate"</li> </ul> <p>To determine the value of the DPI from a digital tag (D) and the quality descriptor (QD):</p> <ul style="list-style-type: none"> <li>• If QD Bit 1 and Bit 2 are not set and D is 0, then DPI=1</li> <li>• If QD Bit 1 and Bit 2 are not set and D is 1, then DPI=2</li> <li>• If QD Bit 2 is set, then DPI=0</li> <li>• If QD Bit 3 is set, then DPI=3</li> </ul>	<p>0 = do not store the digital point in the quality descriptor.</p> <p>1 = store the digital point in the quality descriptor.</p>	0
UpdateNonTimestampedTagWithTimestampedData	This parameter determines whether the non-timestamped tag value is updated when timestamped data is received. By default when a timestamped value is received, the values for	0 or 1	1

	<p>timestamped and non-timestamped tags will be updated. But in case the parameter is set to 0, only the timestamped tag value be updated.</p> <p><b>Note:</b> You can set this parameter for every device connected to a specific port. See <a href="#">Port-specific Parameters</a>.</p>		
UpdateTimestampedTagWithNonTimestampedData	<p>This parameter determines whether the timestamped tag value is updated with non-timestamped data. If it is set to 1, the timestamped tag will be updated with non-timestamped data until timestamped data is available.</p> <p><b>Note:</b> You can set this parameter for every device connected to a specific port. See <a href="#">Port-specific Parameters</a>.</p>	0 or 1	0

## Write Parameters



### UNINTENDED EQUIPMENT OPERATION

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** At times during the life cycle of a driver, circumstances may call for the creation of specific `citect.ini` parameters that remain undocumented. Always seek the advice of Technical Support regarding the possible implementation of undocumented parameters before you make any changes to a `citect.ini` file.

[IEC870IP] Parameter	Description	Allowable Values	Default Value
AppTimeoutMs	Sets the application-level timeout for the device. This is used for a select confirmation after a	0-32767 (milliseconds)	4000

	<p>control select has been issued.</p> <p><b>Note:</b> You can set this parameter for every device connected to a specific port.</p> <p>See <a href="#">Port-specific Parameters</a>.</p>		
DefaultSetPtType	<p>Determines if the value of the set point defaults to:</p> <ul style="list-style-type: none"> <li>• "normalized value" ("NORM"), or</li> <li>• "scaled value" ("SCALE").</li> </ul> <p><b>Note:</b> You can modify this parameter for an individual output point in the IOA configuration file.</p> <p>Determines the file name and path of the configuration file containing individual IOA parameters.</p> <p>See <a href="#">Configuring Commands for Individual IOAs</a>.</p>	"NORM" or "SCALE"	"NORM"
DefaultSEMode	<p>Determines if the output should default to:</p> <ul style="list-style-type: none"> <li>• "direct execute" (0), or</li> <li>• "select/execute" (1).</li> </ul> <p><b>Note:</b> You can modify this parameter for an individual output point in the IOA configuration file.</p> <p>Determines the file name and path of the configuration file containing individual IOA parameters.</p> <p>See <a href="#">Configuring Commands for Individual IOAs</a>.</p>	0 or 1	0
DefaultDOQual	Sets the default value of output qualifier for digital	0 – Digital controls default to "no additional"	0

	<p>control.</p> <p><b>Note:</b> You can modify this parameter for an individual output point in the IOA configuration file. Determines the file name and path of the configuration file containing individual IOA parameters.</p> <p>See <a href="#">Configuring Commands for Individual IOAs</a>.</p>	<p>definition".</p> <p>1 – Digital controls default to "short duration pulse".</p> <p>2 – Digital controls default to "long duration pulse".</p> <p>3 – Digital controls default to "persistent output".</p>	
IOAConfigFileName	<p>Determines the file name and path of the configuration file containing individual IOA parameters.</p> <p>See <a href="#">Configuring Commands for Individual IOAs</a>.</p>	Any valid path	Plant SCADA\bin
UseIOAConfigFile	<p>Determines whether the IOA configuration file is used.</p> <p>If set to zero (0), the file will not be used.</p> <p>See <a href="#">Configuring Commands for Individual IOAs</a>.</p>	0 or 1	0

## Counter Parameters



### UNINTENDED EQUIPMENT OPERATION

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** At times during the life cycle of a driver, circumstances may call for the creation of specific citect.ini parameters that remain undocumented. Always seek the advice of Technical Support regarding the possible implementation of undocumented parameters before you make any changes to a citect.ini file.

[IEC870IP] Parameter	Description	Allowable Values	Default Value
CounterPollTimeSec	Determines how often a counter interrogation is performed.	0 - 32767 (in seconds)	1800
CounterPollType	Determines the type of counter interrogation message that is sent.	0 – No freeze or reset 1 – Freeze without reset 2 – Freeze with reset 3 – Reset only	1

### Time Synchronization Parameters

#### **WARNING**

##### **UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** At times during the life cycle of a driver, circumstances may call for the creation of specific citect.ini parameters that remain undocumented. Always seek the advice of Technical Support regarding the possible implementation of undocumented parameters before you make any changes to a citect.ini file.

[IEC870IP] Parameter	Description	Allowable Values	Default Value
TimeSyncOnInit	Determines whether or not the time is set for a device as it comes online.  <b>Note:</b> You can set this parameter for every device connected to a specific port. See <a href="#">Port-specific Parameters</a> .	0 - do not set the time for a device as it comes online.  1 - set the time for a device as it comes online.	1
TimeSyncPollTimeSec	Determines how often the time is set for a device (in seconds).	0-32767 (seconds)	1800
TimeSyncDelayAcquire	Determines if a delay acquisition command is sent when a time synchronization response is received.  <b>Note:</b> You can set this parameter for every device connected to a	0 - do not send a delay acquisition command.  1 - send a delay acquisition command.	0

	specific port. See <a href="#">Port-specific Parameters</a> .		
--	---	--	--

## Polling Parameters

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** At times during the life cycle of a driver, circumstances may call for the creation of specific citect.ini parameters that remain undocumented. Always seek the advice of Technical Support regarding the possible implementation of undocumented parameters before you make any changes to a citect.ini file.

[IEC870IP] Parameter	Description	Allowable Values	Default Value
ReadPollTimeMs	<p>This parameter controls polling interval used by the polling manager.</p> <p>See <a href="#">Using Read Commands to Retrieve Data</a>.</p> <p><b>Note:</b> You can set this parameter for every device connected to a specific port. See <a href="#">Port-specific Parameters</a>.</p>	50 - 86400000 (milliseconds)	100

## Logging Parameters

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** At times during the life cycle of a driver, circumstances may call for the creation of specific citect.ini parameters that remain undocumented. Always seek the advice of Technical Support regarding the possible implementation of undocumented parameters before you make any changes to a citect.ini file.

[IEC870IP] Parameter	Description	Allowable Values	Default Value
DebugLevel	The trace level used for	ALL or any combination of	-

	the log file.	the following, separated by a pipe character ( ): WARN ERROR TRACE DEBUG See <a href="#">Logging</a> for examples.	
DebugCategory	The categories used for the log file.	ALL or any combination of the following, separated by a pipe character ( ): DRI MISC PROT DCB See <a href="#">Logging</a> for examples.	-

**Note:** These parameters do not have a value specified by default. To enable logging for this driver, you need to manually configure these parameters in your INI file.

## See Also

[Logging Messages](#)

## Input Tags

The IEC870IP driver defines a number of tag addresses that allow process information to be passed to Plant SCADA in a monitoring direction. The input tags are read only.

<IOA> is used to represent an information object address, defined as a decimal number in the range 0 — 16777215.

**Note:** Tags with the following addresses are only updated when the driver receives a time-stamped value directly from a device.

- SD<IOA>
- SI<IOA>
- SL<IOA>
- SF<IOA>
- SQD<IOA>

See [Timestamp Handling](#) for more information on the correct syntax to use for your input tags.

Address format	Plant SCADA type	IEC 60870 type	Description
D<IOA> SD<IOA>	DIGITAL	Single point Double point	Digital value: <ul style="list-style-type: none"> <li>• Value: digital value</li> <li>• Quality: value quality</li> <li>• Timestamp: value timestamp</li> </ul>

Address format	Plant SCADA type	IEC 60870 type	Description
			Example: D100
I<IOA> SI<IOA>	INTEGER	Measured value: normalized and scaled value  Step Position  Double Point	Integer value: <ul style="list-style-type: none"><li>• Value: integer value</li><li>• Quality: value quality</li><li>• Timestamp: value timestamp</li></ul> Example: I200
L<IOA> SL<IOA>	LONG	Integrated totals	Long value: <ul style="list-style-type: none"><li>• Value: long value</li><li>• Quality: value quality</li><li>• Timestamp: value timestamp</li></ul> Example: L300
F<IOA> SF<IOA>	FLOAT	Short Floating Point	Float value: <ul style="list-style-type: none"><li>• Value: float value</li><li>• Quality: value quality</li><li>• Timestamp: value timestamp</li></ul> Example: F350
QD<IOA> SQD<IOA>	INTEGER	Quality descriptor	<p>The set of bitflags described in <a href="#">Quality Descriptors</a>.</p> <p>This includes IEC 60870 quality flags and the IEC870IP driver-specific flags:</p> <ul style="list-style-type: none"> <li>• Value: quality descriptor value</li> <li>• Quality: quality good</li> <li>• Timestamp: time when value with this quality descriptor was received</li> </ul> <p>Example: QD350</p>

**Note:** The address format used in the table above demonstrates unstructured tag addresses (i.e. "<IOA>"). The IEC870IP driver also supports structured tag addresses (e.g. "<IOA1>.<IOA2>.<IOA3>"). For more information, see [Structured Tag Addressing](#).

## Port-specific Parameters

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** At times during the life cycle of a driver, circumstances may call for the creation of specific citect.ini parameters that remain undocumented. Always seek the advice of Technical Support regarding the possible implementation of undocumented parameters before you make any changes to a citect.ini file.

Any parameters placed under the [IEC870IP] section of the Citect.ini file will apply globally to every device using the IEC870IP driver. However, you can also set parameters for every device connected to a specified port.

To do this, use the following section name within the Citect.ini file.

[<Protocol>.<PortName>]

Where:

- <Protocol> is IEC870IP
- <PortName> is the 'Port Name' defined for a specified port.

For example, any parameters set within the section [IEC870IP.PrimaryPort1] will impact every device connected to the port defined as "PrimaryPort1".

The following list includes the parameters you can set for all devices on a specific port.

[Data Interrogation Parameters:](#)

- ManufacturedIsUTC
- ManufactureTimestamp
- UpdateNonTimestampedTagWithTimestampedData
- UpdateTimestampedTagWithNonTimestampedData

[Time Synchronization Parameters:](#)

- TimeSyncOnInit
- TimeSyncDelayAcquire
- TimeSyncPollTimeSec

[Communication Parameters:](#)

- CommandTimeout
- ConnectionTimeout
- DelayChannelOffline
- DRIResponseTimeout
- GIPollTimeSec

[Polling Parameters:](#)

- ReadPollTimeMs

[Write Parameters:](#)

- AppTimeoutMs

[Counter Parameters:](#)

- CounterPollTimeSec
- CounterPollType

## Configuring Commands for Individual IOAs

The IEC870IP driver can be supported by an IOA configuration file that enables you to use complex addressing on a per-IOA basis. When implemented, this file allows output points to be individually operated using IEC 60870-5-104 output commands.

For example, a single digital output could be independently configured to "select/execute" or "direct execute". It could be "short pulse", "long pulse", "persistent" or "predefined". Any IOAs not included in the file will continue to use default settings.

## Using the "IECioa.dbf" file

An IOA configuration file (called "IECioa.dbf" by default) is available for modification in the Plant SCADA bin directory. You can use this file as a template to configure your own IOA customizations. To implement the file:

1. Locate the file "IECioa.dbf" in Plant SCADA bin directory.  
**Note:** It is recommended you take a copy of this file and move it to the associated project directory, as this will avoid the file being overwritten if the driver is reinstalled.
2. Configure the parameters UseIOAConfigFile and IOAConfigFileName (see [Write Parameters](#)). These parameters let the driver know an IOA configuration file is available, and confirm the name and location.  
**Note:** If the configuration file specified by these parameters cannot be loaded, the associated unit will go offline.
3. Edit the configuration file as required. It contains one IOA per row, each defining the output qualifiers to use for the specified IO point.

The following table describes the available fields:

Column	Description
CHANNEL	Plant SCADA Port name
COMMADDR	Plant SCADA I/O device address (common address of ASDU)
IOA	Information Object Address for this point
TYPE	For output points only, leave blank for other point

Column	Description
	<p>types:</p> <p>Single or Double Control Output, or Regulating Step Command:</p> <ul style="list-style-type: none"> <li>• "SDO" Single Digital Output (default)</li> <li>• "DDO" Double Digital Output</li> <li>• "RSO" Regulating Step Output Set Point Command</li> <li>• "NORM" Sent as a Normalized Value</li> <li>• "SCALE" Sent as a Scaled Value</li> </ul> <p>The default setpoint is specified by the driver configuration parameter DefaultSetPtType.</p>
QUALIF	<p>Control Qualifier for Output points only, leave blank for other point types.</p> <p>Single or Double Control Output, or Regulating Step Command:</p> <ul style="list-style-type: none"> <li>0 - No additional definition</li> <li>1 - Short Duration Pulse</li> <li>2 - Long Duration Pulse</li> <li>3 - Persistent Output</li> <li>4..31 - Reserved, as per IEC Standard</li> </ul> <p>The default mode is specified by the driver configuration parameter DefaultDOQual.</p> <p>Set Point Command:</p> <ul style="list-style-type: none"> <li>0 - Default</li> <li>1..127 - Reserved, as per IEC Standard</li> </ul>
SELEX	<p>Select/execute mode for output points leave blank for other point types.</p> <ul style="list-style-type: none"> <li>0 - Direct execute (single stage)</li> <li>1 - Select/Execute (three stage).</li> </ul> <p>The default is specified by the driver configuration parameter DefaultSEMode.</p>

**Note:** If a protocol command is set for an IOA that does not support the specified operation, an error will result.

## Troubleshooting

The following topics provide information about the Plant SCADA tools available to diagnose and resolve any unexpected behavior within the runtime system.

- [Driver Statistics](#)
- [Logging](#)

## Driver Statistics

You can use the following driver statistic counters to help you debug the IEC870IPdriver. These statistics can be viewed within the Plant SCADA Kernel using the Page Driver command and then by pressing the 'v' key to select verbose mode.

See the topic *Runtime > Monitor Runtime > The Kernel > Kernel Commands > Page Driver* in the main Plant SCADA help for more information.

Number	Statistic	Description
1	Data CRC Error	The number of messages received that had a bad checksum.
2	Unit Offline	The number of times any device has been made offline.
3	Negative Reply Rx	The number of times a device has indicated in a Command Response that there was something wrong with the command sent to it.
4	NOT IMPL Replies	The number of times the driver has received a NOT IMPLEMENTED message.
5	Busy Replies	The number of times the driver has received a BUSY message.
6	FCB Errors	The number of times the driver has received a message with an incorrect Frame Control Bit (FCB), this indicates that not all messages are being received correctly.
7	Unknown Addr Rx	The number of times the driver has received a message containing invalid address information. Indicates that the slave device may be incorrectly configured.
8	Intercharacter timeouts	The number of times message have been rejected because the time between bytes in a message was too large. This may indicate that communication parameters need to be adjusted.
9	CMD Not Supported	The number of times the driver has received a Command Not Supported message.

Number	Statistic	Description
10	Request Link Count	The number of request link status messages that have been transmitted since the I/O server started.
11	General Int Count	The number of general interrogation messages that have been transmitted since the I/O server started.
12	Time Sync Count	The number of time synchronization messages that have been transmitted since I/O server started.
13	Test Link Count	The number of test link messages that have been transmitted since the I/O server started.

## Logging

The IEC870IP driver can log combinations of trace levels across different categories. This is achieved by setting the following INI parameters.

### [IEC870IP]DebugLevel

This parameter allows you to define the trace level. The options include:

Option	Description
WARN	Output warning messages.
ERROR	Output error messages.
TRACE	Output trace messages.
DEBUG	Output debug messages.
ALL	All of the above.

### [IEC870IP]DebugCategory

This parameter allows you to enable logging for a particular category of trace. The options include:

Option	Description
DRI	Driver Runtime Interface trace.
MIISC	Any other debugging.
PROT	Protocol level debug.
DCB	Front end driver trace.
ALL	All of the above.

In both cases, you can use any combination of the available options separated by a pipe character ( | ).

### Example

```
[IEC870IP]
DebugLevel=WARN | ERROR | TRACE
DebugCategory=ALL
```

In most cases, you can set the DebugLevel and DebugCategory parameters to ALL. If this results in a large, cumbersome log file that causes older data to be overwritten, you can consider refining the debug options.

See [Logging Messages](#) for a list of the error and warning messages supported by the driver.

---

**Note:** These parameters do not have a value specified by default. To enable logging for this driver, you need to manually configure these parameters in your INI file.

The events generated by the IEC870IP driver are logged in the following Plant SCADA syslog file:

```
syslog.IOServer.<Cluster name>.<IO Server name>.dat
```

This file is located in the directory specified by the Citect.ini parameter **[CtEdit]Logs**. By default, this location will be:

```
%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Logs
```

The size of the syslog file and its rollover process is configured via the **[Debug]SysLogSize** and **[Debug]SysLogArchive** INI parameters. For more information, see the topic *Debug Parameters* in the Plant SCADA documentation.

---

**Note:** If the number of messages being processed for a driver exceeds the value specified by the Plant SCADA parameter **[Kernel]ErrorBuffers**, the pending messages will not be logged. If this occurs, you can increase the value of this parameter so that all the messages generated by the driver are reported. The number of messages that are not logged can be monitored via the Kernel's General window under the item "Lost Errors". If you adjust **[Kernel]ErrorBuffers** and Lost Errors are still occurring, you may need to adjust the level of system logging.

## See Also

[Logging Parameters](#)

### Logging Messages

The following tables include the error and warning messages that can be logged by the IEC870IP driver.

### LEVEL\_ERROR messages

Debug category	Message
DRI	DRI error message: {error description}
MISC	Cancelling DCB on channel unsuccessful
MISC	Logical Channel Open unsuccessful
MISC	Channel offline
MISC	CreateTimer(OnConnectTimeout) unsuccessful
MISC	COMSetVector(DATA_ALL_EVENTS) returned error message
MISC	COMSetVector(CONN_READY) returned error message
MISC	COMOpenEx returned error message {error description} for channel
MISC	Total number of Channels exceeds the driver maximum
MISC	Opening port {port name} unsuccessful
MISC	Initializing channel {channel} unsuccessful, Channel does not exist
MISC	Stopping channel {channel number} unsuccessful, Channel does not exist
MISC	Closing channel {channel number} unsuccessful, Channel does not exist
MISC	InitUnit unsuccessful {unit name}
MISC	Stopping unit on channel {channel number} unsuccessful
MISC	Performing read on channel {channel number} unsuccessful, Channel does not exist
MISC	Performing write on channel {channel number} unsuccessful, Channel does not exist
MISC	Total number of I/O devices exceeds the driver maximum. Possible mismatch with IECioa.dbf.
MISC	IOAConfig File Open unsuccessful. Filename: {file name } Error message={error}
MISC	AIOConfig Read Num Records unsuccessful. Filename:

	{file name }
MISC	IOAConfig Position to first Rec unsuccessful. Filename: {file name }
MISC	IOAConfig Read of COMMADDR field unsuccessful
MISC	IOAConfig Read of IOA field unsuccessful
MISC	IOAConfig Read of TYPE field unsuccessful
MISC	IOAConfig Read of QUALIF field unsuccessful
MISC	IOAConfig Read of SOEENABL field unsuccessful
MISC	IOAConfig read Event Log Tagname unsuccessful
MISC	IOAConfig read event log Raw Zero Scale unsuccessful
MISC	IOAConfig read event log Raw Full Scale unsuccessful
MISC	IOAConfig read event log Engineering Zero Scale unsuccessful
MISC	IOAConfig read event log Engineering Full Scale unsuccessful
MISC	IOAConfig File Close unsuccessful
PROT	Bad channel number
PROT	Opening port {port } unsuccessful = {error}
PROT	Could not create Command Timeout Timer
PROT	Could not create Intercharacter Timeout
PROT	Invalid Message contains 0 elements
PROT	Invalid timestamp received, using current time.
DCB	Timer creation unsuccessful in Port {port }
DCB	COMConnectEx returned error message {error }
DCB	Total number of I/O devices exceeds the driver maximum.
DCB	Caught protocol Error message in SDA Channel {channel number}, error message = {error}
DCB	Invalid DCB passed to BuildRequest in Port {port}

DCB	CancelDCB unsuccessful
DCB	ProtoReplyError hit default case, proto_err={error}
DCB	ProtoReplyError hit default case, proto_err = {error}

**LEVEL\_WARNING messages**

Debug category	Message
CAT_VALIDATE	AddrConfig Unit has 2 entries, discarding 2nd
DCB	Canceling DCB...
PROT	Receive message for unknown address
DCB	FormatDCB called. Format string: {string}
DCB	Cancel DCB called

**See Also**[Logging](#)[Logging Parameters](#)**Appendix**

This appendix includes the [IEC 870-5-104 Interoperability List](#), a list of the parameters defined in the IEC 60870-5-101 protocol that shows which options are supported by the Plant SCADA IEC870IP driver.

**IEC 870-5-104 Interoperability List**

This interoperability list refers to section 9 of the IEC 60870-5-104 International Standard (reference number CEI/IEC 60870-5-104:2006).

It is a complete list of the parameters defined in the IEC 60870-5-101 protocol, with the options supported by the IEC870IP driver defined using the following formatting:

<input type="checkbox"/> <description>	This formatting indicates that the described parameter is supported by the 104 variation of the protocol, though it is not supported by this driver.
<input checked="" type="checkbox"/> <description>	A check mark indicates that the described parameter is supported by the 104 variation of the protocol and implemented in this driver.
<input checked="" type="checkbox"/> <description>	This formatting indicates that the described parameter

	is usually not supported by the 104 variation of the protocol, but is implemented in this driver.
<input checked="" type="checkbox"/> <description>	This formatting indicates that the described parameter is not supported by the 104 variation of the protocol and is not supported by the driver.

## 9.1 System or device

<input type="checkbox"/> System definition
<input checked="" type="checkbox"/> Control station definition (master)
<input type="checkbox"/> Control station definition (slave)

## 9.2 Network configuration (network-specific parameters)

<input checked="" type="checkbox"/> Point-to-point	<input checked="" type="checkbox"/> Multipoint party line
<input checked="" type="checkbox"/> Multiple point-to-point	<input checked="" type="checkbox"/> Multipoint star

## 9.3 Physical layer (network-specific parameter)

### Transmission speed (control direction)

Unbalanced interchange circuit V.24/V.28 Standard	Unbalanced interchange circuit V.24/V.28 Recomm. if >1200bit/s	Balanced interchange circuit X.24/X.27	
<input checked="" type="checkbox"/> 100bit/s	<input checked="" type="checkbox"/> 2400 bit/s	<input checked="" type="checkbox"/> 2400 bit/s	<input checked="" type="checkbox"/> 56000bit/s
<input checked="" type="checkbox"/> 200bit/s	<input checked="" type="checkbox"/> 4800 bit/s	<input checked="" type="checkbox"/> 4800 bit/s	<input checked="" type="checkbox"/> 64000bit/s
<input checked="" type="checkbox"/> 300bit/s	<input checked="" type="checkbox"/> 9600 bit/s	<input checked="" type="checkbox"/> 9600 bit/s	
<input checked="" type="checkbox"/> 600bit/s		<input checked="" type="checkbox"/> 19200 bit/s	
<input checked="" type="checkbox"/> 1200bit/s		<input checked="" type="checkbox"/> 38400 bit/s	

### Transmission speed (monitor direction)

Unbalanced interchange circuit V.24/V.28 Standard	Unbalanced interchange circuit V.24/V.28 Recomm. if >1200bit/s	Balanced interchange circuit X.24/X.27	
<input checked="" type="checkbox"/> 100 bit/s	<input checked="" type="checkbox"/> 2400 bit/s	<input checked="" type="checkbox"/> 2400 bit/s	<input checked="" type="checkbox"/> 56000 bit/s

<input type="checkbox"/> 200 bit/s	<input type="checkbox"/> 4800 bit/s	<input type="checkbox"/> 4800 bit/s	<input type="checkbox"/> 64000 bit/s
<input type="checkbox"/> 300 bit/s	<input type="checkbox"/> 9600 bit/s	<input type="checkbox"/> 9600 bit/s	
<input type="checkbox"/> 600 bit/s		<input type="checkbox"/> 19200 bit/s	
<input type="checkbox"/> 1200 bit/s		<input type="checkbox"/> 38400 bit/s	

## 9.4 Link layer

Frame format FT 1.2, single character 1 and the fixed time out interval are used exclusively in this companion standard.

Link Transmission Procedure	Address Field of the Link
<input type="checkbox"/> Balanced transmission	<input type="checkbox"/> Not present (balanced transmission only)
<input type="checkbox"/> Unbalanced transmission	<input type="checkbox"/> One octet
	<input type="checkbox"/> Two octets
<b>Frame Length</b>	<input type="checkbox"/> Structured
<input type="checkbox"/> Max.length L (number of octets)	<input type="checkbox"/> Unstructured

When using an unbalanced link layer, the following ASDU types are returned in class 2 messages (low priority) with the indicated causes of transmission:

<input type="checkbox"/> The standard assignment of ASDUs to class 2 messages is used as follows:	
<b>Type identification</b>	<b>Cause of transmission</b>
9, 11, 13, 21	<1>
<input type="checkbox"/> A special assignment of ASDUs to class 2 messages is used as follows:	
<b>Type identification</b>	<b>Cause of transmission</b>

## 9.5 Application layer

### Transmission Mode for Application Data

Mode 1 (least significant octet first), as defined in clause 4.10 of IEC 870-5-4, is used exclusively in this companion standard.

### Common address of ASDU (System-specific parameter)

<input checked="" type="checkbox"/> One octet	<input type="checkbox"/> Two octets
---	-------------------------------------

### Information Object Address (System-specific parameter)

<input type="checkbox"/> One octet	<input checked="" type="checkbox"/> Structured
<input type="checkbox"/> Two octets	<input checked="" type="checkbox"/> Unstructured
<input checked="" type="checkbox"/> Three octets	

### Cause of Transmission (System-specific parameter)

<input type="checkbox"/> One octet	<input checked="" type="checkbox"/> Two octets (with originator address)
------------------------------------	--

### Length of APDU

<input checked="" type="checkbox"/> 253	Max.length L (number of octets)
---	---------------------------------

### Selection of standard ASDUs

Process information in monitor direction (station-specific parameter)

<input checked="" type="checkbox"/> <1>:=	Single-point information	M_SP_NA_1
<input checked="" type="checkbox"/> <2>:=	Single-point information with time tag	M_SP_TA_1
<input checked="" type="checkbox"/> <3>:=	Double-point information	M_DP_NA_1
<input checked="" type="checkbox"/> <4>:=	Double-point information with time tag	M_DP_TA_1
<input checked="" type="checkbox"/> <5>:=	Step position information	M_ST_NA_1
<input checked="" type="checkbox"/> <6>:=	Step position information with time tag	M_ST_TA_1
<input type="checkbox"/> <7>:=	Bitstring of 32 bit	M_BO_NA_1
<input checked="" type="checkbox"/> <8>:=	Bitstring of 32 bit with time tag	M_BO_TA_1
<input checked="" type="checkbox"/> <9>:=	Measured value, normalized value	M_ME_NA_1
<input checked="" type="checkbox"/> <10>:=	Measured value, normalized value with time tag	M_ME_TA_1

<input checked="" type="checkbox"/> <11>:=	Measured value, scaled value	M_ME_NB_1
<input checked="" type="checkbox"/> <12>:=	Measured value, scaled value with time tag	M_ME_TB_1
<input checked="" type="checkbox"/> <13>:=	Measured value, short floating point value	M_ME_NC_1
<input checked="" type="checkbox"/> <14>:=	Measured value, short floating point value with time tag	M_ME_TC_1
<input checked="" type="checkbox"/> <15>:=	Integrated totals	M_IT_NA_1
<input checked="" type="checkbox"/> <16>:=	Integrated totals with time tag	M_IT_TA_1
<input checked="" type="checkbox"/> <17>:=	Event of protection equipment with time tag	M_EP_TA_1
<input checked="" type="checkbox"/> <18>:=	Packed start event of protection equipment with time tag	M_EP_TB_1
<input checked="" type="checkbox"/> <19>:=	Packed output circuit information of protection equipment with time tag	M_EP_TC_1
<input type="checkbox"/> <20>:=	Packed single-point information with status change detection	M_PS_NA_1
<input checked="" type="checkbox"/> <21>:=	Measured value, normalised value without quality descriptor	M_ME_ND_1
 <input checked="" type="checkbox"/> <30>:=	Single-point information with time tag CP56Time2a	M_SP_TB_1
<input checked="" type="checkbox"/> <31>:=	Double-point information with time tag CP56Time2a	M_DP_TB_1
<input checked="" type="checkbox"/> <32>:=	Step position information with time tag CP56Time2a	M_ST_TB_1
<input type="checkbox"/> <33>:=	Bitstring of 32 bit with time tag CP56Time2a	M_BO_TB_1
<input checked="" type="checkbox"/> <34>:=	Measured value, normalised value with CP56Time2a	M_ME_TD_1
<input checked="" type="checkbox"/> <35>:=	Measured value, scaled value with CP56Time2a	M_ME_TE_1
<input checked="" type="checkbox"/> <36>:=	Measured value, short floating point value with CP56Time2a	M_ME_TF_1

<input checked="" type="checkbox"/> <37>:=	Integrated totals with CP56Time2a	M_IT_TB_1
<input type="checkbox"/> <38>:=	Event of protection equipment with CP56Time2a	M_EP_TD_1
<input type="checkbox"/> <39>:=	Packed start events of protection equipment with CP56Time2a	M_EP_TE_1
<input type="checkbox"/> <40>:=	Packed output circuit information of protection equipment with CP56Time2a	M_EP_TF_1

Process information in control direction (station-specific parameter)

Either the ASDUs of the set <45> - <51> or the set <58> - <64> are used.

<input checked="" type="checkbox"/> <45>:=	Single command	C_SC_NA_1
<input checked="" type="checkbox"/> <46>:=	Double command	C_DC_NA_1
<input checked="" type="checkbox"/> <47>:=	Regulating step command	C_RC_NA_1
<input checked="" type="checkbox"/> <48>:=	Set point command, normalized value	C_SE_NA_1
<input checked="" type="checkbox"/> <49>:=	Set point command, scaled value	C_SE_NB_1
<input checked="" type="checkbox"/> <50>:=	Set point command, short floating point value	C_SE_NC_1
<input type="checkbox"/> <51>:=	Bitstring of 32 bit	C_BO_NA_1
<input checked="" type="checkbox"/> <58>:=	Single command with time tag CP56Time2a	C_SC_TA_1
<input checked="" type="checkbox"/> <58>:=	Single command with time tag CP56Time2a	C_SC_TA_1
<input checked="" type="checkbox"/> <59>:=	Double command with time tag CP56Time2a	C_DC_TA_1
<input checked="" type="checkbox"/> <60>:=	Regulating step command with time tag CP56Time2a	C_RC_TA_1
<input checked="" type="checkbox"/> <61>:=	Set point command, normalized value with time tag CP56Time2a	C_SE_TA_1
<input checked="" type="checkbox"/> <62>:=	Set point command, scaled value with time tag CP56Time2a	C_SE_TB_1
<input checked="" type="checkbox"/> <63>:=	Set point command, short floating point value with time tag CP56Time2a	C_SE_TC_1
<input type="checkbox"/> <64>:=	Bitstring of 32 bit with time tag	C_BO_TA_1

	CP56Time2a	
--	------------	--

System information in monitor direction (station-specific parameter)

<input type="checkbox"/> <70>:=	End of initialization	M_EI_NA_1
---------------------------------	-----------------------	-----------

System information in control direction (station-specific parameter)

<input checked="" type="checkbox"/> <100>:=	Interrogation command	C_IC_NA_1
<input checked="" type="checkbox"/> <101>:=	Counter interrogation command	C_CI_NA_1
<input checked="" type="checkbox"/> <102>:=	Read command	C_RD_NA_1
<input checked="" type="checkbox"/> <103>:=	Clock synchronization command	C_CS_NA_1
<input type="checkbox"/> <104>:=	Test command	C_TS_NB_1
<input type="checkbox"/> <105>:=	Reset process command	C_RP_NC_1
<input type="checkbox"/> <106>:=	Delay acquisition command	C_CD_NA_1
<input type="checkbox"/> <107>:=	Test comand with time tag CP56Time2a	C_TS_TA_1

Parameter in control direction (station-specific parameter)

<input type="checkbox"/> <110>:=	Parameter of measured value, normalized value	P_ME_NA_1
<input type="checkbox"/> <111>:=	Parameter of measured value, scaled value	P_ME_NB_1
<input type="checkbox"/> <112>:=	Parameter of measured value, short floating point value	P_ME_NC_1
<input type="checkbox"/> <113>:=	Parameter activation	P_AC_NA_1

File transfer (station-specific parameter)

<input type="checkbox"/> <120>:=	File ready	F_FR_NA_1
<input type="checkbox"/> <121>:=	Section ready	F_SR_NA_1
<input type="checkbox"/> <122>:=	Call directory, select file, call file, call section	F_SC_NA_1
<input type="checkbox"/> <123>:=	Last section, last segment	F_LS_NA_1
<input type="checkbox"/> <124>:=	Ack file, ack section	F_AF_NA_1
<input type="checkbox"/> <125>:=	Segment	F_SG_NA_1

<input type="checkbox"/> <126>:=	Directory	F_DR_TA_1
<input type="checkbox"/> <127>:=	Query log - request archive file	F_SC_NB_1

## 9.6 Basic application functions

**Station initialization** (station-specific parameter)

- |   |
|---|
| <input checked="" type="checkbox"/> Remote initialization |
|---|

**Cyclic data transmission** (station-specific parameter)

- |   |
|---|
| <input type="checkbox"/> Cyclic data transmission |
|---|

**Read procedure** (station-specific parameter)

- |   |
|---|
| <input type="checkbox"/> Read procedure |
|---|

**Spontaneous transmission** (station-specific parameter)

- |   |
|---|
| <input type="checkbox"/> Spontaneous transmission |
|---|

**Double transmission of information objects with cause of transmission spontaneous** (station-specific parameter)

- |   |
|---|
| <input type="checkbox"/> Single-point information M_SP_NA_1, M_SP_TA_1, M_SP_TB_1 and M_PS_NA_1                     |
| <input type="checkbox"/> Double-point information M_DP_NA_1, M_DP_TA_1 and M_DP_TB_1                                |
| <input type="checkbox"/> Step position information M_ST_NA_1, M_ST_TA_1 and M_ST_TB_1                               |
| <input type="checkbox"/> Bitstring of 32 bit M_BO_NA_1, M_BO_TA_1 and M_BO_TB_1 (if defined for a specific project) |
| <input type="checkbox"/> Measured value, normalized value M_ME_NA_1, M_ME_TA_1, M_ME_ND_1 and M_ME_TD_1             |
| <input type="checkbox"/> Measured value, scaled value M_ME_NB_1, M_ME_TB_1 and M_ME_TE_1                            |
| <input type="checkbox"/> Measured value, short floating point number M_ME_NC_1, M_ME_TC_1 and M_ME_TF_1             |

**Station interrogation** (system- or station-specific parameter)

<input checked="" type="checkbox"/> global		
<input type="checkbox"/> group 1	<input type="checkbox"/> group 7	<input type="checkbox"/> group 13
<input type="checkbox"/> group 2	<input type="checkbox"/> group 8	<input type="checkbox"/> group 14
<input type="checkbox"/> group 3	<input type="checkbox"/> group 9	<input type="checkbox"/> group 15
<input type="checkbox"/> group 4	<input type="checkbox"/> group 10	<input type="checkbox"/> group 16
<input type="checkbox"/> group 5	<input type="checkbox"/> group 11	

<input type="checkbox"/> group 6	<input type="checkbox"/> group 12	addresses per group have to be defined
----------------------------------	-----------------------------------	--

**Clock synchronization** (station-specific parameter)

<input checked="" type="checkbox"/> Clock synchronization
<input type="checkbox"/> Day of week used
<input type="checkbox"/> RES 1, GEN (time tag substituted/not substituted) used
<input type="checkbox"/> SU-bit (summertime) used

**Command Transmission** (object-specific parameter)

<input checked="" type="checkbox"/> Direct command transmission
<input checked="" type="checkbox"/> Direct set point command transmission
<input checked="" type="checkbox"/> Select and execute command
<input checked="" type="checkbox"/> Select and execute set point command
<input type="checkbox"/> C_SE ACTTERM used
<input checked="" type="checkbox"/> No additional definition
<input checked="" type="checkbox"/> Short pulse duration (duration determined by a system parameter in the outstation)
<input checked="" type="checkbox"/> Long pulse duration (duration determined by a system parameter in the outstation)
<input checked="" type="checkbox"/> Persistent output
<input type="checkbox"/> Supervision of maximum delay in command direction of commands and set point commands
<input type="checkbox"/> Maximum allowable delay of commands and set point commands

**Transmission of integration totals** (station- or object-specific parameter)

<input type="checkbox"/> Mode A: Local freeze with spontaneous transmission
<input type="checkbox"/> Mode B: Local freeze with counter interrogation
<input type="checkbox"/> Mode C: Freeze and transmit by counter-interrogation commands
<input type="checkbox"/> Mode D: Freeze by counter-interrogation command, frozen values reported
<input checked="" type="checkbox"/> Counter read
<input checked="" type="checkbox"/> Counter freeze without reset
<input checked="" type="checkbox"/> Counter freeze with reset

Counter reset General request Request counter group 1 Request counter group 2 Request counter group 3 Request counter group 4**Parameter loading** (object-specific parameter) Threshold value Smoothing factor Low limit for transmission of measured value High limit for transmission of measured value**Parameter activation** (object-specific parameter) Act/deact of persistent cyclic or periodic transmission of the addressed object**File transfer** (object-specific parameter)

File transfer in monitor direction

 Transparent file Transmission of disturbance data of protection equipment Transmission of sequences of events Transmission of sequences of recorded analog values

File transfer in control direction

 Transparent file**Background scan** (station-specific parameter) Background scan**Acquisition of transmission delay** (object-specific parameter) Acquisition of transmission delay**Definition of timeouts**

Parameter	Default Value	Remarks	Selected Value	Associated Citect.ini parameter
t <sub>0</sub>	30 s	Time-out of connection establishment	5	ConnectionTimeout
t <sub>1</sub>	15 s	Time-out of send or test APDUs	2	CommandTimeout
t <sub>2</sub>	10 s	Time-out for acknowledges in case of no data messages t <sub>2</sub> < t <sub>1</sub>	2	
t <sub>3</sub>	20 s	Time-out for sending test frames in case of a long idle state	5	TxTestLink

**Maximum number of outstanding I format APDUs k and latest acknowledge APDUs (w)**

Parameter	Default Value	Remarks	Selected Value
k	12 APDUs	Maximum difference receive sequence number to send state variable	1
w	8 APDUs	Latest acknowledge after receiving w I format APDUs	1

**Portnumber**

Parameter	Value	Default Value	Remarks
Portnumber	(set in the configuration of the TCP port)	2404	1

**Redundant connections**

The number of redundancy group connections used is dependent on the Plant SCADA project configuration.

## KNX Driver

The KNX driver enables Plant SCADA to communicate with KNX networks and devices using Ethernet and UDP.

KNX is an open standard that has emerged from Europe to service the building and home automation industry. It can be used to control a wide range of equipment and applications associated with building management, including lighting, heating, ventilation, air-conditioning, monitoring systems, alarms, security and load management.

KNX represents an extension of the European Integration Bus (EIB) standard, with additional physical layers and configuration modes built on to the EIB communication stack.

The KNX driver is based on a subset of the standards described in KNX Version 1.1. It is designed for run-time operation only (programming is not supported) for the purpose of reading and writing group address values.

**Note:** The KNX driver has been tested on Citect SCADA v7.20 SP4 and Citect SCADA v7.40 SP2 for backward compatibility.

## See Also

[Communicating with KNX Networks](#)

[Tag Addressing](#)

[KNX Data Type Mappings](#)

[KNX Tag Import](#)

[Advanced Configuration and Maintenance](#)

[Troubleshooting](#)

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

#### DANGER

**DANGER** indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.

#### WARNING

**WARNING** indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.

**⚠ CAUTION**

**CAUTION** indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.

**NOTICE**

NOTICE used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.

**Please Note**

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

**Before You Begin**

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ WARNING****LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

## Communicating with KNX Networks

Plant SCADA KNX driver supports communicating with KNX IP devices via KNXnet/IP routers configured for Unicast or Multicast mode.

The routers need to be properly configured to route messages on to the IP network.

---

**Note:** The KNX driver should be capable of communicating with any KNXnet/IP-based KNX-to-IP router that has the routing specification implemented. However, you should perform specific testing to confirm the driver's ability to communicate effectively with a particular device.

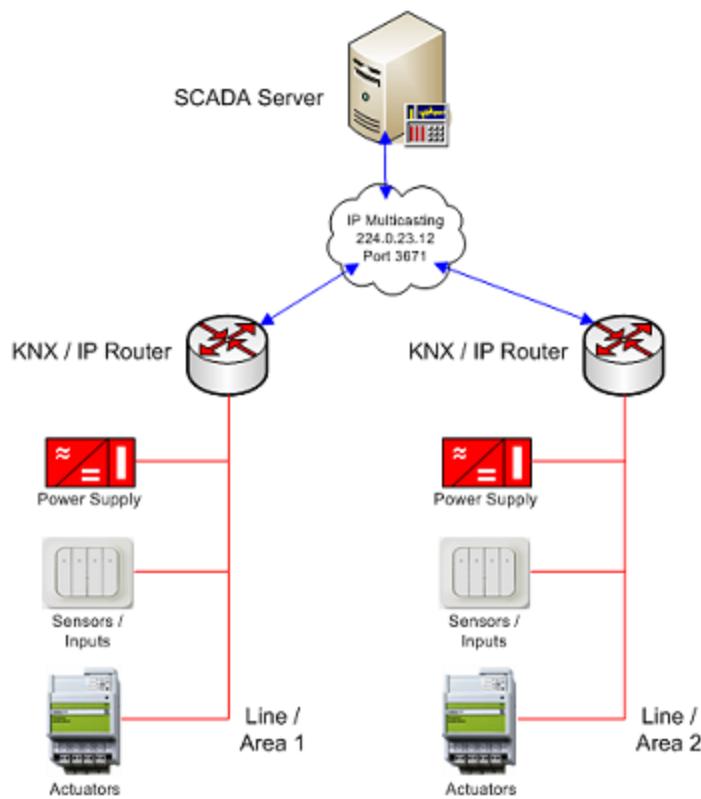
---

When communicating with large and complex configurations it is important to configure the I/O devices appropriately. For example, when multiple floors of a building have identical installation of routers, devices and so on, you need to configure one or more I/O devices for each floor.

### Multicast Mode

In Multicast mode, the driver connects to the router using the Multicast address and multicast port configured for KNX/IP routing. A single device in Plant SCADA can communicate with KNX devices on multiple routers. The routers must be configured for Multicast mode communication.

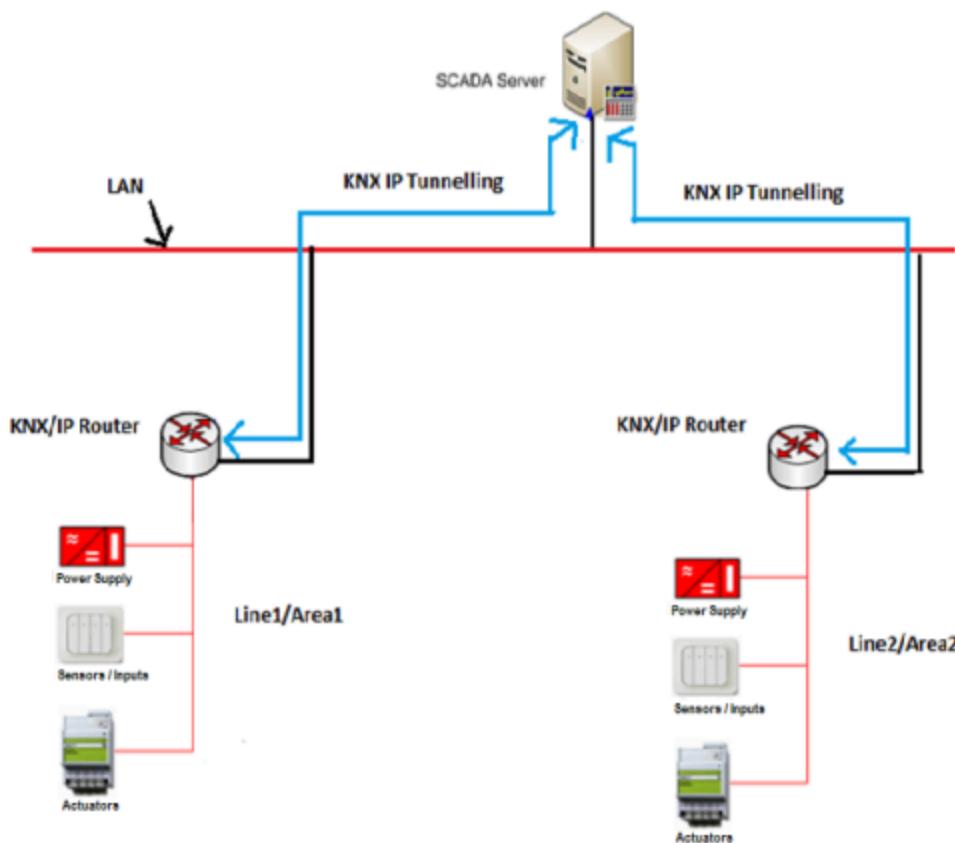
The following diagram illustrates a general layout of the KNX communication in multicast mode.



## Unicast Mode

An entire group of KNX devices is configured as a single device in Plant SCADA. This requires all the devices to be on a single router which is configured for Unicast mode communication.

The following diagram illustrates a general layout of the KNX communication in unicast mode.



#### Connection sequence in Unicast mode:

1. Driver sends a **CONNECT\_REQUEST** to the router.
2. After sending connect request, driver waits for **TunnelConnectionTimeout** to receive a response [**CONNECT\_RESPONSE**] from the router.  
Once the connect response time is ended, any response for the previous connect request will be ignored by the driver and it will request another connection [**CONNECT\_REQUEST**] if there is any pending retries (configured through **TunnelConnectionRetries**), else it makes the device offline.
3. Driver starts sending **TUNNELLING\_REQUEST** packets to the router requesting data for respective group addresses configured in the Plant SCADA project to populate the cache.
4. After the **InitTimeout** time, if at least one tag is good or when all the tags become good whichever occurs first the driver makes the device online and displays the tag values.
5. Driver continuously polls the router status with a **CONNECTSTATE\_REQUEST** to check the connection status with the router. The polling interval can be configured in **TunnelConnectionActive** INI parameter. The router will reply with a **CONNECTSTATE\_RESPONSE** message to confirm the status.
6. The Driver/Router sends the **DISCONNECT\_REQUEST** packet to close the connection if:
  - The device stops
  - The I/O server is stopped
  - The router receives any malformed packet
  - The connection is broken.

## See Also

[Communications Settings](#)

[Tag Addressing](#)

## Communication Settings

To establish communication with a device, an I/O Device need to be correctly configured in Plant SCADA Studio's **Topology** activity, as well as any associated Boards and Ports. If you use the Express Communications Wizard to connect to a device, the required settings will be automatically configured for you. However, if you need to manually configure these settings, use the values outlined below.

## Boards

Field	Value
Board Name	A unique name (up to 16 alphanumeric characters) per server for the computer board being used to connect with the device. This is used by Plant SCADA in the Ports form.
Board Type	KNX
Address	Set field to 0 (zero).
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.
Comment	Any useful comment.

## Ports

Field	Value
Port Name	A unique name (up to 16 alphanumeric character) for the computer port being used to connect with the device. This is used by Plant SCADA in the Devices form.
Port Number	Leave this field blank.
Board Name	The name of the board this port is attached to as defined on the Boards form.
Baud Rate	Leave this field blank.

Field	Value
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank.
Special Options	<p>Fill this field with IP address of the router and Unicast port number.</p> <p>Use the following format:            -la -Pn            where:            a = the IP address in standard Internet dot format. (For example 192.9.2.60)            n = 3671 (This is the reserved KNX port number for Unicast and Multicast communication)</p> <p><b>Note:</b> Leave this field blank for Multicast communication.</p>
Comment	Any useful comment.

## I/O Devices

Field	Value
Name	A unique name (up to 16 alphanumeric character) for the I/O Device being identified. Assign each I/O Device with a unique name in the Plant SCADA system.
Number	A unique number for the I/O Device (0-16383). Assign each I/O Device a unique number in the Plant SCADA system, (unless redundancy of the I/O Device is being used).
Address	Leave this field blank.
Protocol	KNX
Port Name	Refers to the port defined in the Ports form.
Comment	Any useful comment.

## Tag Addressing

A separate variable tag needs to be added to your Plant SCADA project for each group address within the KNX system you would like Plant SCADA to communicate with.

Variable tag addresses are configured according to the following formats:

**x/y/z:a[!w]**

or

**x/y/z:a.b[!w]**

Where:

- **x** is the main group number [0-31].
- **y** is the sub group number [0-7].
- **z** is the group address [0-255].
- **a** is the main encoding type.
- **b** is the minor encoding type.
- **!w** is an optional flag to define a tag as write only.

The main, sub and group address fields are specified in decimal. The encoding types are specified in decimal, with leading zeros to pad to three characters for the minor encoding type.

The minor encoding type is optional. If the minor encoding is not specified, then an encoding that matches as close as possible to an unscaled memory copy shall be chosen. See [KNX Data Type Mappings](#) for a full list of Plant SCADA data type mappings.

## Tag Ranges

Some data types will not map directly to Plant SCADA data types. In some cases, a tag may not span the entire range of values of a given type. Instead, it uses a reduced set of values that represent the actual values within that range.

For example, 3-bit controlled types are typically used in applications such as opening or closing shades. The controller can be instructed to open or close based on a percentage amount. This is mapped through to an INT tag in Plant SCADA that will take a value in the range -100, -50, -25, -12, -6, -3, -1, 0, 1, 3, 6, 12, 25, 50, 100, representing how much to open or close by.

Plant SCADA will convert a representative value used by a device to reflect the actual value, allowing for more meaningful interpretation.

## Examples

Plant SCADA tag address	Values range
0/0/0:3.007	Values fall within the following range: -100, -50, -25, -12, -6, -3, -1, 0, 1, 3, 6, 12, 25, 50, 100.
0/0/0:3.008	They represent the percentage a value increased or decreased by. If the operator inputs a value other those that appear in this range, the driver will find closest match before sending the command to the device. Example, -44 will become -50, 19 will become 25 and so on.

Plant SCADA tag address	KNX group address
1/1/123:1	Main Group 1, Middle Group 1, Group Address 123 interpreted as a Boolean Switch Plant SCADA type: DIGITAL
1/1/123:1!w	Main Group 1, Middle Group 1, Group Address 123 interpreted as a Boolean Switch Plant SCADA type: DIGITAL <b>Note:</b> This variable is a write only tag.
2/3/44:1.009	Main Group 2, Middle Group 3, Group Address 44 interpreted as a Boolean Open/Close signal Plant SCADA type: DIGITAL
2/3/44:1.009!w	Main Group 2, Middle Group 3, Group Address 44 interpreted as a Boolean Open/Close signal Plant SCADA type: DIGITAL <b>Note:</b> This variable is a write only tag.
1/1/55:5	Main Group 1, Middle Group 1, Group Address 55 interpreted as an 8 bit unsigned relative position Plant SCADA type: INT <b>Note:</b> This variable will have a range of [0..255]. Values outside this range will result in an error.
1/1/55:5.001	Main Group 1, Middle Group 1, Group Address 55 interpreted as an 8 bit unsigned scaling value Plant SCADA type: BYTE <b>Note:</b> This variable will have a range of [0..100]. Values outside this range will result in an error.
7/5/202:16	Main Group 7, Middle Group 5, Group Address 202 interpreted as an 8 bit encoded string Plant SCADA type: STRING <b>Note:</b> The maximum length of the string is 13 characters plus a null char terminator.

## KNX Data Type Mappings

The Date and Time data types will be read and written using the standard Plant SCADA time formats.

The driver will enforce the data ranges as specified for each data type.

When a write is issued to a controlled data type the controlled bit will be set to 1.

Generic KNX Type	Specific KNX Type	Plant SCADA Data Type
Boolean	1 (defaults to 1.001)	DIGITAL
Boolean	1.001	DIGITAL
Boolean	1.002	DIGITAL
Boolean	1.003	DIGITAL
Boolean	1.004	DIGITAL
Boolean	1.005	DIGITAL
Boolean	1.006	DIGITAL
Boolean	1.007	DIGITAL
Boolean	1.008	DIGITAL
Boolean	1.009	DIGITAL
Boolean	1.010	DIGITAL
Boolean	1.011	DIGITAL
Boolean	1.012	DIGITAL
Boolean	1.013	DIGITAL
Boolean	1.014	DIGITAL
2-Bit Controlled	2(defaults to 2.001)	BYTE
2-Bit Controlled	2.001	BYTE
2-Bit Controlled	2.002	BYTE

2-Bit Controlled	2.003	BYTE
2-Bit Controlled	2.004	BYTE
2-Bit Controlled	2.005	BYTE
2-Bit Controlled	2.006	BYTE
2-Bit Controlled	2.007	BYTE
2-Bit Controlled	2.008	BYTE
2-Bit Controlled	2.009	BYTE
2-Bit Controlled	2.010	BYTE
2-Bit Controlled	2.011	BYTE
2-Bit Controlled	2.012	BYTE
3-Bit Controlled	3 (defaults to 3.007)	INT
3-Bit Controlled	3.007	INT
3-Bit Controlled	3.008	INT
3-Bit Controlled	3.009	(currently not supported)
Character Set	4 (defaults to 4.002)	BYTE
Character Set	4.001	BYTE
Character Set	4.002	BYTE

8-Bit Unsigned Value	5 (defaults to 5.010)	INT
8-Bit Unsigned Value	5.001	BYTE
8-Bit Unsigned Value	5.003	INT
8-Bit Unsigned Value	5.004	INT
8-Bit Unsigned Value	5.010	INT
8-Bit Signed Value	6 (defaults to 6.010)	INT
8-Bit Signed Value	6.010	INT
Status with Mode	6.020	(currently not supported)
2-Octet Unsigned Value	7 (defaults to 7.001)	UINT
2-Octet Unsigned Value	7.001	UINT
2-Octet Unsigned Value	7.002	UINT
2-Octet Unsigned Value	7.003	UINT
2-Octet Unsigned Value	7.004	UINT
2-Octet Unsigned Value	7.005	UINT
2-Octet Unsigned Value	7.006	UINT
2-Octet Unsigned Value	7.007	UINT
2-Octet Unsigned Value	7.010	UINT
2-Octet Unsigned Value	7.011	UINT

2-Octet Unsigned Value	7.012	UINT
2-Octet Unsigned Value	7.013	UINT
2-Octet Signed Value	8 (defaults to 8.001)	INT
2-Octet Signed Value	8.001	INT
2-Octet Signed Value	8.002	INT
2-Octet Signed Value	8.003	INT
2-Octet Signed Value	8.004	INT
2-Octet Signed Value	8.005	INT
2-Octet Signed Value	8.006	INT
2-Octet Signed Value	8.007	INT
2-Octet Signed Value	8.010	INT
2-Octet Signed Value	8.011	INT
2-Octet Float Value	9 (defaults to 9.002)	REAL
2-Octet Float Value	9.001	REAL
2-Octet Float Value	9.002	REAL
2-Octet Float Value	9.003	REAL
2-Octet Float Value	9.004	REAL
2-Octet Float Value	9.005	REAL

2-Octet Float Value	9.006	REAL
2-Octet Float Value	9.007	REAL
2-Octet Float Value	9.008	REAL
2-Octet Float Value	9.010	REAL
2-Octet Float Value	9.011	REAL
2-Octet Float Value	9.020	REAL
2-Octet Float Value	9.021	REAL
2-Octet Float Value	9.022	REAL
2-Octet Float Value	9.023	REAL
2-Octet Float Value	9.024	REAL
2-Octet Float Value	9.025	REAL
2-Octet Float Value	9.026	REAL
2-Octet Float Value	9.027	REAL
2-Octet Float Value	9.028	REAL
Time	10 (defaults to 10.001)	(currently not supported)
Date	11 (defaults to 11.001)	(currently not supported)
4-Octet Unsigned Value	12 (defaults to 12.001)	ULONG

4-Octet Signed Value	13 (defaults to 13.001)	LONG
4-Octet Float Value	14 (defaults to 14.005) The range supported for DPT14 is 14.000-14.079	REAL
Access	15 (defaults to 15.001)	LONG
String	16 (defaults to 16.001)	STRING
String	16.001	STRING
8-Bit Unsigned Value	20.102	BYTE
4-Octet Unsigned Value	232.600	LONG
2-Octet Signed Value	237.600	INT
8-Bit Unsigned Value	238.001	BYTE
8-Bit Unsigned Value	238.600	BYTE

## KNX Tag Import

Plant SCADA can automatically generate variable tags for all supported KNX group addresses configured in the KNX Device. Tags are imported per I/O device from an XML file, exported from ETS5. It is important that group addresses exported from ETS5 are for device objects linked to a router, configured for Unicast mode communication with the I/O device.

### To generate variable tags using tag import corresponding to KNX device properties:

1. In ETS5, export the group address to an export file using xml format.

---

**Note:** Make sure that each group address has a data type association.

2. Open Plant SCADA Studio, and select **Import Tags** from the **Topology** activity.
3. The **Import Variable Tags** dialog box opens.
4. In the **Destination** panel, select the **I/O Device**.

---

**Note:** Make sure the I/O device is configured for the KNX protocol. Otherwise you will not be able to

---

configure the device connection. Ensure that the I/O device name does not contain comma, else an error message is displayed saying tags cannot be imported. If I/O device name contains any other special character, then compilation errors will occur.

5. In the **Source** panel, set the **Database type** to KNX.
6. In the **External database** field, click **Browse** button to open KNX Device Configuration dialog box. To configure the KNX device:
  - a. In **Tag Import Options**, identify the required XML file to import in the **XML File Path** field. Click **Browse** button to locate the file.

**Note:** The tag import functionality is available for XML file format generated by ETS5 configuration tool only. XML file format from previous versions such as ETS4 or before does not contain the data type.
  - b. In the **Advanced Options** panel, if you select **Use GroupAddress Name as Tag Name**, variable tags are generated in the following format:

<IODevice>\<GroupAddressName> For example, "IODev1\sensor"

Else, the tags will be generated in the following format:

<IODevice>\<Tag<n>> For example, "IODev1\Tag1", "IODev1\Tag2" and so on.
  - c. Click **OK** to close the KNX Device Configuration dialog box.
7. Click **Import** button from the Import Variable Tags dialog box.

Plant SCADA reads the group addresses and data types from the XML file, and generates the variable tags for the KNX group addresses.

---

**Note:** At present, the tag import functionality is limited to generating the variable tags and the corresponding tag address. Here, the tag address contains a group address and a data type. The description associated with the group address will be copied in to the comment field of the corresponding Plant SCADA tag.

It does not support creating different value ranges for variable tags of a particular data type where the range is defined by minor encoding. For example: 5.001, 5.004. Here 5.001 refers to percentage from 1 to 100 and 5.004 refers to percentage from 1 to 225. If the value ranges are different, you have to configure manually in Plant SCADA.

---

## Tag import log file

Each time a tag import is executed a log file is created in KNXImportLog folder, under the Logs folder, at the location specified in the [CtEdit] section of ini file. The log file will be overwritten each time.

Below are some of the special scenarios of tag import:

- **Group addresses sharing identical name and communicating via same router.**

In each of the following scenario, the XML file will have duplicate group names:

1. If **Use GroupAddress Name as Tag Name** option is selected on the KNX Import File Browser dialog box  
In this case the XML file contains multiple group addresses with duplicate group address name. After import, you need to rename tags created with duplicate names, else compiler errors will occur.
2. If **Use GroupAddress Name as Tag Name** option is not selected on the KNX Import File Browser dialog box  
In this case the XML file contains multiple group addresses with duplicate address name. Tags with separate names will be created for these group addresses.

- **Special characters in group address name.**

Consider an example where a group address name contains special characters and the tag import option **Use GroupAddress Name as Tag Name** is selected. In this case, KNX driver will skip the special characters and generate the tags, as special characters are not allowed in tag names. Hence there is a chance of duplication of tags. In such case, you need to rename the tags, else compiler errors will occur. Refer tag import log file for details of group address names with special characters.

---

**Note:** Backslash and under scores are allowed.

---

- **Group address without a data type.**

Any group address with data type not configured will not be considered during tag import. Refer tag import log for details of group addresses with data type not configured.

## Advanced Configuration and Maintenance

This section provides further information about the implementation of KNX devices within the Plant SCADA.

- [Configuring Redundancy](#)
- [Quality and Timestamps](#)
- [Background Polling](#)
- [Write Requests](#)
- [Data Type Mismatches](#)
- [Maximum Transmission Rate Control](#)
- [Customize a Project using Citect.ini Parameters.](#)

### Configuring Redundancy

Redundancy for the KNX driver in unicast communication supports the traditional redundancy model for Plant SCADA I/O devices as described in the I/O Server Redundancy section of Plant SCADA help.

Redundancy in multicast communication is different from the traditional redundancy model for Plant SCADA I/O devices. In this case one I/O device represents an entire KNX network.

The use of IP multicasting as the communications medium between IP router devices means that only one connection per network interface is possible. In this context, only one I/O device can be defined per I/O server, unless the server supports multiple interfaces and the **LocalIPAddress** parameter is used to bind different I/O devices to different interfaces.

The use of IP multicast, and thus the lack of a direct connection to the KNX IP router devices, means that the I/O server is very limited in the extent to which it can signal a device to be offline.

The KNX driver will only signal a device as offline under the following conditions:

A media disconnect event is received by the socket, i.e. network cable pulled out

All of the group addresses on the device has timeout and become bad

It is possible for two I/O devices on two servers to be configured as a redundant pair. In this case, the two devices will act as a hot-standby, with both servers maintaining a live cache.

### Quality and Timestamps

The KNX driver implements quality and timestamp information, with **GOOD** and **BAD** quality levels used to

reflect the state of a cache value.

A tag will be flagged as bad quality on the basis of either a data type/size mismatch, or for exceeding the cache lifetime.

The quality and timestamp values are used in relation to cache lifetime management. Because no two group addresses can be blocked together, the driver will use the quality value to return #COM for an individual tag.

See [KNX Parameters](#).

## Background Polling

If configured to route telegrams properly, all IP routers on a network will send a message each time a group address is written to by an input device. The driver processes these telegrams and updates the cache value for the group address, thus updating the timestamp, value and quality.

However, the KNX driver does not maintain a continuous connection to each network; there can be multiple KNX to IP routers and it is impossible for Plant SCADA to tell if telegrams from devices have been lost. This means the delivery of on-change notifications can be unreliable.

Because of this, a slow background poll of all group addresses will be conducted. As each group addresses reaches half of the cache life time value a read request will be issued for the group address.

## Write Requests

Write requests issued to KNX IP routers in 'routing' mode are unconfirmed. As such, write requests issued to the driver are delivered in a best-effort manner.

It is not possible for the driver to confirm write requests using a read command because another device may have also written a different value to the group address in the interim.

To improve the accuracy of cache values, a read request for the group address value will be issued immediately after the write request.

## Data Type Mismatches

The configuration between Plant SCADA and the KNX network in regard to group address data types must match. If the KNX driver receives a group address telegram containing data in a format incompatible with the Plant SCADA configuration, the address will be considered bad quality and a read request will return a GENERIC\_INVALID\_DATA\_FORMAT error.

See [KNX Data Type Mappings](#).

## Maximum Transmission Rate Control

By default, the KNX driver uses a bandwidth rate limiting mechanism to restrict outbound traffic to approximately 20 percent of its full capability. This is because KNX uses a 9600 bps serial bus, and there is a possibility the packet buffers of individual IP routers may overflow.

It is possible to set this value higher using the [KNX Parameters](#) parameter, however, this is only recommended for a large KNX network on a single line containing many IP router devices with concise filter tables. If a value other than the default is used, you should monitor and confirm the delivery of telegrams to and from the KNX network.

This parameter only governs outbound traffic, so it should be set at a rate that allows enough bandwidth to accommodate a reply to each read request.

You can also use the [KNX Parameters](#) parameter to adjust the minimum timing delay between successive transmitted frames by the driver.

## Customize a Project using Citect.ini Parameters

The KNX driver supports the following parameter categories:

- [KNX Parameters](#)
- [Logging Parameters](#).

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

- Do not under any circumstances change or remove any citect.ini parameters without an informed understanding of the possible implications of your actions.
- Do not delete sections of the citect.ini file without an informed understanding of the possible implications of your actions.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** At times during the life cycle of a driver, circumstances may call for the creation of specific citect.ini parameters that remain undocumented. Always seek the advice of Technical Support regarding the possible implementation of undocumented parameters before you make any changes to a citect.ini file.

## KNX Parameters

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

The table describes the KNX driver-specific parameters:

Driver Specific Parameter	Description	Allowable Values	Default Value
CacheLifeTime	The time a cache value will remain active for. Once an item reaches half its cache life time, the driver will process a read	Any positive integer representing number of seconds.	300 (seconds)

Driver Specific Parameter	Description	Allowable Values	Default Value
	request for the group address to refresh the cache value.		
IgnoreRouterDisconnect	When the value of this parameter is set to 1, the driver will not take the I/O device offline if a disconnect request is received from the router.	0 or 1	0
InitTimeout	The maximum time the I/O server is allowed to wait for replies to be received from all group addresses to populate the cache before either aborting the InitUnit or bringing the unit online.	Any positive integer representing number of seconds.	120 (seconds)
InterFrameDelay	The minimum timing delay between successive transmitted frames by the driver. This can be used to avoid flooding the buffers of gateway device.  See <a href="#">Maximum Transmission Rate Control</a> .	Any positive integer representing number of milliseconds.	50 (ms)
LocalIPAddress	Sets the local IP address the I/O server will attempt to bind to.	Any four octet dotted notation IP address. e.g. 192.168.1.1	0.0.0.0 (Instructs the IP stack to bind to any/ all available network interfaces)
LocalPort	Sets the local port number the I/O server will attempt to bind to. For the KNXnetRouter, 3671 is the standard KNXnet/IP Port). If LocalPort is not configured it will default to the MulticastPort only for multicast mode of communication.	Any valid port number	3671

Driver Specific Parameter	Description	Allowable Values	Default Value
MaxTransmitRate	<p>The maximum traffic rate that the driver is allowed to generate in bits per second. The KNX wire protocol is a 9600 bps serial protocol. This parameter allows the maximum traffic rate generated by the KNX driver to be controlled.</p> <p>This only governs outbound traffic. As such it should be set to a lower value than might be expected, as read requests will require bandwidth for a reply.</p> <p>In a large network with many IP router devices with concise filter tables, it may be helpful to set this value much higher than 9600, provided the engineer has tested and confirmed that the packet buffers of any individual IP router will not be overflowed.</p> <p>See <a href="#">Maximum Transmission Rate Control</a>.</p>	Any positive integer representing a number of bits per second. (A value less than 200 is not recommended.)	2000 bps (approx. 20% of maximum bus throughput)
MulticastAddress	Sets the multicast IP address the I/O server will attempt to communicate with.	Any four octet dotted notation IP address within the range of 224.0.0.0 to 239.255.255.255 e.g. 224.168.1.1	224.0.23.12 (Standard KNXnet/IP multicast address)
MulticastPort	Sets the multicast port number the I/O server will attempt to send packets to. For the KNXnetRouter, 3671 is the standard KNXnet/IP Port.	Any valid port number	3671
OfflinePollTime	The polling period (in	Any positive integer	CacheLifeTime/2

Driver Specific Parameter	Description	Allowable Values	Default Value
	seconds) of tags which have bad quality.	representing number of seconds.	
PhysicalAddress	<p>Sets the physical address for the Plant SCADA I/O server in the KNX network. All messages sent from the Plant SCADA I/O server will be sent with this address as the source address.</p> <p>If the value is larger than 65535, it will be masked to only use the lower 16 bits. The value represents Area Number (bits 12-15), Line Number (bits 8-11), Device Number (bits 0-7).</p> <p>The value has to be specified in decimal. If hexadecimal notation is used the address will resolve to 0.</p>	Any 16 bit value	0
Retry	<p>Determines the number of retries the driver should try to send the read request when there is no response from router. After exceeding the number of retries, the driver will set the quality of the tags in the read request to 'Bad' and stop sending read requests for these tags until reaching half of its cache life time.</p> <p><b>Note:</b> Changing the default value of this parameter is only useful to decrease the network traffic between the driver and KNX router caused by unsuccessful read requests.</p>	Any positive integer representing the number of retries.	Zero (0). The driver will not set the tag quality to 'Bad' and will continue sending read requests after exceeding the number of retries.

Driver Specific Parameter	Description	Allowable Values	Default Value
Timeout	Specifies the time delay between each read request.	0 to 32000 (milliseconds)	5000 milliseconds (5 seconds)
TTL	Sets the maximum time-to-live (TTL) on outbound multicast IP packets.  This should only be adjusted on the advice of network administrator. It would only be necessary to control the routing of multicast packets from one IP subnet to another.	Any positive integer	16
UnitType	Determines the type of bus interface unit the device should communicate with. 'KNXnetRouter' represents an KNXnet/IP device configured for routing.	KNXnetRouter	KNXnetRouter

The table describes the KNX INI parameters used specifically for Unicast communication:

INI Parameter	Description	Allowable Values	Default Value
TunnelConnectionRetries 1	Determines the number of retries the driver should try to connect with the router when the initial connection request is unsuccessful.  For example, If the value is 1 then it will retry to send the connect request one more time after the initial connect request is unsuccessful.  Once the number of retries ends, the driver will not make further attempts to connect and will report the unit offline.	0 - 5	3
TunnelConnectionTimeout	Connection related	Any positive integer	10000 (ms)

INI Parameter	Description	Allowable Values	Default Value
t <sup>1</sup>	packets are time bound. If no response received from router within this timeout period, then the driver will ignore any later responses. The timeout option helps driver to determine either to send the packets again or to make the unit offline.	representing number of milliseconds.	
TunnelConnectionActive	Defines the interval to periodically check the connection state with the router. Driver sends a CONNECTSTATE_REQUEST to the router to verify that the connection is active. These parameters can also be configured at unit level. The unit has to be configured for unicast communication.	5 - 100 seconds	60 (seconds)

1. The **InitTimeout** needs to be greater than **TunnelConnectionRetries** x **TunnelConnectionTimeout**.

### Logging Parameters

#### **WARNING**

##### **UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

The table describes the KNX Logging parameters:

Driver Specific Parameter	Description	Allowable Values	Default Value
DebugLevel	Controls which levels of log message are written to the log file.	ALL or any combination of the following values separated by pipe character ( ) :  WARN ERROR	Empty string (no logging).

Driver Specific Parameter	Description	Allowable Values	Default Value
		TRACE DEBUG See <a href="#">Logging</a> for examples.	
DebugCategory	Controls which categories of log message are written to the log file.	ALL or any combination of the following values separated by pipe character ( ) :  PROT DCB THREAD MISC SOCK ENTRY UNIT. See <a href="#">Logging</a> for examples.	Empty string (no logging).

## See Also

[KNX Parameters](#)

## Troubleshooting

The following topics include information about the Plant SCADA tools provided to help resolve problems with communication and configuration.

- [Logging](#)
- [Specific Errors](#)
- [Frequently Asked Questions.](#)

## Logging

The KNX driver can log combinations of trace levels across different categories. This is achieved by setting the following INI parameters.

## [KNX]DebugLevel

This parameter allows you to define the trace level. The options include:

Option	Description
WARN	Output warning messages.
ERROR	Output error messages.
TRACE	Output trace messages.
DEBUG	Output debug messages.
ALL	All of the above.

## [KNX]DebugCategory

This parameter allows you to enable logging for a particular category of trace. The options include:

Option	Description
PROT	Trace of communication between the driver and the lower level protocol.
DCB	Front end driver trace.
THREAD	Thread trace.
MISC	Any other debugging.
SOCK	Socket trace.
ENTRY	Driver entry trace.
UNIT	I/O device trace.
ALL	All of the above.

In both cases, you can use any combination of the available options separated by a pipe character ( | ).

### Example

```
[KNX]
DebugLevel=WARNING|ERROR|TRACE
DebugCategory=ALL
```

In most cases, you can set the DebugLevel and DebugCategory parameters to ALL. If this results in a large, cumbersome log file that causes older data to be overwritten, you can consider refining the debug options.

---

**Note:** These parameters do not have a value specified by default. To enable logging for this driver, you need to manually configure these parameters in your INI file.

The events generated by the KNX driver are logged in the following Plant SCADA syslog file:

`syslog.IOServer.<Cluster name>.<IO Server name>.dat`

This file is located in the directory specified by the Citect.ini parameter **[CtEdit]Logs**. By default, this location will be:

%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Logs

The size of the syslog file and its rollover process is configured via the **[Debug]SysLogSize** and **[Debug]SysLogArchive** INI parameters. For more information, see the topic *Debug Parameters* in the Plant SCADA documentation.

**Note:** If the number of messages being processed for a driver exceeds the value specified by the Plant SCADA parameter [Kernel]ErrorBuffers, the pending messages will not be logged. If this occurs, you can increase the value of this parameter so that all the messages generated by the driver are reported. The number of messages that are not logged can be monitored via the Kernel's General window under the item "Lost Errors". If you adjust [Kernel]ErrorBuffers and Lost Errors are still occurring, you may need to adjust the level of system logging.

## See Also

[Logging Parameters](#)

## Specific Errors

Plant SCADA driver errors can occur when a device:

- does not respond
- is disconnected
- is offline
- returns an error.

Plant SCADA has two kinds of driver errors - generic and specific.

Generic errors are hardware errors 0-31, and are common to all protocols (for more information, see the topic *Error Messages* in the Reference Information section of the main Plant SCADA documentation).

Specific errors are unique to a particular driver, and may not be recognized by the hardware alarm system. The driver converts the specific errors into generic errors so they can be identified by the I/O Server. This means when a driver becomes inoperable, there is often both a specific error and a corresponding generic error.

When a hardware error occurs, Plant SCADA generates an alarm, and displays the alarm on the hardware alarm page (in the alarm description of the hardware alarm). To see the error number, verify that you have Alarm Category 255 defined with a display format that includes {ErrDesc,15} {ErrPage,10}.

## KNX Specific errors

Error value	Error definition	Description
0x20000001	ERR_DEMO_EXPIRED	Unlicensed demo system has timed out
0x20000002	ERR_INIT_UNIT_IN_PROGRESS	Notification only – the unit initialization is still in progress
0x20000003	ERR_INIT_UNIT_TIMEOUT	The unit initialization process has timed out

Error value	Error definition	Description
0x20000004	ERR_INVALID_INTERNAL_STATE	Internal software error
0x20000005	ERR_ALL_TAGS_BAD_QUALITY	The unit has been taken offline because all tags on the unit are in a bad quality state
0x20000006	ERR_TAG_NOT_FOUND	The requested tag could not be found
0x20000007	ERR_TAG_QUALITY_BAD	The tag has a bad quality value

## Frequently Asked Questions

This section addresses problems you might encounter when setting up and operating the KNX driver:

- **Why does it take so long for my KNX IP router to come online?**

At startup, a read request is issued for each group address to initially populate the cache. Ideally, the IP router should come online as soon as a response has been received back from each group address.

However, there is no way for Plant SCADA to determine if a read request has been lost due to an unreliable destination address.

Under these circumstances, you would like any valid addresses to be acknowledged, and for communication to commence. The lost addresses should not hold up communication with the rest of the system.

For this reason, the KNX driver has a initialization timeout period set to 120 seconds. After this period of time, all validated group addresses (a minimum of at least one) will commence communicating; all invalid addresses will be terminated with an error written to the driver log.

Therefore, if your IP router is taking a couple of minutes to come online, it probably means invalid group addresses are forcing the device to the end of the initialization timeout period. You should check the driver log to determine which addresses are causing the problem.

- **Everything is plugged in to the network, but I can't see any packets from Plant SCADA in the bus monitor.**

If this is occurring, check the following:

- If you have multiple network cards (or virtual NICs just from having VMWare installed) you need to confirm that the parameter [KNX]LocalIPAddress is set to the correct out bound interface.
- Confirm that the routers are configured correctly to route messages to and from the KNX network.
- Confirm that all group addresses are configured as readable.

- **When I try to write to a group address, the new value flickers and then reverts to the old address.**

Confirm that the group address is configured to be writable.

- **Unicast mode communication to the KNX network not established. Unit status is offline and all tags show BAD quality**

Make sure the Router is configured for Unicast communication mode. Also check that special option field of port form is configured with the IP address and Unicast port of the router.

## MELSECQ Driver

The MELSECQ driver supports TCP/IP communication with the Mitsubishi Electric [MelsecQ via TCP/IP](#), [MelsecQnA via TCP/IP](#) and [Melsec iQ-R](#) PLCs.

**Note:** The MelsecQ driver supersedes the MelsecQnA driver.

### TCP/IP Protocol Setup

The driver will communicate with the Ethernet card in the SCADA computer using the Plant SCADA TCP/IP driver (TCPPIP.DLL).

This protocol requires Winsock v1.1 TCP/IP software to be installed in your computer. This protocol does not require any software from Mitsubishi Electric.

The software needs to be properly installed before configuring this protocol. In particular, the correct values for IP address, subnet mask and default gateway need to be set before proceeding.

### See Also

[Supported Devices](#)

[Customize a Project using Citect.ini Parameters](#)

[MELSECQ Driver Specific Errors](#)

[Logging](#)

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

### **DANGER**

**DANGER** indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.

**⚠ WARNING**

**WARNING** indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.

**⚠ CAUTION**

**CAUTION** indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.

**NOTICE**

**NOTICE** used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.

**Please Note**

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

**Before You Begin**

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ WARNING****LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

## Supported Devices

- [MelsecQ via TCP/IP](#)

The MELSECQ driver communicates to a Melsec Q PLC through Ethernet interface modules QJ71E71-100, QJ71E71 and QJ71E71-B2.

The Ethernet module is an interface on the PLC side for connecting the Q series PLC with the host system, such as a personal computer and a work station where Plant SCADA is installed, and between PLCs using the TCP/IP or UDP/IP communication protocol via Ethernet (100BASE-TX, 10BASE-T, 10BASE5, 10BASE2).

- [MelsecQnA via TCP/IP](#)

The MELSECQ driver communicates to a Melsec QnA PLC through Ethernet interface modules AJ71QE71-B5, A1SJ71QE71-B2 and A1SJ71QE71-B5.

The Melsec-QnA driver can connect with up to 8 TCP/IP channels per AJ71QE71 interface module. One channel (port) is used to communicate with a primary Melsec-QnA PLC; the other ports can then be used for setting up extra Ethernet communication links to remote I/O modules.

The primary Melsec QnA PLC can also communicate with other PLC CPUs via a MELSECNET network (MELSECNET/10, MELSECNET/II or MELSECNET/B) connected to the primary PLC. The limit of 8 channels per Ethernet AJ71QE71 is set by the number of fixed buffers on this module. The driver itself has an upper limit of 32 communication channels; this driver setting can be changed if necessary.

- [Melsec iQ-R PLC](#)

---

**Note:** The same Ethernet card that is being used for Plant SCADA communication can be used for PLC communication, though this may lower performance.

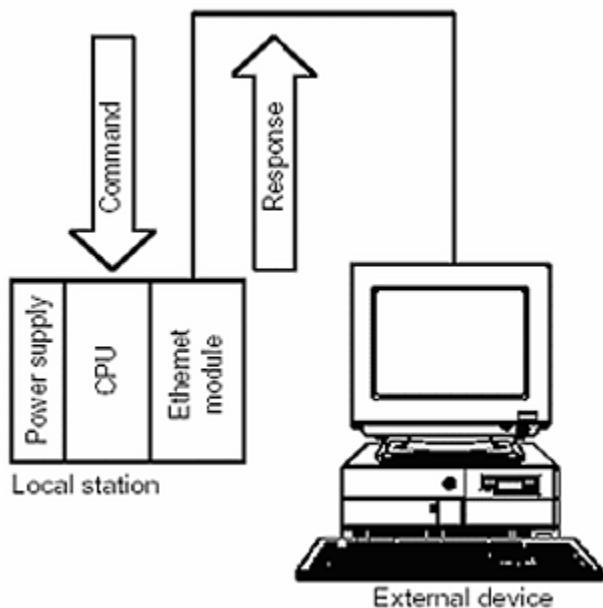
---

The maximum request length for the MELSECQ protocol is 2048 bits (256 bytes).

### MelsecQ via TCP/IP

The Ethernet module (models QJ71E71-100, QJ71E71 and QJ71E71-B2) is an interface module on the PLC side for connecting the Q series PLC with the host system, such as a personal computer and a work station where Plant SCADA is installed, and between PLCs using the TCP/IP or UDP/IP communication protocol via Ethernet (100BASE-TX, 10BASE-T, 10BASE5, 10BASE2).

The following diagram illustrates how the MelsecQ Ethernet module communicates with the external device.



### MelsecQ - Device Address

The information you enter here will be placed in the Special Options field of the Ports settings, with the following format:

-la.b.c.d -Pn -T

where:

- *a.b.c.d* - is the destination IP address using decimal numbers (for example 192.9.2.60).
- *n* - the destination Port number. Often one physical port has several virtual ports, used for different purposes.
- -T - forces the driver to use TCP, rather than UDP (-U).

See the topic [MelsecQ via TCP/IP](#) for setup details.

## MelsecQ - Hardware Setup

Apart from 100BASE-TX, 10BASE-T, 10BASE5, or 10BASE2 Ethernet cabling, no special hardware setup is required for a Plant SCADA interface.

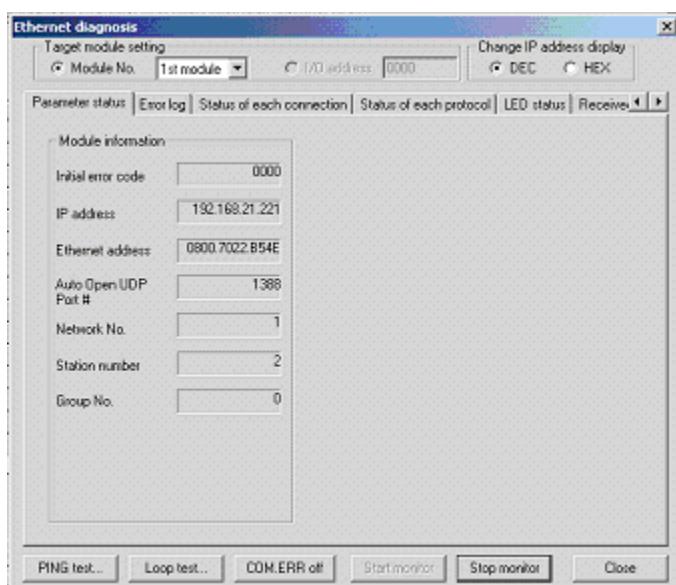
**Note:** MelsecQ devices support an AutoUDP port (port 5001) that opens when a module is initialized with an IP address. Using this port for SCADA communications is not recommended, as other applications may attempt to access the same device via the AutoUDP port.

## Plant SCADA Computer Setup

You should set up your specific Ethernet card as required and confirm basic network communications before proceeding with the communications configuration for the MelsecQ driver. Refer to your hardware documentation for instructions.

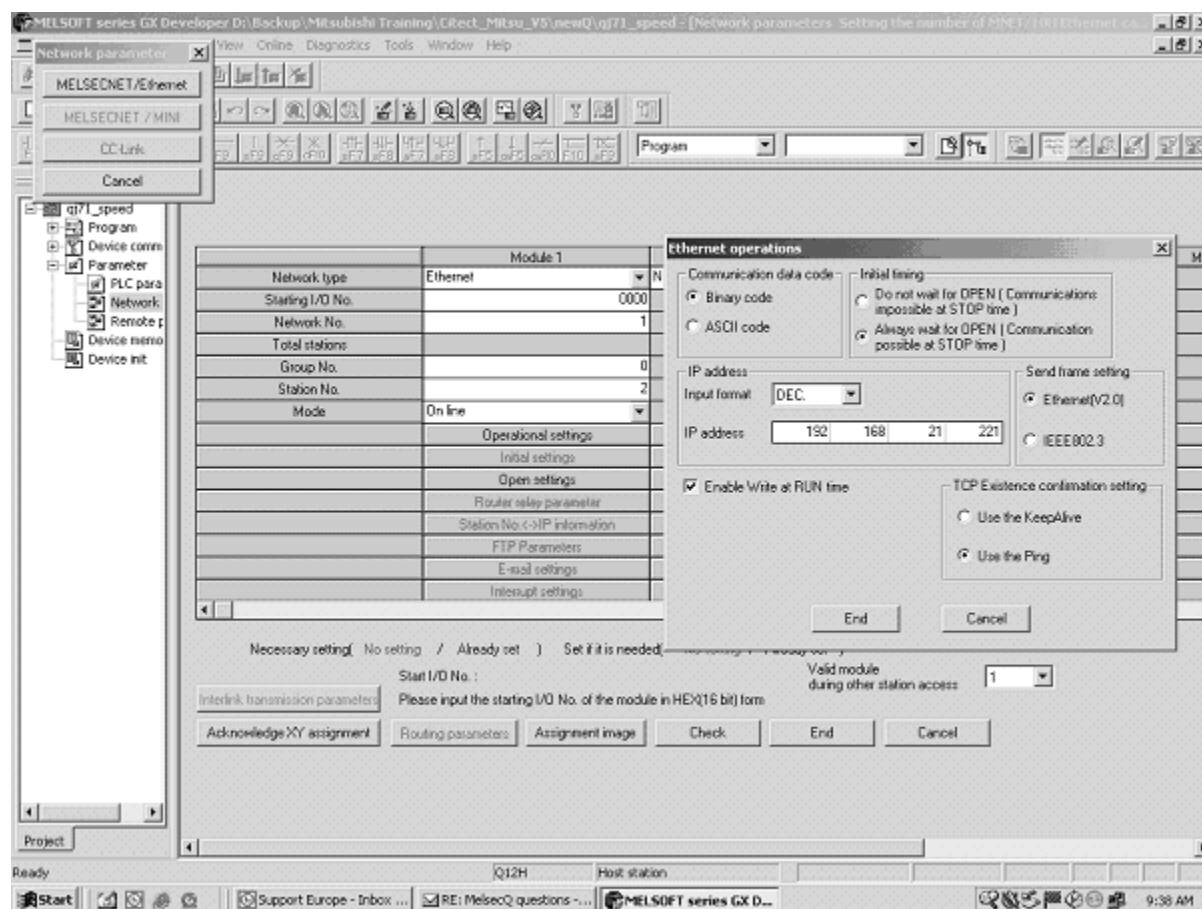
### To set up the MelsecQ software

1. Open Mitsubishi GX Developer, and then double-click Network parameters to display the dialog box.
2. Click MELSECNET/Ethernet . The Ethernet Operations dialog box appears.
3. Enter the parameters in the module listbox.
4. Click Operational Settings .
5. Enter the IP address .
6. Click End .
7. Save and write data to the PLC.
8. Manually reboot the PLC to reset the program memory.
9. Choose Ethernet Diagnostics on the Diagnostics menu to check that IP address is correct. The Ethernet diagnosis dialog box should look like this:



Use the following screenshot of the Mistubishi GX Developer to help you set up the PLC and Ethernet module.

After entering your settings, the dialog boxes should look like this:



## AJ71QE71-B5, A1SJ71QE71-B2 and A1SJ71QE71-B5 Hardware

The B2 has a Coax cable connector for 10Base2 and the B5 has an AUI connector for 10Base5 type connections. Mitsubishi recommend using a Transceiver with SQTEST of HeartBeat functionality if using the B5 model.

There are no switches on the Ethernet module models QJ71E71-100, QJ71E71 and QJ71E71-B2. Configuration is performed via the Mitsubishi GX Developer PLC software.

### Indicators on the front of the module

LED name (QJ71E71-100)	LED name (QJ71E71-B5,QJ71E 71-B2)	Display Description	LED is on	LED is off
RUN	RUN	Normal operation display	Normal	Abnormal
INIT	INIT	Initial processing status display	Normal completion	Not processed
OPEN	OPEN	Open processing status display	Normally opened connection available	Normally opened connection not available

SD	SD	Data sending display	Data being sent	Data is not sent
ERR	ERR	Setting abnormal display	ERR. ERR. Abnormal	Normal setting
COM.ERR.	COM.ERR.	Communication abnormal display	Communication abnormal occurrence	Normal communication in progress
100 M	(Not used)	Transmission speed display	100Mbps	10Mbps/When not connected
RD	RD	Data receiving status display	Data being received	Data not received

The [OPEN] LED turns on/off depending on the open states of user connections 1 to 16.

(The open states of the system connections, for example, automatic open UDP port are not included.)

The [ERR.] LED turns on in the following cases:

- When setting values in GX Developer (mode, station number, and/or network number) are incorrect.
- When an error has occurred in the Ethernet module or PLC CPU and the operation is disabled due to the error.

### MelsecQ - Hints and Tips

If the MELSECQ driver writes a string value to a PLC, it will use a null terminated ASCII value. If a third party driver or application does not correctly read the null character, buffered values may appear as additional junk characters. If this occurs, you may need to use Cicode to pad the values being written to a string tag.

An example of how to do this is provided in the Plant SCADA Knowledge Base article "Q4841: MELSECQ driver writes junk characters".

### MelsecQ - Communication Settings

To establish communication with a device, configure the **Boards**, **Ports** and **I/O Devices** settings in the Plant SCADA Studio's Topology activity. If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure these, use the information outlined below.

## Boards

Field	Value
Board Type	Enter TCPIP.
Address	Enter 0.
I/O Port	Leave this field blank.

Field	Value
Interrupt	Leave this field blank.
Special Options	This field is not used.

## Ports

For details about initializing the port, see [MelsecQ - Hints and Tips](#).

Field	Value
Port Number	Leave this field blank.
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	This field is not used.
Special Options	Enter the AJ71QE71 Ethernet module's TCP/IP address of the controller.  See "I/O Device Address" below for details about this option.

## I/O Devices

For details about initializing the I/O Device, see [MelsecQ - Hints and Tips](#).

Field	Value
I/O Device Address	The CPU number (0-64 or 255). You can specify the CPU monitoring time using the optional parameter "/Tt" after the CPU number. It should be possible to specify the network number (0-239 or 255) by using optional parameter "/Nn" after the CPU number (only useful for MELSECNET/10 PLC networks). Finally you can specify the ASCII Mode for data exchange by using optional parameter /A (the default is binary mode).  The address format is: <b>a [/Tt] [/Nn] [/Pp OR /Qq OR /Ss] [/A]</b> Where: <b>a</b> = CPU number 0 to 64 or 255 in decimal <b>t</b> = CPU monitoring time parameter (units 250 ms). The default is 10.

Field	Value
	<p><b>n</b> = Network number 0 to 239 or 255 in decimal. The default network number is 0.</p> <p><b>p</b> = PLC number, default is 0 for control PLC</p> <p><b>s</b> = System CPU</p> <ul style="list-style-type: none"> <li>1 - Control System CPU</li> <li>2 - Standby System CPU</li> <li>3 - System A CPU</li> <li>4 - System B CPU</li> </ul> <p><b>q</b> = External device ID for multidrop Q mode C24 devices.</p> <p>/P, /Q and /S parameters are mutually exclusive. The precedence of them is /P &gt; /Q &gt; /S.</p> <p><b>Note:</b> An in-depth description of the Network and CPU monitoring policy can be found in section 10.1 of the "MELSEC Q series Programmable Controller User's Manual - QnA Ethernet Interface Module type AJ71QE71(B5) - A1SJ71QE71-B2 - A1SJ71QE71-B5". Manual number IB (NA) 66661-C</p>
I/O Device Protocol	Enter MELSECQ.

### MelsecQ - Data Types

Data Types	Plant SCADA Data Format	Plant SCADA Data Type	Descriptions/ Special Usages/ Limitations/ Valid Ranges
Input relay	X<hexno>	DIGITAL	Read / Write
Output relay	Y<hexno>	DIGITAL	Read / Write
Link input	DX<hexno>	DIGITAL	Read / Write
Link output	DY<hexno>	DIGITAL	Read / Write
Internal relay	M<no>	DIGITAL	Read / Write
Link relay	B<hexno>	DIGITAL	Read / Write
Latch relay	L<no>	DIGITAL	Read / Write
Annunciator	F<no>	DIGITAL	Read / Write
Edge relay	V<no>	DIGITAL	Read / Write

Data Types	Plant SCADA Data Format	Plant SCADA Data Type	Descriptions/ Special Usages/ Limitations/ Valid Ranges
Step relay	S<no>	DIGITAL	Read / Write
Special relay	SM<no>	DIGITAL	Read / Write
Special relay for link	SB<hexno>	DIGITAL	Read / Write
Timer (Contact)	TS<no>	DIGITAL	Read / Write
Timer (Coil)	TC<no>	DIGITAL	Read / Write
Integrating Timer (Contact)	SS<no>	DIGITAL	Read / Write
Intergating Timer (Coil)	SC<no>	DIGITAL	Read / Write
Counter (Contact)	CS<no>	DIGITAL	Read / Write
Counter (Coil)	CC<no>	DIGITAL	Read / Write
Timer (Current Value)	TN<no>	INT, LONG, STRING, REAL, BCD, LONGBCD	Read / Write
Integrating Timer (Current Value)	SN<no>	INT, LONG, STRING, REAL, BCD, LONGBCD	Read / Write
Counter (Current value)	CN<no>	INT, LONG, STRING, REAL, BCD, LONGBCD	Read / Write
Data register	D<no>	INT, LONG, STRING, REAL, BCD, LONGBCD	Read / Write
Link register	W<hexno>	INT, LONG, STRING, REAL, BCD, LONGBCD	Read / Write
Special register for link	SW<hexno>	INT, LONG, STRING, REAL, BCD, LONGBCD	Read / Write
Special register	SD<no>	INT, LONG, STRING, REAL, BCD, LONGBCD	Read / Write
File Register (Block Access)	R<no>	INT, LONG, STRING, REAL, BCD, LONGBCD	Read / Write
File Register (Seq. Access)	ZR<hexno>	INT, LONG, STRING, REAL, BCD, LONGBCD	Read / Write
Index register	Z<no>	INT, LONG, STRING, REAL, BCD, LONGBCD	Read / Write

Where:

<no>	decimal number
<hexno>	hexadecimal number

DX & DY are DIRECT read and writes of the I/O values. X & Y are read / written at the end of the scan. Thus the X / DX , Y / DY areas are one of the same.

## EXAMPLES:

Data Type	INTEGER
Address	V77
Comment	Stator current
Data Type	DIGITAL
Address	X107
Comment	Running feedback indicator

## MelsecQnA via TCP/IP

The MELSECQ Driver supports TCP/IP communication to the MelsecQnA PLC, manufactured by Mitsubishi Electric, over Ethernet. Using this method you can connect to single MelsecQnA PLCs or to multiple MelsecQnA PLCs, using 10BASE5 or 10BASE2 Ethernet cabling.

Communication to the MelsecQnA PLC is specifically through the Melsec QnA interface module type AJ71QE71, but AJ71QE71-B5, A1SJ71QE71-B2 and A1SJ71QE71-B5 Ethernet interface modules are also supported. See the section on [MelsecQnA - Hardware Setup](#) for more information on these modules.

The primary MelsecQnA PLC is also able to communicate with other PLC CPUs via a MELSECNET network (MELSECNET/10, MELSECNET/II or MESLCNET/B) connected to the primary PLC.

## MelsecQnA - Device Address

The information you enter here will be placed in the Special Options field of the Ports form, with the following format:

-la.b.c.d -Pn -T

where:

- *a.b.c.d* is the destination IP address using decimal numbers (for example 192.9.2.60).
- *n* The destination Port number. Often one physical port has several virtual ports, used for different purposes.
- -T Forces the driver to use TCP, rather than UDP (-U).

See the topic [MelsecQnA via TCP/IP](#) for details about setting up MelsecQnA PLCs.

## MelsecQnA - Hardware Setup

### MelsecQnA PLC Setup

Apart from 10BASE5 or 10BASE2 Ethernet cabling, no special hardware setup is required for a Plant SCADA interface.

The PLC has to be programmed to operate the AJ71QE71 interface. Following is an example of the ladder logic to enable the TCP/IP port to work. This example uses ports 1020 & 1021. This code needs to exist in the initethet section of the QNA code.

---

**Note:** MelsecQnA devices support an AutoUDP port (port 5001) that opens when a module is initialized with an IP address. Using this port for SCADA communications is not recommended, as other applications may attempt to access the same device via the AutoUDP port.

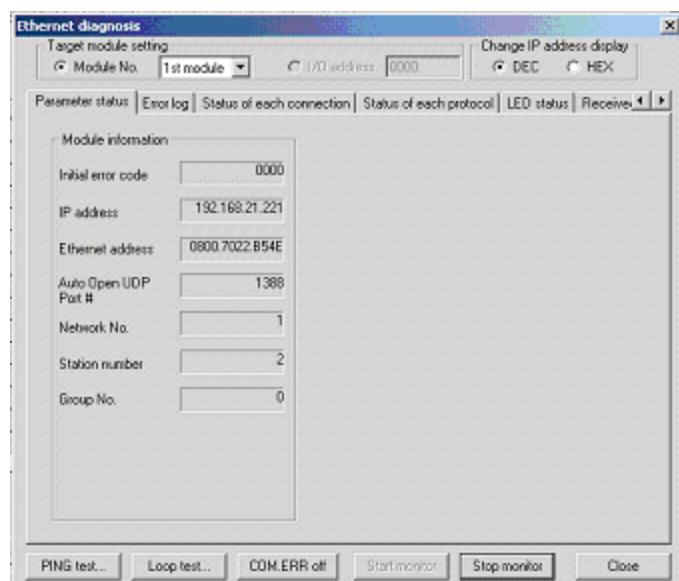
---

### Plant SCADA Computer Setup

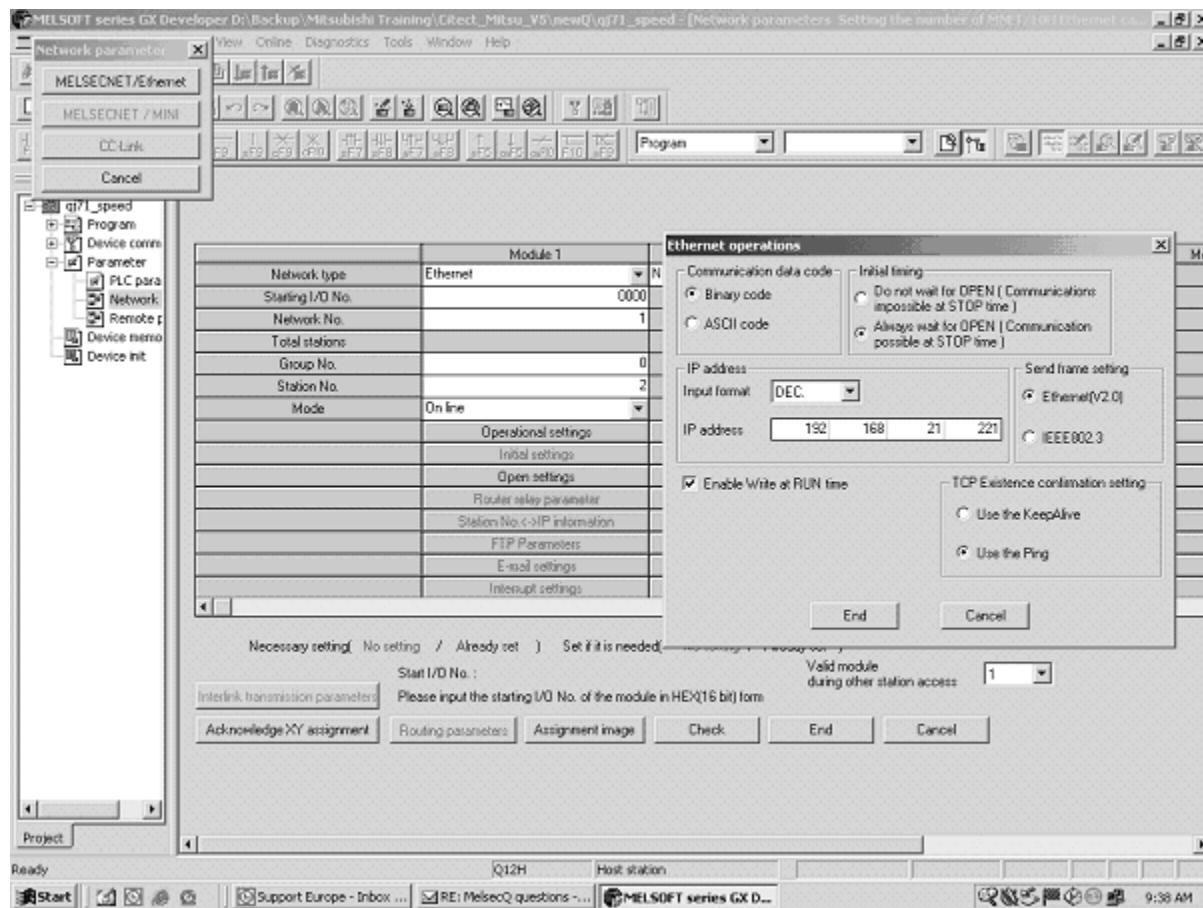
It is recommended that you setup your specific Ethernet card as required and confirm basic network communications before proceeding with the communications configuration for the MELSECQ driver. Refer to the documentation accompanying your hardware for instructions.

#### To set up the MelsecQ software

1. Open Mitsubishi GX Developer, and then double-click Network parameters to display the dialog box.
2. Click MELSECNET/Ethernet . The Ethernet Operations dialog box appears.
3. Enter the parameters in the module listbox.
4. Click Operational Settings .
5. Enter the IP address .
6. Click End.
7. Save and write data to the PLC.
8. Manually reboot the PLC to reset the program memory.
9. Choose Ethernet Diagnostics on the Diagnostics menu to check that IP address is correct. The Ethernet diagnosis dialog box should look like this:



Use the following screenshot of the Mitsubishi GX Developer to help you set up the PLC and Ethernet module. After entering your settings, the dialog boxes should look like this:



## AJ71QE71-B5, A1SJ71QE71-B2 and A1SJ71QE71-B5 Hardware

The B2 has a Coax cable connector for 10Base2 and the B5 has an AUI connector for 10Base5 type connections.

Mitsubishi recommend using a Transceiver with SQTEST or HeartBeat functionality if using the B5 model.

### Indicators on the front of the module

RUN	ON = Normal, OFF = Bad Thing (Normal Operation Display)
RDY	Flashing = On-Line and Good (Standard Display of communication exchange completion)
BSY	ON = Executing, OFF= Not Executing (Display during execution of communication exchange processing)
SW ERR	CPU error, CPU type error, or rotary switch error display ON = Error, OFF= Normal
COM ERR	ON = Communication Exchange Abnormality (Bad Thing)
CPU R/W	ON = Exchanging data with CPU
TRANS.S	ON = Data link command request executing
TRANS.R	ON = Data link RECV command request waiting
FTP	ON = FTP server function operating
B1 - B8	Communication line connection status ON = Open completed, OFF= Closed state
TEST	ON = Self diagnosis executing
TEST.ERR	ON = Error during self diagnosis

### Operation Mode Switch

0	On-line
1	Off-line
2	Execute a self-diagnosis test using a self-loopback test
3	Execute a RAM test
4	Execute a ROM test

5	Execute am EEPROM test
6	F not used

## Communication Exchange Condition Setting Switches

Recommended settings have been marked in blue. See also [MelsecQnA - Hints and Tips](#) for a summary of switch settings.

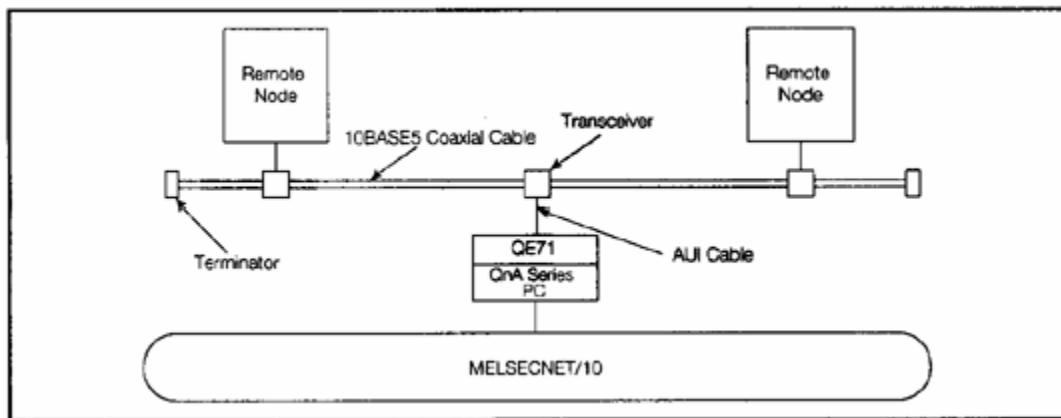
SW1	TCP Timeout error temporary circuit process selection. Selection of Circuit processing at the time of a transmission of a TCP time out error. OFF = Close the circuit ON = Do Not Close the Circuit For Plant SCADA keep it OFF
SW2	Data Code Selection Type of Data Communications supported: OFF= Binary Mode ON = ASCII Mode Mitsubishi has released a Windows version of their programming software that uses ASCII Mode for communicating with the Module. Plant SCADA normally uses Binary Mode, but there is also the ability to use the ASCII Mode.
SW3	Self start mode setting Selects the start mode when the QE71 is booted. OFF= Runs following Y19 (initial processing request signal) ON = Reads the parameters in the EEPROM buffer memory regardless of the Y19 (initial processing request signal) after power has been turned on or the module reset and then conducts initial processing of the contents. This switch should be OFF.
SW4 - SW6	Usage not possible (fixed to OFF).
SW7	CPU communications exchange timing setting Write Protect the PLC. OFF = Writing Prohibited ON = Writing allowed. If you want to write to the PLC then set this to ON.
SW8	Initial Timing Setting Select the Timing which starts Initial Processing. OFF= Quick Start (Start without Delay) when entirely

constructed in a single network.  
OFF = Normal Start (start after a 20 sec delay) when entirely constructed in multiple networks.  
For Plant SCADA, keep it OFF. Select the Timing which starts Initial Processing.  
OFF= Quick Start (Start without Delay) when entirely constructed in a single network.  
OFF = Normal Start (start after a 20 sec delay) when entirely constructed in multiple networks.  
For Plant SCADA, keep it OFF.

### MelsecQnA - Wiring Diagram 1

The following functional/layout diagrams illustrate the connection between Plant SCADA and the Melsec-QnA Ethernet interface module type AJ71QE7. In these configurations, the primary QnA Series PLC itself is connected to a MELSECNET/10 network. The nodes which are marked as remote can be occupied by a Plant SCADA Workstation.

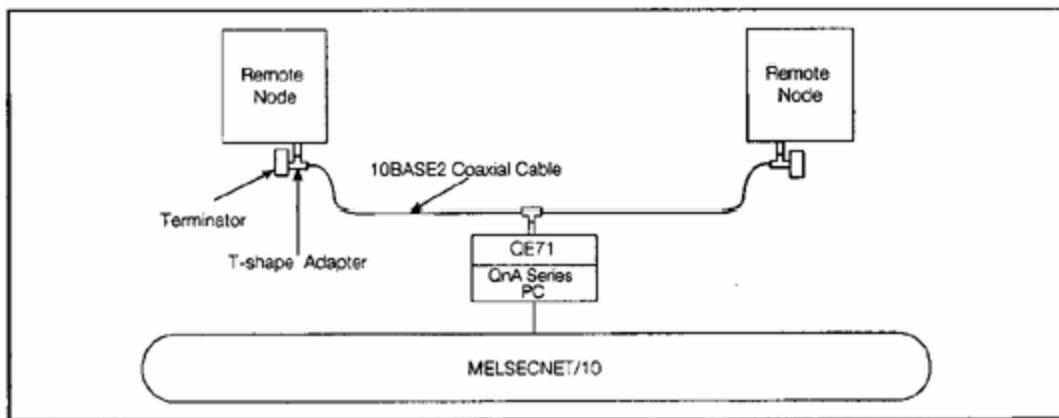
The following diagram shows connection using 10BASE5 Ethernet cabling:



### MelsecQnA - Wiring Diagram 2

The following functional/layout diagrams illustrate the connection between Plant SCADA and the Melsec-QnA Ethernet interface module type AJ71QE7. In these configurations, the primary QnA Series PLC itself is connected to a MELSECNET/10 network. The nodes which are marked as remote can be occupied by a Plant SCADA Workstation.

The following diagram shows connection using 10BASE2 "CheaperNet" cabling:



### MelsecQnA - Hints and Tips

- To initialize the ports, the Melsec-QnA driver will open an Ethernet channel on an AJ71QE71 interface module by using the IP addresses you specify on the Ports Form. You may specify up to 8 TCP/IP ports for one AJ71QE71; this limit is set by the amount of fixed buffers on the Ethernet interface module. Throughput of this driver may be improved by specifying different ports for different remote I/O modules (ie. spread the traffic across several TCP/IP connections, as the driver only supports one pending message per TCP/IP connection). This limit of one pending message per TCP/IP connection finds its roots in the simple request/response protocol used to communicate with PLCs.
- To initialize the I/O Device, Plant SCADA will call CTDRV\_INIT\_UNIT with information from the driver database. This is also the appropriate place to set the data exchange mode (Binary Mode, ASCII Mode) for the IO Device.
- Concerning data exchange, the driver will support both Binary Mode (SW2 on front panel of AJ71QE71 OFF) and ASCII Mode (SW2 on front panel of AJ71QE71 ON).

Normally SW1 and SW8 will be OFF, while SW3 and SW6 will be ON. SW4 - 6 are not used.

SW1 - OFF

SW2 - OFF (ON)

SW3 - OFF

SW7 - ON

SW8 - OFF

Mode = 0

SW1 is TCP Timeout control

SW2 is Communications Type.

Mitsubishi Programming Software MMPlus needs this to be set to ASCII Mode (ON).

Plant SCADA normally uses Binary Mode (OFF), but a parameter allows the selection of ASCII Mode (ON).

SW3 is Self-start enabling/disabling of PLC. OFF = Self-start disabled, ON = Self-start enabled.

SW7 is write protecting the PLC. OFF - Writing prohibited, ON = writing approved.

SW8 is Initial timing setting of PLC. OFF = Quick start, ON = Normal start (start after a delay of 20 seconds).

- If the MELSECQ driver writes a string value to a PLC, it will use a null terminated ASCII value. If a third party driver or application does not correctly read the null character, buffered values may appear as additional junk characters. If this occurs, you may need to use Cicode to pad the values being written to a string tag. An example of how to do this is provided in the Plant SCADA Knowledge Base article "Q4841: MELSECQ driver writes junk characters".

## MelsecQnA - Communication Settings

To establish communication with a device, configure the **Boards**, **Ports** and **I/O Devices** in the Plant SCADA Studio's **Topology** activity. If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure these, use the information outlined below.

### Boards

Field	Value
Board Type	Enter TCPIP.
Address	Enter 0.
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	This field is not used.

### Ports

For details about initializing the port, see [MelsecQ - Hints and Tips](#).

Field	Value
Port Number	Leave this field blank.
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	This field is not used.
Special Options	Enter the AJ71QE71 Ethernet module's TCP/IP address of the controller.  See "I/O Device Address" below for details about this option.

### I/O Devices

For details about initializing the I/O Device, see [MelsecQ - Hints and Tips](#).

Field	Value
I/O Device Address	<p>The CPU number (0-64 or 255). You can specify the CPU monitoring time using the optional parameter "/Tt" after the CPU number. It should be possible to specify the network number (0-239 or 255) by using optional parameter "/Nn" after the CPU number (only useful for MELSECNET/10 PLC networks). Finally you can specify the ASCII Mode for data exchange by using optional parameter /A (the default is binary mode).</p> <p>The address format is:</p> <p><b>a [/Tt] [/Nn] [/A]</b></p> <p>Where:</p> <p><b>a</b> = CPU number 1 to 64 or 255 in decimal</p> <p><b>t</b> = CPU monitoring time parameter (units 250 ms). The default is 10.</p> <p><b>n</b> = Network number 0 to 239 or 255 in decimal. The default network number is 0.</p> <p><b>Note:</b> An in-depth description of the Network and CPU monitoring policy can be found in section 10.1 of the "MELSEC Q series Programmable Controller User's Manual - QnA Ethernet Interface Module type AJ71QE71(B5) - A1SJ71QE71-B2 - A1SJ71QE71-B5". Manual number IB (NA) 66661-C</p>
I/O Device Protocol	Enter MELSECQ.

### MelsecQnA - Data Types

Data Types	Plant SCADA Data Format	Plant SCADA Data Type	Descriptions/ Special Usages/ Limitations/ Valid Ranges
Input relay	X<hexno>	DIGITAL	Read / Write
Output relay	Y<hexno>	DIGITAL	Read / Write
Link input	DX<hexno>	DIGITAL	Read / Write
Link output	DY<hexno>	DIGITAL	Read / Write
Internal relay	M<no>	DIGITAL	Read / Write
Link relay	B<hexno>	DIGITAL	Read / Write
Latch relay	L<no>	DIGITAL	Read / Write

Data Types	Plant SCADA Data Format	Plant SCADA Data Type	Descriptions/ Special Usages/ Limitations/ Valid Ranges
Annunciator	F<no>	DIGITAL	Read / Write
Edge relay	V<no>	DIGITAL	Read / Write
Step relay	S<no>	DIGITAL	Read / Write
Special relay	SM<no>	DIGITAL	Read / Write
Special relay for link	SB<hexno>	DIGITAL	Read / Write
Timer (Contact)	TS<no>	DIGITAL	Read / Write
Timer (Coil)	TC<no>	DIGITAL	Read / Write
Integrating Timer (Contact)	SS<no>	DIGITAL	Read / Write
Intergating Timer (Coil)	SC<no>	DIGITAL	Read / Write
Counter (Contact)	CS<no>	DIGITAL	Read / Write
Counter (Coil)	CC<no>	DIGITAL	Read / Write
Timer ( Current Value)	TN<no>	DIGITAL, INT, LONG, STRING, REAL	Read / Write
Integrating Timer (Current Value)	SN<no>	DIGITAL, INT, LONG, STRING, REAL	Read / Write
Counter (Current value)	CN<no>	DIGITAL, INT, LONG, STRING, REAL	Read / Write
Data register	D<no>	DIGITAL, INT, LONG, STRING, REAL	Read / Write
Link register	W<hexno>	DIGITAL, INT, LONG, STRING, REAL	Read / Write
Special register for link	SW<hexno>	DIGITAL, INT, LONG, STRING, REAL	Read / Write
Special register	SD<no>	DIGITAL, INT, LONG, STRING, REAL	Read / Write
File Register (Block Access)	R<no>	DIGITAL, INT, LONG, STRING, REAL	Read / Write
File Register	ZR<hexno>	DIGITAL, INT, LONG,	Read / Write

Data Types	Plant SCADA Data Format	Plant SCADA Data Type	Descriptions/ Special Usages/ Limitations/ Valid Ranges
(Seq. Access)		STRING, REAL	
Index register	Z<no>	DIGITAL, INT, LONG, STRING, REAL	Read / Write

Where:

<no>	decimal number
<hexno>	hexadecimal number

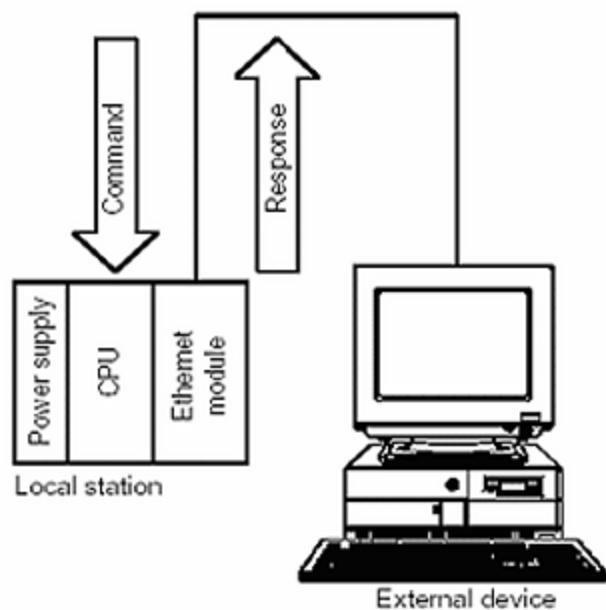
DX & DY are DIRECT read and writes of the I/O values. X & Y are read / written at the end of the scan. Thus the X / DX , Y / DY areas are one of the same.

## EXAMPLES:

Data Type	INTEGER
Address	V77
Comment	Stator current
Data Type	DIGITAL
Address	X107
Comment	Running feedback indicator

## Melsec iQ-R via TCP/IP

The MELSECQ Driver supports TCP/IP communication with the Melsec iQ-R, manufactured by Mitsubishi Electric. The following diagram illustrates how the Melsec iQ-R module communicates with the external device.



### Melsec iQ-R - Device Address

The information you enter here will be placed in the Special Options field of the Ports settings, with the following format:

`-la.b.c.d -Pn -T`

where:

- *a.b.c.d* - is the destination IP address using decimal numbers (for example 192.9.2.60).
- *n* - the destination Port number. Often one physical port has several virtual ports, used for different purposes.
- `-T` - forces the driver to use TCP, rather than UDP (`-U`).

See the topic [Melsec iQ-R via TCP/IP](#) for setup details.

### Melsec iQ-R - Communications Settings

To establish communication with a device, configure the **Boards**, **Ports** and **I/O Devices** settings in the Plant SCADA Studio's Topology activity. If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure these, use the information outlined below.

## Boards

Field	Value
Board Type	Enter TCPIP.
Address	Enter 0.

Field	Value
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	This field is not used.

## Ports

Field	Value
Port Number	Leave this field blank.
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	This field is not used.
Special Options	Enter the AJ71QE71 Ethernet module's TCP/IP address of the controller.  See "I/O Device Address" below for details about this option.

## I/O Devices

Field	Value
I/O Device Address	The CPU number (0-64 or 255). You can specify the CPU monitoring time using the optional parameter "/Tt" after the CPU number. It should be possible to specify the network number (0-239 or 255) by using optional parameter "/Nn" after the CPU number (only useful for MELSECNET/10 PLC networks). Finally you can specify the ASCII Mode for data exchange by using optional parameter /A (the default is binary mode).  The address format is: <b>a [/Tt] [/Nn] [/Pp OR /Qq OR /Ss] [/A]</b> Where: <b>a</b> = CPU number 0 to 64 or 255 in decimal

Field	Value
	<p><b>t</b> = CPU monitoring time parameter (units 250 ms). The default is 10.</p> <p><b>n</b> = Network number 0 to 239 or 255 in decimal. The default network number is 0.</p> <p><b>p</b> = PLC number, default is 0 for control PLC</p> <p><b>s</b> = System CPU</p> <ul style="list-style-type: none"> <li>1 - Control System CPU</li> <li>2 - Standby System CPU</li> <li>3 - System A CPU</li> <li>4 - System B CPU</li> </ul> <p><b>q</b> = External device ID for multidrop Q mode C24 devices.</p> <p>/P, /Q and /S parameters are mutually exclusive. The precedence of them is /P &gt; /Q &gt; /S.</p> <p><b>Note:</b> An in-depth description of the Network and CPU monitoring policy can be found in section 10.1 of the "MELSEC Q series Programmable Controller User's Manual - QnA Ethernet Interface Module type AJ71QE71(B5) - A1SJ71QE71-B2 - A1SJ71QE71-B5". Manual number IB (NA) 66661-C</p>
I/O Device Protocol	Enter MELSECQ2.

### Melsec iQ-R - Data Types

Data Types	Plant SCADA Data Format	Plant SCADA Data Type	Descriptions/ Special Usages/ Limitations/ Valid Ranges
Input relay	X<hexno>	DIGITAL	Read / Write
Output relay	Y<hexno>	DIGITAL	Read / Write
Link input	DX<hexno>	DIGITAL	Read / Write
Link output	DY<hexno>	DIGITAL	Read / Write
Internal relay	M<no>	DIGITAL	Read / Write
Link relay	B<hexno>	DIGITAL	Read / Write

Data Types	Plant SCADA Data Format	Plant SCADA Data Type	Descriptions/ Special Usages/ Limitations/ Valid Ranges
Latch relay	L<no>	DIGITAL	Read / Write
Announcer	F<no>	DIGITAL	Read / Write
Edge relay	V<no>	DIGITAL	Read / Write
Step relay	S<no>	DIGITAL	Read / Write
Special relay	SM<no>	DIGITAL	Read / Write
Special relay for link	SB<hexno>	DIGITAL	Read / Write
Timer (Contact)	TS<no>	DIGITAL	Read / Write
Timer (Coil)	TC<no>	DIGITAL	Read / Write
Counter (Contact)	CS<no>	DIGITAL	Read / Write
Counter (Coil)	CC<no>	DIGITAL	Read / Write
Timer (Current Value)	TN<no>	INT, LONG, STRING, REAL, BCD, LONGBCD	Read / Write
Counter (Current value)	CN<no>	INT, LONG, STRING, REAL, BCD, LONGBCD	Read / Write
Data register	D<no>	INT, LONG, STRING, REAL, BCD, LONGBCD	Read / Write
Link register	W<hexno>	INT, LONG, STRING, REAL, BCD, LONGBCD	Read / Write
Special register for link	SW<hexno>	INT, LONG, STRING, REAL, BCD, LONGBCD	Read / Write
Special register	SD<no>	INT, LONG, STRING, REAL, BCD, LONGBCD	Read / Write
File Register (Block Access)	R<no>	INT, LONG, STRING, REAL, BCD, LONGBCD	Read / Write
File Register (Seq. Access)	ZR<hexno>	INT, LONG, STRING, REAL, BCD, LONGBCD	Read / Write
Index register	Z<no>	INT, LONG, STRING, REAL, BCD, LONGBCD	Read / Write
Counter contact	LCS<no>	DIGITAL	Read / Write

Data Types	Plant SCADA Data Format	Plant SCADA Data Type	Descriptions/ Special Usages/ Limitations/ Valid Ranges
Counter coil	LCC<no>	DIGITAL	Read / Write
Integrating timer Contact	STS<no>	DIGITAL	Read / Write
Integrating timer coil	STC<no>	DIGITAL	Read / Write
Counter current value	LCN<no>	LONG	Read / Write
Timer current value	LTN<no>	LONG	Read only
Retentive timer current value	LSTN<no>	LONG	Read only
Index register	LZ<no>	LONG	Read / Write
Data register	RD<no>	INT	Read / Write
Integrating timer Current value	STN<no>	INT	Read / Write

Where:

<no>	decimal number
<hexno>	hexadecimal number

DX & DY are DIRECT read and writes of the I/O values. X & Y are read / written at the end of the scan. Thus the X / DX , Y / DY areas are one of the same.

## EXAMPLES:

Data Type	INTEGER
Address	V77
Comment	Stator current
Data Type	DIGITAL
Address	X107
Comment	Running feedback indicator

## Customize a Project using Citect.ini Parameters

The MELSECQ driver supports the following parameter categories:

- MELSECQ Driver Parameters
- Logging Parameters.

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

- Do not under any circumstances change or remove any citect.ini parameters without an informed understanding of the possible implications of your actions.
- Do not delete sections of the citect.ini file without an informed understanding of the possible implications of your actions.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** At times during the life cycle of a driver, circumstances may call for the creation of specific citect.ini parameters that remain undocumented. Always seek the advice of Technical Support regarding the possible implementation of undocumented parameters before you make any changes to a citect.ini file.

## **MELSECQ Driver Parameters**

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

MELSECQ Parameter	Description	Allowable Values	Default Value
Block	A value (bytes) used by the I/O server to determine if two or more packets can be blocked into one data request before being sent to the I/O device.  For example, if you set the value to 10, and the I/O server receives two simultaneous data requests - one for byte 3, and another for byte 8 - the two requests will be blocked into a single	5 to 256	256

	physical data request packet.  This single request packet is then sent to the I/O Device, saving on bandwidth and processing.		
Delay	The period (in milliseconds) to wait between receiving a response and sending the next command.	0 to 300 (milliseconds)	0
MaxPending	The maximum number of pending commands that the driver holds ready for immediate execution.	1 to 32	1
PollTime	The interrupt or polling service time (in milliseconds). Setting the polling time to 0 puts the driver in interrupt mode.	0 to 300 (milliseconds)	0
Retry	The number of times to retry a command after a timeout.	0 to 8	2
Timeout Allowable Values: Default Value:	Specifies how many milliseconds to wait for a response before displaying an error message.	0 to 32 000 (milliseconds)	10000 milliseconds (Mitsubishi recommends this value)
WatchTime	The frequency (in seconds) that the driver uses to check the communications link to the I/O Device.	0 to 128 (seconds)	30 seconds <b>Note:</b> This value should be increased to lower CPU load. Mitsubishi recommends a default value of 3.

## See Also

[Logging Parameters](#)

## Logging Parameters

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

MELSECQ Parameter	Description	Allowable Values	Default Value
DebugCategory	Controls which categories of log messages are written to the log file.	ALL or any combination of the following values separated by a pipe character (   ):  PROT UNIT MSG ENTRY DCB MISC  See <a href="#">Logging</a> for examples.	Empty string (no logging).
DebugLevel	Controls which levels of log message are written to the log file.	ALL or any combination of the following values separated by a pipe character (   ):  WARN ERROR TRACE DEBUG  See <a href="#">Logging</a> for examples.	Empty string (no logging).

**See Also**

[MELSECQ Driver Parameters](#)

## MELSECQ Driver Specific Errors

When a hardware error occurs, Plant SCADA generates an alarm, and displays the alarm on the hardware alarm page (in the alarm description of the hardware alarm). To see the error number, you need to have Alarm Category 255 defined with a display format that includes {ErrDesc,15} {ErrPage,10} .

There are a number of generic errors that are common to many protocols. In some cases, only the generic error is available, though often both the generic error and a specific error are given.

The following errors, listed in (hexadecimal) sequence, are specific to this protocol. Plant SCADA displays the error number and description for common protocol-specific errors. Uncommon errors are not contained in the

Plant SCADA error database, in which case Plant SCADA will only display the error number.

You may require additional information to enable you to rectify an error. This information should be detailed in the documentation that accompanied the I/O Device (or network). If, after reviewing the documentation, you cannot rectify an error, contact Technical Support for this product.

## 10000H + CODE

Code is the error code. Consult the QnA Ethernet Interface Module User's Manual for details on the error code.

## Logging

The MELSECQ driver can log combinations of trace levels across different categories. This is achieved by setting the following INI parameters.

### [MELSECQ]DebugLevel

This parameter allows you to define the trace level. The options include:

Option	Description
WARN	Output warning messages.
ERROR	Output error messages.
TRACE	Output trace messages.
DEBUG	Output debug messages.
ALL	All of the above.

### [MELSECQ]DebugCategory

This parameter allows you to enable logging for a particular category of trace. The options include:

Option	Description
PROT	Trace of communication between the driver and the lower level protocol.
UNIT	I/O device trace.
ENTRY	Driver entry trace.
DCB	Front end driver trace.
MISC	Any other debugging.
ALL	All of the above.

In both cases, you can use any combination of the available options separated by a pipe character ( | ).

#### Example

```
[MELSECQ]
DebugLevel=WARN|ERROR|TRACE
DebugCategory=ALL
```

In most cases, you can set the DebugLevel and DebugCategory parameters to ALL. If this results in a large, cumbersome log file that causes older data to be overwritten, you can consider refining the debug options.

**Note:** These parameters do not have a value specified by default. To enable logging for this driver, you need to manually configure these parameters in your INI file.

The events generated by the MELSECQ driver are logged in the following Plant SCADA syslog file:

```
syslog.IOServer.<Cluster name>.<IO Server name>.dat
```

This file is located in the directory specified by the Citect.ini parameter **[CtEdit]Logs**. By default, this location will be:

```
%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Logs
```

The size of the syslog file and its rollover process is configured via the **[Debug]SysLogSize** and **[Debug]SysLogArchive** INI parameters. For more information, see the topic *Debug Parameters* in the Plant SCADA documentation.

**Note:** If the number of messages being processed for a driver exceeds the value specified by the Plant SCADA parameter **[Kernel]ErrorBuffers**, the pending messages will not be logged. If this occurs, you can increase the value of this parameter so that all the messages generated by the driver are reported. The number of messages that are not logged can be monitored via the Kernel's General window under the item "Lost Errors". If you adjust **[Kernel]ErrorBuffers** and Lost Errors are still occurring, you may need to adjust the level of system logging.

## See Also

[Logging Parameters](#)

## MODBUS Driver

The MODBUS Driver supports serial communication with many I/O devices (including the Modicon 484, 584, 884, and 984 programmable controllers, and Moore Products IOExpress).

**Note:** The **MODBUS** and **MODBUSA** drivers are closely related. They only differ from each other in the data framing technique used: MODBUS is RTU-based, while MODBUSA is ASCII-based. You should determine which your system requires and make sure you proceed using the appropriate driver.

The maximum request length for the MODBUS driver is 2000 bits.

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.
<b> DANGER</b>	
<b>DANGER</b> indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.	
<b> WARNING</b>	
<b>WARNING</b> indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.	
<b> CAUTION</b>	
<b>CAUTION</b> indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.	
<b>NOTICE</b>	
<b>NOTICE</b> used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.	

## Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

## Before You Begin

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ WARNING****LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

## Supported Devices

The MODBUS Driver supports serial communication with the following devices:

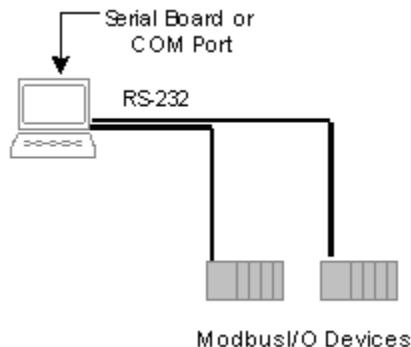
- Generic devices
- Modicon 484 PLCs
- Modicon 884 PLCs
- Modicon 984 PLCs
- Moore Products IOExpress Data Collection Units.

Also see [Working with non-standard devices](#).

## MODBUS Generic Devices

Being an established industry standard, the Modbus protocol supports many I/O devices from a variety of manufacturers. The MODBUS Driver therefore can support communication between Plant SCADA and any generic Modbus devices.

Communication is via a Modbus port on the PLC. Using this method, you can connect to multiple PLCs as in the following diagram:



The following topics provide information required to connect Plant SCADA to a generic Modbus device.

### MODBUS Generic Devices - Device Address

The I/O Device Address for a PLC is its station number. Try 1 to 10.

- **1** specifies station number **1**.
- **9** specifies station number **9**.

### MODBUS Generic Devices - Hardware Setup

Set your PLC to the communication setting you require. The default hardware settings for the device are as follows:

Setting	Value
Baud Rate	19200
Data Bits	8
Stop Bits	1
Parity	2 (Even)

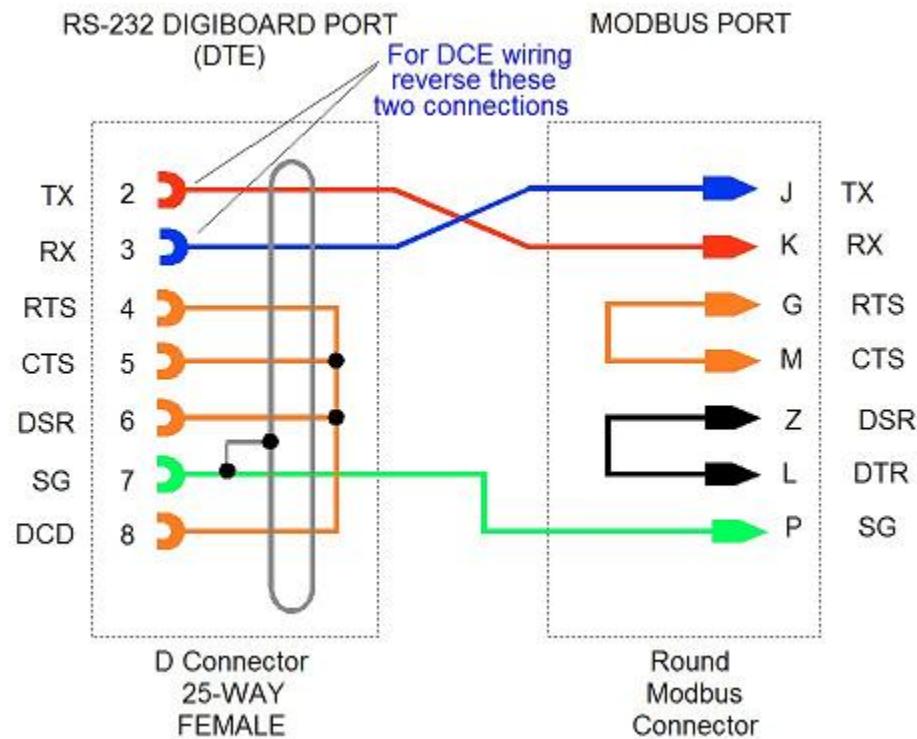
**Note:** These settings are recommendations only. If you use the Communications Express Wizard, these default settings are configured in your project automatically, though your hardware may support other values. If you do set the baud rate, data bits, stop bits or parity to another value, you must manually set the new value(s) in your project.

Set the following configuration options on the Modbus port on the PLC:

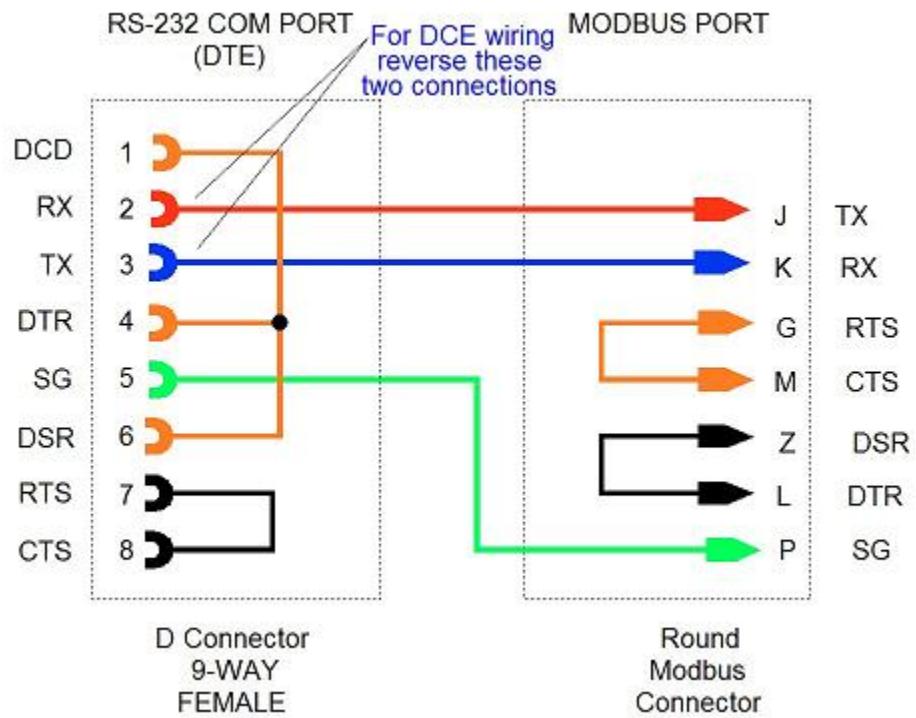
- Station number

- RTU mode (This is not selected if the MODBUSA protocol is being used)
- Delay time to 1

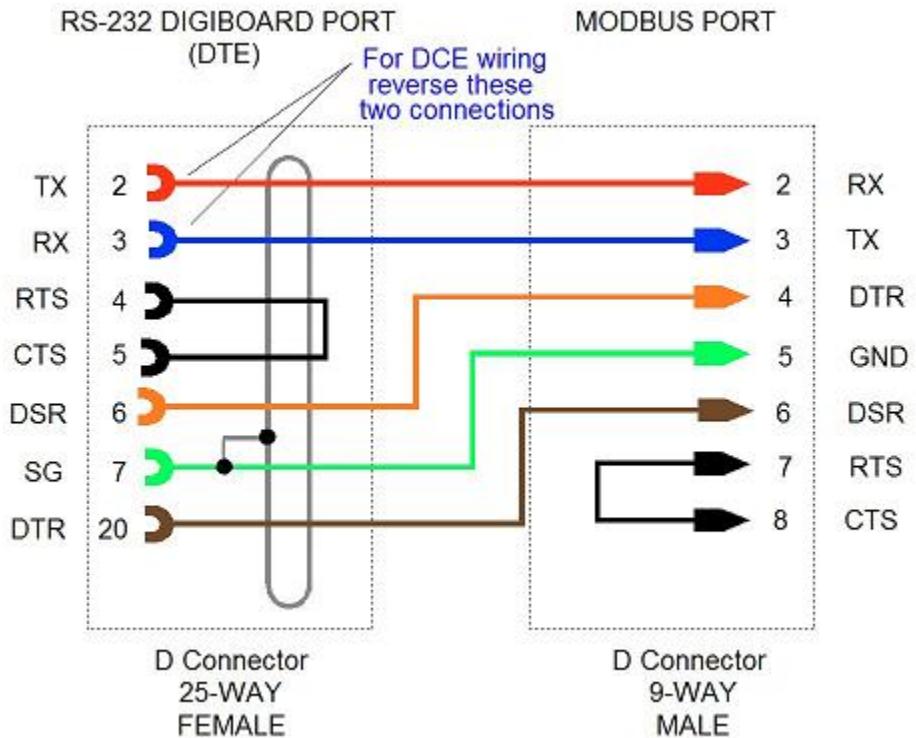
### MODBUS Generic Devices - Wiring Diagram 1



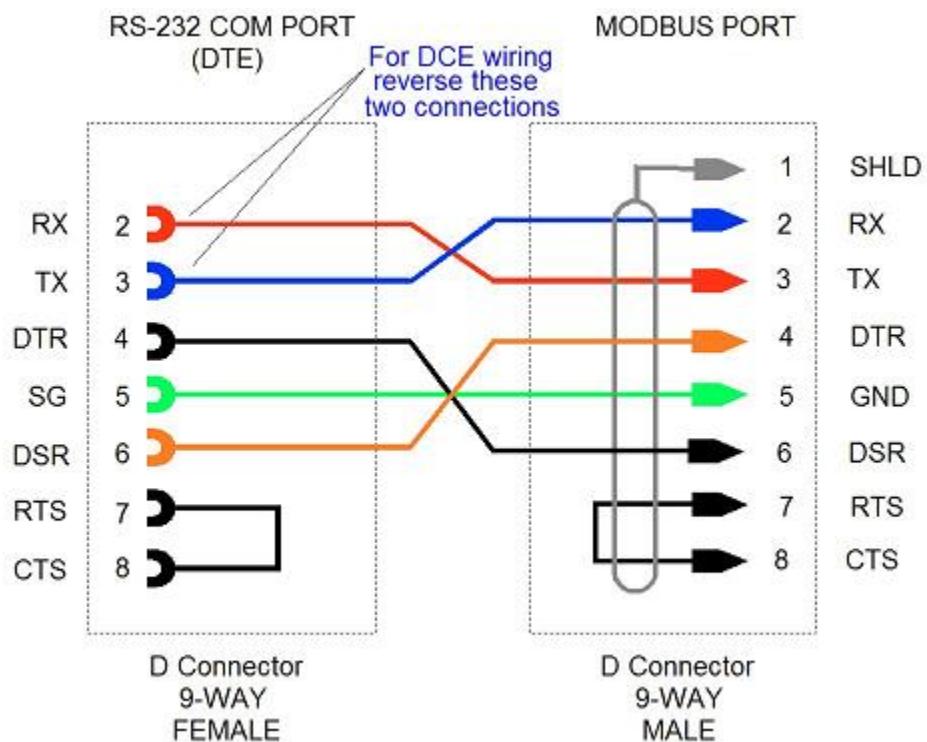
## MODBUS Generic Devices - Wiring Diagram 2



## MODBUS Generic Devices - Wiring Diagram 3



## MODBUS Generic Devices - Wiring Diagram 4



## MODBUS Generic Devices - Communication Settings

To establish communication with a device, configure the Boards, Port and I/O Device in Plant SCADA Studio's **Topology** activity. If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure these, use the settings outlined below.

### Board Settings

Typically, you would use a serial board or COM port for this communication. Refer to the instructions for setting up and using COM ports or serial boards. Or complete the Board settings with the following specific information.

Field	Value
Board Type	If using a serial board or COM port, enter COMx.
Address	If using a serial board or COM port, enter 0.
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.

## Port Settings

Complete the Port settings with the following specific information.

Field	Value
Port Number	This value matches the COM port number. This number is defined in the Ports section of the Control Panel.
Baud Rate	This value matches the setting of the unit - 19200.
Data Bits	Enter 8.
Stop Bits	Enter 1.
Parity	This value matches the setting of the unit - EVEN_P.
Special Options	You may want to use the special options for the COMx Driver if you are using a modem (or similar) and want the driver to perform differently. For more information, see the topic <i>COMx Driver Special Options Reference</i> in the <i>Communicating with I/O Devices</i> chapter of the main help.

## I/O Device Settings

Complete the I/O Device settings with the following specific information.

Field	Value
I/O Device Address	The PLC station number configured on the PLC. Try 1 to 10. 1 specifies station number 1. 9 specifies station number 9 .
I/O Device Protocol	Enter MODBUS if you want to use the Modbus RTU (binary) protocol. Enter MODBUSA if you want to use the Modbus ASCII protocol. (See the topic <a href="#">Protocol Variants</a> to determine if a non-standard protocol variant is required.)

## MODBUS Generic Devices - Data Types

---

**Note:** The address ranges specified in the following table will be appropriate for most generic Modbus devices. However, you should check the capabilities of a device to confirm if it will support the full extent of the ranges suggested.

---

Data Types	Address Format	Plant SCADA Data Type
<b>DIGITAL</b>		
Output Coils	000001 to 065536	DIGITAL ( <b>Note:</b> Leading zero is required.)
Input Status	100001 to 165536	DIGITAL (Read Only)
<b>INTEGER</b>		
Input Registers	300001 to 365536	INT / LONG / STRING / REAL (Read Only)
Output Registers	400001 to 465536	INT / LONG / STRING / REAL
<b>EXTENDED REGISTERS</b>		
(6xxxxx range)	Ef:r.b <i>where:</i> f = file no. (1 - 999) r = register no. (0 - 9999) b = bit no. (1 - 16)	DIGITAL / INT / LONG / STRING / REAL

**Examples**

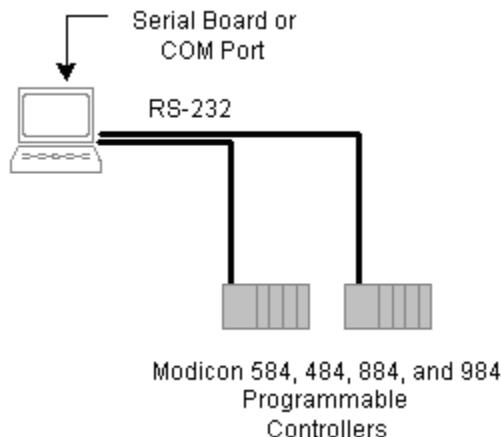
Data Type	DIGITAL
Address	00001
Comment	Output Coil 00001
Data Type	INT
Address	40001
Comment	Output Register 40001
Data Type	INT
Address	E1:00001
Comment	Modbus Address 600001

**Note:** LONG data types default to a simplified implementation, with a shortened range of 0 to 655,359,999. To increase the LONG data type to the full range, set the LongDataType parameter to either 1 or 3.

**Modicon 484 PLCs**

The serial method of communication to Modicon 484 PLCs, uses either the MODBUS or MODBUSA protocol. Communication is via the Modbus port on the PLC. Using this method, you can connect to single or multiple PLCs

as in the following diagram:



### Modicon 484 PLCs - Device Address

The I/O Device Address for a 484 PLC is PLC station number. Try 1 to 10.

- **1** specifies station number **1**.
- **9** specifies station number **9**.

### Modicon 484 PLCs - Hardware Setup

Set your PLC to the communication settings you require. The default hardware settings for the 484 are as follows:

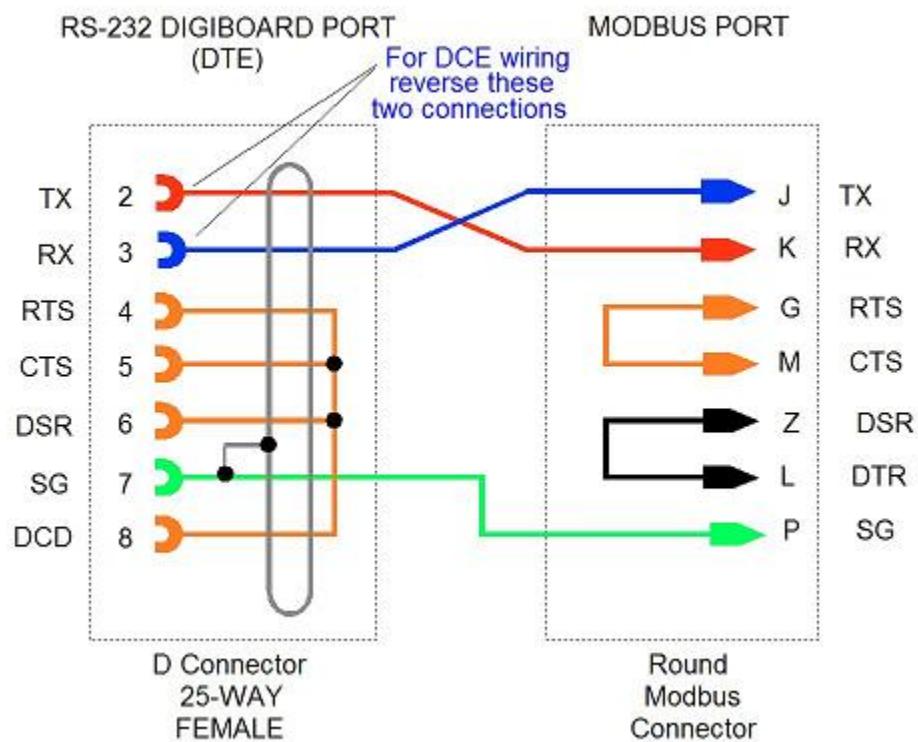
Setting	Value
Baud Rate	19200
Data Bits	8
Stop Bits	1
Parity	2 (Even)

**Note:** These settings are recommendations only. If you use the Communications Express Wizard, these default settings are configured in your project automatically, though your hardware may support other values. If you do set the baud rate, data bits, stop bits or parity to another value, you must manually set the new value(s) in your project.

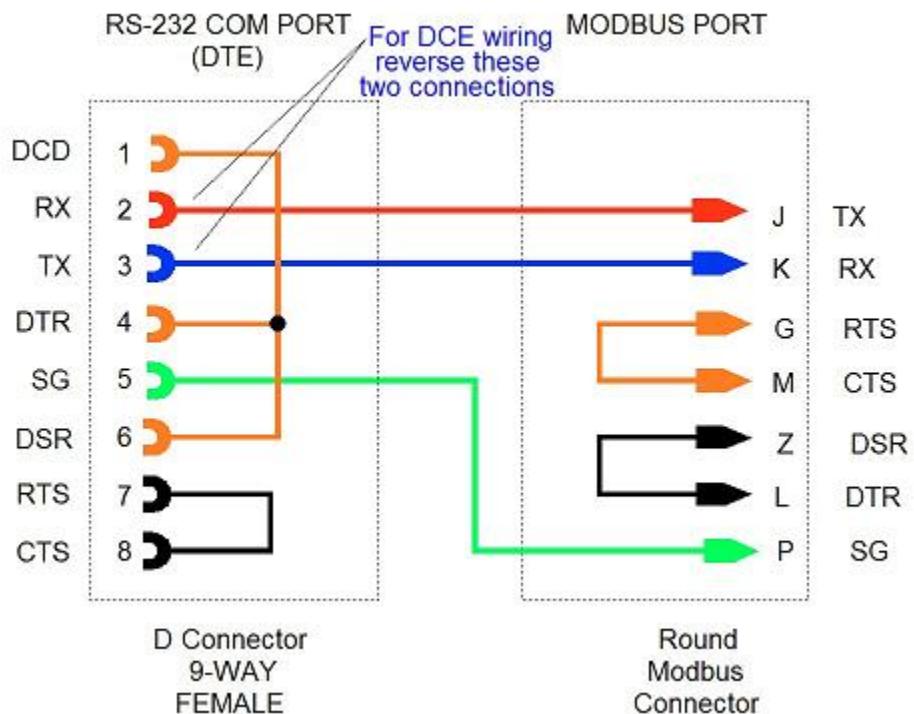
Set the following configuration options on the Modbus port on the Modicon PLC:

- Station number
- RTU mode (If this is not selected then the MODBUSA protocol is used)
- Delay time to 1

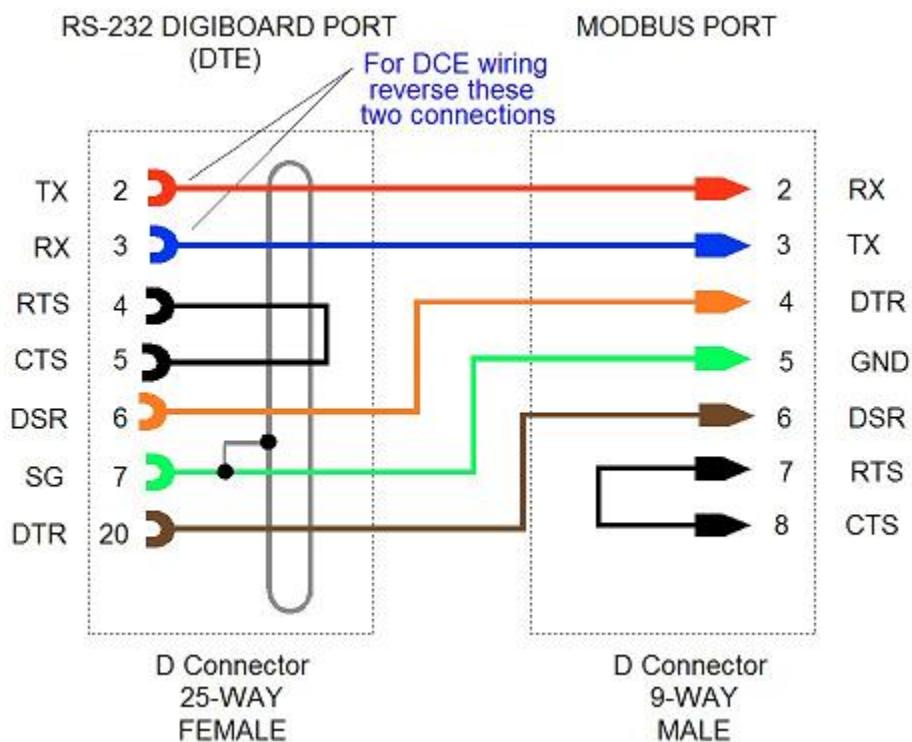
## Modicon 484 PLCs - Wiring Diagram 1



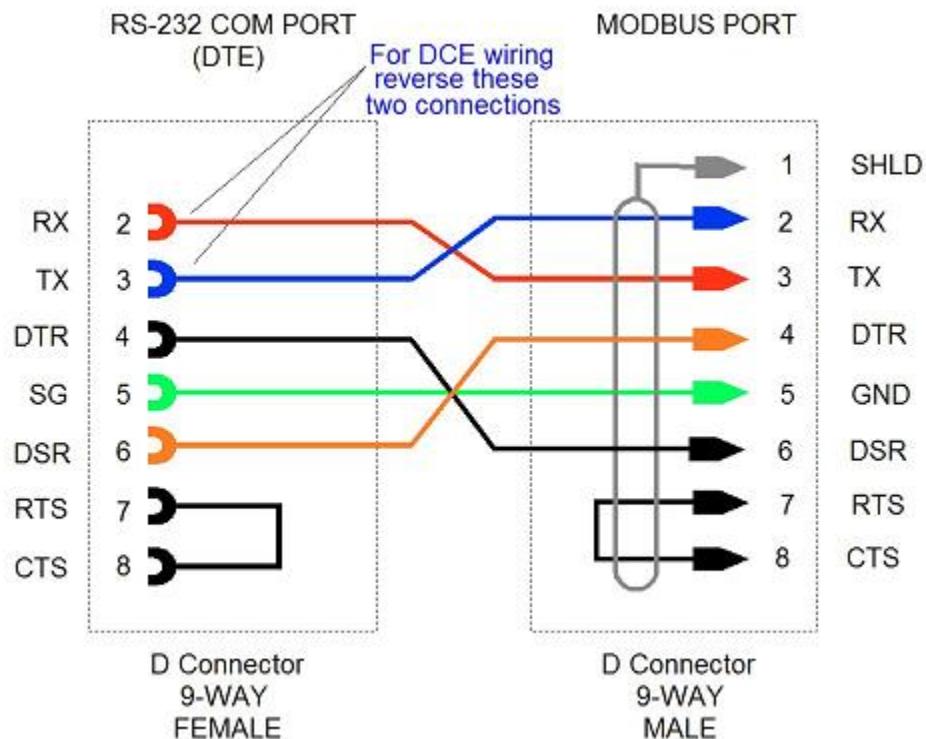
## Modicon 484 PLCs - Wiring Diagram 2



## Modicon 484 PLCs - Wiring Diagram 3



## Modicon 484 PLCs - Wiring Diagram 4



## Modicon 484 PLCs - Communication Settings

To establish communication with a device, configure the Boards, Port and I/O Device in Plant SCADA Studio's **Topology** activity. If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure these, use the settings outlined below.

### Board Settings

Typically, you would use a serial board or COM port for this communication. Refer to the instructions for setting up and using COM ports or serial boards. Or complete the Board settings with the following specific information.

Field	Value
Board Type	If using a serial board or COM port, enter COMx.
Address	If using a serial board or COM port, enter 0.
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.

### Port Settings

Complete the Port settings with the following specific information.

Field	Value
Port Number	This value matches the COM port number. This number is defined in the Ports section of the Control Panel.
Baud Rate	This value matches the setting of the unit - 19200.
Data Bits	Enter 8.
Stop Bits	Enter 1.
Parity	This value matches the setting of the unit - EVEN_P.
Special Options	You may want to use the special options for the COMx Driver if you are using a modem (or similar) and want the driver to perform differently.  For more information, see the topic <i>COMx Driver Special Options Reference</i> in the <i>Communicating with I/O Devices</i> chapter of the main help.

## I/O Device Settings

Complete the I/O Device settings with the following specific information.

Field	Value
I/O Device Address	The PLC station number configured on the PLC. Try 1 to 10. 1 specifies station number 1. 9 specifies station number 9.
I/O Device Protocol	Enter MODBUS if you want to use the Modbus RTU (binary) protocol. Enter MODBUSA if you want to use the Modbus ASCII protocol. (See the topic <a href="#">Protocol Variants</a> to determine if a non-standard protocol variant is required.)

## Modicon 484 PLCs - Data Types

Data Types	Address Format	Plant SCADA Data Type
DIGITAL		
Output Coils	000001 to 065536	DIGITAL ( <b>Note:</b> Leading zero is required.)
Input Status	100001 to 165536	DIGITAL (Read Only)
INTEGER		
Input Registers	300001 to 365536	INT / LONG / STRING / REAL (Read Only)
Output Registers	400001 to 465536	INT / LONG / STRING / REAL

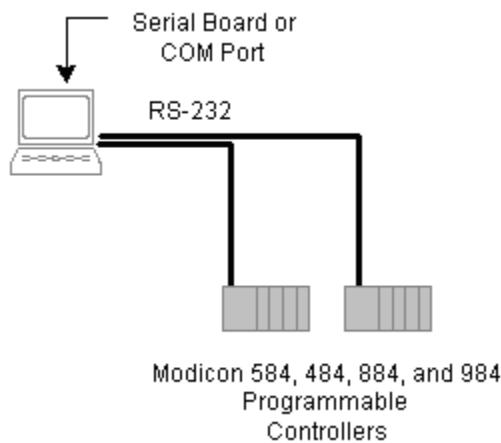
### Examples

Data Type	DIGITAL
Address	00001
Comment	Output Coil 00001
Data Type	INT
Address	40001
Comment	Output Register 40001

**Note:** In the MODBUS protocol, LONG data types default to a simplified implementation, with a shortened range of 0 to 655,359,999. To increase the LONG data type to the full range, set the [MODBUS]LongDataType parameter to either 1 or 3.

## Modicon 584 PLCs

The serial method of communication to Modicon 584 PLCs, uses either the MODBUS or MODBUSUSA protocol. Communication is via the Modbus port on the PLC. Using this method, you can connect to single PLCs or to multiple PLCs as in the following diagram:



## Modicon 584 PLCs - Device Address

The I/O Device Address for a 584 PLC is PLC station number. Try 1 to 10.

- **1** specifies station number **1**.
- **9** specifies station number **9**.

## Modicon 584 PLCs - Hardware Setup

Set your PLC to the communication settings you require. The default hardware settings for the 584 are as follows:

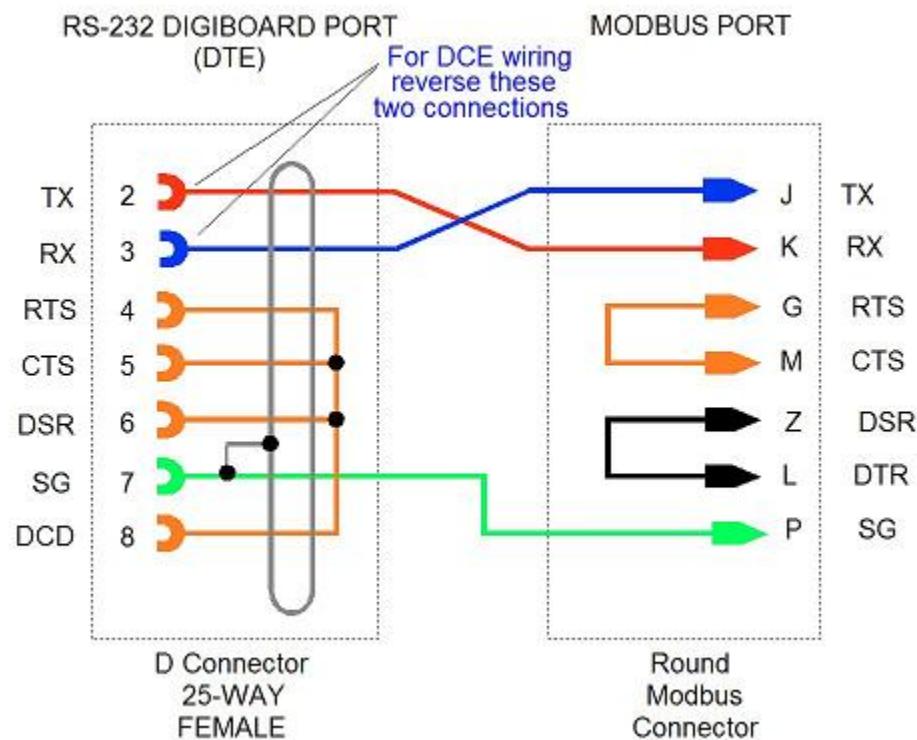
Setting	Value
Baud Rate	19200
Data Bits	8
Stop Bits	1
Parity	2 (Even)

**Note:** These settings are recommendations only. If you use the Communications Express Wizard, these default settings are configured in your project automatically, though your hardware may support other values. If you do set the baud rate, data bits, stop bits or parity to another value, you must manually set the new value(s) in your project.

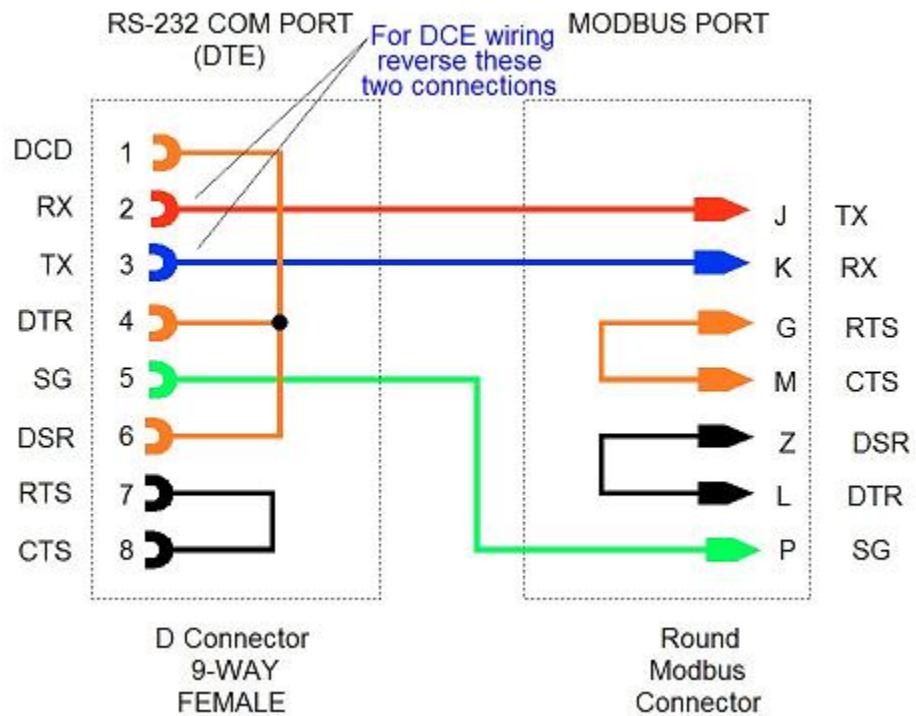
Set the following configuration options on the Modbus port on the Modicon PLC:

- Station number
- RTU mode (If this is not selected then the MODBUSA protocol is used)
- Delay time to 1

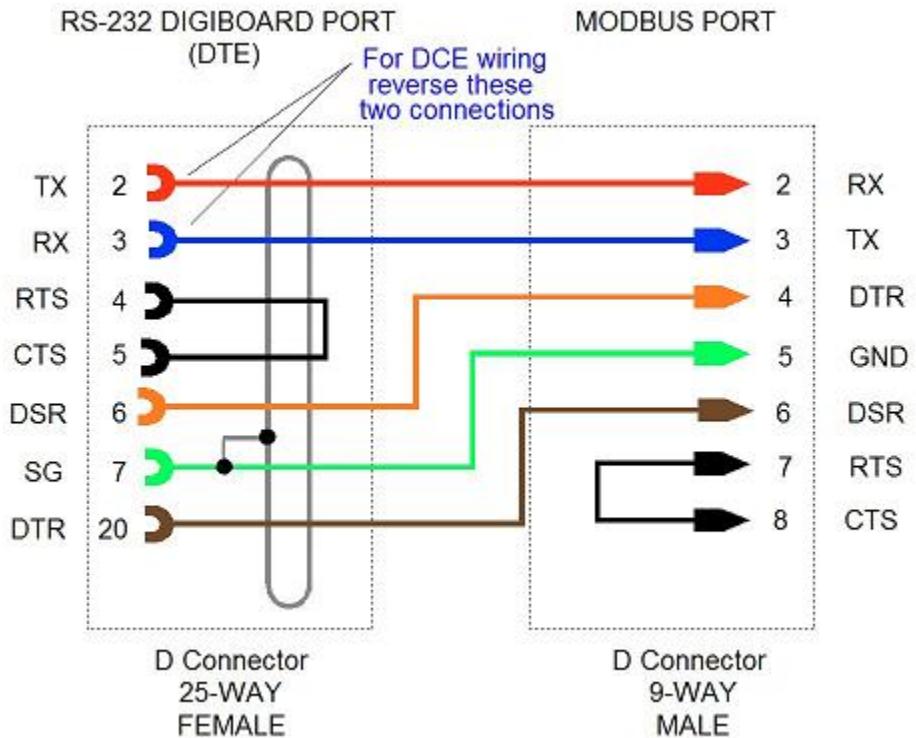
### Modicon 584 PLCs - Wiring Diagram 1



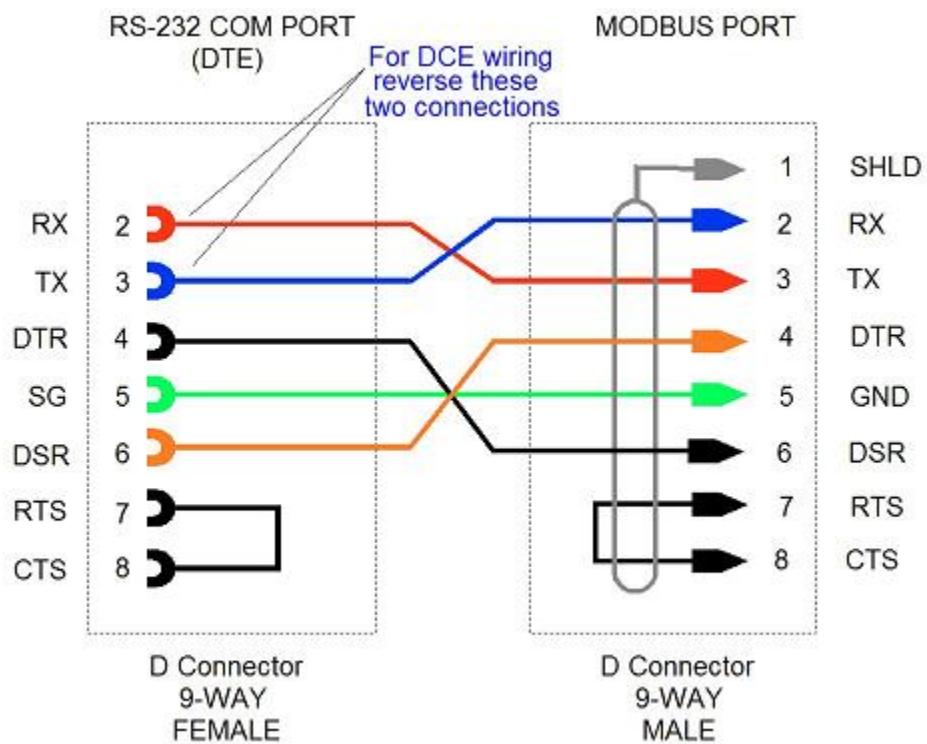
## Modicon 584 PLCs - Wiring Diagram 2



## Modicon 584 PLCs - Wiring Diagram 3



## Modicon 584 PLCs - Wiring Diagram 4



## Modicon 584 PLCs - Communication Settings

To establish communication with a device, configure the Boards, Port and I/O Device in Plant SCADA Studio's **Topology** activity. If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure these, use the settings outlined below.

### Board Settings

Typically, you would use a serial board or COM port for this communication. Refer to the instructions for setting up and using COM ports or serial boards. Or complete the Board settings with the following specific information.

Field	Value
Board Type	If using a serial board or COM port, enter COMx.
Address	If using a serial board or COM port, enter 0.
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.

## Port Settings

Complete the Port settings with the following specific information.

Field	Value
Port Number	This value matches the COM port number. This number is defined in the Ports section of the Control Panel.
Baud Rate	This value matches the setting of the unit - 19200.
Data Bits	Enter 8.
Stop Bits	Enter 1.
Parity	This value matches the setting of the unit - EVEN_P.
Special Options	You may want to use the special options for the COMx Driver if you are using a modem (or similar) and want the driver to perform differently. For more information, see the topic <i>COMx Driver Special Options Reference</i> in the <i>Communicating with I/O Devices</i> chapter of the main help.

## I/O Device Settings

Complete the I/O Device settings with the following specific information.

Field	Value
I/O Device Address	The PLC station number configured on the PLC. Try 1 to 10. 1 specifies station number 1. 9 specifies station number 9.
I/O Device Protocol	Enter MODBUS if you want to use the Modbus RTU (binary) protocol. Enter MODBUSA if you want to use the Modbus ASCII protocol. (See the topic <a href="#">Protocol Variants</a> to determine if a non-standard protocol variant is required.)

## Modicon 584 PLCs - Data Types

Data Types	Address Format	Plant SCADA Data Type
<b>DIGITAL</b>		
Output Coils	000001 to 065536	DIGITAL ( <b>Note:</b> Leading zero is required.)
Input Status	100001 to 165536	DIGITAL (Read Only)
<b>INTEGER</b>		
Input Registers	300001 to 365536	INT / LONG / STRING / REAL (Read Only)
Output Registers	400001 to 465536	INT / LONG / STRING / REAL

**Examples**

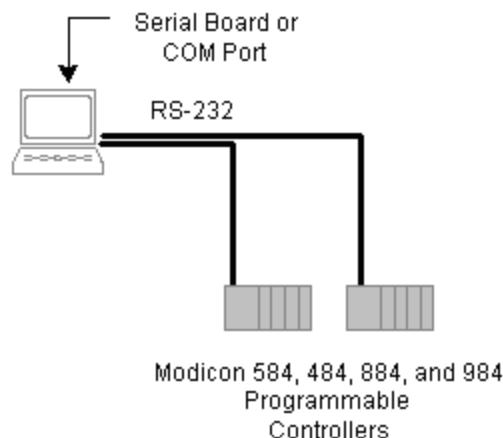
Data Type	DIGITAL
Address	00001
Comment	Output Coil 00001
Data Type	INT
Address	40001
Comment	Output Register 40001

**Note:**

- 1) Modicon 584 PLCs support remapping of both reads and writes.
- 2) In the MODBUS protocol, LONG data types default to a simplified implementation, with a shortened range of 0 to 655,359,999. To increase the LONG data type to the full range, set the [MODBUS]LongDataType parameter to either 1 or 3.

**Modicon 884 PLCs**

The serial method of communication to Modicon 884 PLCs, uses either the MODBUS or MODBUSA protocol. Communication is via the Modbus port on the PLC. Using this method you can connect to single or multiple PLCs as in the following diagram:



### Modicon 884 PLCs - Device Address

The I/O Device Address for a 884 PLC is PLC station number. Try 1 to 10.

- **1** specifies station number **1**.
- **9** specifies station number **9**.

### Modicon 884 PLCs - Hardware Setup

Set your PLC to the communication settings you require. The default hardware settings for the 884 are as follows:

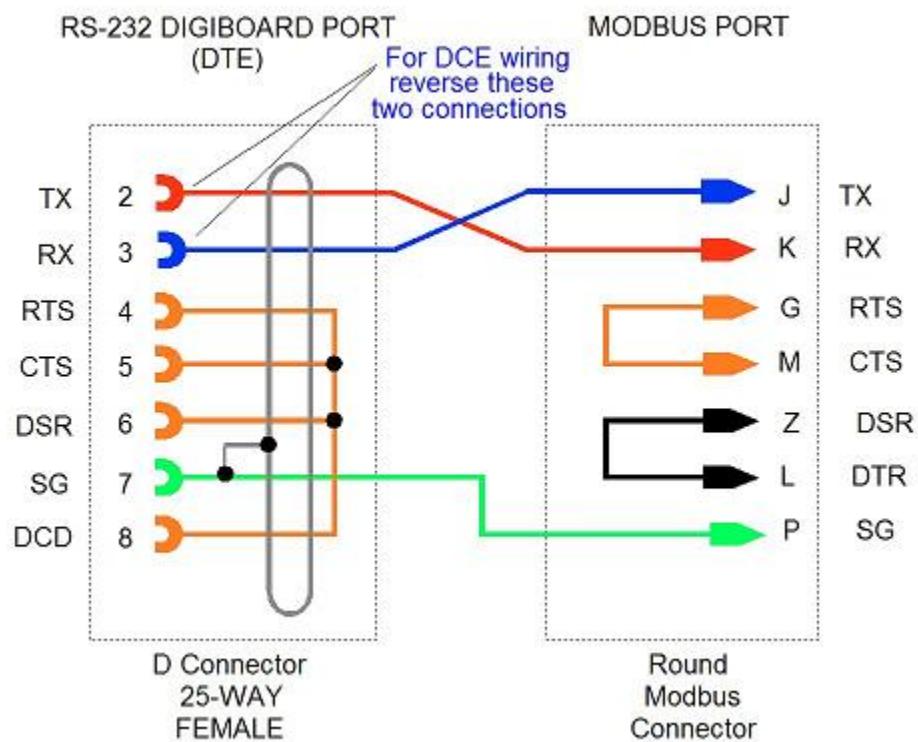
Setting	Value
Baud Rate	19200
Data Bits	8
Stop Bits	1
Parity	2 (Even)

**Note:** These settings are recommendations only. If you use the Communications Express Wizard, these default settings are configured in your project automatically, though your hardware may support other values. If you do set the baud rate, data bits, stop bits or parity to another value, you must manually set the new value(s) in your project.

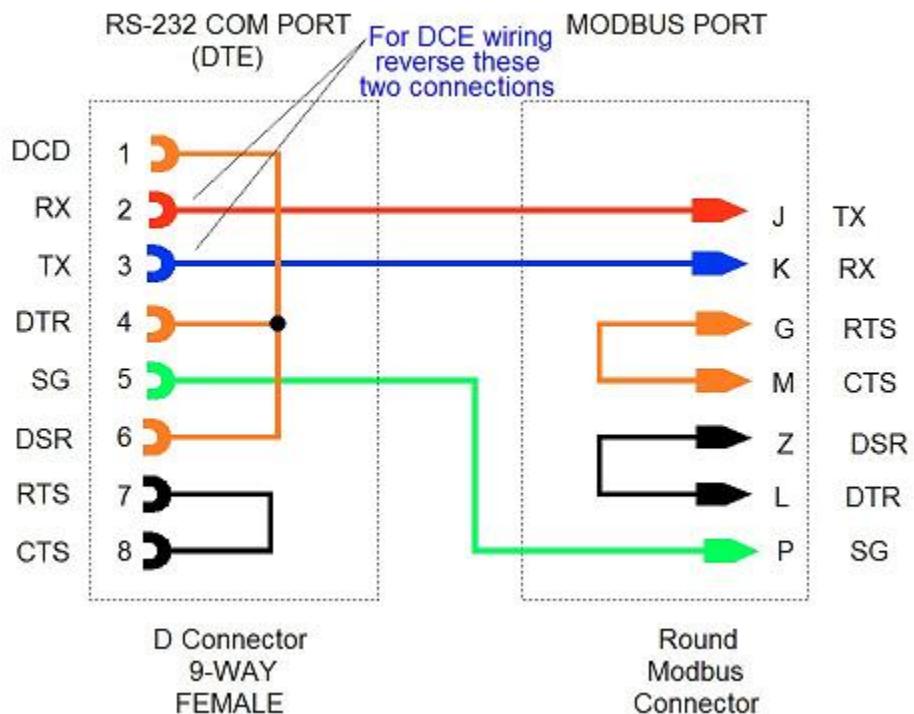
Set the following configuration options on the Modbus port on the Modicon PLC:

- Station number
- RTU mode (If this is not selected then the MODBUSA protocol is used)
- Delay time to 1

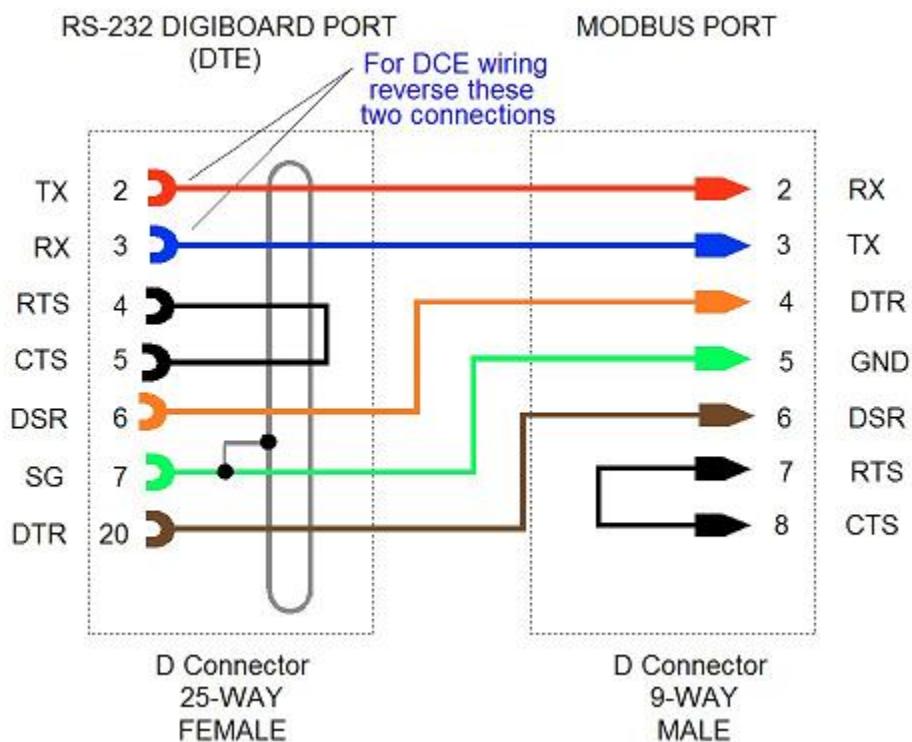
## Modicon 884 PLCs - Wiring Diagram 1



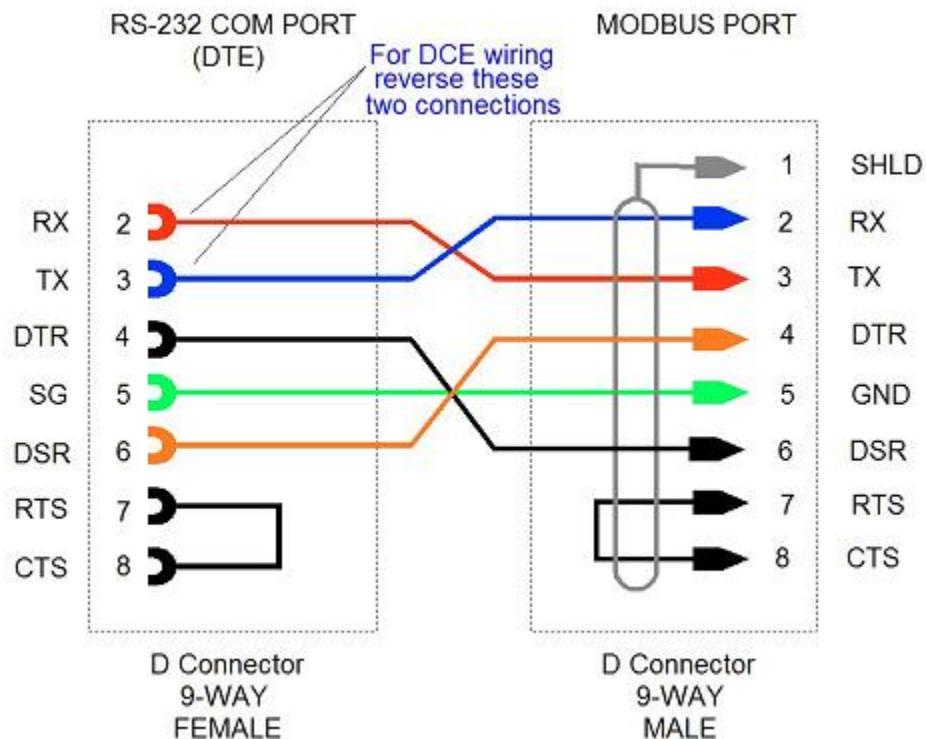
## Modicon 884 PLCs - Wiring Diagram 2



## Modicon 884 PLCs - Wiring Diagram 3



## Modicon 884 PLCs - Wiring Diagram 4



## Modicon 884 PLCs - Communication Settings

To establish communication with a device, configure the Boards, Port and I/O Device in Plant SCADA Studio's **Topology** activity. If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure these, use the settings outlined below.

### Board Settings

Typically, you would use a serial board or COM port for this communication. Refer to the instructions for setting up and using COM ports or serial boards. Or complete the Board settings as instructed with the following specific information.

Field	Value
Board Type	If using a serial board or COM port, enter COMx.
Address	If using a serial board or COM port, enter 0.
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.

### Ports Settings

Complete the Ports settings with the following specific information.

Field	Value
Port Number	This value matches the COM port number. This number is defined in the Ports section of the Control Panel.
Baud Rate	This value matches the setting of the unit - 19200.
Data Bits	Enter 8.
Stop Bits	Enter 1.
Parity	This value matches the setting of the unit - EVEN_P.
Special Options	You may want to use the special options for the COMx Driver if you are using a modem (or similar) and want the driver to perform differently. For more information, see the topic <i>COMx Driver Special Options Reference</i> in the <i>Communicating with I/O Devices</i> chapter of the main help.

## I/O Device Settings

Complete the I/O Device settings with the following specific information.

Field	Value
I/O Device Address	The PLC station number configured on the PLC. Try 1 to 10. 1 specifies station number 1. 9 specifies station number 9.
I/O Device Protocol	Enter MODBUS if you want to use the Modbus RTU (binary) protocol. Enter MODBUSA if you want to use the Modbus ASCII protocol. (See the topic <a href="#">Protocol Variants</a> to determine if a non-standard protocol variant is required.)

## Modicon 884 PLCs - Data Types

Data Types	Address Format	Plant SCADA Data Type
<b>DIGITAL</b>		
Output Coils	000001 to 065536	DIGITAL ( <b>Note:</b> Leading zero is required.)
Input Status	100001 to 165536	DIGITAL (Read Only)
<b>INTEGER</b>		
Input Registers	300001 to 365536	INT / LONG / STRING / REAL (Read Only)
Output Registers	400001 to 465536	INT / LONG / STRING / REAL

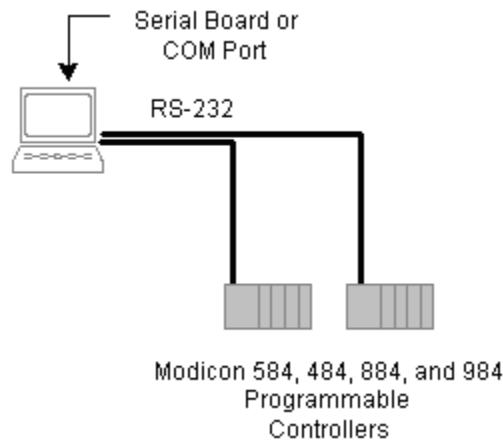
### Examples

Data Type	DIGITAL
Address	00001
Comment	Output Coil 00001
Data Type	INT
Address	40001
Comment	Output Register 40001

**Note:** In the MODBUS protocol, LONG data types default to a simplified implementation, with a shortened range of 0 to 655,359,999. To increase the LONG data type to the full range, set the [MODBUS]LongDataType parameter to either 1 or 3.

## Modicon 984 PLCs

The serial method of communication to Modicon 984 PLCs, uses either the MODBUS or MODBUSUSA protocol. Communication is via the Modbus port on the PLC. Using this method you can connect to single PLCs or to multiple PLCs as in the following diagram:



## Modicon 984 PLCs - Device Address

The I/O Device Address for a 984 PLC is PLC station number. Try 1 to 10.

- **1** specifies station number **1**.
- **9** specifies station number **9**.

## Modicon 984 PLCs - Hardware Setup

Set your PLC to the communication settings you require. The default hardware settings for the 984 are as follows:

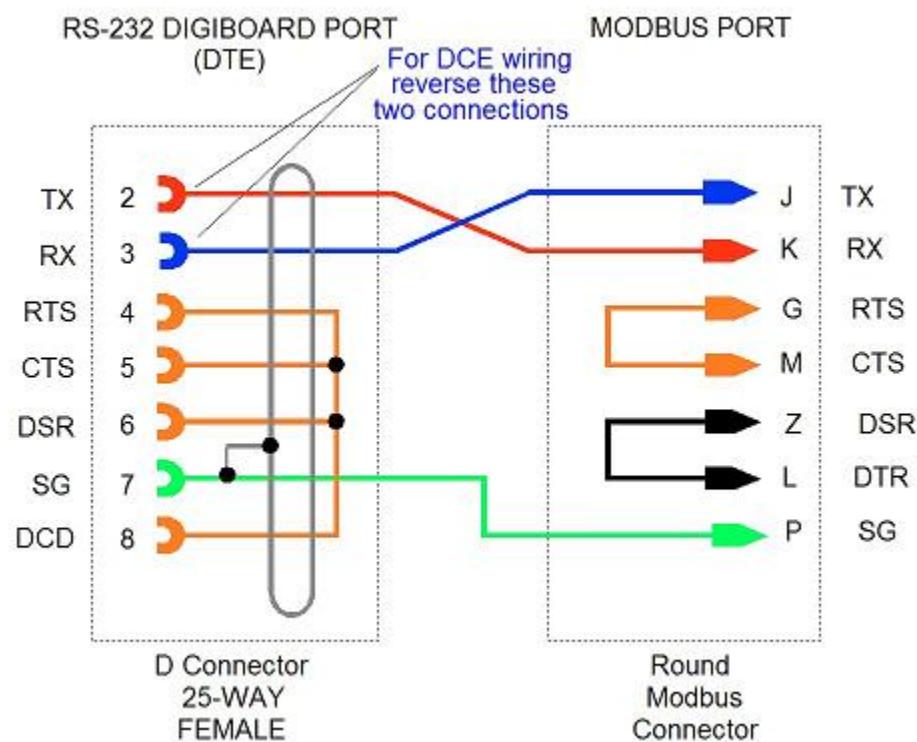
Setting	Value
Baud Rate	19200
Data Bits	8
Stop Bits	1
Parity	2 (Even)

**Note:** These settings are recommendations only. If you use the Communications Express Wizard, these default settings are configured in your project automatically, though your hardware may support other values. If you do set the baud rate, data bits, stop bits or parity to another value, you must manually set the new value(s) in your project.

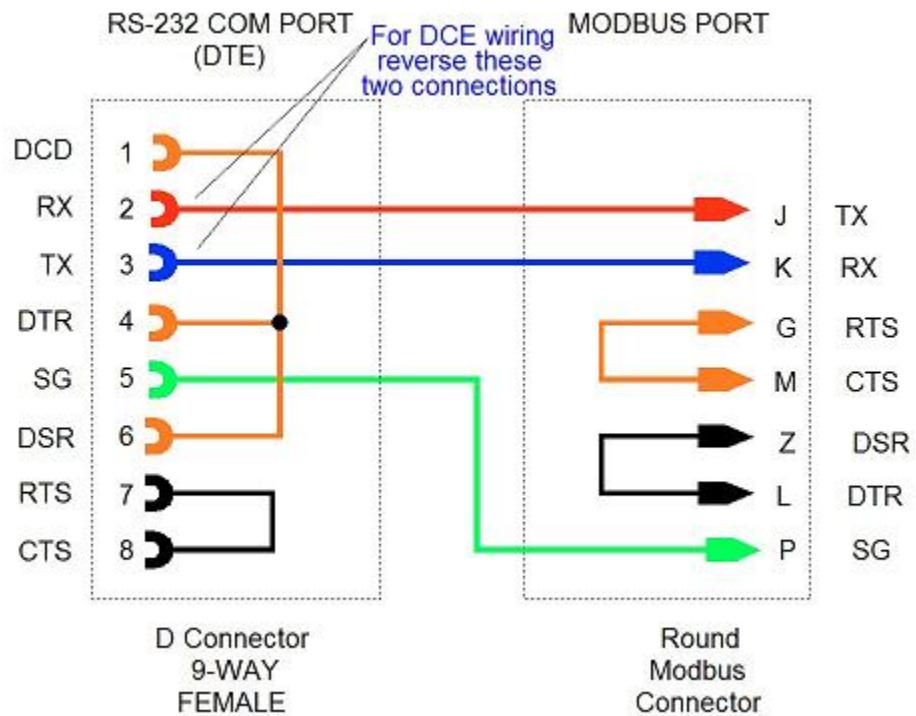
Set the following configuration options on the Modbus port on the Modicon PLC:

- Station number
- RTU mode (If this is not selected then the MODBUSA protocol is used)
- Delay time to 1

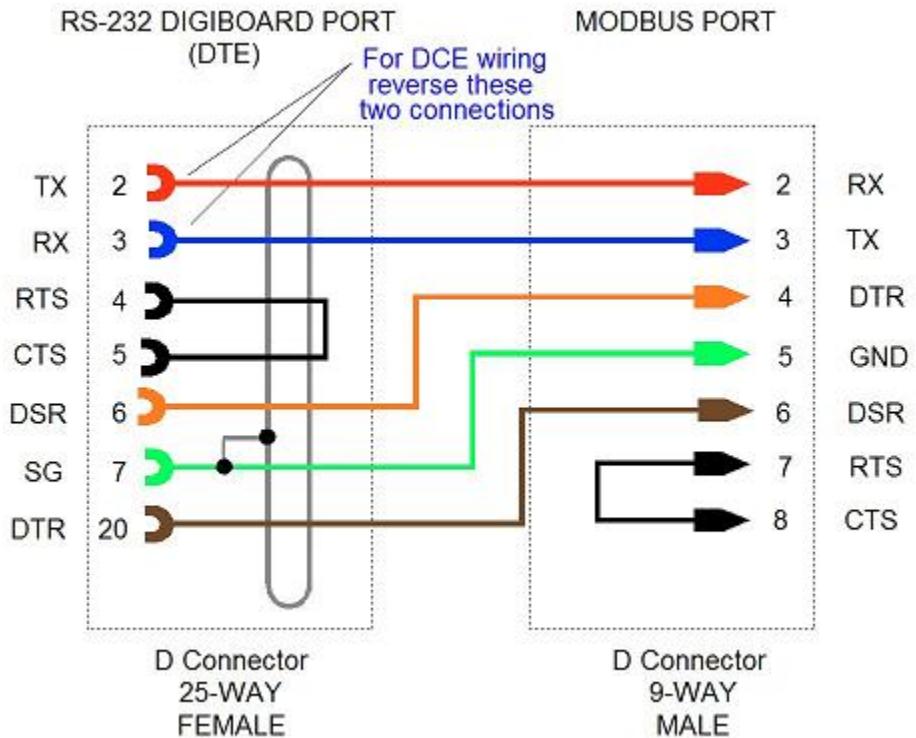
### Modicon 984 PLCs - Wiring Diagram 1



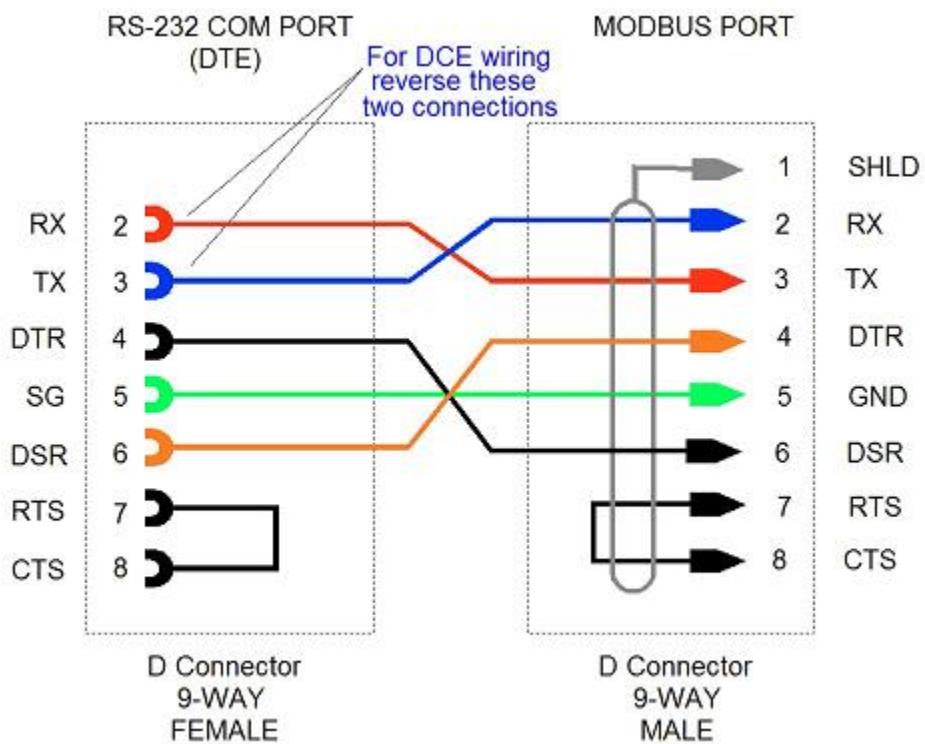
## Modicon 984 PLCs - Wiring Diagram 2



## Modicon 984 PLCs - Wiring Diagram 3



## Modicon 984 PLCs - Wiring Diagram 4



## Modicon 984 PLCs = Communication Settings

To establish communication with a device, configure the Boards, Port and I/O Device in Plant SCADA Studio's **Topology** activity. If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure these, use the settings outlined below.

**Board Settings**

Typically, you would use a serial board or COM port for this communication. Refer to the instructions for setting up and using COM ports or serial boards. Or complete the Board settings as instructed with the following specific information.

Field	Value
Board Type	If using a serial board or COM port, enter COMx.
Address	If using a serial board or COM port, enter 0.
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.

## Port Settings

Complete the Port settings with the following specific information.

Field	Value
Port Number	This value matches the COM port number. This number is defined in the Ports section of the Control Panel.
Baud Rate	This value matches the setting of the unit - 19200.
Data Bits	Enter 8.
Stop Bits	Enter 1.
Parity	This value matches the setting of the unit - EVEN_P.
Special Options	You may want to use the special options for the COMx Driver if you are using a modem (or similar) and want the driver to perform differently. For more information, see the topic <i>COMx Driver Special Options Reference</i> in the <i>Communicating with I/O Devices</i> chapter of the main help.

## I/O Device Settings

Complete the I/O Device settings with the following specific information.

Field	Value
I/O Device Address	The PLC station number configured on the PLC. Try 1 to 10. 1 specifies station number 1. 9 specifies station number 9.
I/O Device Protocol	Enter MODBUS if you want to use the Modbus RTU (binary) protocol. Enter MODBUSA if you want to use the Modbus ASCII protocol. (See the topic <a href="#">Protocol Variants</a> to determine if a non-standard protocol variant is required.)

## Modicon 984 PLCs - Data Types

Data Types	Address Format	Plant SCADA Data Type
<b>DIGITAL</b>		
Output Coils	000001 - 065536	DIGITAL ( <b>Note:</b> Leading zero is required.)
Input Status	100001- 165536	DIGITAL (Read Only)
<b>INTEGER</b>		
Input Registers	300001- 365536	INT / LONG / STRING / REAL (Read Only)
Output Registers	400001- 465536	INT / LONG / STRING / REAL

**Examples**

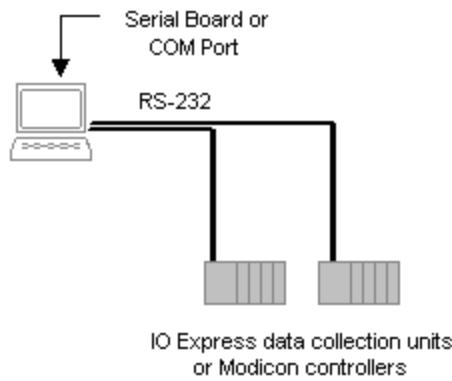
Data Type	DIGITAL
Address	00001
Comment	Output Coil 00001
Data Type	INT
Address	40001
Comment	Output Register 40001

**Note:**

- 1) Modicon 984 PLCs support remapping of both reads and writes.
- 2) In the MODBUS protocol, LONG data types default to a simplified implementation, with a shortened range of 0 to 655,359,999. To increase the LONG data type to the full range, set the [MODBUS]LongDataType parameter to either 1 or 3.

**Moore Products IOExpress**

The serial method of communication to IOExpress data collection units, manufactured by Moore Products, uses the MODBUS protocol. Using this method, you can connect to single units or to multiple units as in the following diagram:



### Moore Products IOExpress - Device Address

The I/O Device Address for a IOExpress unit station address configured on the IOExpress unit. Try 1-10, for example:

- **1** specifies station address **1**.
- **9** specifies station address **9**.

### Moore Products IOExpress - Hardware Setup

Set your unit to the communication settings you require. The default hardware settings for the IOExpress are as follows:

Setting	Value
Baud Rate	19200
Data Bits	8
Stop Bits	1
Parity	2 (Even)

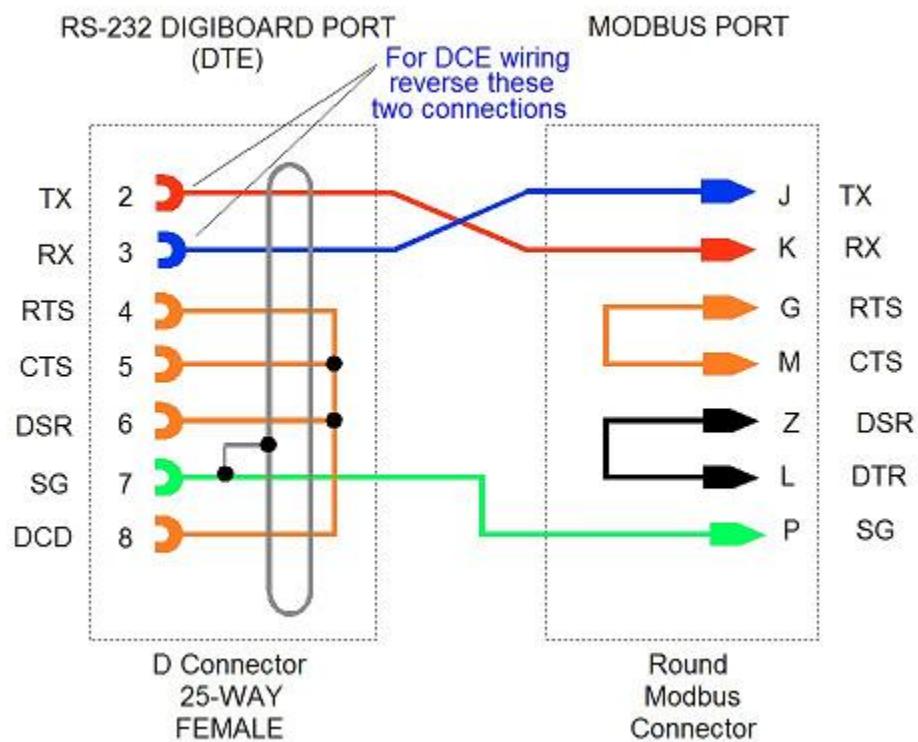
**Note:** These settings are recommendations only. If you use the Communications Express Wizard, these default settings are configured in your project automatically, though your hardware may support other values. If you do set the baud rate, data bits, stop bits or parity to another value, you must manually set the new value(s) in your project.

Set the following configuration options on the unit:

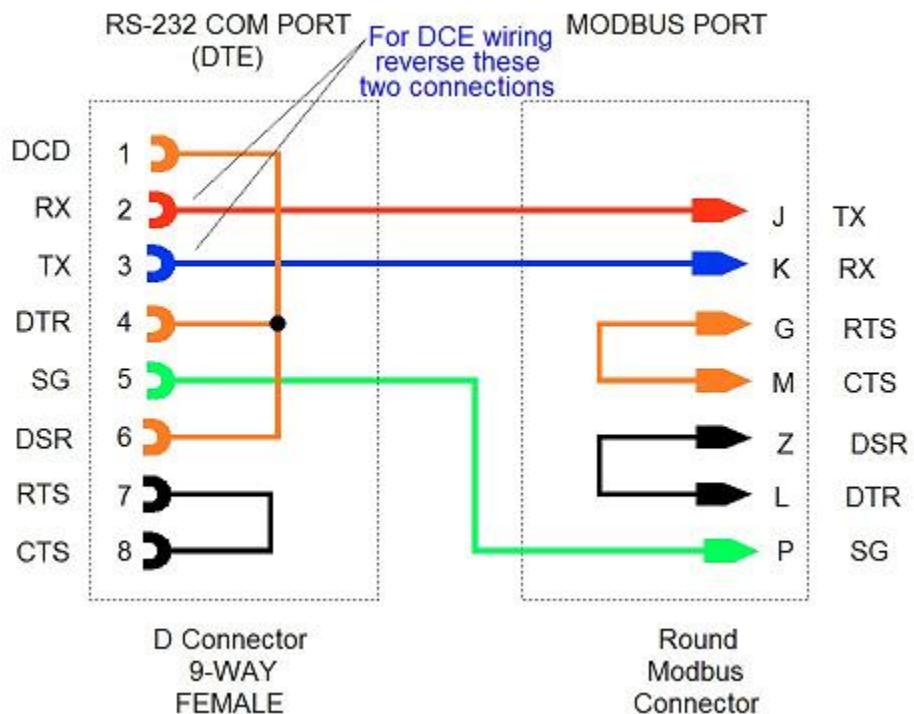
- Station number
- RTU mode (If this is not selected then the MODBUSA protocol must be used)
- Delay time to 1

The IOExpress unit emulates the MODBUS protocol, but has some slight differences. See [Working with Non-standard Devices](#).

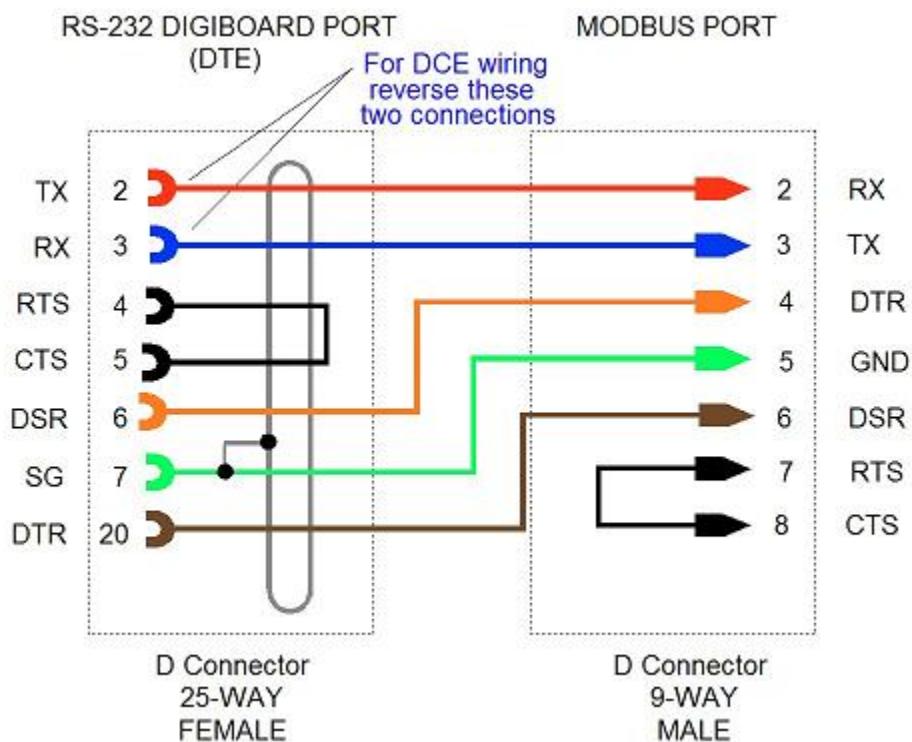
## Moore Products IOExpress - Wiring Diagram 1



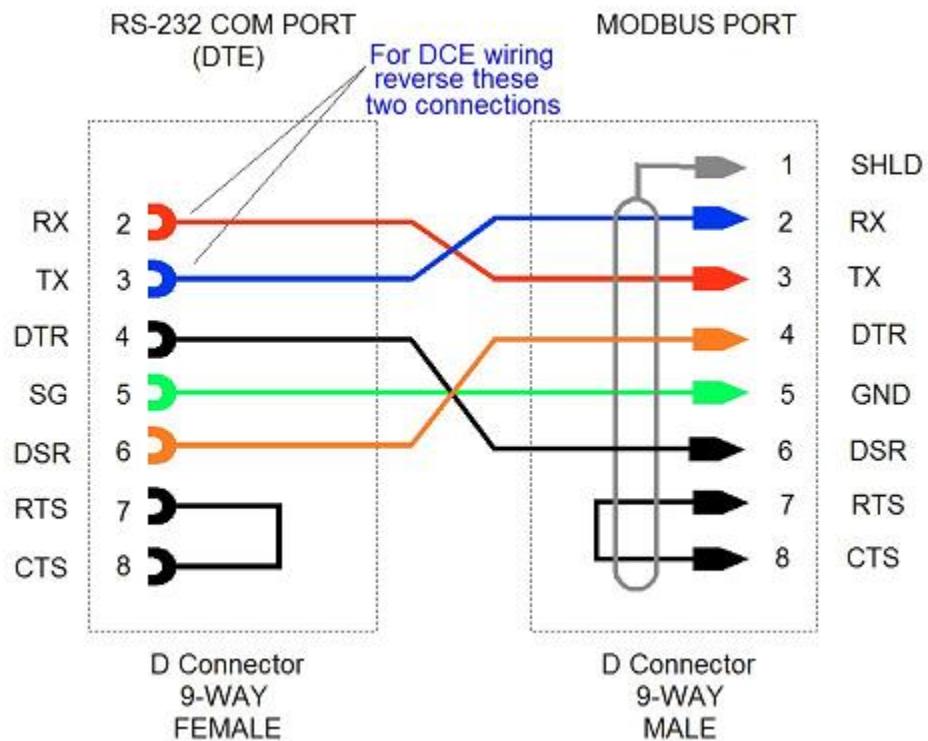
## Moore Products IOExpress - Wiring Diagram 2



## Moore Products IOExpress - Wiring Diagram 3



## Moore Products IOExpress - Wiring Diagram 4



## Moore Products IOExpress - Communication Settings

To establish communication with a device, configure the Boards, Port and I/O Device in Plant SCADA Studio's **Topology** activity. If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure these, use the settings outlined below.

### Board Settings

Typically, you would use a serial board or COM port for this communication. Refer to the instructions for setting up and using COM ports or serial boards, or complete the Board settings as instructed with the following specific information.

Field	Value
Board Type	If using a serial board or COM port, enter COMx.
Address	If using a serial board or COM port, enter 0.
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.

### Port Settings

Complete the Port settings with the following specific information.

Field	Value
Port Number	This value matches the COM port number. This number is defined in the Ports section of the Control Panel.
Baud Rate	This value matches the setting of the unit - 19200.
Data Bits	Enter 8.
Stop Bits	Enter 1.
Parity	This value matches the setting of the unit - EVEN_P.
Special Options	You may want to use the special options for the COMx Driver if you are using a modem (or similar) and want the driver to perform differently. For more information, see the topic <i>COMx Driver Special Options Reference</i> in the <i>Communicating with I/O Devices</i> chapter of the main help.

## I/O Device Settings

Complete the I/O Device settings with the following specific information.

Field	Value
I/O Device Address	The PLC station number configured on the PLC. Try 1 to 10. 1 specifies station number 1. 9 specifies station number 9.
I/O Device Protocol	Enter MODBUS. (See the topic <a href="#">Protocol Variants</a> to determine if a non-standard protocol variant is required.)

### Moore Products IOExpress - Data Types

Data Types	Address Format	Plant SCADA Data Type
<b>DIGITAL</b>		
Output Coils	000001 to 065536	DIGITAL ( <b>Note:</b> Leading zero is required.)
Input Status	100001 to 165536	DIGITAL (Read Only)
<b>INTEGER</b>		
Input Registers	300001 to 365536	INT / LONG / STRING / REAL (Read Only)
Output Registers	400001 to 465536	INT / LONG / STRING / REAL

#### Examples

Data Type	DIGITAL
Address	00001
Comment	Output Coil 00001
Data Type	INT
Address	40001
Comment	Output Register 40001

## Working With Non-standard Devices

Some non-standard PLCs support the Modbus protocol. However, some of these I/O devices do not support the protocol correctly. When Plant SCADA tries to communicate with these I/O devices, you might get errors on startup or during communication.

To handle these non-standard I/O devices, the Modbus protocol has some special options to change its operation:

1. On startup, Plant SCADA tries to read some digital inputs to verify that the PLC does exist. If Plant SCADA cannot read these bits, it assumes the PLC is bad, and remains off line. Some non-standard I/O devices do not support digital bits, so this command results in an error. With the [\[MODBUS\]InitType](#) parameter, you can change the type of variable the Modbus protocol tries to read on startup, to allow communication to start.
2. The standard MODBUS driver allows reading of 2000 bits in one request. Some non-standard I/O devices do not support such large reads. You can change the maximum read size with the [\[MODBUS\]MaxBits](#) parameter. (The default is 2000.)

Change the MAX\_LENGTH field for the Modbus protocol (in the \Plant SCADA\BIN\PROTDIR.DBF file) to the same value. Re-compile your project after changing this option. Be aware that if you reinstall Plant SCADA, the PROTDIR.DBF file is overwritten and you will need to repeat the change.

3. Some I/O devices that operate over radio modems also require the padding of protocol messages. The padding characters wake the radio modem to allow the rest of the message to be transmitted. Set the [\[MODBUS\]Pad](#) parameter to the number of characters to pad, and the [\[MODBUS\]PadChar](#) parameter to the ASCII code of the character to pad. For example, to pad 20 0xFF characters to the start of the message, set:

```
[MODBUS]
Pad=20
PadChar=255
```

4. In the Modbus protocol, LONG data types default to a simplified implementation, with a shortened range of 0 to 655,359,999. To increase the LONG data type to the full range, set the [\[MODBUS\]LongDataType](#) parameter to either 1 or 3.

As a LONG is composed from two registers, the register order used by Plant SCADA must match the order used by the PLC. Refer to the help accompanying the [\[MODBUS\]LongDataType](#) parameter to determine the appropriate value.

---

**Note:** The MODBUS driver supports a number of variants of the MODBUS protocol to communicate with some non-standard PLCs. See the topic MODBUS protocol variants [MODBUS Protocol Variants](#) to determine which variant you should use when configuring a device.

---

## Protocol Variants

Many devices from different vendors support the Modbus protocol. Unfortunately, addressing structures used by these devices vary. The following information will help you understand which protocol you should select on the I/O devices dialog box to make the addresses you enter into the variables form match those entered into the programming software for your device.

You will need to know the following:

- What is the lowest register / coil number in the device?

Some devices (such as a TSX Quantum or Modicon 948) have register / coil 1 as the lowest register / coil

number. Other devices (such as the TSX Premium) have register / coil 0 as the lowest register / coil number

- When addressing an individual bit from a register, what is the lowest bit number?

Some devices refer to the bits within a register as bits 0-15, others refer to them as bits 1-16.

Based on the above information you can select a protocol using the following table:

Range of bits in a register	Lowest coil / register number	Protocol
1-16	1	MODBUS (default)
1-16	0	MODBUS1
0-15	1	MODBUS2
0-15	0	MODBUS3

If you are using a PLC that supports extended registers (6xxxxx addresses in a TSX Quantum for example) then the following table should be used:

Range of bits in a register	Lowest coil / register number	Protocol
1-16	1	MODBUS10
1-16	0	MODBUS11
0-15	1	MODBUS12
0-15	0	MODBUS13

Two special protocols exist for use with the TSX Quantum and TSX Premium PLC ranges. These protocol options will select the correct addressing modes, select certain INI values and other settings to provide compatibility with these devices without the need to set any other special parameters.

These protocols also allow you to access memory digitals and memory words using %M7 or %MW3 or %MW4.6 style addressing. Select a protocol as follows:

PLC Model	Protocol
TSX Quantum	MODBUS20
TSX Premium	MODBUS30

## Customizing a Project using Citect.ini Parameters

Citect.ini parameters are used to tune the performance of the Plant SCADA MODBUS driver and to perform runtime maintenance diagnostics.

You can customize the way Plant SCADA communicates with the Modbus system (and even individual PLCs) by creating or editing the [MODBUS] section of the citect.ini file for your project.

There are some common Plant SCADA driver settings as well as custom MODBUS driver settings.

When Plant SCADA starts at runtime, it reads configuration values from the citect.ini file that is stored locally. Therefore, any MODBUS configuration settings must be included in the citect.ini file located on the computer

acting as the I/O server to the Modbus system.

By default, Plant SCADA looks for the citect.ini file in the Plant SCADA project \bin directory. If it can't find the file there, it searches the default Windows directory.

## MODBUS Driver Parameters

These parameters default to a value which have shown to be suitable for the majority of circumstances.

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

With certain parameters, the MODBUS driver can apply different initialization parameter values to specific I/O devices or groups of I/O devices. For details see [Device/Group-specific Parameters](#).

## [MODBUS]Block

A value (bytes) used by the I/O server to determine if two or more packets can be blocked into one data request before being sent to the I/O device. For example, if you set the value to 10 and the I/O server receives two simultaneous data requests (one for byte 3 and another for byte 8) the two requests are blocked into a single physical data request packet. This single request packet is then sent to the I/O device, saving on bandwidth and processing.

**Allowable Values:** 5 to 256

**Default Value:** 50

## [MODBUS.<Port\_Name>.<IODevice\_Name>]Broadcast

Sends a write request to a specified device on a Modbus network.

**Note:** This parameter can only be used as a device-specific parameter, for example, [MODBUS.<Port\_Name>.<IODevice\_Name>]Broadcast=1. It should not be used globally. For more details, see [Device/group-specific parameters](#).

To use this feature:

1. Define an I/O device with address 0 (zero) for the Channel of interest.
2. Define the value for the BroadcastDelay parameter in the Citect.ini file (default is 50ms). This parameter defines the amount of time the driver will delay after submitting a broadcast request before it will send another request to the PLCs.
3. Define tags (Coils and Registers) associated with this I/O device.
4. Implement a Plant SCADA application that requests writes to the tags (via Cicode, application display, and so on).

Requests for Variable Tag READs to the I/O device with the address zero (0) will return a Driver Error of Unknown

Command (driver error 15 decimal). Requests for Variable Tag WRITES to the I/O device with the address zero (0) will be issued to ALL controllers on the SAME Channel (port) as the I/O device with the assigned address of zero. The Modbus protocol does not provide a response to broadcast requests so inspect the appropriate coils or registers in each controller to observe completion of the request.

**Allowable Values:**

0 = Broadcast disabled

1 = write request will be issued

**Default Value:** 1

## [MODBUS]BroadcastDelay

Defines the amount of time the driver will delay after submitting a broadcast request before sending another request to the PLCs.

**Allowable Values:** 0 to 300 (milliseconds)**Default Value:** 50

---

**Note:** This parameter is only available at the MODBUS level, i.e. [MODBUS]

---

## [MODBUS]Delay

The period (in milliseconds) to wait between receiving a response and sending the next command.

**Allowable Values:** 0 to 32767 (milliseconds)**Default Value:** 0

## [MODBUS]DoCRC

Enables or disables the Modbus Cyclic Redundancy Check (CRC), a communications check. The MODBUS driver does a CRC on incoming data and provides a CRC remainder in outgoing data blocks. In some limited cases, such as the testing of "slave" drivers, it may be necessary to disable the CRC.

**Allowable Values:** 1 or 0, where:

1 = enable CRC

0 = disable CRC

**Default Value:** 1

The CRC should remain "enabled" under normal circumstances. If the CRC is disabled, no checks are performed on incoming data and the CRC field in the data transmitted by the driver remains 0. This will mean the driver cannot detect lack off communication between itself and the I/O device.

## [MODBUS]FileNumber

If the parameter InitType is set to 20 to implement extended register access, use this parameter to identify the extended memory file number.

**Allowable Values:** 1 to 65535**Default value:** 1

## [MODBUS]FailOnBadData

Used by the MODBUS driver to determine whether or not to force the display of good data within a block read which contains bad tags during tag read.

**Allowable Values:** 1 or 0, where:

1 = A block read which contains a bad tag would result in the whole block returning #BAD

0 = The MODBUS driver will not return "#BAD" for a block of data if at least one valid value is in the read block.

**Default Value:** 1

## [MODBUS]FloatMode

---

**Note:** This parameter is not applicable if you are using the [protocol variant](#) MODBUS20 or MODBUS30 to communicate with a device. Under these circumstances, a value of 0 (zero) will be hardcoded for this parameter.

---

This INI parameter sets the byte order for floating point values (the MODBUS driver supports floating point values). Some systems expect to use a different byte order for their floating point data.

**Allowable Values:** 0 to 3, where:

0 - Byte order = 1 0 3 2

1 - Byte order = 3 2 1 0

2 - Byte order = 0 1 2 3

3 - Byte order = 2 3 0 1

**Default Value:** 0

With this parameter, you can set different values for specific I/O devices or groups of I/O devices. See [Device/group-specific parameters](#).

## [MODBUS]ForceMultiCoilsOnly

Forces the use of only function code 15 (multiple coils) for coil writes.

**Allowable Values:**

0 - (function code 5 and 15)

1 - (function code 15 only)

**Default Value:** 0

## [MODBUS]InitType

The type of variable the MODBUS protocol tries to read on startup, to allow communication to start.

**Allowable Values:** 1 to 4, or 20 (for extended registers access).

InitType	Address	Variable
1	00001 - 00017	Output status
2	10001 - 10017	Input status
3	40001	Output registers

4	30001	Input registers
20		General reference

**Default Value:** 2

If setting this parameter to 20 for extended register access, you will need to adjust the FileNumber parameter to identify the extended memory file number.

With this parameter, you can set different values for specific I/O devices or groups of I/O devices. See [Device/group-specific parameters](#).

**[MODBUS]InitUnitAddress**

The MODBUS driver reads the citect.ini file to determine the correct unit address for initialization of a PLC. InitUnitAddress is the parameter used to set the unit address.

**Allowable Values:** 0 to 65535**Default Value:** 0

With this parameter, you can set different values for specific I/O devices or groups of I/O devices. See [Device/group-specific parameters](#).

**[MODBUS]LongDataType**


---

**Note:** This parameter is not applicable if you are using the [protocol variant](#) MODBUS20 or MODBUS30 to communicate with a device. Under these circumstances, a value of 3 will be hardcoded for this parameter.

---

In the Modbus protocol, LONG data types default to a simplified implementation, with a shortened range of 0 to 99,999,999 - mode 0. Mode 2 has the same range as mode 0, but with the register order swapped. Mode 1 supports the complete LONG range of -2,147,483,648 to +2,147,483,647. Mode 3 has the same range as mode 1, but with the register order swapped.

**Allowable Values:** 0 to 3, where:

- 0 = 10000 x low register + high register
- 1 = 65536 x low register + high register
- 2 = 10000 x high register + low register
- 3 = 65536 x high register + low register

**Default Value:** 3

With this parameter, you can set different values for specific I/O devices or groups of I/O devices. See [Device/group-specific parameters](#).

**[MODBUS]MaxBits**


---

**Note:** This parameter is not applicable if you are using the [protocol variant](#) MODBUS20 or MODBUS30 to communicate with a device. If MODBUS20 is being used, a value of 1904 will be hardcoded for this parameter. If MODBUS30 is being used, a value of 1024 will be hardcoded.

---

The maximum read size in one request. Decrease the value for non-standard I/O devices that do not support large reads.

**Allowable Values:** 8 to 32767

**Default Value:** 1904

### [MODBUS]MaxBitsExt

---

**Note:** This parameter is not applicable if you are using the [protocol variant](#) MODBUS20 or MODBUS30 to communicate with a device. If MODBUS20 is being used, a value of 1904 will be hardcoded for this parameter. If MODBUS30 is being used, a value of 1024 will be hardcoded.

This INI parameter sets the maximum bits blocked for the Quantum PLCs "Ex:y[.z]" address type.

**Allowable Values:** 8 to 32767

**Default Value:** 1904

### [MODBUS]MaxPending

The maximum number of pending commands that the driver holds ready for immediate execution.

**Allowable Values:** 1 to 32

**Default Value:** 2

### [MODBUS]Pad

The number of characters with which to pad protocol messages. The padding characters wake the radio modem to allow the rest of the message to be transmitted. A value of 20 is recommended for non-standard I/O devices that operate over radio modems.

**Allowable Values:** 0 to 65535

**Default Value:** 0

### [MODBUS]PadChar

The ASCII code of the padding character with which to pad protocol messages. The padding characters wake the radio modem to allow the rest of the message to be transmitted.

**Allowable Values:** 0 to 255

**Default Value:** 255

### [MODBUS]PollTime

The interrupt or polling service time (in milliseconds). Setting the polling time to 0 puts the driver in interrupt mode.

**Allowable Values:** 0 to 300 (milliseconds)

**Default Value:** 0

### [MODBUS]PresetMultiRegistersOnly

Forces the use of only function code 16 (multiple registers) for register writes.

**Allowable Values:** 0 or 1:

0 - (function code 6 and 16)  
1 - (function code 16 only)

**Default Value:** 0

## [MODBUS]RegisterBitReverse

---

**Note:** This parameter is not applicable if you are using the [protocol variant](#) MODBUS20 or MODBUS30 to communicate with a device. Under these circumstances, a value of 1 will be hardcoded for this parameter.

This INI parameter allows reverse interpretation of Most Significant Bit (MSB) and Least Significant Bit (LSB) in words. The Modbus specification defines bit 1 of a word as the MSB, and bit16 of the word as the LSB. Some devices use a different order, causing bit 1 of a word to be the LSB and bit 16 to be the MSB.

**Allowable Values:** 0 or 1, where:

1 - Bit 1= LSB, Bit 16 = MSB  
0 - Bit 1 = MSB, Bit 16 = LSB

**Default Value:** 1

With this parameter, you can set different values for specific I/O devices or groups of I/O devices. See [Device/group-specific parameters](#).

## [MODBUS]Retry

The number of times to retry a command after a timeout.

**Allowable Values:** 0 to 8

**Default Value:** 0

## [MODBUS]SendBCDSwap

Reverses the byte order for BCDs on writes (the MODBUS driver supports BCDs). Some systems expect to use a different byte order for their data; in the case of BCDs, this causes "1234" to be written to the device as "3412".

**Allowable Values:** 0 or 1, where:

1 = 1 2 3 4  
0 = 3 4 1 2

**Default Value:** 1

With this parameter, you can set different values for specific I/O devices or groups of I/O devices. See [Device/group-specific parameters](#).

## [MODBUS]Status

Sometimes you can read data from a device even though the processor module is not running. In view of this, it can be useful to detect this situation by monitoring a continuously changing register or a digital. If the register is unable to change in a given period or the digital becomes zero, the processor can be assumed to have stopped running.

If enabled, the MODBUS driver will check on startup and approximately every watchtime the status of a user specified variable in the PLC is changing. If a digital is specified, the value should be on, otherwise the unit is put offline. If an analog is specified, on startup two consecutive reads with an interval of 1000ms (specified by

[MODBUS]Timeout) should return different values before the unit can be put online. At every watchtime, if the digital is off or the analog data does not change from the last value read, the driver returns an error and the unit is put offline.

**Allowable Values:**

RawType

BitWidth

UnitType

UnitAddress

UnitAddress

where:

RawType	BitWidth	UnitType	UnitAddress	UnitCount
0 = Digital	1 = Digital	See variable specification .dbf files	The item number or bit number	1 = analog 16 = digital
1 = Int	16 = Int			
4 = Long	32 = Long			
8 = Byte	8 = Byte			

**Default Value:**

Zero length string

**Examples**

To specify Modbus address 40001 as the analog tag variable which must change to indicate the device is online, use:

```
[MODBUS]
status=1,16,3,0,1
```

To use Modbus address 00001 as the digital tag variable which must remain set to 1 to indicate the device is online, then use:

```
[MODBUS]
status=0,1,1,0,16
```

## [MODBUS]StringReverse

Reverse byte order for strings. The MODBUS driver supports strings. Some systems expect to use a different byte order for their data, which in the case of string variables, causes "ABCD" to appear as "BADC".

**Allowable Values:** 0 or 1

**Default Value:** 0 (do not reverse byte order)

With this parameter, you can set different values for specific I/O devices or groups of I/O devices. See [Device/group-specific parameters](#).

## [MODBUS]Timeout

Specifies how many milliseconds to wait for a response before displaying an error message.

**Allowable Values:** 0 to 32000 (milliseconds)

**Default Value:** 3000

**Note:** This parameter is only available at the MODBUS Portlevel, i.e. [MODBUS.Port] but values are overridden by the default values in theyslog and Kernel page.

## [MODBUS]WatchTime

The frequency (in seconds) that the driver uses to check the communications link to the I/O device.

**Allowable Values:** 0 to 128 (seconds)

**Default Value:** 30

## Device/Group-specific Parameters

The MODBUS driver can apply different initialization parameter values to specific I/O devices or groups of I/O devices. This means the user can specify:

Global parameters that apply to all devices.

Channel- (port-) level parameters that apply to all devices on the specified port.

Group-level parameters that apply to all devices in a specified group.

Device-level parameters that apply only to the specified device.

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

This feature can be implemented in the citect.ini for the following [MODBUS Parameters](#):

- Broadcast
- BroadcastDelay
- FloatMode
- ForceMultiCoilsOnly
- InitType
- InitUnitAddress
- LongDataType
- PresetMultiRegistersOnly
- RegisterBitReverse
- SendBCDSwap

- StringReverse

To set parameters for a particular port, group, or device, you must create a new section in the citect.ini file. Label it with the driver name followed by a period (.) character and the name of the particular port, group, or device you want to specify the parameter setting for.

For example:

- [MODBUS.<Port\_Name>]: applies the parameter settings to the specified port.
- [MODBUS.<Group\_Name>]: applies the parameter settings to the specified group.
- [MODBUS.<Port\_Name>,<IODevice\_Name>]: applies to the specified device.

Any parameters you then define in the following section of the citect.ini file relate only to the specified device or device group.

#### Example

The following citect ini file format is an example of how the InitType parameter could be specified differently for different I/O devices communicating using the Modbus protocol.

Assume that two ports are used: PORT1 and PORT2. PORT1 has three I/O devices attached:

- DEV1A DEV1B EV1C

POR2 also has three devices:

- DEV2A DEV2B DEV2C

Assume that the user has specified that DEV1C and DEV2C belong to GROUPZ. The citect.ini file contains the following entries:

```
[MODBUS]
InitType=1
[MODBUS.PORT1]
InitType=2
[MODBUS.PORT2]
InitType=2
[MODBUS.GROUPZ]
InitType=3
[MODBUS.PORT1.DEV1A]
InitType=1
[MODBUS.PORT2.DEV2B]
InitType=4
```

The resultant InitType for the I/O devices will be as follows:

DEV1A:	1	as a result of [MODBUS.PORT1.DEV1A]
DEV1B:	2	as a result of [MODBUS.PORT1]
DEV1C:	3	as a result of [MODBUS.GROUPZ]
DEV2A:	2	as a result of [MODBUS.PORT2]

DEV2B:	4	as a result of [MODBUS.PORT2.DEV2B]
DEV2C:	3	as a result of [MODBUS.GROUPZ]

As the above example shows, there is a hierarchy that determines the outcome of such settings. In simple terms, specific parameter settings overwrite general level settings. Therefore, parameters written in the scope of I/O devices will overwrite those set for groups; parameters set for groups will overwrite global settings, and so on.

## MODBUS Driver Errors

MODBUS has two kinds of protocol driver errors:

- **Generic errors**, which are hardware errors 0-31 and common to all protocols. Some generic errors are common to all protocols. Sometimes only the generic error is available, though often both the generic error and a specific error are given.
- **Specific errors**, which can be unique and therefore cannot be recognized by the hardware alarm system. The drivers convert their specific errors into generic errors that can be identified by the I/O server. For example, when a driver experiences an error, there is often both a protocol-specific error and a corresponding generic error.

When a hardware error occurs, Plant SCADA generates an alarm and displays the alarm on the Hardware Alarm page (in the alarm description of the hardware alarm). To see the error number, make sure you have Alarm Category 255 defined with a display format that includes {ErrDesc,15} {ErrPage,10}.

## Specific Errors

The following errors, listed in (hexadecimal) sequence, are specific to this driver. Plant SCADA displays the error number and description for common protocol-specific errors. Uncommon errors are not contained in the Plant SCADA error database, in which case Plant SCADA only displays the error number.

You might need additional information to rectify an error. This information should be detailed in the documentation that accompanied the I/O device (or network). If, after reviewing all documentation, you cannot rectify an error, contact Technical Support for this product.

Error (in hex)	Description
0x0101 Illegal function	The message function received is not allowed for the addressed slave.
0x0102 Illegal data address	The address referenced is out of range.
0x0103 Illegal data value	Invalid data is being written to the PLC.
0x0108 Memory parity error	An error has been encountered in the computer's memory.
0x28	Bad response from PLC
0x104	Device inoperative

0x105	Acknowledge
0x106	Busy – rejected
0x107	Negative Acknowledge

## MODBUSA Driver

The MODBUSA Driver supports serial communication with many I/O devices (including the Modicon 484, 584, 884, and 984 programmable controllers, and Moore Products IOExpress).

---

**Note:** The **MODBUS** and **MODBUSA** drivers are closely related. They only differ from each other in the data framing technique used: MODBUS is RTU-based, while MODBUSA is ASCII-based. You should determine which your system requires and make sure you proceed using the appropriate driver.

---

The maximum request length for the MODBUSA driver is 2000 bits.

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

#### DANGER

**DANGER** indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.

#### WARNING

**WARNING** indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.

**⚠ CAUTION**

**CAUTION** indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.

**NOTICE**

NOTICE used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.

**Please Note**

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

**Before You Begin**

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ WARNING****LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

## Supported Devices

The MODBUSA Driver supports serial communication with the following devices:

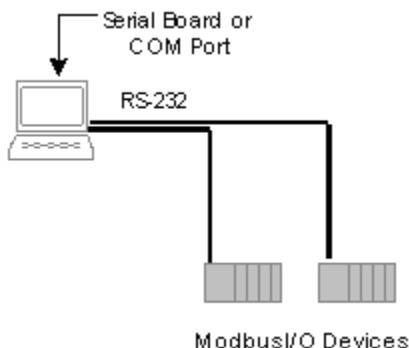
- Generic devices.

Also see [Working with Non-standard Devices](#).

### Generic Devices

Being an established industry standard, the Modbus protocol supports many I/O devices from a variety of manufacturers. The MODBUSA Driver therefore can support communication between Plant SCADA and any generic Modbus devices.

Communication is via a Modbus port on the PLC. Using this method, you can connect to multiple PLCs as in the following diagram:



The following topics provide information required to connect Plant SCADA to a generic Modbus device.

### Generic Devices - Device Address

The I/O Device Address for a PLC is its station number. Try 1 to 10.

- **1** specifies station number **1**.
- **9** specifies station number **9**.

### Generic Devices - Hardware Setup

Set your PLC to the communication setting you want. The default hardware settings for the device are as follows:

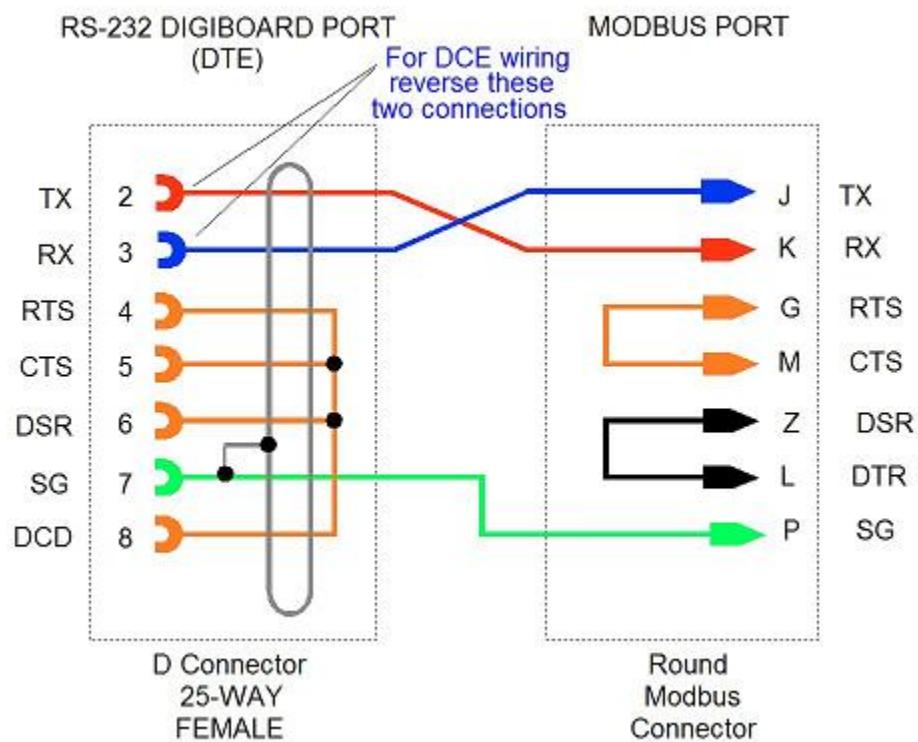
Setting	Value
Baud Rate	19200
Data Bits	8
Stop Bits	1
Parity	2 (Even)

**Note:** These settings are recommendations only. If you use the Communications Express Wizard, these default settings are configured in your project automatically, though your hardware may support other values. If you do set the baud rate, data bits, stop bits or parity to another value, you must manually set the new value(s) in your project.

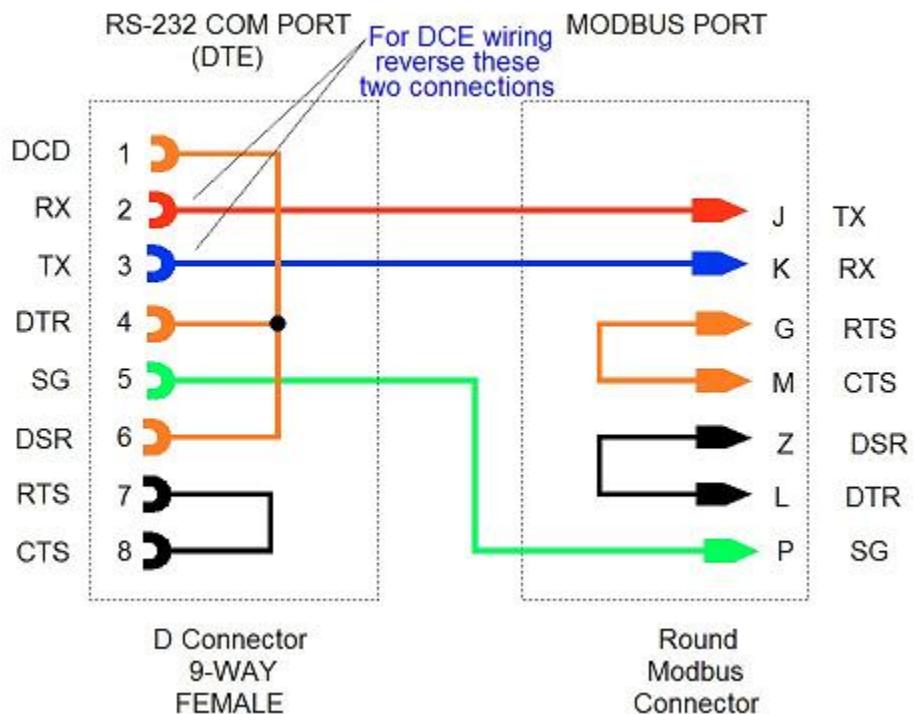
Set the following configuration options on the Modbus port on the PLC:

- Station number
- RTU mode (This is not selected if the MODBUSA protocol is being used)
- Delay time to 1

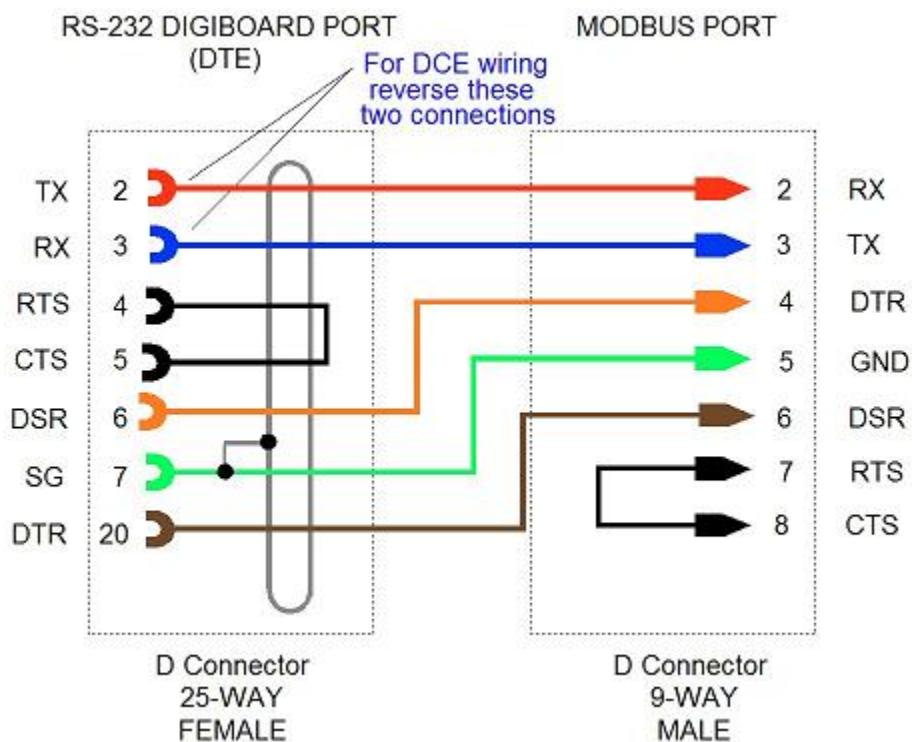
## Generic Devices - Wiring Diagram 1



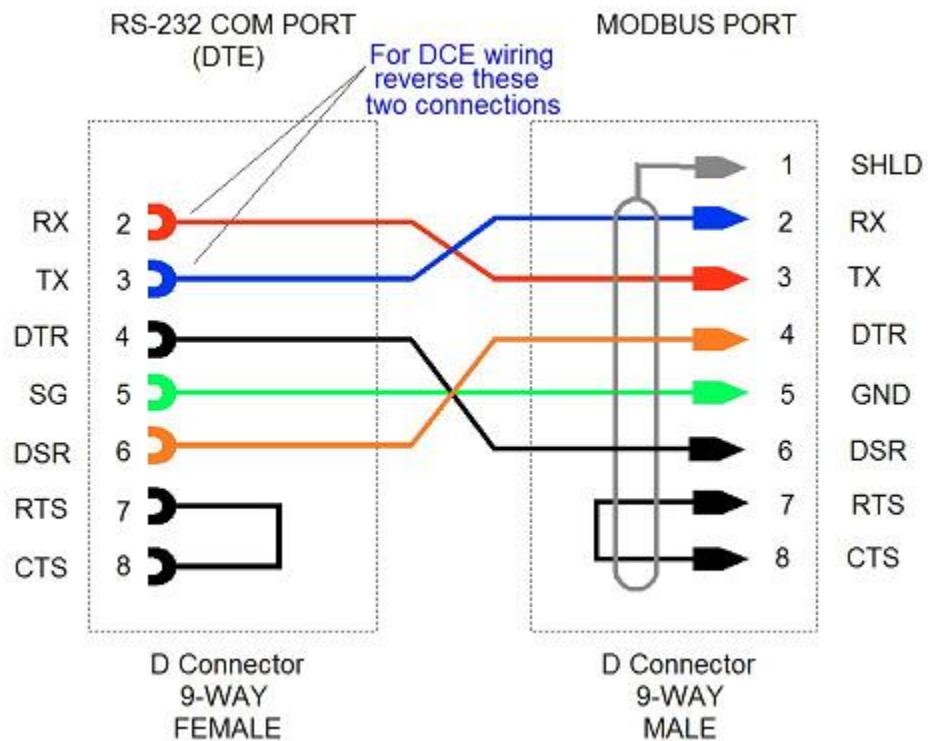
## Generic Devices - Wiring Diagram 2



## Generic Devices - Wiring Diagram 3



## Generic Devices - Wiring Diagram 4



## Generic Devices - Communications Settings

To establish communication with a device, the associated Board, Port and I/O Device need to be correctly configured in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure them, use the information outlined below.

### Boards

Typically, you would use a serial board or COM port for this communication. Refer to the instructions for setting up and using COM ports or serial boards. Or, complete the Board settings with the following information.

Field	Value
Board Type	If using a serial board or COM port, enter COMx.
Address	If using a serial board or COM port, enter 0.
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.

### Ports

Field	Value
Port Number	This value matches the COM port number. This number is defined in the Ports section of the Control Panel.
Baud Rate	This value matches the setting of the unit - 19200.
Data Bits	Enter 8.
Stop Bits	Enter 1.
Parity	This value matches the setting of the unit - EVEN_P.
Special Options	You may want to use the special options for the COMx Driver if you are using a modem (or similar) and want the driver to perform differently. For more information, see the topic <i>COMx Driver Special Options Reference</i> in the <i>Configure I/O Device Communications</i> chapter of the main help.

### I/O Devices

Field	Value
Port Number	This value matches the COM port number. This number is defined in the Ports section of the Control Panel.
Baud Rate	This value matches the setting of the unit - 19200.
Data Bits	Enter 8.
Stop Bits	Enter 1.
Parity	This value matches the setting of the unit - EVEN_P.
Special Options	You may want to use the special options for the COMx Driver if you are using a modem (or similar) and want the driver to perform differently. For more information, see the topic <i>COMx Driver Special Options Reference</i> in the <i>Configure I/O Device Communications</i> chapter of the main help.

## Generic Devices - Data Types

**Note:** The address ranges specified in the following table will be appropriate for most generic Modbus devices. However, you should check the capabilities of a device to confirm if it will support the full extent of the ranges suggested.

Data Types	Address Format	Plant SCADA Data Type
<b>DIGITAL</b>		
Output Coils	000001 to 065536	DIGITAL ( <b>Note:</b> Leading zero is required.)
Input Status	100001 to 165536	DIGITAL (Read Only)
<b>INTEGER</b>		
Input Registers	300001 to 365536	INT / LONG / STRING / REAL (Read Only)
Output Registers	400001 to 465536	INT / LONG / STRING / REAL
<b>EXTENDED REGISTERS</b>		
(6xxxxx range)	Ef:r.b <i>where:</i> f = file no. (1 - 999) r = register no. (0 - 9999) b = bit no. (1 - 16)	DIGITAL / INT / LONG / STRING / REAL

**Examples**

Data Type	DIGITAL
Address	00001
Comment	Output Coil 00001
Data Type	INT
Address	40001
Comment	Output Register 40001
Data Type	INT
Address	E1:00001
Comment	Modbus Address 600001

**Note:** LONG data types default to a simplified implementation, with a shortened range of 0 to 655,359,999. To increase the LONG data type to the full range, set the LongDataType parameter to either 1 or 3.

**Working With Non-standard Devices**

Some non-standard PLCs support the Modbus protocol. However, some of these I/O devices do not support the protocol correctly. When Plant SCADA tries to communicate with these I/O devices, you might get errors on startup or during communication.

To handle these non-standard I/O devices, the Modbus protocol has some special options to change its operation:

1. On startup, Plant SCADA tries to read some digital inputs to verify that the PLC does exist. If Plant SCADA cannot read these bits, it assumes the PLC is bad, and remains off line. Some non-standard I/O devices do not support digital bits, so this command results in an error. With the [MODBUSA]InitType parameter, you can change the type of variable the Modbus protocol tries to read on startup, to allow communication to start.
2. The standard MODBUSA driver allows reading of 2000 bits in one request. Some non-standard I/O devices do not support such large reads. You can change the maximum read size with the [MODBUSA]MaxBits parameter. (The default is 2000.)

Change the MAX\_LENGTH field for the Modbus protocol (in the \Plant SCADA\BIN\PROTDIR.DBF file) to the same value. Re-compile your project after changing this option. Be aware that if you reinstall Plant SCADA, the PROTDIR.DBF file is overwritten and you will need to repeat the change.

3. Some I/O devices that operate over radio modems also require the padding of protocol messages. The padding characters wake the radio modem to allow the rest of the message to be transmitted. Set the [MODBUSA]Pad parameter to the number of characters to pad, and the [MODBUSA]PadChar parameter to the ASCII code of the character to pad. For example, to pad 20 0xFF characters to the start of the message, set:

[MODBUSA]  
Pad=20

PadChar=255

4. In the Modbus protocol, LONG data types default to a simplified implementation, with a shortened range of 0 to 655,359,999. To increase the LONG data type to the full range, set the [MODBUSA]LongDataType parameter to either 1 or 3.

As a LONG is composed from two registers, the register order used by Plant SCADA must match the order used by the PLC. Refer to the help accompanying the [MODBUSA]LongDataType parameter to determine the appropriate value.

**Note:** The MODBUSA driver supports a number of variants of the MODBUS protocol to communicate with some non-standard PLCs. See the topic to determine which variant you should use when configuring a device.

## Protocol Variants

Many devices from different vendors support the Modbus protocol. Unfortunately, addressing structures used by these devices vary. The following information will help you understand which protocol you should select on the I/O devices dialog box to make the addresses you enter into the variables form match those entered into the programming software for your device.

You will need to know the following:

- What is the lowest register / coil number in the device?

Some devices (such as a TSX Quantum or Modicon 948) have register / coil 1 as the lowest register / coil number. Other devices (such as the TSX Premium) have register / coil 0 as the lowest register / coil number

- When addressing an individual bit from a register, what is the lowest bit number?

Some devices refer to the bits within a register as bits 0-15, others refer to them as bits 1-16.

Based on the above information you can select a protocol using the following table:

Range of bits in a register	Lowest coil / register number	Protocol
1-16	1	MODBUS (default)
1-16	0	MODBUS1
0-15	1	MODBUS2
0-15	0	MODBUS3

If you are using a PLC that supports extended registers (6xxxxx addresses in a TSX Quantum for example) then the following table should be used:

Range of bits in a register	Lowest coil / register number	Protocol
1-16	1	MODBUS10
1-16	0	MODBUS11
0-15	1	MODBUS12
0-15	0	MODBUS13

Two special protocols exist for use with the TSX Quantum and TSX Premium PLC ranges. These protocol options will select the correct addressing modes, select certain INI values and other settings to provide compatibility with these devices without the need to set any other special parameters.

These protocols also allow you to access memory digitals and memory words using %M7 or %MW3 or %MW4.6 style addressing. Select a protocol as follows:

PLC Model	Protocol
TSX Quantum	MODBUS20
TSX Premium	MODBUS30

## Customizing a Project using Citect.ini Parameters

Citect.ini parameters are used to tune the performance of the Plant SCADA MODBUSA driver and to perform runtime maintenance diagnostics.

You can customize the way Plant SCADA communicates with the Modbus system (and even individual PLCs) by creating or editing the [MODBUSA] section of the citect.ini file for your project.

There are some common Plant SCADA driver settings as well as custom MODBUSA driver settings.

When Plant SCADA starts at runtime, it reads configuration values from the citect.ini file that is stored locally. Therefore, any MODBUSA configuration settings must be included in the citect.ini file located on the computer acting as the I/O server to the Modbus system.

By default, Plant SCADA looks for the citect.ini file in the Plant SCADA project \bin directory. If it can't find the file there, it searches the default Windows directory.

### MODBUSA Driver Parameters

These parameters default to a value which have shown to be suitable for the majority of circumstances.

#### **WARNING**

##### **UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

With certain parameters, the MODBUSA driver can apply different initialization parameter values to specific I/O devices or groups of I/O devices. For details see Device/group-specific parameters.

### [MODBUSA]Block

A value (bytes) used by the I/O server to determine if two or more packets can be blocked into one data request before being sent to the I/O device. For example, if you set the value to 10 and the I/O server receives two simultaneous data requests (one for byte 3 and another for byte 8) the two requests are blocked into a single physical data request packet. This single request packet is then sent to the I/O device, saving on bandwidth and

processing.

**Allowable Values:** 5 to 256

**Default Value:** 50

## [MODBUSA.<Port\_Name>.<IODevice\_Name>]Broadcast

Sends a write request to a specified device on a Modbus network.

---

**Note:** This parameter can only be used as a device-specific parameter, for example, [MODBUSA.<Port\_Name>.<IODevice\_Name>]Broadcast=1. It should not be used globally. For more details, see [Device/Group-specific Parameters](#).

---

To use this feature:

Define an I/O device with address 0 (zero) for the Channel of interest.

Define the value for the BroadcastDelay parameter in the Citect.ini file (default is 50ms). This parameter defines the amount of time the driver will delay after submitting a broadcast request before it will send another request to the PLCs.

Define tags (Coils and Registers) associated with this I/O device.

Implement a Plant SCADA application that requests writes to the tags (via Cicode, application display, and so on).

Requests for Variable Tag READs to the I/O device with the address zero (0) will return a Driver Error of Unknown Command (driver error 15 decimal). Requests for Variable Tag WRITEs to the I/O device with the address zero (0) will be issued to ALL controllers on the SAME Channel (port) as the I/O device with the assigned address of zero. The Modbus protocol does not provide a response to broadcast requests so inspect the appropriate coils or registers in each controller to observe completion of the request.

**Allowable Values:**

0 = Broadcast disabled

1 = write request will be issued

**Default Value:** 1

## [MODBUSA]BroadcastDelay

Defines the amount of time the driver will delay after submitting a broadcast request before sending another request to the PLCs.

**Allowable Values:** 0 to 300 (milliseconds)

**Default Value:** 50

---

**Note:** This parameter is only available at the MODBUSA level, i.e. [MODBUSA]

---

## [MODBUSA]Delay

The period (in milliseconds) to wait between receiving a response and sending the next command.

**Allowable Values:** 0 to 32767 (milliseconds)

**Default Value:** 0

## [MODBUS]DoCRC

Enables or disables the Modbus Cyclic Redundancy Check (CRC), a communications check. The MODBUSA driver does a CRC on incoming data and provides a CRC remainder in outgoing data blocks. In some limited cases, such as the testing of "slave" drivers, it may be necessary to disable the CRC.

**Allowable Values:** 1 or 0, where:

1 = enable CRC

0 = disable CRC

**Default Value:** 1

The CRC should remain "enabled" under normal circumstances. If the CRC is disabled, no checks are performed on incoming data and the CRC field in the data transmitted by the driver remains 0. This will mean the driver cannot detect lack off communication between itself and the I/O device.

## [MODBUS]FailOnBadData

Used by the MODBUSA driver to determine whether or not to force the display of good data within a block read which contains bad tags during tag read.

**Allowable Values:** 1 or 0, where:

1 = A block read which contains a bad tag would result in the whole block returning #BAD

0 = The MODBUSA driver will not return "#BAD" for a block of data if at least one valid value is in the read block.

**Default Value:** 1

## [MODBUS]FloatMode

---

**Note:** This parameter is not applicable if you are using the protocol variant MODBUS20 or MODBUS30 to communicate with a device. Under these circumstances, a value of 0 (zero) will be hardcoded for this parameter.

This INI parameter sets the byte order for floating point values (the MODBUSA driver supports floating point values). Some systems expect to use a different byte order for their floating point data.

**Allowable Values:** 0 to 3, where:

0 - Byte order = 1 0 3 2

1 - Byte order = 3 2 1 0

2 - Byte order = 0 1 2 3

3 - Byte order = 2 3 0 1

**Default Value:** 0

With this parameter, you can set different values for specific I/O devices or groups of I/O devices. See [Device/Group-specific Parameters](#).

## [MODBUS]ForceMultiCoilsOnly

Forces the use of only function code 15 (multiple coils) for coil writes.

**Allowable Values:**

0 - (function code 5 and 15)

1 - (function code 15 only)

**Default Value:** 0

### [MODBUS]InitType

The type of variable the MODBUSA protocol tries to read on startup, to allow communication to start.

**Allowable Values:** 1 to 4.

InitType	Address	Variable
1	00001 - 00017	Output status
2	10001 - 10017	Input status
3	40001	Output registers
4	30001	Input registers

**Default Value:** 2

With this parameter, you can set different values for specific I/O devices or groups of I/O devices. See [Device/Group-specific parameters](#).

### [MODBUS]InitUnitAddress

The MODBUSA driver reads the citect.ini file to determine the correct unit address for initialization of a PLC. InitUnitAddress is the parameter used to set the unit address.

**Allowable Values:** 0 to 65535

**Default Value:** 0

With this parameter, you can set different values for specific I/O devices or groups of I/O devices. See [Device/Group-specific Parameters](#).

### [MODBUS]LongDataType

**Note:** This parameter is not applicable if you are using the protocol variant MODBUS20 or MODBUS30 to communicate with a device. Under these circumstances, a value of 3 will be hardcoded for this parameter.

In the Modbus protocol, LONG data types default to a simplified implementation, with a shortened range of 0 to 99,999,999 - mode 0. Mode 2 has the same range as mode 0, but with the register order swapped. Mode 1 supports the complete LONG range of -2,147,483,648 to +2,147,483,647. Mode 3 has the same range as mode 1, but with the register order swapped.

**Allowable Values:** 0 to 3, where:

- 0 = 10000 x low register + high register
- 1 = 65536 x low register + high register
- 2 = 10000 x high register + low register
- 3 = 65536 x high register + low register

**Default Value:** 3

With this parameter, you can set different values for specific I/O devices or groups of I/O devices. See [Device/Group-specific Parameters](#).

## [MODBUS]MaxBits

**Note:** This parameter is not applicable if you are using the protocol variant MODBUS20 or MODBUS30 to communicate with a device. If MODBUS20 is being used, a value of 1904 will be hardcoded for this parameter. If MODBUS30 is being used, a value of 1024 will be hardcoded.

The maximum read size in one request. Decrease the value for non-standard I/O devices that do not support large reads.

**Allowable Values:** 8 to 32767

**Default Value:** 1904

## [MODBUS]MaxBitsExt

**Note:** This parameter is not applicable if you are using the protocol variant MODBUS20 or MODBUS30 to communicate with a device. If MODBUS20 is being used, a value of 1904 will be hardcoded for this parameter. If MODBUS30 is being used, a value of 1024 will be hardcoded.

This INI parameter sets the maximum bits blocked for the Quantum PLCs "Ex:y[.z]" address type.

**Allowable Values:** 8 to 32767

**Default Value:** 1904

## [MODBUS]MaxPending

The maximum number of pending commands that the driver holds ready for immediate execution.

**Allowable Values:** 1 to 32

**Default Value:** 2

## [MODBUS]Pad

The number of characters with which to pad protocol messages. The padding characters wake the radio modem to allow the rest of the message to be transmitted. A value of 20 is recommended for non-standard I/O devices that operate over radio modems.

**Allowable Values:** 0 to 65535

**Default Value:** 0

## [MODBUS]PadChar

The ASCII code of the padding character with which to pad protocol messages. The padding characters wake the radio modem to allow the rest of the message to be transmitted.

**Allowable Values:** 0 to 255

**Default Value:** 255

## [MODBUS]PollTime

The interrupt or polling service time (in milliseconds). Setting the polling time to 0 puts the driver in interrupt

mode.

**Allowable Values:** 0 to 300 (milliseconds)

**Default Value:** 0

## [MODBUSA]PresetMultiRegistersOnly

Forces the use of only function code 16 (multiple registers) for register writes.

**Allowable Values:** 0 or 1:

0 - (function code 6 and 16)

1 - (function code 16 only)

**Default Value:** 0

## [MODBUSA]RegisterBitReverse

---

**Note:** This parameter is not applicable if you are using the protocol variant MODBUS20 or MODBUS30 to communicate with a device. Under these circumstances, a value of 1 will be hardcoded for this parameter.

This INI parameter allows reverse interpretation of Most Significant Bit (MSB) and Least Significant Bit (LSB) in words. The Modbus specification defines bit 1 of a word as the MSB, and bit16 of the word as the LSB. Some devices use a different order, causing bit 1 of a word to be the LSB and bit 16 to be the MSB.

**Allowable Values:** 0 or 1, where:

1 - Bit 1= LSB, Bit 16 = MSB

0 - Bit 1 = MSB, Bit 16 = LSB

**Default Value:** 1

With this parameter, you can set different values for specific I/O devices or groups of I/O devices. See [Device/Group-specific Parameters](#).

## [MODBUSA]Retry

The number of times to retry a command after a timeout.

**Allowable Values:** 0 to 8

**Default Value:** 0

## [MODBUSA]SendBCDSwap

Reverses the byte order for BCDs on writes (the MODBUSA driver supports BCDs). Some systems expect to use a different byte order for their data; in the case of BCDs, this causes "1234" to be written to the device as "3412".

**Allowable Values:** 0 or 1, where:

1 = 1 2 3 4

0 = 3 4 1 2

**Default Value:** 1

With this parameter, you can set different values for specific I/O devices or groups of I/O devices. See [Device/Group-specific Parameters](#).

**[MODBUSA]Status**

Sometimes you can read data from a device even though the processor module is not running. In view of this, it can be useful to detect this situation by monitoring a continuously changing register or a digital. If the register is unable to change in a given period or the digital becomes zero, the processor can be assumed to have stopped running.

If enabled, the MODBUSA driver will check on startup and approximately every watchtime the status of a user specified variable in the PLC is changing. If a digital is specified, the value should be on, otherwise the unit is put offline. If an analog is specified, on startup two consecutive reads with an interval of 1000ms (specified by [MODBUSA]Timeout) should return different values before the unit can be put online. At every watchtime, if the digital is off or the analog data does not change from the last value read, the driver returns an error and the unit is put offline.

**Allowable Values:**

RawType

BitWidth

UnitType

UnitAddress

UnitAddress

where:

RawType	BitWidth	UnitType	UnitAddress	UnitCount
0 = Digital	1 = Digital	See variable specification .dbf files	The item number or bit number	1 = analog 16 = digital
1 = Int	16 = Int			
4 = Long	32 = Long			
8 = Byte	8 = Byte			

**Default Value:**

Zero length string

**Examples**

To specify Modbus address 40001 as the analog tag variable which must change to indicate the device is online, use:

```
[MODBUSA]
status=1,16,3,0,1
```

To use Modbus address 00001 as the digital tag variable which must remain set to 1 to indicate the device is online, then use:

```
[MODBUSA]
status=0,1,1,0,16
```

## [MODBUSA]StringReverse

Reverse byte order for strings. The MODBUSA driver supports strings. Some systems expect to use a different byte order for their data, which in the case of string variables, causes "ABCD" to appear as "BADC".

**Allowable Values:** 0 or 1

**Default Value:** 0 (do not reverse byte order)

With this parameter, you can set different values for specific I/O devices or groups of I/O devices. See [Device/Group-specific Parameters](#).

## [MODBUSA]Timeout

Specifies how many milliseconds to wait for a response before displaying an error message.

**Allowable Values:** 0 to 32000 (milliseconds)

**Default Value:** 3000

**Note:** This parameter is only available at the MODBUSA Portlevel, i.e. [MODBUSA.Port] but values are overridden by the default values in the syslog and Kernel page.

## [MODBUSA]WatchTime

The frequency (in seconds) that the driver uses to check the communications link to the I/O device.

**Allowable Values:** 0 to 128 (seconds)

**Default Value:** 30

## Device/group-specific parameters

The MODBUSA driver can apply different initialization parameter values to specific I/O devices or groups of I/O devices. This means the user can specify:

- Global parameters that apply to all devices.
- Channel- (port-) level parameters that apply to all devices on the specified port.
- Group-level parameters that apply to all devices in a specified group.
- Device-level parameters that apply only to the specified device.

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

This feature can be implemented in the citect.ini for the following [MODBUSA Parameters](#):

- Broadcast

- BroadcastDelay
- FloatMode
- ForceMultiCoilsOnly
- InitType
- InitUnitAddress
- LongDataType
- PresetMultiRegistersOnly
- RegisterBitReverse
- SendBCDSwap
- StringReverse

To set parameters for a particular port, group, or device, you must create a new section in the citect.ini file. Label it with the driver name followed by a period (.) character and the name of the particular port, group, or device you want to specify the parameter setting for.

For example:

- [MODBUS.<Port\_Name>]: applies the parameter settings to the specified port.
- [MODBUS.<Group\_Name>]: applies the parameter settings to the specified group.
- [MODBUS.<Port\_Name>.<IODevice\_Name>]: applies to the specified device.

Any parameters you then define in the following section of the citect.ini file relate only to the specified device or device group.

#### Example

The following citect ini file format is an example of how the InitType parameter could be specified differently for different I/O devices communicating using the Modbus protocol.

Assume that two ports are used: PORT1 and PORT2.

PORT1 has three I/O devices attached:

- DEV1A DEV1B EV1C

PORT2 also has three devices:

- DEV2A DEV2B DEV2C

Assume that the user has specified that DEV1C and DEV2C belong to GROUPZ. The citect.ini file contains the following entries:

```
[MODBUS]
InitType=1
[MODBUS.PORT1]
InitType=2
[MODBUS.PORT2]
InitType=2
[MODBUS.GROUPZ]
InitType=3
[MODBUS.PORT1.DEV1A]
```

InitType=1  
[MODBUS.PORT2.DEV2B]  
InitType=4

The resultant InitType for the I/O devices will be as follows:

DEV1A:	1	as a result of [MODBUS.PORT1.DEV1A]
DEV1B:	2	as a result of [MODBUS.PORT1]
DEV1C:	3	as a result of [MODBUS.GROUPZ]
DEV2A:	2	as a result of [MODBUS.PORT2]
DEV2B:	4	as a result of [MODBUS.PORT2.DEV2B]
DEV2C:	3	as a result of [MODBUS.GROUPZ]

As the above example shows, there is a hierarchy that determines the outcome of such settings. In simple terms, specific parameter settings overwrite general level settings. Therefore, parameters written in the scope of I/O devices will overwrite those set for groups; parameters set for groups will overwrite global settings, and so on.

## MODBUSA Driver Errors

MODBUSA has two kinds of protocol driver errors:

- **Generic errors**, which are hardware errors 0-31 and common to all protocols. Some generic errors are common to all protocols. Sometimes only the generic error is available, though often both the generic error and a specific error are given.
- **Specific errors**, which can be unique and therefore cannot be recognized by the hardware alarm system. The drivers convert their specific errors into generic errors that can be identified by the I/O server. For example, when a driver experiences an error, there is often both a protocol-specific error and a corresponding generic error.

When a hardware error occurs, Plant SCADA generates an alarm and displays the alarm on the Hardware Alarm page (in the alarm description of the hardware alarm). To see the error number, make sure you have Alarm Category 255 defined with a display format that includes {ErrDesc,15} {ErrPage,10}.

## Specific Errors

The following errors, listed in (hexadecimal) sequence, are specific to this driver. Plant SCADA displays the error number and description for common protocol-specific errors. Uncommon errors are not contained in the Plant SCADA error database, in which case Plant SCADA only displays the error number.

You might need additional information to rectify an error. This information should be detailed in the documentation that accompanied the I/O device (or network). If, after reviewing all documentation, you cannot rectify an error, contact Technical Support for this product.

Error (in hex)	Description
0x0101 Illegal function	The message function received is not allowed for the addressed slave.
0x0102 Illegal data address	The address referenced is out of range.
0x0103 Illegal data value	Invalid data is being written to the PLC.
0x0108 Memory parity error	An error has been encountered in the computer's memory.
0x28	Bad response from PLC
0x104	Device inoperative
0x105	Acknowledge
0x106	Busy – rejected
0x107	Negative Acknowledge

## MODNET Driver

The MODNET driver supports TCP/IP communication with Modicon™ TSX Quantum™ and Premium™ PLCs, as well as a number of TCP-compatible generic MODBUS devices.

The maximum request length for the MODNET driver is 1920.

### See Also

[Supported Devices](#)

[MODNET Protocol Variants](#)

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.
<b>⚠ DANGER</b>	<b>DANGER</b> indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.
<b>⚠ WARNING</b>	<b>WARNING</b> indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.
<b>⚠ CAUTION</b>	<b>CAUTION</b> indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.
<b>NOTICE</b>	NOTICE used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.

## Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

## Before You Begin

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ WARNING****LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

## Supported Devices

The MODNET driver supports MODBUS communications via TCP/IP for the following devices:

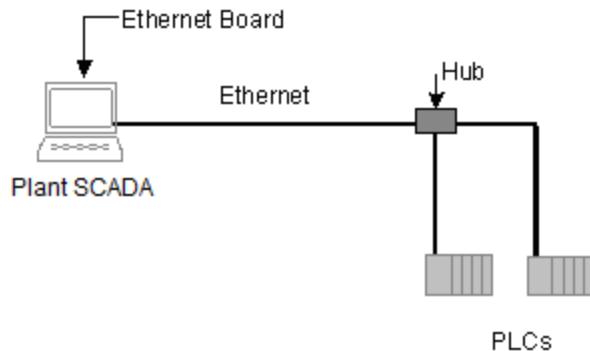
- Generic MODBUS TCP/IP Devices
- Modicon™ M340, TSX Quantum™ and Premium™ PLCs.

---

**Note:** To support a wide range of Modbus/TCP devices, there are a number of variants of the MODNET protocol. See the topic [MODNET Protocol Variants](#) to determine which variant you should use when configuring a device.

### Generic MODBUS TCP/IP Devices

The MODNET Driver supports TCP/IP communication to devices supporting MODBUS/TCP over Ethernet. Using this method you can connect to single or multiple PLCs as in the following diagram:



**Note:** The same Ethernet card that is being used for Plant SCADA communication can be used for PLC communication though this may lower performance.

## See Also

- [Generic MODBUS TCP/IP Devices - Device Address](#)
- [Generic MODBUS TCP/IP Devices - Hardware Setup](#)
- [Generic MODBUS TCP/IP Devices - Communication Settings](#)
- [Generic MODBUS TCP/IP Devices - Data Types](#)

### Generic MODBUS TCP/IP Devices - Device Address

The information you enter here will be placed in the Special Options field of the Port settings, with the following format:

-la -Pn -T

where:

a = the IP address in standard Internet dot format. (For example 192.9.2.60)

n = the destination Port number. Often one physical port has several virtual ports, used for different purposes. The default is 502.

-T -- forces the driver to use TCP, rather than UDP (-U).

See [Generic MODBUS TCP/IP Devices](#) for more information about this setup.

### Generic MODBUS TCP/IP Devices - Hardware Setup

## Plant SCADA Computer Setup

It is recommended that you setup your specific Ethernet card as required and confirm basic network communications before proceeding with the communications configuration for the MODNET driver. Refer to the documentation accompanying your hardware for instruction.

### Generic MODBUS TCP/IP Devices - Communication Settings

The following tables show the settings for communication with Generic MODBUS TCP/IP Devices. To establish communication with a device, configure the Boards, Ports and I/O Devices settings in Plant SCADA Studio's

**Topology** activity correctly. If you use the Express Communications Wizard to connect to a device, these will be automatically configured for you. However, if you need to configure them manually, or verify that the fields are correct, use the settings outlined below.

## Boards

Field	Value
Board Type	Enter TCPIP .
Address	Enter 0 .
I/O Port	Leave this field blank .
Interrupt	Leave this field blank.
Special Options	Leave this field blank.

## Ports

Field	Value
Port Number	A valid unique number.
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank.
Special Options	<p>Enter the destination IP address of the PLC. Use the following format:</p> <p><b>-Ia -Pn -T</b></p> <p>where:</p> <p><i>a</i> = the IP address in standard Internet dot format. (For example 192.9.2.60)</p> <p><i>n</i> = the destination Port number. Often one physical port has several virtual ports, used for different purposes. Use this option only if you want to override the default of 502.</p> <p><b>-T</b> = forces the driver to use TCP, rather than UDP (-U).</p>

## I/O Devices

Field	Value
I/O Device Address	If the connection to the I/O device is point-to-point using TCP/IP, leave this field blank to assign a default value of 0. Or, you can specify the slave address if the device requires it.  If the connection to the I/O device is through MODBUS/TCP to MODBUS serial line gateway, use the slave address of I/O device.  <b>Note:</b> To enable broadcast through MODBUS/TCP to MODBUS serial line gateway, an I/O device should be created with address value 0. All the broadcast requests should be sent to this I/O device. Refer to the Broadcast and BroadcastDelay parameters for details (see <a href="#">MODNET Driver Parameters</a> ).
I/O Device Protocol	Enter the MODNET protocol variant. (See <a href="#">MODNET Protocol Variants</a> to determine which variant is required.)

### Generic MODBUS TCP/IP Devices - Data Types

**Note:** The address ranges specified in the following table will be appropriate for most generic Modbus devices. However, you should check the capabilities of a device to confirm if it will support the extent of the ranges suggested.

For information on array support, see [Generic MODBUS TCP/IP Devices - Array Support](#).

Data Types	Valid Address Range	Address Format	Plant SCADA Data Type
Output Coils	000001 - 065536	<valid address> (e.g. 001000)	DIGITAL
Input Status	100001 - 165536	<valid address> (e.g. 103000)	DIGITAL
Input Registers	300001 - 365536	<valid address> (e.g. 301350)	INT/ REAL/ LONG/ BCD/ LONGBCD
Input Registers (DIGITAL)	300001 - 365536	<valid address>.b where b is the bit number between 1 and 16 inclusive	DIGITAL

Data Types	Valid Address Range	Address Format	Plant SCADA Data Type
		(e.g. 302300.2)	
Output Registers	400001 - 465536	<valid address> (e.g. 401190)	INT/ REAL/ LONG/ BCD/ LONGBCD
Output Registers (DIGITAL)	400001 - 465536	<valid address>.b where b is the bit number between 1 and 16 inclusive (e.g. 40001.4)	DIGITAL
Extended registers	-	<b>EF:R</b> where: <b>E</b> denotes the extended register. <b>F</b> is the file number between 1 and 16 inclusive <b>R</b> is the register between 0 and 9999 inclusive	INT/ REAL/ LONG/ BCD/ LONGBCD/ STRING
Individual Output Coil Register	000001 – 065536	G<valid address> (e.g. G 001000)	DIGITAL
Individual Input Coil Status	100001 – 165536	G<valid address> (e.g. G103000)	DIGITAL
Individual Input Register	300001 – 365536	G<valid address> (e.g. G 301350)	INT / REAL / LONG / BCD / LONGBCD / STRING
Individual Output Register	400001 – 465536	G<valid address> (e.g. G401190)	INT / REAL / LONG / BCD / LONGBCD / STRING
Individual Extended Register		G<EF:R> where: <b>E</b> denotes the extended register. <b>F</b> is the file number between 1 and 16 inclusive <b>R</b> is the register between 0 and 9999 inclusive	INT/ REAL / LONG / BCD / LONGBCD / STRING

**Note:** When writing DIGITAL variables to Output registers, the individual bits in a register (1..16) can be written to with the format **n .BitPosition** (e.g. 40001.1). Writing to sequential bits requires individual write commands. The user can specify the bit order direction in the INI parameter [MODNET]RegisterBitReverse.

#### EXAMPLES:

Data Type	DIGITAL
Address	000006
Comment	Output Coil
Data Type	LONGBCD
Address	300019
Comment	Input Register

## Generic MODBUS TCP/IP Devices - Array Support

Plant SCADA Data Type	Address	Array size
DIGITAL	[Address][Array size]	1-2032
INT	[Address][Array size]	1-127
LONG	[Address][Array size]	1-63
REAL	[Address][Array size]	1-63
BCD	[Address][Array size]	1-127
LONGBCD	[Address][Array size]	1-63
STRING	[Address][Array size]	1-127 (2-254 characters)

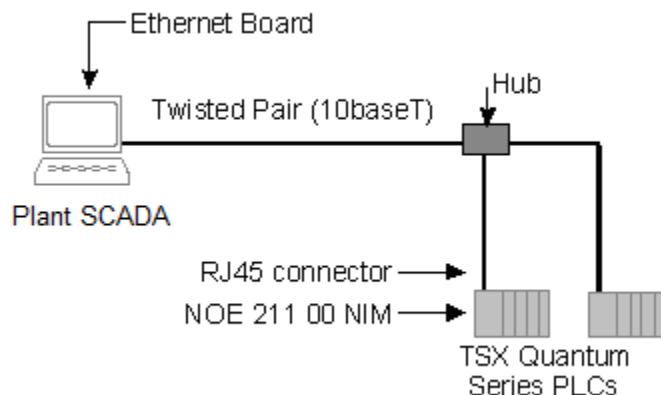
### EXAMPLES:

Data Type	DIGITAL
Address	000006[2000]
Data Type	LONGBCD
Address	300019[5]
Data Type	INT
Address	400001[1]
Data Type	REAL
Address	400100[12]

Data Type	STRING
Address	400001[2]

## M340, TSX Quantum™ and Premium™ PLCs

The MODNET Driver supports TCP/IP method of communication to the Modicon M340, TSX Quantum and Premium PLCs. Using this method you can connect to single or multiple PLCs as in the following diagram:



### Note:

- 1) The same Ethernet card that is being used for Plant SCADA communication can be used for PLC communication though this may lower performance.
- 2) The hub is necessary even if connecting to only one PLC.

## See Also

- [M340, TSX Quantum™ and Premium™ PLCs - Device Address](#)
- [M340, TSX Quantum™ and Premium™ PLCs - Hardware Setup](#)
- [M340, TSX Quantum™ and Premium™ PLCs - Communication Settings](#)
- [M340, TSX Quantum™ and Premium™ PLCs - Data Types](#)

## M340, TSX Quantum and Premium PLCs - Device Address

The information you enter here will be placed in the Special Options field of the Ports form, with the following format:

-la -Pn -T

where:

a = the IP address in standard Internet dot format. (For example 192.9.2.60)

n = the destination Port number. Often one physical port has several virtual ports, used for different purposes. The default is 502.

-T = forces the driver to use TCP, rather than UDP (-U).

See the topic [M340, TSX Quantum™ and Premium™ PLCs](#) for more information about this setup.

## M340, TSX Quantum and Premium PLCs - Hardware Setup

### Plant SCADA Computer Setup

It is recommended that you setup your specific Ethernet card as required and confirm basic network communications before proceeding with the communications configuration for the MODNET driver. Refer to the documentation accompanying your hardware for instruction.

## M340, TSX Quantum and Premium PLCs - Communication Settings

The following tables show the settings for communication with TSX Quantum™ and Premium™ PLCs. To establish communication with a device, configure the Boards, Ports and I/O Devices settings in Plant SCADA Studio's **Topology** activity. If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to configure them manually, or verify that the fields are correct, use the settings outlined below.

### Boards

Field	Value
Board Type	Enter TCPIP.
Address	Enter 0.
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.

### Ports

Field	Value
Port Number	A valid unique number.
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank.
Special Options	Enter the destination IP address of the PLC. Use the following format:

Field	Value
	<p><b>-Ia -Pn -T</b></p> <p>where:</p> <p><i>a</i> = the IP address in standard Internet dot format. (For example 192.9.2.60)</p> <p><i>n</i> = the destination Port number. Often one physical port has several virtual ports, used for different purposes. Use this option only if you want to override the default of 502.</p> <p><b>-T</b> = forces the driver to use TCP, rather than UDP (-U).</p>

## I/O Device Settings

Field	Value
I/O Device Address	Leave this field blank. A value of 0 is assigned by default.
I/O Device Protocol	Enter the MODNET protocol variant. (See <a href="#">MODNET Protocol Variants</a> to determine which variant is required.)

## M340, TSX Quantum and Premium PLCs - Data Types

Data Types	Valid Address Range	Address Format	Plant SCADA Data Type
Output Coils	000001 - 065536	<valid address> (e.g. 01000)	DIGITAL
Output Coils (Memory)	000001 – 065536	%M<valid address> (e.g. %M7)	DIGITAL
Input Status	100001 - 165536	<valid address> (e.g. 11001)	DIGITAL
Input Registers	300001 - 365536	<valid address> (e.g. 31350)	INT/ REAL/ LONG/ BCD/ LONGBCD
Input Registers (DIGITAL)	300001 - 365536	<valid address>.b where <b>b</b> is the bit number between 1 and 16 inclusive	DIGITAL

Data Types	Valid Address Range	Address Format	Plant SCADA Data Type
		(e.g. 32300.2)	
Output Holding Registers (ANALOG)	400001 - 465536	<valid address> (e.g. 41190)	INT/ REAL/ LONG/ BCD/ LONGBCD
Output Registers (Memory)	000001 – 065536	%MW<valid address> (e.g. %MW3)	INT/ REAL/ LONG/ BCD/ LONGBCD
Output Holding Registers (DIGITAL)	400001 - 465536	<valid address>.b where b is the bit number between 1 and 16 inclusive (e.g. 40001.4)	DIGITAL
Output Registers (DIGITAL) (Memory)	000001 – 065536	%MW<valid address>.b where b is the bit number between 1 and 16 inclusive (e.g. %MW4.6)	DIGITAL
Extended registers		<b>EF:R</b> where: <b>E</b> denotes the extended register. <b>F</b> is the file number between 1 and 16 inclusive. <b>R</b> is the register between 0 and 9999 inclusive.	INT/ REAL/ LONG/ BCD/ LONGBCD/ STRING
Individual Output Coil Register	000001 – 065536	G<valid address> (e.g. G001000)	DIGITAL
Individual Input Coil Status	100001 – 165536	G<valid address> (e.g. G103000)	DIGITAL
Individual Input Register	300001 – 365536	G<valid address> (e.g. G301350)	INT / REAL / LONG / BCD / LONGBCD / STRING
Individual Output Register	400001 – 465536	G<valid address> (e.g. G401190)	INT / REAL / LONG / BCD / LONGBCD / STRING
Individual Extended Register		G<EF:R> where: <b>E</b> denotes the extended register. <b>F</b> is the file number	INT/ REAL / LONG / BCD / LONGBCD / STRING

Data Types	Valid Address Range	Address Format	Plant SCADA Data Type
		between 1 and 16 inclusive <b>R</b> is the register between 0 and 9999 inclusive	

**Note:** When writing DIGITAL variables to output registers, the individual bits in a register (1..16) can be written to with the format **n.BitPosition** (e.g. 40001.1). Writing to sequential bits requires individual write commands. The user can specify the bit order direction in the INI parameter [MODNET]RegisterBitReverse.

#### EXAMPLES

Data Type	DIGITAL
Address	00036
Comment	Output Coil
Data Type	LONGBCD
Address	30019
Comment	Input Register

## MODNET Protocol Variants

Many devices from different vendors support the Modbus/TCP protocol. Unfortunately addressing structures used by these devices vary. The following information will help you understand which protocol you should select on the I/O devices dialog box to make the addresses you enter into the variables form match those entered into the programming software for your device.

You will need to know the following:

- **What is the lowest register / coil number in the device?**

Some devices (such as a TSX Quantum or Modicon 948) have register / coil 1 as the lowest register / coil number. Other devices (such as the TSX Premium) have register / coil 0 as the lowest register / coil number

- **When addressing an individual bit from a register, what is the lowest bit number?**

Some devices refer to the bits within a register as bits 0-15, others refer to them as bits 1-16.

Based on the above information you can select a protocol using the following table:

Range of bits in a register	Lowest coil / register number	Protocol
1-16	1	MODNET (default)
1-16	0	MODNET1
0-15	1	MODNET2

0-15	0	MODNET3
------	---	---------

If you are using a PLC that supports extended registers (6xxxxx addresses in a TSX Quantum for example) then the following table should be used:

Range of bits in a register	Lowest coil / register number	Protocol
1-16	1	MODNET10
1-16	0	MODNET11
0-15	1	MODNET12
0-15	0	MODNET13

Two special protocols exist for use with the TSX Quantum and TSX Premium PLC ranges. These protocol options will select the correct addressing modes, select certain INI values and other settings to provide compatibility with these devices without the need to set any other special parameters.

These protocols also allow you to access memory digitals and memory words using %M7 or %MW3 or %MW4.6 style addressing. Select a protocol as follows:

PLC Model	Protocol
TSX Quantum	MODNET20
TSX Premium	MODNET30
M340	MODNET30

## MODNET Driver Parameters

### **WARNING**

#### UNINTENDED EQUIPMENT OPERATION

- Do not under any circumstances change or remove any of the undocumented citect.ini parameters.
- Before deleting sections of the citect.ini file, confirm that no undocumented parameters will be deleted.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

[MODNET] Parameter name	Description	Allowable values	Default value
Broadcast	Enables or disables broadcast through a gateway.	1 – Write request will be issued. 0 – Broadcast disabled.	0

[MODNET] Parameter name	Description	Allowable values	Default value
	<p>The 'broadcast' mechanism is designed to allow an I/O Device configured with address '0' to come online and be used to send write requests to the Modbus network through a gateway. Some devices will interpret this address as a 'broadcast' address and accept the write request. Consequently, there will be no acknowledgement of the message.</p> <p>The ini parameter is configured as:</p> <p>[MODNET.&lt;port_name&gt;.&lt;iodevice_name&gt;] Broadcast</p>		
BroadcastDelay	Time delay before replying to broadcast write requests in milliseconds.	1-65535	50
ConnTimeout	Timeout period for channel initialization in milliseconds.	1 - 65535 (milliseconds)	15000
Debug	<p>Allows extra debug information to be displayed in the kernel window and logged into the syslog.dat file.</p> <p><b>Note:</b> This parameter is only applicable if DebugStr parameter is configured.</p>	<p>1 - turn on the extra debug information option 0 - turn off the extra debug information option</p>	0 (zero)
DebugStr	Enables logging of debug information to be displayed in the kernel window and logged into the syslog.dat file.	<p>&lt;port name&gt; ALL – Enables logging for a specific port. * ALL –</p>	Disabled

[MODNET] Parameter name	Description	Allowable values	Default value
	This ini parameter is configured as: [MODNET] DebugStr = <port name> ALL	Enables logging for all the ports configured.	
Delay	The period (in milliseconds) to wait between receiving a response and sending the next command.	0 to 300 (milliseconds)	0 (zero)
FailOnBadData	Controls error reporting when BCD or LONG values received from the device are outside the valid range for that type.	1 – Reports error 0 – No error reported	1
FloatMode	<p>Specifies the order of bytes in REAL variable types.</p> <p><b>Note:</b> This parameter is not applicable if you are using the <a href="#">MODNET Protocol Variants</a> MODNET20 or MODNET30 to communicate with a device. Under these circumstances, a value of 0 (zero) will be hardcoded for this parameter.</p> <p>With this parameter, you can set different values for specific I/O devices or groups of I/O devices.</p> <p>See <a href="#">Device/Group-specific Parameters</a>.</p>	0: Order of bytes = 1 0 3 2 1: Order of bytes = 3 2 1 0 2: Order of bytes = 0 1 2 3 3: Order of bytes = 2 3 0 1	0 (zero)
ForceMultiCoilsOnly	<p>Forces the use of only function code 15 (multiple coils) for coil writes.</p> <p>With this parameter, you can set different values for specific I/O devices or groups of I/O devices.</p>	0 - function code 5 and 15 1 - function code 15 only	0 (zero)

[MODNET] Parameter name	Description	Allowable values	Default value
	See <a href="#">Device/Group-specific Parameters</a> .		
InitVar (With the InitVarType parameter configured.)	<p>Address of the register to read at startup to confirm connections.</p> <p>If the InitVarType parameter is configured, InitVar must have the register address and InitVarType must provide the register type. InitVar will be able to support the values 0-65535.</p> <p>Example:</p> <pre>[Modnet] InitVar = 39000 InitVarType = 4</pre> <p>With this configuration during start-up and re-connection the driver will attempt to read Input Register 39000</p> <p>See <a href="#">Device/Group-specific Parameters</a>.</p>	0 – 65535	40001
InitVar (Without the InitVarType parameter configured.)	<p>Address of the register to read at startup to confirm connections.</p> <p>If the InitVarType parameter is not configured, InitVar must have the complete address including the register type.</p> <p>InitVar should be configured as format FR.</p> <p>F : Register type (range: 1-4, 1-Digital read output, 2-Digital read input, 3-Integer read output, 4-Integer read input)</p> <p>R : Register address</p>	(1-4) (0-65535)	040001

[MODNET] Parameter name	Description	Allowable values	Default value
	<p>(range : 0-65535)</p> <p>Example :</p> <p>[Modnet] InitVar = 12345</p> <p>By default, InitVarType will be assigned value 0.</p> <p>With this configuration during start-up and re-connection the driver will attempt to read Input Coil 2345.</p> <p>With this parameter, you can set different values for specific I/O devices or groups of I/O devices.</p> <p>See <a href="#">Device/Group-specific Parameters</a>.</p>		
InitVarType	<p>The type of register the MODNET driver tries to read on startup to allow communication to start. It is configured with the parameter InitVar.</p> <p>With this parameter, you can set different values for specific I/O devices or groups of I/O devices.</p> <p>See <a href="#">Device/Group-specific Parameters</a>.</p>	0 to 4, or 20 (for extended registers access) 1 - Digital read output 2 - Digital read input 3 - Integer read output 4 - Integer read input 20 – Extended register 0 - the value defined for the parameter InitVar will be considered a complete address, having register type.	0
LongDataType	<p>In the MODNET protocol, LONG data types default to a simplified implementation, with a shortened range of 0 to 99,999,999 for mode 0.</p> <p>Mode 2 has the same range as mode 0, but with the register order swapped.</p>	0 - implies 10,000 * low register + high register 1 - implies 65,536 * low register + high register 2 - implies 10,000 * high register + low register 3 - implies 65,536 * high register + low register	3

[MODNET] Parameter name	Description	Allowable values	Default value
	<p>Mode 1 supports the complete LONG range of -2,147,483,648 to +2,147,483,647.</p> <p>Mode 3 has the same range as mode 1, but with the register order swapped.</p> <p>When using the MODBUS protocol, Plant SCADA combines two registers to store a long data type in a PLC. The way in which it does this is defined by this parameter.</p> <p><b>Note:</b> This parameter is not applicable if you are using the <a href="#">MODNET Protocol Variants</a> MODNET20 or MODNET30 to communicate with a device. Under these circumstances, a value of 3 will be hardcoded for this parameter.</p> <p>With this parameter, you can set different values for specific I/O devices or groups of I/O devices.</p> <p>See <a href="#">Device/Group-specific Parameters</a>.</p>		
MaxBits	<p>Sets the maximum number of bits in a block for dynamic blocking. This is normally the same as the MODNET protdir.dbf "max_bits" setting.</p> <p><b>Note:</b> For <a href="#">MODNET Protocol Variants</a> MODNET20, this parameter is not applicable. The default</p>	8 to 2000	2000

[MODNET] Parameter name	Description	Allowable values	Default value
	<p>value 2000 is used for blocking, ignoring any configured value.</p> <p>For protocol variant MODNET30, this parameter is applicable. The default value is 1024 in this case.</p> <p>With this parameter, you can set different values for specific I/O devices or groups of I/O devices.</p> <p>See <a href="#">Device/Group-specific Parameters</a>.</p>		
MaxBitsExt	<p>This parameter sets the maximum bits blocked for extended registers.</p> <p><b>Note:</b> This parameter is not applicable if you are using the <a href="#">MODNET Protocol Variants</a> MODNET20 or MODNET30 to communicate with a device. If MODNET20 is being used, a value of 1920 will be hardcoded for this parameter. If MODNET30 is being used, a value of 1024 will be hardcoded.</p>	8 to 32767	1920
MaxOutstanding	<p>Specifies the maximum number of outstanding requests (that is, the maximum number of requests allowed to be sent to the PLC at a given time).</p> <p>This parameter should be adjusted to coincide with the number of requests the PLC can process in one scan cycle. The</p>	1 to 32	1

[MODNET] Parameter name	Description	Allowable values	Default value
	<p>appropriate value can be obtained by analysing a "debug * all" dump to determine the number of PLC replies received in one PLC scan. If the PLC is running at short scan rates, the scan time might have to be temporarily stretched to 30 milliseconds in order to identify the different PLC scans easily. Empirical testing indicates that longer PLC scan times allows the PLC to process more requests per scan.</p> <p>Setting the MaxOutstanding parameter unnecessarily high will lower throughput. The number of requests the PLC can process per scan determines the driver's performance. This value is dependent on both the tuning of the PLC scan times and on the PLC's overheads such as Modbus communications.</p> <p><b>Note:</b> MaxPending needs to be greater than MaxOutstanding.</p> <p>If this parameter is placed under the section [MODNET] then all ports in the project will have this parameter set to this number. If this parameter is placed under the following section: [MODNET.&lt;Port_name&gt;] the specified port will have this parameter set to</p>		

[MODNET] Parameter name	Description	Allowable values	Default value
	<p>this number and previous number will be overwritten.</p> <p>A [MODNET.&lt;Port_name&gt;] parameter has the highest precedence.</p> <p>If the MaxOutstanding parameter is missing, the number of the outstanding requests will be set to the default of 1 for all Modnet ports in the project.</p> <p><b>Examples</b></p> <p>[MODNET.TCPPORT1] MaxOutstanding =3 [MODNET.TCPPORT2] MaxOutstanding =4</p> <p>In the above example, TCPPORT1 is set to 3 and port TCPPORT2 is set to 4</p> <p>[MODNET] MaxOutstanding =3 [MODNET. TCPPORT1] MaxOutstanding =5</p> <p>In this example, TCPPORT1 is set to 5 and all other ports in the project (if any) are set to 3.</p>		
MaxPending	<p>The maximum number of pending commands that the driver holds ready for immediate execution.</p> <p><b>Note:</b> MaxPending needs to be greater than MaxOutstanding.</p>	1 to 32	2
NonCriticalExceptionMask	Defines the exceptions to be considered non-critical exception. If a non-critical exception occurs, the	0x1 – 0xFFFFFFFF	0x20 Exception code 6 – Sever busy

[MODNET] Parameter name	Description	Allowable values	Default value
	<p>driver will retry to send the request.</p> <p>Its value is a exception mask, where bit position value maps to exception code.</p> <p>Example:</p> <p>The value 0x20 sets the mask for bit position 6, that represents exception code 6, i.e. server busy.</p>		
OnlineTestExceptionMask	<p>Defines the exceptions to be reported that are received while initializing or checking status of IO Device.</p> <p>Its value is a exception mask, where bit position value maps to exception code.</p> <p>Example:</p> <p>The value 0xFFFFFFF9 sets the mask of bit positions except bit position 2 and 3, that represents exception code 2 (Illegal data address) and 3 (Illegal data value). Hence these exceptions will be ignored to report IO Device online status.</p>	0x1 – 0xFFFFFFFF	0xFFFFFFF9
PollTime	<p>The interrupt or polling service time (in milliseconds). Setting the polling time to 0 puts the driver in interrupt mode.</p> <p><b>Note:</b> It is recommended that you do not modify the PollTime value, which is set to 0 (zero) by default.</p>	0 to 300 (milliseconds)	0
PresetMultiRegistersOnly	Forces the use of only	0 - function code 6 and 16	1

[MODNET] Parameter name	Description	Allowable values	Default value
	<p>function code 16 (multiple registers) for register writes. This function code setting is normal for a Class 0 Modbus TCP/IP device.</p> <p>With this parameter, you can set different values for specific I/O devices or groups of I/O devices.</p> <p>See <a href="#">Device/Group-specific Parameters</a>.</p>	1 - function code 16 only	
ReadMinTwoRegisters	<p>By default, the driver reads a minimum of one register. This parameter allows the driver to check the read request size and make sure the driver reads at least two registers.</p>	0 – The driver reads a minimum of one register. 1 - The driver reads at least two registers.	0
RegisterBitReverse	<p>This parameter defines the order in which bits in a register are individually addressed.</p> <p><b>Note:</b> This parameter is not applicable if you are using the <a href="#">MODNET Protocol Variants</a> MODNET20 or MODNET30 to communicate with a device. Under these circumstances, a value of 1 will be hardcoded for this parameter.</p> <p>With this parameter, you can set different values for specific I/O devices or groups of I/O devices.</p> <p>See <a href="#">Device/Group-specific Parameters</a>.</p>	0 - setting Bit 16 (only) will result in a register value of 1 1 - setting Bit 1 (only) will result in a register value of 1	0
Retry	The number of times to	0 to 8	0

[MODNET] Parameter name	Description	Allowable values	Default value
	<p>retry a command after a timeout.</p> <p>With this parameter, you can set different values for specific I/O devices or groups of I/O devices.</p> <p>See <a href="#">Device/Group-specific Parameters</a>.</p>		
SendBCDSwap	<p>Defines the order of bytes in a written word. This will not affect the order of bytes read.</p> <p><b>Note:</b> This parameter is not applicable to long BCD data types.</p> <p>With this parameter, you can set different values for specific I/O devices or groups of I/O devices.</p> <p>See <a href="#">Device/Group-specific Parameters</a>.</p>	1 - Big Endian format (Motorola or Network format) 0 - Small Endian format (Intel format)	1
SetNRDDDisconnectExceptionErrorAsOffline	<p>Indicates how the non-standard exception error 0x1d is handled.</p> <p>This error occurs when a MODNET device is disconnected from an NRDD Gateway.</p>	0 - The error is treated as a general error 1 - The error will cause the device to go offline	0
StringReverse	<p>Reverses the order of bytes in string variable types, for example, "ABCD" "BADC".</p> <p>With this parameter, you can set different values for specific I/O devices or groups of I/O devices.</p> <p>See <a href="#">Device/Group-specific Parameters</a>.</p>	0 - No reverse 1 - Reverse	0
TransactionExceptionMask	Defines which exception to be considered	0x1 – 0xFFFFFFFF	0x400

[MODNET] Parameter name	Description	Allowable values	Default value
	<p>transaction difficulties. Driver will report unit offline if this exception is received.</p> <p>Its value is a exception mask, where bit position value maps to exception code.</p> <p>Example: The value 0x400 sets the mask for bit position 11, that represent exception code 0xB.</p>		
Timeout	<p>Specifies how many milliseconds to wait for a response before displaying an error message.</p> <p>With this parameter, you can set different values for specific I/O devices or groups of I/O devices.</p> <p>See <a href="#">Device/Group-specific Parameters</a>.</p>	0 to 32000 (milliseconds)	1000
WatchTime	<p>The period (in seconds) with which the driver checks the communications link to the I/O device.</p> <p>With this parameter, you can set different values for specific I/O devices or groups of I/O devices.</p> <p>See <a href="#">Device/Group-specific Parameters</a>.</p>	0 to 128 (seconds)	30

## Device/Group-specific Parameters

The MODNET driver can apply different initialization parameter values to specific I/O devices or groups of I/O devices. This means the user can specify:

Global parameters that apply to devices.

Channel-level (port-level) parameters that apply to devices on the specified port.

Group-level parameters that apply to devices in a specified group.

Device-level parameters that apply only to the specified device.

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

This feature can be implemented in the citect.ini for the following MODNET parameters:

- MaxBits
- InitVar
- InitVarType
- StringReverse
- RegisterBitReverse
- FloatMode
- LongDataType
- SendBCDSwap
- ForceMultiCoilsOnly
- PresetMultiRegistersOnly
- Retry
- Timeout

To set parameters for a particular port, group, or device, you need to create a new section in the citect.ini file. Label it with the driver name followed by a period (.) character and the name of the particular port, group, or device you want to specify the parameter setting for.

For example:

[MODNET.<Port\_Name>]: applies the parameter settings to the specified port.

[MODNET.<Group\_Name>]: applies the parameter settings to the specified group.

[MODNET.<Port\_Name>.<IODevice\_Name>]: applies to the specified device.

Any parameters you then define in the following section of the citect.ini file relate only to the specified device or device group.

### **Example**

The following citect ini file format is an example of how the InitVarType parameter could be specified differently for different I/O devices communicating using the MODNET protocol.

Assume that two ports are used: PORT1 and PORT2. PORT1 has three I/O devices attached:

- DEV1A
- DEV1B

- EV1C

PORTE also has three devices:

- DEV2A
- DEV2B
- DEV2C

Assume that the user has specified that DEV1C and DEV2C belong to GROUPZ. The citect.ini file contains the following entries:

```
[MODNET]
InitVarType=1
[MODNET.PORT1]
InitVarType=2
[MODNET.PORT2]
InitVarType=2
[MODNET.GROUPZ]
InitVarType=3
[MODNET.PORT1.DEV1A]
InitVarType=1
[MODNET.PORT2.DEV2B]
InitVarType=4
```

The resultant InitVarType for the I/O devices will be as follows:

DEV1A:	1	as a result of [MODNET.PORT1.DEV1A]
DEV1B:	2	as a result of [MODNET.PORT1]
DEV1C:	3	as a result of [MODNET.GROUPZ]
DEV2A:	2	as a result of [MODNET.PORT2]
DEV2B:	4	as a result of [MODNET.PORT2.DEV2B]
DEV2C:	3	as a result of [MODNET.GROUPZ]

As the above example shows, there is a hierarchy that determines the outcome of such settings. In simple terms, specific parameter settings overwrite general level settings. Therefore, parameters written in the scope of I/O devices will overwrite those set for groups; parameters set for groups will overwrite global settings, and so on.

## MODNET Driver Specific Errors

The following errors, listed in (hexadecimal) sequence, are specific to this protocol. Plant SCADA displays the error number and description for common protocol-specific errors. Uncommon errors are not contained in the Plant SCADA error database, in which case Plant SCADA only displays the error number.

You might need additional information to rectify an error. This information should be detailed in the documentation that accompanied the I/O device (or network). If, after reviewing the documentation, you cannot

rectify an error, contact Technical Support for this product.

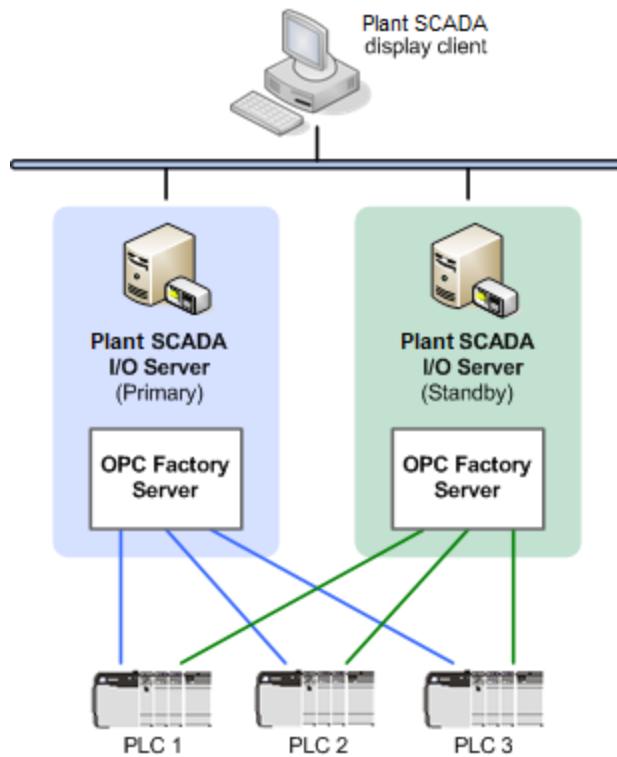
Error (in hex)	Description
0x0101 Illegal function	The message function received is not allowed for the addressed slave.
0x0102 Illegal data address	The address referenced is out of range.
0x0103 Illegal data value	Invalid data is being written to the PLC.
0x0108 Memory parity error	An error has been encountered in the computer's memory.
0x28	Bad response from PLC.
0x104	Device inoperative.
0x105	Acknowledge.
0x106	Busy – rejected.
0x107	Negative Acknowledge.
0x2100	Function Code Mismatch.
0x2101	Invalid Command.
0x100	Point-To-Point and gateway units are on the same port.

## OFSOPC Driver

The OFSOPC Driver enables Plant SCADA to communicate with OPC Factory Server (OFS), providing a gateway to a network of OFS-compatible devices.

For each Plant SCADA I/O server, a single connection to a local instance of OFS is established, allowing tight integration between the driver and the OFS configuration environment. Individual devices are addressed within a Plant SCADA project using OFS alias names.

The following diagram demonstrates a simple system installation using I/O server and PLC network redundancy.



**Note:** With the release of version 7.20, Plant SCADA supports the retrieval of time-stamped data directly from field devices. This capability is enabled by the Driver Runtime Interface (DRI), a component of Plant SCADA that is used by the OFSOPC driver to directly update time-stamped digital alarms, time-stamped analog alarms and event-based trends. For more information, see the topic *Retrieving Time-stamped Data from Field Devices* in the main Plant SCADA help.

## See Also

- [System Requirements](#)
- [Setting up Device Communication](#)
- [Configuring Variable Tags](#)
- [Advanced Configuration and Maintenance](#)
- [Performance Tuning](#)
- [Troubleshooting](#)

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.
<b> DANGER</b>	
<b>DANGER</b> indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.	
<b> WARNING</b>	
<b>WARNING</b> indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.	
<b> CAUTION</b>	
<b>CAUTION</b> indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.	
<b>NOTICE</b>	
<b>NOTICE</b> used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.	

## Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

## Before You Begin

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ WARNING****LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

**⚠ WARNING****INTENDED USAGE**

The integration of diagnostic buffer support into Plant SCADA is provided as a supplementary source of alarm information and should not be used to monitor critical alarms for a live production system.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Alarms generated by PLCs set to different date and time values will be sorted according to their time information, but the resulting order may not reflect their real chronological order of appearance (i.e. as defined per a shared clock). Depending on the synchronization accuracy required at system level, different synchronization methods should be used.

**⚠ WARNING****ALARMS NOT DISPLAYED IN TRUE CHRONOLOGICAL ORDER**

Set Plant SCADA and all PLCs to the same system date and time.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## System Requirements

Each Plant SCADA I/O server needs to have a local installation of OPC Factory Server (OFS) version 3.33 or later. Version 1.02.12.001 (or later) of the OFSOPC driver will only use the **#PLCQualStatus** tag to determine the status of a device.

## OFS configuration file locations

At startup, the OFSOPC driver might need to access the OFS configuration file **DeviceConfig.xml** to gather push data information.

Once OFS has been installed, this file can be found in the following directory:

%PROGRAMDATA%\AVEVA Plant SCADA 2020 R2\OFS\<ver>

Where <ver> is the version of OFS installed.

## Setting up Device Communication

The OFSOPC driver handles device communication through the integration of OPC Factory Server (OFS) on the I/O Server. Individual devices are addressed through the use of OFS alias names in the configuration of your Plant SCADA variable tags.

Setting up communication within a Plant SCADA project is therefore a straightforward process of connecting to OFS locally on each I/O server.

See [Communication Settings](#) for the default settings used by the Express Communication Wizard when configuring the connection to OFS as a device.

---

**Note:** If you want to use the OFSOPC driver on an I/O server that is running as a Windows® service, you need to make some additional configuration settings on the I/O server.

## See Also

[Device Address](#)

[Data Dictionary Support](#)

## Device Address

The address to use for a device using the OFSOPC driver is the alias name configured in the OFS configuration for the remote PLC.

## See Also

[Communication Settings](#)

## Communication Settings

To establish communication with a device, the associated **Board**, **Port** and **I/O Device** need to be correctly configured in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure them, use the settings outlined below.

### Boards

Field	Value
Board Name	This field is user defined (e.g. 'Board1').
Board Type	Enter <b>OFSOPC</b> .
Address	Enter zero (0)
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.
Comment	Any useful comment. This field is user defined and not required by the driver.

### Ports

Field	Value
Port Name	This field is user defined.
Port Number	Leave this field blank.
Board Name	Refers to the board previously defined in Board settings (e.g. 'Board1').
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank.
Special Options	Leave this field blank.

Comment	Any useful comment. This field is user defined and not required by the driver.
---------	--

## I/O Devices

Field	Value
Name	This field is user defined.
Number	I/O device number which needs to be unique for each logical device, but identical for redundant devices.
Address	The alias name configured in the OFS configuration for the remote PLC.
Protocol	Enter OFSOPC.
Port Name	Refers to the port previously defined in the Port settings.
Comment	Any useful comment. This field is user defined and not required by the driver.

## Data Dictionary Support

To support OFS Data Dictionary, OFS version 3.6 (or later) needs to be installed.

OFS version 3.6 introduced a new specific item, #PLCDaDataDicReady, which provides the status of Data Dictionary in OFS for each device alias.

The #PLCDaDataDicReady item can be equal to the following values:

- **0:** indicates that the PLC Data Dictionary is not configured for this alias (in the OFS configuration tool).
- **1:** indicates that the PLC Data Dictionary is not available. This may occur if OFS is still in the process of loading the Data Dictionary, or the Data Dictionary is not set in the UnityPro application (which means it is not embedded in the PLC).
- **2:** indicates that the PLC Data Dictionary is available.

The driver checks the status of the #PLCDaDataDicReady item. If the value of the item is equal to **0** (data dictionary is not enabled in OFS configuration tool), or **2** (data dictionary is loaded), it adds the OPC items and sets the I/O device online.

**Note:** Data Dictionary must be either enabled or disabled in both the OFS configuration tool and the UnityPro application, otherwise the value of #PLCDaDataDicReady will be equal to 1 and the driver will not put the I/O device online.

## Configuring Variable Tags

Variable tags are addressed using the alias names defined in OPC Factory Server (OFS). The following table demonstrates the variable tag address formats you can use to read values.

When specifying a data type, consider whether the server will be able to convert data to the requested type.

Address	Plant SCADA type	OFS Type	Example
[<alias>!]<item_id>	digital	vt_bool	bool_symbol or plc1!bool_symbol
[<alias>!]<item_id>	byte	vt_ui1	byte_symbol or plc1!byte_symbol
[<alias>!]<item_id>	int	vt_i2	int_symbol or plc1!int_symbol
[<alias>!]<item_id>	long	vt_i4	long_symbol or plc1!long_symbol
[<alias>!]<item_id>	real	vt_r4	real_symbol or plc1!real_symbol
[<alias>!]<item_id>	string	VT_ARRAY of VT_UI1	string_symbol or plc1!string_symbol

Plant SCADA also allows a collection of variables to be associated with an OFS array tag. The following table demonstrates the variable tag address formats used to read values from OFS array tags.

See also [Array Tag Support](#).

Address	Plant SCADA type	OFS Type	Example
[<alias>!]<item_id>!a[size]	digital	array of vt_bool	dig_array!a[32] or plc1!dig_array!a[32]
[<alias>!]<item_id>!a[size]	byte	array of vt_ui1	byte_array!a[10] or plc1!byte_array!a[10]
[<alias>!]<item_id>!a[size]	int	array of vt_i2	int_array!a[10] or plc1!int_array!a[10]
[<alias>!]<item_id>!a[size]	long	array of vt_i4	long_array!a[10] or plc1!long_array!a[10]
[<alias>!]<item_id>!a[size]	real	array of vt_r4	real_array!a[10] or plc1!real_array!a[10]

## Tag addressing for bits

The following table defines the syntax for configuring the tag that represents the bit of a WORD or INT UnityPro

data type.

Address	Plant SCADA type	UnityPro Type	Example
[<alias>!]<item_ID>.<bit number>	digital	INT or WORD	MyInt.0 Or PLC!MyInt.1

The item\_id above can be either a symbolic or direct address.

The direct address syntax is shown below:

- Single variable: [<alias>!]<direct\_address>  
For example: PLC1!%MW1
- Array of variables: [<alias>!]<direct\_address>:<size>!A[<size>]  
For example: PLC1!%MW1:10!A[10]

---

**Note:** The driver will poll some of the OFS special items instead of subscribing to the item value updates. For more information, refer to [Poll Engine for #Specific Tags](#).

---

## See Also

[Timestamp and Quality Support](#)

[Tag Addressing for Derived Data Types \(DDT\)](#)

## Array Support

The OFSOPC driver allows you to write to an individual array item without updating the whole array. This operation is performed by creating a temporary subscription to the individual sub-item and issuing a write to that item.

For example, the following expression indicates to the OFSOPC driver to assign a value of 8 to the fifth element of the "MyArray" tag in OFS:

MyArray [5] = 8

---

**Note:** Plant SCADA indexes an array starting from zero, while some OFS applications allow you to specify an array's lower bound (as a positive or negative value). The OFSOPC driver will dynamically determine an array's lower bound and map the Plant SCADA array to it.

---

## Large array support

The maximum size of an array tag is limited to 256 bytes due to the underlying infrastructure of Plant SCADA. For example, if the type is integer (2 bytes), Plant SCADA only allows the array tag to be configured with 128 elements.

However, special array offset syntax can be used to address multiple array tags configured with the same address as sub-arrays. A variable tag can therefore represent a sub-array with the particular offset and range within a large array.

The following demonstrates the variable tag address syntax used to facilitate this:

[<alias>!]<ITEM\_ID>\<OFFSET>!A[SIZE]

#### Example

The following provides an example of how sub-arrays can be used to map to a large array in an OFS system.

To address **IntArray[17]** in OFS, you would use the variable tag address **LargeArrayPart1[17]**.

To address **IntArray[193]**, you would use **LargeArrayPart2[65]** (as  $65 = 193 - 128$ ).

## Timestamp and Quality Support

When used with Plant SCADA version 7.0 or later, the OFSOPC driver supports the use of additional variable tags to obtain quality and timestamp values. Quality or timestamp tags are duplicates of a variable tag, with an appended "!Q" or "!T" or "!M" in the address field to define their purpose.

For example:

Address	Plant SCADA Type	Description
[<alias>!]<item_ID>!Q	INT	The quality reported for the item
[<alias>!]<item_ID>!T	LONG	The seconds component of the timestamp reported for the item (in UTC seconds)
[<alias>!]<item_ID>!M	LONG	The milliseconds component of the timestamp reported for the item

## Tag Addressing for Derived Data Types (DDT)

The following table defines the syntax for configuring the tag that represents the field of a structure.

Address	Example
[<alias>!]<item_ID>.<field>	MyStruct.MyField1 or PLC1!MyStruct.MyField1

For example, if the Derived Data Type (DDT) "MyStruct" is defined in UnityPro as the following:

Field Name	UnityPro Type
MyField1	INT
MyField2	INT
MyField3	INT

And **MyStructInstance** is configured as an instance of **MyStruct** in UnityPro®, then you would need to configure the following tags to access the fields of **MyStructInstance**:

Tag Name	Tag Address	Plant SCADA Type
MyStructInstance_MyField1	MyStructInstance.MyField1	INT
MyStructInstance_MyField2	MyStructInstance.MyField2	INT
MyStructInstance_MyField3	MyStructInstance.MyField3	INT

## Poll Engine for #Specific Tags

Generally, the OFSOPC driver activates the item specified in the tag address and relies on the value updates from OFS, but some special OFS items do not provide value updates via subscription. The driver needs to poll these items for the value updates. If the tag address contains any of the following strings, the driver will poll it for value updates:

- #AppliName
- #AppliVersion
- #NbrRequest
- #RefreshDevice
- #AppliOMC
- <>system>#!ClientAlive
- #TSEventSynchro
- #DeviceIdentity

---

**Note:** The OFSOPC driver will poll for value updates only if these tags are subscribed in Plant SCADA. The Plant SCADA tag subscription rate defines the poll rate.

---

## Advanced Configuration and Maintenance

This section provides advanced configuration instructions.

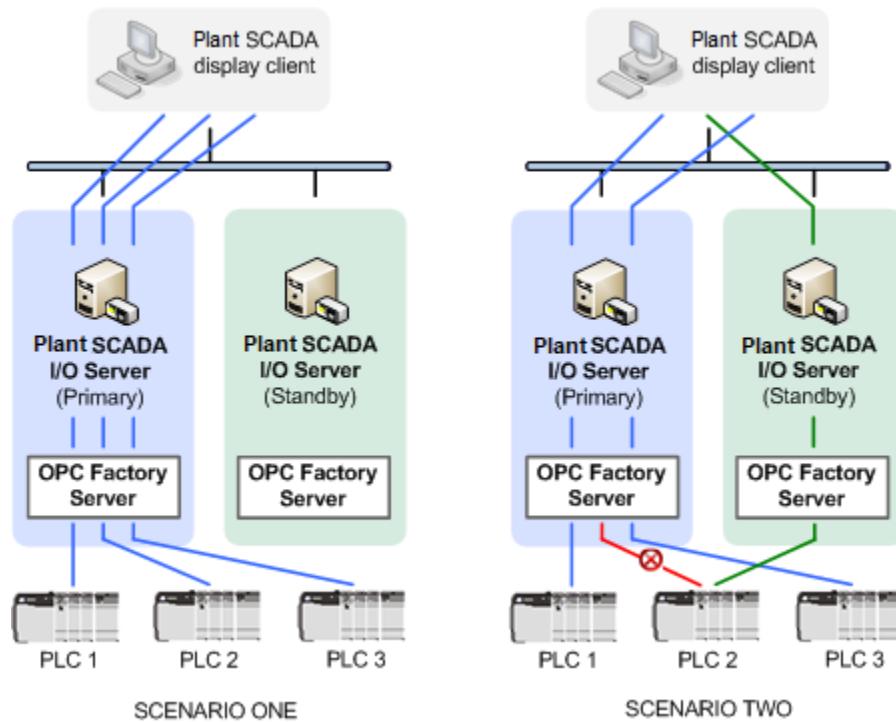
- Configuring Redundancy
- Using the Batch-write Mechanism
- Push Data Support
- OFS Group Management
- Source Timestamped Events Support
- Diagnostic Buffer Support
- Customizing a Project using Citect.ini Parameters

## Configuring Redundancy

The OFSOPC driver supports fully transparent redundancy across the following architectures:

- Multiple I/O server redundancy, including support for load balancing across one or more computers
- Device redundancy, where a PLC is managed by more than one I/O server
- Communication redundancy, where each PLC is on a redundant network.

To facilitate a complex redundant architecture, the OFSOPC driver represents each PLC as a separate I/O device. This allows a switchover to occur on a per-PLC basis, as demonstrated by the diagram below.



Scenario one demonstrates the system in normal operating state, with the client connected to the three PLCs via the primary server. In scenario two, the link from the primary server to PLC2 is broken. Under these circumstances, the client is able to re-establish communication with PLC2 via the standby server.

The OFSOPC driver subscribes to **#PLCQualStatus** and system bit %S0 on each PLC to determine the current operational status of a device. If these tags report bad quality, the I/O device will be declared offline and a switchover to the standby will occur.

---

**Note:** It is also possible to specify a custom item value that the driver will check in addition to #PLCQualStatus. See [Dynamic Status Item Parameters](#).

---

## Using the Batch-write Mechanism

The OFSOPC driver supports a batch-write mechanism that allows a defined set of tags to be updated at the same time.

A batch-write is implemented by creating a special tag with the address "BatchWrite". This functionality will be implemented on a per-PLC basis (you need to create one special tag per device).

Once BatchWrite has been written to with a value of 1, subsequent writes to the I/O device will be deferred until a write operation to BatchWrite with value 0 (zero) is received. At that time, the deferred writes will be performed together to the OFS server.

If a 0 value is not received within a pre-defined timeout period, the batch write operation will be performed on the deferred writes currently held by the driver. The BatchWrite tag will be reset with to a value of 0 (zero) to

help prevent infinite waiting. This timeout period (set to 30 seconds by default), can be adjusted using the **BatchWriteTimeout** parameter.

---

**Note:** The OFSOPC driver does not support a batch write operation during any kind of redundancy switchover. Under these circumstances, the batch-write operation will not work.

---

## Push Data Support

OPC Factory Server (OFS) supports "push data", which allows a remote device to connect as a Modbus/TCP server and write data in a format that includes a timestamp.

The OFSOPC driver supports this functionality by implementing push-data items as time-stamped digital alarms, time stamped analog alarms and event trends.

## Subscribing to a push-data item

The OFSOPC driver initially gathers push-data configuration information from the OFS configuration file **DeviceConfig.xml**. For each item, the address is checked to determine if it is located within a push-data area. If this is the case, the item is added to the OFS group that has the fastest subscription rate to help ensure that changes will be transmitted from OFS to Plant SCADA as efficiently as possible.

## OFS Group Management

The OFSOPC driver uses OFS groups to prioritize and manage device communication.

For each device instance, the driver creates three groups in OFS, each with a different update rate. Items are then allocated to a group according to a detected polling rate. For example, status tags and heavily sampled items fall into the fastest group, rarely accessed items are added to the slowest group.

The subscription rate used for each Plant SCADA tag is then adjusted to match the polling rate set for the group to which the related OFS item is allocated. Tags will be dynamically moved between the groups in response to changes in demand for an item.

It is possible to manually adjust the update rate for each group either globally or per I/O device. This requires adjustments to the `citect.ini` file, and is only recommended for advanced users.

See [Group Management Parameters](#).

## Source Timestamped Events Support

### Source Timestamped (TS) Event Group Support

Time stamped functionality is currently implemented and supported only for digital events. For configuration details refer to [TS Quality Support](#).

The driver creates a new OPC group to manage all OPC items with TS Event support. There is only one TS Event OPC group created per driver instance.

The driver creates and activates an OPC group called `##TSEventsGroup##` at start up and deactivates and removes that group at shutdown.

After the TS Event OPC group has been created, the driver periodically (every 500ms) signals that it is active by writing to the `<<system>>! #ClientAlive` OPC item.

As start up the driver reads the TS Event Support item property (Id: 5012) and adds those items which support TS Events to the TS Event Group.

After TS Event items have been added and activated the driver writes to the #TSEventSynchro item to get the current values of all TS Event items.

## Plant SCADA Configuration for Source TS Support - Ports and I/O Devices

There must be only one logical device and one port configured to communicate with all available TS Sources. This is to ensure the following

- Single client connection to OFS
- Driver issues only a single request to OFS for initial data from all available TS sources.

Accordingly, only one primary IO Device/Port and one Standby IO Device/Port can exist in the project in the case of redundancy.

## Compatibility

The driver initializes TS Event support only if the following conditions are true:

- Plant SCADA v7.20 or higher version is used.
- OFS Server v3.40 or higher version is used.
- EventServiceSupport setting in DeviceConfig.xml is set to True for a specific device alias.

**Note:** When the source event buffer in the PLC is full, any new events will not be stored. In this case the value of SOE\_Uncertain variable becomes TRUE. When the buffer becomes available again, the PLC will provide the values of all time-stamped event variables. As these values are timestamped with a current time, the time quality of these values will be set to Invalid. The SOE\_Uncertain is a variable in a time stamped event source whose value becomes TRUE when there is no space in the event buffer.

### **WARNING**

#### **LOSS OF ALARMS**

- To be able to detect that the event buffer is full, configure a tag and an alarm tag associated with the SOE\_Uncertain parameter in UnityPro.
- Respond quickly to a buffer full alarm if it appears, as this will avoid a situation where the buffer becomes inoperable.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## TS Quality Support

TS Event OPC item have not only the value quality, but also the time quality associated with it. The time quality is provided in the high 8 bits of the OPC Quality Word which are available for vendor specific use:

- bits 7 to 0 - OPC quality flags,
- bits 15 to 8 - Timestamp quality.

The time quality is available in SCADA v7.20 only via a separate tag and it will be stored and displayed by the alarm system in SCADA v7.30.

Time quality is defined as follows:

- bits 4 to 0 – Time Accuracy

These bits represent a time accuracy (values 2-24) or one of the error codes:

- 27 Clock In Sync
- 28 TSInit
- 29 IO Channel error
- 30 Invalid
- 31 Unspecified
- bit 5 - Clock Not Synchronized
- bit 6 - Clock Failure
- bit 7 - Leaps Seconds Known

## Support for Version 7.20

Active Alarm page can be used to display time stamped event data received from the event source. However, to display the time quality of a time stamped event on the Active Alarm page, a separate variable tag needs to be configured in addition to the time stamped digital variable tag. This new variable tag should be created with the address <item\_ID>!Q and it represents the overall quality of the original digital variable tag, where:

<item\_ID> refers to the original digital variable tag address.

The value of this tag will include time quality information: the low 8 bits represent the Quality, Substatus and Limit status according to the OPC specification; the high 8 bits represent time quality. Details of the time quality values can be found in the section on Time Quality.

In addition, a custom Cicode function will need to be written to return a string representation of time quality. To display time quality in the runtime environment this Cicode function can be placed in the "Alarm Desc" field when configuring the time stamped digital alarm in configuration time. This time quality will be displayed at runtime under the "Desc" column on the Active Alarm page.

The Cicode function should utilize the high 8 bits and translate this to a string representing the description of the time quality.

For example:

```
STRING
FUNCTION GetTimeQuality(INT iQuality)
INT iHighByte = HighByte(iQuality);
SELECT CASE iHighByte
CASE 2 TO 24
RETURN "Time Accuracy ("+IntToStr(iHighByte)+")";
CASE 27
RETURN "Clock In Sync ("+IntToStr(iHighByte)+")";
CASE 28
```

```
RETURN "TSInit ("+IntToStr(iHighByte)+"")";  
CASE 29  
RETURN "IO Channel Error ("+IntToStr(iHighByte)+"")";  
CASE 30  
RETURN "Invalid ("+IntToStr(iHighByte)+"")";  
CASE 31  
RETURN "Unspecified ("+IntToStr(iHighByte)+"")";  
CASE 32 TO 63  
RETURN "Clock Not Synchronized ("+IntToStr(iHighByte)+"")";  
CASE 64 TO 127  
RETURN "Clock Failure ("+IntToStr(iHighByte)+"")";  
CASE 128 TO 255  
RETURN "Leaps Seconds Known ("+IntToStr(iHighByte)+"")";  
CASE ELSE  
RETURN "Unknown ("+IntToStr(iHighByte)+"")";  
END SELECT  
END
```

For the alarm configuration, enter the following in the alarm's "Desc" field:

GetTimeQuality(*Tag1\_TSQuality*),

where:

*Tag1\_TSQuality* is the associated time quality tag.

## Support for Version 7.30

Citect SCADA 7.30 introduced a Sequence of Events (SOE) page which displays historical events in a sequence of events (SOE) format. Each record represents a single change of an event. This page can be utilized to display time stamped event data including time quality.

There are two columns which can be used to display time quality on the Active Alarm and SOE pages:

1. "TSQuality" column. This column displays a string representation of a synthesis of the time quality  
The column value can be one of the following:
  - Time Good,
  - Time Uncertain,
  - Clock Not Synchronized.
2. "Quality" column. This column displays a numeric value of OPC quality flags: the low 8 bits represent the Quality, Substatus and Limit status according to the OPC specification; the high 8 bits represent time quality.  
Note that time quality is not displayed on the Active Alarm or SOE pages by default, the user will need to use one of the following options:
  - Modify [Format]Alarm and [Format]SOE ini parameters in order to display it.
  - Add these columns from "Add Columns" list on the page at run time and save the changes.

The user will have to include "TSQuality" and "Quality" column names in the list of columns in [Format]Alarm and [Format]SOE ini parameters.

For example:

[Format]Alarm = {Time, 101}{OnTime,101}{Tag,100}{Name,181}{TSQuality,100}{Quality, 50},

[Format]SOE = {Time, 101}{Message,250}{Source,100}{Name,181}{TSQuality,100}{Quality, 50}.

## Timestamp Event Replication

The driver implements the replication of event data between redundant driver instances in order to be certain that all TS events are pushed to the Alarm and Tran systems.

Event replication helps in a case when there is a lack of communication between the I/O server which receives events from the OFS and the Alarm Server.

For example in a system which has primary I/O server, standby I/O server and alarm server, the primary I/O server receives events from the OFS server. It pushes them to the alarm server and also replicates them to the standby I/O server. The standby I/O server stores these events in the event buffer.

In the case of a lack of communication between the primary I/O server and the alarm server, the primary I/O server will not be able to push events to the alarm server.

If during that time switchover happens (for example the primary I/O server was shut down), the standby I/O server will push its event buffer to the alarm server. The standby I/O server will store events data for 10 minutes. Also the size of the event buffer is limited to 300,000 events.

In a normal scenario when there is no lack of communication, the primary I/O server sends a confirmation message to the standby server after an event has been successfully propagated through the alarm system. On receiving that message, the standby I/O server deletes the corresponding event from the event buffer.

Also, if there is a lack of communication between the primary and standby I/O servers and the replication of events is not possible, the primary I/O server stores events received from the OFS for a limited time (10 min) and sends them to the standby I/O server after connection has been restored.

## Diagnostic Buffer Support

A diagnostic buffer is a list of alarm records that is stored internally on a number of PLCs that support a diagnostic buffer. This is configured and managed by the software application Unity Pro™.

For diagnostic buffer management, two sets of functions have been defined:

- [Record Browsing Interface](#)  
Functions designed to read diagnostic records from a PLC.
- [Device Resynchronization Interface](#)  
Functions designed to force the driver to resend the whole content of the diagnostic buffer.

## See Also

[Diagnostic Buffer Parameters](#)

### Record Browsing Interface

The following functions are designed to read diagnostic records from a PLC that supports diagnostic buffer.

- [GetNextEvent](#) - gets the next diagnostic buffer record and sets it as the current record.
- [GetEventField](#) - retrieves the value of the field identified by the key in the current record

# GetNextEvent

Gets the next diagnostic buffer record and sets it as the current record. The first call will retrieve the first record.

## Syntax

*GetNextEvent*

(This function has no parameters.)

## Return value

0 - if there are no more records

1 - if a new record has been read

# GetEventField

Retrieves the value of the field identified by the key in the current record.

This function assumes a previous call to GetNextEvent (to set the current record), otherwise it will return an empty string.

## Syntax

**GetEventField**(*Alarm key, Error ID, Network Num, ON Time, OFF Time, Ack Time, Status, Area, Fbcomment, FBInstanceName + FBPinName*)

- *Alarm key*

The key used to identify the alarm this event belongs to. This key is generated internally using instance name, pin name, and comment

- *Error ID*

Error ID generated by the PLC, used for acknowledgement

- *Network Num*

Allow to retrieve the PLC (DEVICE) that emits this record

- *ON Time*

Time of the occurrence of the event in the PLC (UTC seconds timestamp)

- *OFF Time*

Time of disappearance of event in the PLC (UTC seconds timestamp)

- *Ack Time*

Time of acknowledgement of the event in the PLC (UTC seconds timestamp)

- *Status*

Current status of the event. The status value can be interpreted as:

Bit	Title	0	1
0	Status	Disappeared	Active
1	Acknowledgment	Not acknowledged or acknowledgment not requested	Acknowledged
2	Need Acknowledge	Acknowledgment is not required	Acknowledgment is required
3	Deleted	Not removed	Removed
4	New	Existing alarm	New alarm
5	Auto dereg	no	Detected error has been simultaneously activated and deactivated

- *Area*

Area as defined in the PLC diagnostic buffer. This may or may not match the definition of area in Plant SCADA.

- *Fbcomment*

Comment associated with the instance of the FB for this anomaly

- *FBInstanceName + FBPinName*

Name of the instance of the FB that caused this anomaly + the pin name

## Return value

String value of the field.

## Device Resynchronization Interface

These methods force the driver to read every entry for a given device.

- [GetResendState](#) - indicates if a "ResendAll" operation for a particular device has been successful.
- [ResendAll](#) - triggers an explicit resend of any existing records in the current diagnostic buffer.

## GetResendState

Returns a value that indicates when a "ResendAll" operation for a particular device is complete and if it has been successful.

## Syntax

GetResendState(*in*)

- *in*  
Network number (short)

## Return value

Return value	Field key value	Description
Successful	0	Last ResendAll call was successful for the device.
Unsuccessful	1	Last ResendAll call was unsuccessful for the device.
In progress	2	A ResendAll operation is currently running on the specified device.
Invalid parameter	-1	Device not found.

# ResendAll

Triggers an explicit resend of existing diagnostic buffer records for the specified device.

## Syntax

ResendAll(*in*)

- *in*  
Network number (short)

## Return value

None

## Customizing a Project using Citect.ini Parameters

You can customize the way Plant SCADA communicates with the OFS system by creating or editing the [OFSOPC] section of the Citect.ini file for your project. However, the default values for these parameters have been determined by driver development and testing to be optimal for the OFSOPC driver.

- Driver-specific Parameters
- Group Management Parameters
- Logging Parameters
- BatchWrite Parameters
- Diagnostic Buffer Parameters
- Dynamic Status Item Parameters

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

- Do not under any circumstances change or remove any of the undocumented citect.ini parameters.
- Before deleting sections of the citect.ini file, confirm that no undocumented parameters will be deleted.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

When Plant SCADA starts at runtime, it reads configuration values from the Citect.ini file stored locally. Therefore, any OFS configuration settings needs to be included in the Citect.ini file located on the computer acting as the I/O server to the OFS system.

By default, Plant SCADA looks for the ini file in the Plant SCADA project \bin directory. If it can't find it there, it searches the default Windows directory.

### **Driver-specific Parameters**

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

- Do not under any circumstances change or remove any of the undocumented citect.ini parameters.
- Before deleting sections of the citect.ini file, confirm that no undocumented parameters will be deleted.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

[OFSOPC] Parameter	Allowable Values	Default Value	Description
ReadAfterWrite	0 - No read, rely on the normal polling/ OnDataChange mechanism.  1 - Force sync read of item	0	Determines if a force read should be made from the server cache or from the device.

	written from the server cache after write.  2 - Force sync read of item written from device after write.		
--	--	--	--

**Group Management Parameters****UNINTENDED EQUIPMENT OPERATION**

- Do not under any circumstances change or remove any of the undocumented citect.ini parameters.
- Before deleting sections of the citect.ini file, confirm that no undocumented parameters will be deleted.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

[OFSOPC] Parameter	Allowable Values	Default Value	Description
Group1UpdateRate	Any positive value less than Group2UpdateRate	400	Group 1 polling rate in milliseconds
Group2UpdateRate	Any positive value less than Group3UpdateRate	1000	Group 2 polling rate in milliseconds
Group3UpdateRate	Any positive value less than 2147483647	3000	Group 3 polling rate in milliseconds

**Note:** The default value will apply if a specified value does not fall within the range of allowable values.

The following example demonstrates how to override the group management parameter setting for a specific I/O device.

[OFSOPC]

<Cluster>.<IOServerName>.<IODeviceName>.<Parameter>=<Value>

where:

- <Cluster> = the name of the cluster configured in Plant SCADA
- <IOServerName> = the name of the I/O server to which the device is connected
- <IODeviceName> = the name given to the device in your project
- <Parameter> = the group management parameter you would like to overwrite
- <Value> = the update rate for the specified device in milliseconds

## Logging Parameters

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

- Do not under any circumstances change or remove any of the undocumented citect.ini parameters.
- Before deleting sections of the citect.ini file, confirm that no undocumented parameters will be deleted.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

[OFSOPC] Parameter	Description	Allowable Values	Default Value
DebugLevel	The trace level used for the log file.	Any combination of the following, separated by a pipe character ( ):  ERROR WARN TRACE DEBUG ALL  See <a href="#">Logging</a> for examples.	-
DebugCategory	The categories used for the log file.	Any combination of the following, separated by a pipe character ( ):  TAG = tag debug NET = network level debug PROT = protocol level debug DCB = front end debug (i.e. I/O server) BUFF = buffer level debug THRD = thread level debug MISC = any other debugging ALL = all of the above  See <a href="#">Logging</a> for examples.	-

**Note:** These parameters do not have a value specified by default. To enable logging for this driver, you need to manually configure these parameters in your INI file.

## BatchWrite Parameters

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

- Do not under any circumstances change or remove any of the undocumented citect.ini parameters.
- Before deleting sections of the citect.ini file, confirm that no undocumented parameters will be deleted.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

The BatchWriteTimeout parameter specifies the maximum period of time that the OFSOPC driver will wait to receive an 'End Batch-Write' command (i.e. a setting of BatchWrite = 0) following the commencement of a batch-write (BatchWrite = 1). This is required to avoid an infinite wait period.

[OFSOPC]Parameter	Allowable Values	Default Value	Description
BatchWriteTimeout	10 - 3600 (seconds)  (The default value will apply if the specified value exceeds these limits.)	30 (seconds)	Number of seconds to wait for BatchWrite = 0 to be written following the commencement of a batch-write (BatchWrite=1).  If BatchWrite remains set at 1 for this period of time, the system will send the write and set BatchWrite back to 0.

The following example demonstrates how to override the global setting for the BatchWriteTimeout parameter for a specific I/O device.

[OFSOPC]  
*<Cluster>.<IOServerName>.<IODeviceName>.BatchWriteTimeout=<Value>*

Where:

- *<cluster>* = the name of the cluster the I/O server belongs to
- *<IOServerName>* = the name of the I/O server to which the device is connected
- *<IODeviceName>* = the name given to the device in your project
- *<Value>* = the timeout value for the specified device in milliseconds

## Diagnostic Buffer Parameters

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

- Do not under any circumstances change or remove any of the undocumented citect.ini parameters.
- Before deleting sections of the citect.ini file, confirm that no undocumented parameters will be deleted.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

[OFSOPC]Parameter	Allowable Values	Default Value	Description
EnableDiagBuffer	0 = disabled 1 = enabled	0	Enables or disables diagnostic buffer support
DiagnosticArea	Short (0-65535). The value needs to be added in decimal format. The value 0 means all areas are selected (same as 65535).	0	Allows you to filter the diagnostic buffer records according to their areas value.  There are 16 possible areas, and each bit of the word is used to manage one area.  Bit 0 corresponds to Area 0 ...  Bit 15 corresponds to Area 15.

## Dynamic Status Item Parameters

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

- Do not under any circumstances change or remove any of the undocumented citect.ini parameters.
- Before deleting sections of the citect.ini file, confirm that no undocumented parameters will be deleted.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

By default, the driver subscribes to #PLCQualStatus to determine the current operational status of a device. It is also possible to specify a custom item value that the driver will check in addition to #PLCQualStatus. This can be

useful to confirm that the driver will put the I/O device offline if OFS becomes unresponsive.

The item value must be periodically changing in the PLC. If it stops, the driver will put the I/O device offline. This behavior is controlled by the following parameters.

[OFSOPC]Parameter	Allowable Values	Default Value	Description
DynamicStatusItemName	PLC item name	-	<p>This parameter specifies the item name that gets updated in PLC at a certain interval.</p> <p>Some system items are already programmed in the PLC and can be used to avoid changes to the PLC program.</p> <p>For example, %S4, %S5, %S6 have value updates every 5ms, 50ms, or 500ms correspondingly.</p>
DynamicStatusItemInterval	1 - 2147483647 (seconds)	0	<p>This parameter specifies the period which the driver will use to check the status of the item specified by the DynamicStatusItemName parameter.</p> <p>If the item value has not changed during the period specified, the driver will put the I/O device offline.</p> <p>For the system items %S4, %S5, %S6, the parameter can be set to 1 second.</p>

These parameters can only be specified for a specific I/O device using the syntax below.

[OFSOPC]  
*<Cluster>.<IOServerName>.<IODeviceName>.<Parameter>=<Value>*

Where:

- *<Cluster>* = the name of the cluster configured in Plant SCADA.
- *<IOServerName>* = the name of the I/O server to which the device is connected.
- *<IODeviceName>* = the name given to the device in your project.
- *<Parameter>* = the parameter name.
- *<Value>* = the parameter value.

## Performance Tuning

OPC is a standard protocol that is widely used in industrial automation systems. OPC is chosen for use to simplify and standardize PLC communication.

The OPC communications solution is based on an OPC DA server called OFS. OFSOPC is a dedicated driver implemented for Plant SCADA that connects the SCADA system to OFS.

### Communicating with PLCs through OFS

The performance of the communications between a Plant SCADA system and a PLC through OFS involves many factors other than just the driver, because of the number of components involved.

These components, each of which could be a bottleneck, are:

- PLC: depending on the firmware/hardware, the limitations may vary. The cycle time of the application may also have a large impact on the communications performance, because the PLC will process a predefined maximum number of requests per cycle.
- Network (PLC/OFS)
- OFS server: depending on its configuration and the application, OFS can be very demanding of CPU usage.
- OFSOPC Driver: depending on its configuration, it can affect the speed of communications.

These tuning suggestions only deals with the OFS/OFSOPC driver communications. For tuning other components, please refer to other relevant documents or specifications.

### See Also

[Tuning Parameters](#)

### Tuning Parameters

The implementation of the OFSOPC driver is based on the OPC standards but also takes into account specific enhancements in the current implementation of OFS that optimizes communications with Schneider Electric PLCs.

This section explains how to optimize the communications between PLCs and a Plant SCADA system through OFS. It presents generic principles to be adapted for each project depending on its specific needs.

It is written to optimize a configuration with OFSOPC driver 1.02.12.001 or greater, and OFS Server 3.35 or greater.

Tuning for performance consists of adjusting several configurations in both the [OFS Server](#) and the [OFSOPC Driver](#).

### OFS Server Configuration

The values provided in this section are recommended default values for SCADA usage. They can be adapted depending on project needs.

## OFS version

Use the latest version of OFS that is available. This information targets OFS V3.35, or greater.

## Processor affinity

In multi-processor machines, set the processor affinity of the OFS server so that it does not run on the same processor as the Plant SCADA I/O Server.

## Device Settings

These settings can be adjusted on a device-by-device basis (see [Device Settings](#)).

## Communications settings

- Sampling rate on reception
- Minimum group update rate
- QuickSetActive
- Communication Overrun behavior

See [Communications Settings](#).

## Summary of recommended OFS Configuration parameters

In summary, the recommended OFS parameter settings for optimal performance are as follows:

Parameter	Recommended Setting
Communication Overrun behavior	Rate adapt
MaxChannels	Equipment dependent, set to the same as the MaxPending value calculated by OFS.
MaxPending	0 (Equipment dependent. OFS to calculate a value).
Minimum group update rate	Rate <= Fastest Group rate / 2, and Rate is the greatest common divisor of the group rates (provided this doesn't violate the previous rule).
OFS version	3.35 or later
Processor affinity	Different to Plant SCADA I/O server
QuickSetActive	True
Sampling rate on reception	20ms

# Device Settings

## MaxPending

By default, in the OFS configuration, "MaxChannels" is set to 1, and "MaxPending" is set to 0.

A setting of MaxPending=0 means the OFS server automatically detects the PLC's target communications port and decides how many parallel requests may be sent at the same time. OFS will choose the appropriate MaxPending value using a predefined table of communications modules (NOE/COPRO/TSY...).

The runtime value being used by OFS can be seen in the OFS server, network window.

It is preferable to let OFS choose the value of the MaxPending parameter, unless a new release of firmware (after the last OFS server release) changes the capability of the device.

In case of multiple OFS connections to the same PLC (redundant IO Server), it is recommended to set MaxPending to "Used" value / Number of connection. The automatic calculation done by OFS is based on one single OFS-PLC connection.

## Max Channels

For Ethernet communications, it is advisable to set "MaxChannels" to the same value as the "MaxPending" value currently in operation in the OFS server.

Setting the MaxChannels parameter should be done after MaxPending has been optimized according to the recommendations above.

# Communications Settings

## Sampling rate on reception

The lower the value of this parameter, the better the response times for synchronous access will be. However, this parameter only affects CPU load on the PC on which OFS is installed and has no impact on the network bandwidth usage.

Typically, the sampling rate on reception should be set to 20ms.

## Minimum group update rate

OFS manages communications with devices based on the group update rate. This rate is the limit below which OFS will not accept OPC client requests.

To maximize performance, it is recommended to set this rate according to the following rules:

Rate  $\leq$  Fastest Group rate / 2, and

Rate is the greatest common divisor of the group rates (provided this doesn't violate the previous rule)

For example:

Group1UpdateRate = 400 ms

Group2UpdateRate = 800 ms

Group3UpdateRate = 2000 ms

Therefore, the recommended group update rate is 200 ms, (the greatest common divisor of the three group rates, which is also <= 400 /2).

**Note:** If your group rate is too high compared to your system capabilities then OFS will display a message "Polling rate overrun for ..." in OFS diagnostic window when "verbose" mode is set (Menu Misc/verbose).

The Quality of some of the items will then become uncertain (depending on Communication Overrun behavior). This quality will be propagated up to the Plant SCADA system, and therefore should be avoided.

Therefore a general rule is to set the OPC client's refresh rate to twice the calculated "OFS group scan time".

---

The OFS group scan time can be calculated with the formula:

TS=group average time (2) \* group number request (3) / max pending request used (1).

The different parameters of the equation are provided by the OFS server in "Varman" and "Network" windows.

## QuickSetActive

This parameter is new in OFS v3.35. It causes the value of an item to be read immediately after activation instead of waiting for the next cycle. This is useful, for example, in reducing the delay for first display of variables in a popup. Therefore, it should be set active.

## Communication Overrun behavior

This parameter allows deciding whether the overrun in OFS should be notified to Plant SCADA as a "Bad quality" or if the server should instead try to auto-adapt its own polling rate. Set this parameter to "rate adapt" to avoid some false "bad quality" when the load on the OFS server is high.

## OFSOPC Driver Parameters

### OFSOPC Driver Version

Use the latest version of OFSOPC that is available. This document targets OFSOPC 1.2.x, or greater.

## Group Management

The OFSOPC driver defines 3 groups per PLC. These groups are polled at a rate defined by the Group<i>UpdateRate parameter in citect.ini. Set these poll rates such that:

- Group1UpdateRate < Group2UpdateRate < Group3UpdateRate
- Group1UpdateRate < Alarm scan Time
- Group2UpdateRate < Trend sampling Time

The "Group Minimum Update rate" is different for each of these groups and can be set using OFS parameters.

The default values are:

Group1UpdateRate = 400 ms

Group2UpdateRate = 1000 ms

Group3UpdateRate = 3000 ms

Each requested tag will be activated in the group which provides the lowest (e.g. fastest) update rate that is faster than the requested rate. For example, a tag with a requested rate of 2s will be activated in Group 2 if the default values are being used.

If no group provides such a rate then the fastest one is used. For example tags requested at 200 ms with default configuration will be put in 400 ms group.

Usually Group 1 (the fastest group) is associated with tags in the active page and alarms. The Group 2 should be targeted for trends and the group 3 is mostly there for slow trends if they are configured.

With the default Plant SCADA configuration, the Group1 (400 ms) will contain alarms ([Alarm] ScanTime =500ms) and page items ([Page] ScanTime =250ms).

See [Group Management Parameters](#).

## Logging

The OFSOPC driver can log combinations of trace levels across different categories. This is achieved by setting the following Citect.ini parameters:

[OFSOPC]DebugLevel

[OFSOPC]DebugCategory

These settings have an impact on the performance of the driver and therefore it is recommended to switch them off by removing these parameters from the citect.ini file to get optimal performance.

See [Logging Parameters](#).

## MaxPending

This parameter defines the maximum number of pending commands that the OFSOPC driver holds for immediate execution. The default value is now 1200. If this value exists in Citect.ini, it should be updated (or removed to take into account this new default value).

This parameter is common to other drivers it is defined under [OPC] and not [OFSOPC] section in the Citect.ini file.

## Summary of recommended OFSOPC Configuration parameters

In summary, the recommended driver parameters settings for optimal performance are as follows:

Parameter	Recommended Setting
Group management [OFSOPC]Group1UpdateRate=G1 [OFSOPC]Group2UpdateRate=G2 [OFSOPC]Group3UpdateRate=G3	G1< G2< G3 G1 < Alarm scan time G2 < Trend sampling time G3 < slow trend sampling time The above are multiple number of the "Group Minimum Update Rate" in OFS parameters.
Logging [OFSOPC]DebugLevel	Remove/disable

Parameter	Recommended Setting
[OFSOPC]DebugCategory	
Commands	Remove/disable
[OPC]MaxPending	

## Troubleshooting

The following topics provide information about the Plant SCADA tools available to diagnose and resolve any unexpected behavior within the runtime system.

[Driver Errors](#)

[Driver Statistics](#)

[Logging](#)

### Driver Errors

Plant SCADA has two kinds of protocol driver errors: generic and specific.

- **Generic errors** are hardware errors 0-31, which are common to many protocols.
- **Specific errors** are unique to a particular driver.

When a driver specific error occurs, the hardware alarm system will not recognize it. The driver converts the specific error into a generic error that can be identified by the I/O server.

This means a driver error may be represented by a specific error, and a corresponding generic error.

When a hardware error occurs, Plant SCADA generates an alarm and displays it on the hardware alarm page. To see the error number, you need to have Alarm Category 255 defined with a display format that includes {ErrDesc,15} {ErrPage,10}.

Refer to the Plant SCADA Help for more information on generic hardware errors.

See the topic [OFSOPC Driver Specific Errors](#) for information on the errors specific to the OFSOPC driver.

If you need more information to resolve an error, refer to the documentation that accompanied the I/O device (or network). If, after reviewing the documentation, you still cannot solve your problem, contact Technical Support for this product.

### OFSOPC Driver Specific Errors

The following errors, listed in (hexadecimal) order, are specific to the Plant SCADA OFSOPC driver.

Error Value (Hex)	Error Definition	Error Description
0XC0040001	GENERIC_SOFTWARE_ERROR	Invalid handle was passed.
0XC0040002	GENERIC_SOFTWARE_ERROR	A duplicate parameter was passed where one is not allowed.

0xC0040004	GENERIC_INVALID_DATA_FORMAT	Server cannot convert between the passed or requested data type and the canonical type.
0xC0040006	GENERIC_ACCESS_VIOLATION	Item's access rights do not allow the operation.
0xC0040007	GENERIC_SOFTWARE_ERROR	Item definition does not exist within the server's address space.
0xC0040008	GENERIC_SOFTWARE_ERROR	Item definition does not conform to the server's syntax.
0xC004000B	GENERIC_INVALID_DATA_FORMAT	Value passed to WRITE was out of range.
0x80020008	GENERIC_SOFTWARE_ERROR	Bad variable type.
0x8002000A	GENERIC_SOFTWARE_ERROR	Out of present range.
0x80020005	GENERIC_SOFTWARE_ERROR	Type mismatch.
0x100	GENERIC_SOFTWARE_ERROR	Could not access variable dbf.
0x101	DRIVER_BAD_DATA	Read of data value bad.
0x102	GENERIC_GENERAL_ERROR	Write of one or more items not completed.
0x103	DRIVER_UNIT_OFFLINE	Could not resolve the server CLASSID.
0x104	GENERIC_SOFTWARE_ERROR	Could not add one or more items to the Server
0x80010006	GENERIC_CHANNEL_OFFLINE	Connection terminated
0x80010007	GENERIC_CHANNEL_OFFLINE	Server not available.
0x8001000F	GENERIC_INVALID_RESPONSE	Received data is invalid.
0x80010012	GENERIC_SOFTWARE_ERROR	Call did not execute.
0x80010100	GENERIC_SOFTWARE_ERROR	System call aborted.
0x80010101	GENERIC_SOFTWARE_ERROR	Could not allocate some required resource.
0x80010103	GENERIC_BAD_PARAMETER	Requested interface not registered on server object.
0x80010104	GENERIC_SOFTWARE_ERROR	Could not call server.

0x80010105	GENERIC_CHANNEL_OFFLINE	Server threw an exception.
0x80010108	GENERIC_CHANNEL_OFFLINE	Server has disconnected from its clients.
0x8001011B	GENERIC_CHANNEL_OFFLINE	Access denied.
0x8001011C	GENERIC_CHANNEL_OFFLINE	Remote calls not allowed for this process.

## Driver Statistics

You can use the following driver statistic counters to help you debug the OFSOPC driver. These statistics can be viewed within the Plant SCADA Kernel using the Page Driver command and then by pressing the 'v' key to select verbose mode.

See the topic *Runtime > Debug Runtime > The Kernel > Kernel Commands > Page Driver* in the main Plant SCADA help for more information.

Statistic	Description
Total Tags	Number of tags added to the OFSOPC driver, this might include the virtual tags.
# Uncertain Reads	The number of reads that cannot be completed immediately due to a stale cache value. When a tag is read that has uncertain quality, it will trigger item activation on the OFS server, allowing diagnosis of the tag status.
# OPC Item Updates	Number of items updated from OFS server via OnDataChange event.
# Read Unsuccessful	Number of unsuccessful ReadDCBs.
# Writes Unsuccessful	Number of unsuccessful WriteDCBs.
Pending DCB Size	Number of DCB not replied.
Pending Tasks	Number of tasks requested by the front-end and not served by the back-end driver.
Pending Write Tasks	Number of WriteDCBs not replied.
Completed Tasks	Number of tasks completed in the back-end.
Configured OPC Items	Number of OPC items added to the OFS server.
Bad OPC Items	Number of OPC items add to the OFS server unsuccessful.

Statistic	Description
Inactive Items	Number of deactivated OPC items (representing a total of the inactive items in groups 1, 2 and 3).
Temp Group Items	Number of items assigned to OPC temporary group. (This is the internal group used by the OFSOPC driver to hold temporary subscriptions.)
Group1 Items	Number of items assigned to OPC group 1.
Group2 Items	Number of items assigned to OPC group 2.
Group3 Items	Number of items assigned to OPC group 3.
Push Data Items	Number of items assigned to OPC push-data group. An item is considered a push-data item if its address is defined within the configured push-data address range.
Batch Write Units	Number of units that are currently have BatchWrite on mode.
Batch Write Items	Numbers of WriteDCBs in batch-write mode.

## Logging

The OFSOPC driver can log combinations of trace levels across different categories. This is achieved by setting the following INI parameters.

### [OFSOPC]DebugLevel

This parameter allows you to define the trace level. The options include:

Option	Description
WARN	Output warning messages.
ERROR	Output error messages.
TRACE	Output trace messages.
DEBUG	Output debug messages.
ALL	All of the above.

### [OFSOPC]DebugCategory

This parameter allows you to enable logging for a particular category of trace. The options include:

Option	Description
TAG	Tag debug.
NET	Network level debug.
PROT	Protocol level debug.
DCB	Front end driver trace.
BUFF	Buffer level debug.
THRD	Thread level debug.
MISC	Any other debugging.
ALL	All of the above.

In both cases, you can use any combination of the available options separated by a pipe character ( | ).

### Example

```
[OFSOPC]
DebugLevel=WARNING|ERROR|TRACE
DebugCategory=ALL
```

In most cases, you can set the DebugLevel and DebugCategory parameters to ALL. If this results in a large, cumbersome log file that causes older data to be overwritten, you can consider refining the debug options.

---

**Note:** These parameters do not have a value specified by default. To enable logging for this driver, you need to manually configure these parameters in your INI file.

The events generated by the OFSOPC driver are logged in the following Plant SCADA syslog file:

`syslog.IOServer.<Cluster name>.<IO Server name>.dat`

This file is located in the directory specified by the Citect.ini parameter **[CtEdit]Logs**. By default, this location will be:

`%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Logs`

The size of the syslog file and its rollover process is configured via the **[Debug]SysLogSize** and **[Debug]SysLogArchive** INI parameters. For more information, see the topic *Debug Parameters* in the Plant SCADA documentation.

---

**Note:** If the number of messages being processed for a driver exceeds the value specified by the Plant SCADA parameter **[Kernel]ErrorBuffers**, the pending messages will not be logged. If this occurs, you can increase the value of this parameter so that all the messages generated by the driver are reported. The number of messages that are not logged can be monitored via the Kernel's General window under the item "Lost Errors". If you adjust **[Kernel]ErrorBuffers** and Lost Errors are still occurring, you may need to adjust the level of system logging.

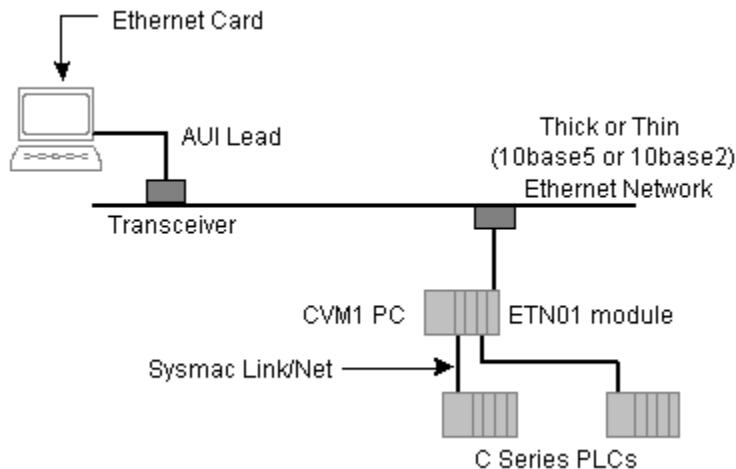
### See Also

[Logging Parameters](#)

## OMFINS Driver

The OMFINS driver allows Plant SCADA to communicate with Omron PLCs via UDP/IP. To enable this type of communication, a PLC needs to be supported by an Ethernet unit, such as a CV500-ETN01 or CV500-ETN21.

Using this method, you can connect to single PLCs or to multiple PLCs as in the following diagram:



**Note:** You can communicate with Omron PLCs using a variety of communication methods and/or protocols. To help decide which method to use for your system, see the topic [Communicating With Omron PLCs](#).

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

### DANGER

**DANGER** indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.

**⚠ WARNING**

**WARNING** indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.

**⚠ CAUTION**

**CAUTION** indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.

**NOTICE**

**NOTICE** used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.

**Please Note**

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

**Before You Begin**

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ WARNING****LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

## Communicating with Omron PLCs

Omron PLCs support a variety of communications methods and protocols including standards such as DeviceNet and Profibus. They also support some native protocols, including:

- Sysmac Way (also called HostLink) - a simple serial protocol supported by every Omron PLC.
- FINS (Factory Intelligent Network Service) - a network protocol using UDP/IP that supports networked PLCs.

Plant SCADA provides the following drivers to support these communication methods:

- **OMRON Driver** - which implements the Sysmac Way serial protocol
- **OMFINS Driver** - which natively implements the FINS protocol
- **FINS Driver** - which uses FINS Gateway middle ware to communicate using the FINS protocol.

To support communication with the FINS gateway middleware, Plant SCADA also includes a low level FINS transport driver.

Complicating the issue further is the fact the range of Omron PLCs is logically split into three distinct series:

Omron PLC Series	PLC Model
Series 1	<b>C Series</b> - including C20, C**P, C**K, C120, C500, C1000H, C2000, C2000H, C**H, CPM1*, CPM2*, CQM1, CQM1H, C200H, C200HS, C200HE, C200HG, C200HX, IDSC and SRM1 PLCs
Series 2	<b>CV, CVM1 Series</b> - including all CV*** and CVM1 CPUs
Series 3	<b>CP, CJ/CS, NSJ Series</b> - including all CP1, CJ1, CS1 and NSJ controllers

The differences between each series is most noted when using Serial communication. Plant SCADA's **OMRON** driver is designed for serial communication with only Series 1 devices. It is based on Omron's native C-Mode HostLink protocol, which is too simplified to support all the tag types required by the Series 2 and 3 devices. Therefore, serial communication with these devices is based on the FINS encapsulation protocol, which requires Plant SCADA's **OMFINS** driver.

Each series also requires Plant SCADA to provide a separate variant of the FINS and OMFINS drivers. Each variant is indicated within a project configuration by a numeric suffix on the driver name, i.e. OMFINS1, OMFINS2, OMFINS3.

The following table should help you decide which Plant SCADA driver you will need to use to communicate with a specific Omron device.

Omron PLC Series	Communication method	Transport Driver	Plant SCADA Driver
Series 1 (C devices)	Serial	COMx	OMRON
	Ethernet	TCP/IP	OMFINS1
	FINS Gateway	FINS	FINS1
Series 2 (CV, CVM1)	Serial	COMx	OMFINS2
	Ethernet	TCP/IP	OMFINS2
	FINS Gateway	FINS	FINS2
Series 3 (CP, CJ/CS, NSJ)	Serial	COMx	OMFINS3
	Ethernet	TCP/IP	OMFINS3
	FINS Gateway	FINS	FINS3

## Omron PLCs via Ethernet

The OMFINS protocol supports UDP/IP communication with the Omron PLCs via the Omron CV500-ETN01/21 Ethernet modules.

Communication is via UDP/IP over Ethernet. The commands within the UDP packets use OMRON Factory Interface Network Service (FINS). The CV-series Ethernet interface uses binary code in its packets - these will be

automatically translated to/from ASCII if passed through a gateway to another SYSMAC network which requires ASCII.

This protocol requires only Winsock TCP/IP software to be installed in your computer. No Omron hardware or software is required on your Plant SCADA workstation. The maximum request length for the OMFINs protocol is 996 words for Ethernet. 512 bytes if also going via other OMRON networks.

The CV500-ETN01/21 has a DIX connector that requires an AUI cable to attach via a transceiver to the thin or thick Ethernet backbone network cable. By installing other network units such as SYSMAC NET Link (SNT) or SYSMAC LINK (SLK) in the Programmable Controller CPU Bus which also contains an ETN unit, Plant SCADA can communicate with any unit on these other networks using FINS commands with the appropriate node/unit addresses.

The same Ethernet card that is being used for Plant SCADA communication can be used for PLC communication.

## Device Address

The information you enter here will be placed in the Special Options field in the Ports settings, with the following format:

**-Ia -Pn -U**

where:

- **a** = the destination IP address in standard Internet dot format. (For example, 192.9.2.60)
- **n** = the destination port (UDP socket) number. Often one physical port has several virtual ports, used for different purposes. The acceptable range is 1024 to 65535, with the default being 9600.
- **-U** = forces the driver to use UDP rather than TCP.

---

**Note:** Declare a separate port for each CV PC on the network.

---

## Unit Address

The I/O Device Address for a Omron CV Series PC is specified as follows:

**w/n/u** (there are no defaults)

Where

- **w** is the FINS network address (1-127)
- **n** is the FINS node number (1-126)
- **u** is the FINS PC Bus unit number (0-15)

Specify the FINS network address, node number, unit number. The FINS node number must be the same as the IP address node number. The FINS network number must be the same for all ETN's on the Ethernet segment.

Note also there is no relationship between FINS network numbers and IP network numbers.

Some examples:

IO Server IP address	IO Server net mask	PC node number	Resultant PC IP address
a.b.c.d	255.255.255.0	10	a.b.c.10
a.b.c.d	255.255.0.0	10	a.b.0.10

## Hardware Setup

Addressing and communication information must be loaded into the system setup area using CVSS or other programming device. Consult the SYSMAC CV-series Ethernet manual for details of installation of the Ethernet unit (ETN) and options available.

## OMRON CV500-ETN01 Settings

Set the front panel switches to the following:

- UNIT (01-15) note that 00 appears to be used for the CPU (e.g.01)
- NODE(1-126) must be unique for each ETN module(s) sharing a FINS network (e.g 06)

Be certain that the OMFINS Source parameter is correctly set (see [OMFINS Parameters](#)).

## Discussion

In the simplest scenario, the last 8 bits of the IP address are the same as the ETN node number - i.e. in the IP address a.b.c.d, 'd' is the same as the node number. The UDP port number must also be set at installation. The same UDP port must be used for all ETN units on a network, and the default port value is 9600.

Communications is done using the FINS protocol inside UDP/IP packets. The addresses used with FINS are the network number, the node number and unit number. The network number is in the range 1-127 (or 0 for local), the node number in the range 1-126 is unique for each Programmable Controller on the network (which may include interconnections to other OMRON networks as well as Ethernet ones). The unit number in the range 0-15 is unique for each CPU bus unit (including the ETN unit) installed in a particular Programmable Controller.

The node number and unit number are set by front panel switches, while the network number is configured with CVSS. As well as supporting FINS over UDP, the ETN unit supports standard TCP, FTP and ICMP (ping).

## Plant SCADA Computer Setup

Set up your specific Ethernet card as required and confirm basic network communications before proceeding with the communications configuration for the OMFINS driver. Refer to the documentation accompanying your hardware for instruction.

---

**Note:** Setup the UDP/IP protocol on your Plant SCADA computer.

---

## Communication Settings

To establish communication with a device, configure the associated Board, Port and I/O Devices in Plant SCADA Studio's **Topology** activity. If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure or verify these settings, use the values outlined below.

## Board Settings

Field	Value
Board Type	Enter TCPIP.
Address	Enter 0.
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.

## Port Settings

Field	Value
Port Number	Leave this field blank.
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank.
Special Options	<p>Enter the destination IP address of Ethernet module. Use the following format:</p> <p><b>-la -Pn -U</b></p> <p>where:</p> <p><b>a</b> = the destination IP address in standard Internet dot format. (For example 192.9.2.60)</p> <p><b>n</b> = the destination port (UDP socket) number. Often one physical port has several virtual ports, used for different purposes. Use this option only if you want to override the default of 9600. The acceptable range is 1024 to 65535.</p> <p><b>-U</b> = forces the driver to use UDP , rather than TCP. Only use this if required.</p> <p>A separate Port should be declared for each CV PC on the network.</p>

## I/O Device Settings

Field	Value
I/O Device Address	<p>The station address for an Omron CV Series PC is specified as follows:</p> <p><b>w/n/u/V</b> for CV mode</p> <p>or</p> <p><b>w/n/u/S</b> for CS1 mode</p> <p>where</p> <p><b>w</b> is the FINS network address (1-127)</p> <p><b>n</b> is the FINS node number (1-126)</p> <p><b>u</b> is the FINS unit address (0)</p> <p>There are no defaults for w, n, or u, but if you do not specify a mode (with a V or S), it will default to CV mode.</p> <p>The FINS node number should be the same as the IP address node number. The FINS network number should be the same for all ETNs on the Ethernet segment.</p> <p>There is no relationship between FINS network numbers and IP network numbers.</p>
I/O Device Protocol	Enter OMFINS

## Examples:

IO Server IP address	IO Server net mask	PC node number	Resultant PC IP address
a.b.c.d	255.255.255.0	10	a.b.c.10
a.b.c.d	255.255.0.0	10	a.b.0.10

## Data Types

Omron's range of PLCs is logically split into three distinct series:

Omron PLC Series	PLC Model
Series 1	<b>C Series</b> - including C20, C**P, C**K, C120, C500, C1000H, C2000, C2000H, C**H, CPM1*, CPM2*, CQM1, CQM1H, C200H, C200HS, C200HE, C200HG, C200HX, IDSC and SRM1 PLCs
Series 2	<b>CV, CVM1 Series</b> - including all CV*** and CVM1 CPUs
Series 3	<b>CP, CJ/CS, NSJ Series</b> - including all CP1, CJ1, CS1 and

Omron PLC Series	PLC Model
	NSJ controllers

Each series has the same memory areas, but different ranges. For this reason, the supported data types for each series are tabled separately.

- Omron 'Series 1' Devices
- Omron 'Series 2' Devices
- Omron 'Series 3' Devices

The following notation is used within the tables:

Enclosing square brackets ([ ]) mean the content is optional and may be omitted.

In some cases, however (where the native tag type can be either DIGITAL or INT), the inclusion of optional content may define the tag type. For example:

- CIO123 is INTEGER
- CIO123.9 becomes DIGITAL (i.e. [.9] changes the tag's type)

**Native type** means default type for a given tag address.

**No Write Command** means that the tag is read-only.

**S.MODE** tag can be written to (unlike other status tags). The allowable write values are "Go to RUN mode" and "Go to MONITOR mode". For the OMFINS driver, RUN=2 and MONITOR=4.

### Omron 'Series 1' Devices

Data Type	Address Format	Native Type	Range
Status (word 1 byte 1) Status (word 1 byte 2) Status (words 3 - 5)	S.STATUS S.MODE S.FATAL S.NFATAL S.ERRCODE	INT (anything else will give #COM)	
Status (error message)	S.ERRMSG	STRING	
CIO Area	[CIO]x[.y]	INT / DIGITAL	x = 0-511 y = 0-15
LR Area	L[R]x[.y]	INT / DIGITAL	x = 0-63 y = 0-15

Data Type	Address Format	Native Type	Range
HR Area	H[R]x[y]	INT / DIGITAL	x = 0-99 y = 0-15
AR Area	A[R]x[y]	INT / DIGITAL	x = 0-27 y = 0-15
Timer Area Completion Flag	TFx	DIGITAL	x = 0-511
Counter Area Completion Flag	CFx	DIGITAL	x = 0-511
Timer Area PV	Tx	INT	x = 0-511
Counter Area PV	Cx	INT	x = 0-511
DM Area	D[M]x[y]	INT / DIGITAL	x = 0-9999 y = 0-15
Expansion DM Area Current Bank	E[M]x[y]	INT / DIGITAL	x = 0-6143 y = 0-15
Expansion DM Area Any Bank	E[M]b:x[y]	INT / DIGITAL	b = 0-15 x = 0-6143 y = 0-15
EM Area Current Bank Number	EMN	INT / DIGITAL	

**Omron 'Series 2' Devices**

Data Type	Address Format	Native Type	Range
Status (word 1 byte 1) Status (word 1 byte 2) Status (words 3 – 6)	S.STATUS S.MODE S.FATAL	INT (anything else will give #COM)	

Data Type	Address Format	Native Type	Range
	S.NFATAL S.MSG S.ERRCODE		
Status (error message)	S.ERRMSG	STRING	
CIO Area	[CIO]x[.y]	INT / DIGITAL	x = 0-2555 y = 0-15
LR Area	L[R]x[.y]	INT / DIGITAL	x = 0-199 y = 0-15
AR Area	A[R]x[.y]	INT / DIGITAL	x = 0-511 y = 0-15
Timer Area Completion Flag	TFx	DIGITAL	x = 0-1023
Counter Area Completion Flag	CFx	DIGITAL	x = 0-1023
Timer Area PV	Tx	INT	x = 0-1023
Counter Area PV	Cx	INT	x = 0-1023
DM Area	D[M]x[.y]	INT / DIGITAL	x = 0-24575 y = 0-15

Data Type	Address Format	Native Type	Range
CPU Bus Link Area	Gx[.y]	INT / DIGITAL	x = 0-255 y = 0-15
TR Area	TR[.y]	INT / DIGITAL	y = 0-15
Step Area	STx	INT	x = 0-1023
Step Area Flag	STFx	DIGITAL	x = 0-1023
Transition Area Flag	TNx	DIGITAL	x = 0-1023
Action Area Flag	ACx	DIGITAL	x = 0-2047
Data Register	DRx	INT	x = 0-2
Index Register	IRx	INT	x = 0-2
Expansion DM Area Current Bank	E[M]x[.y]	INT / DIGITAL	x = 0-32765 y = 0-15
Expansion DM Area Any Bank	E[M]b:x[.y]	INT / DIGITAL	b = 0-7 x = 0-32765 y = 0-15

Data Type	Address Format	Native Type	Range
Expansion DM Area Current Bank Number	EMN	INT	

**Omron 'Series 3' Devices**

Data Type	Address Format	Native Type	Range
Status (word 1 byte 1) Status (word 1 byte 2) Status (words 3 – 6)	S.STATUS S.MODES.FATAL S.NFATAL S.MSG S.ERRCODE	INT (anything else will give #COM)	
Status (error message)	S.ERRMSG	STRING	
CIO Area	[CIO]x[.y]	INT / DIGITAL	x = 0-6143 y = 0-15
LR Area	L[R]x[.y]	INT / DIGITAL	x = 0-199 y = 0-15
HR Area	H[R]x[.y]	INT / DIGITAL	x = 0-511 y = 0-15
AR Area	A[R]x[.y]	INT / DIGITAL	x = 0-959 y = 0-15
Timer Area Completion Flag	TFx	DIGITAL	x = 0-4095
Counter Area Completion Flag	CFx	DIGITAL	x = 0-4095

Data Type	Address Format	Native Type	Range
Timer Area PV	Tx	INT	x = 0-4095
Counter Area PV	Cx	INT	x = 0-4095
DM Area	D[M]x[y]	INT / DIGITAL	x = 0-32767 y = 0-15
Data Register	DRx	INT	x = 0-15
Index Register	IRx	LONG	x = 0-15
Expansion DM Area Current Bank	E[M]x[y]	INT / DIGITAL	x = 0-32767 y = 0-15
Expansion DM Area Any Bank	E[M]b:x[y]	INT / DIGITAL	b = 0-12 x = 0-32767 y = 0-15
Expansion DM Area Current Bank	EMN	INT	
Work Area	W[R]x[y]	INT / DIGITAL	x = 0-511 y = 0-15
Task Flag	TKx	DIGITAL	x = 0-31

## OMFINS Parameters

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

## [OMFINS]Block

A value (bytes) used by the I/O Server to determine if two or more packets can be blocked into one data request before being sent to the I/O Device. For example, if you set the value to 10, and the I/O Server receives two simultaneous data requests - one for byte 3, and another for byte 8 - the two requests will be blocked into a single physical data request packet. This single request packet is then sent to the I/O Device, saving on bandwidth and processing.

**Allowable Values** 5 to 256

**Default Value** 64

## [OMFINS]Delay

The period (in milliseconds) to wait between receiving a response and sending the next command.

**Allowable Values** 0 to 300 (milliseconds)

**Default Value** 0

## [OMFINS]IgnoreNonFatalError

Determines whether Plant SCADA will stay online if the remote rack is powered off.

**Allowable Values** 0 (recoverable errors not ignored) or 1 (recoverable errors ignored)

**Default Value** 0

## [OMFINS]MaxPending

The maximum number of pending commands that the driver holds ready for immediate execution.

**Allowable Values** 1 to 32

**Default Value** 2

## [OMFINS]PollTime

The interrupt or polling service time (in milliseconds). Setting the polling time to 0 puts the driver in interrupt mode.

**Allowable Values** 0 to 300 (milliseconds)

**Default Value** 0

## [OMFINS]Retry

The number of times to retry a command after a timeout.

**Allowable Values** 0 to 8

**Default Value** 1

## [OMFINS]Source

Specify the FINS network address, node number and unit number for the Plant SCADA I/O server. This will be the source address of messages from Plant SCADA, and the destination address of responses from the PCs.

**Format** w/n/u where:

w is s the FINS network address (1-127)

n is the FINS node number (1-126)

u is the FINS unit number (0-15)

**Default Value** 1/1/0

## [OMFINS]Timeout

Specifies how many milliseconds to wait for a response before displaying an error message.

**Allowable Values** 0 to 32000 (milliseconds)

**Default Value** 1000

## [OMFINS]WatchTime

The frequency (in seconds) that the driver uses to check the communications link to the I/O Device.

**Allowable Values** 0 to 128 (seconds)

**Default Value** 30

## Port/IO Device-specific Parameters

To apply a parameter globally to every device, place it under the [OMFINS] section of the Citect.ini file.

To apply a parameter to a specific device, or every device connected to a specific port, use one of the following sections in the Citect.ini file:

- [OMFINS.<PortName>]
  - <PortName> is the name specified for the port in your Plant SCADA project.  
For example, parameters set within the section [OMFINS.PrimaryPort1] impact every device connected to the port defined as "PrimaryPort1".
- [OMFINS.<PortName>.<DeviceName>]
  - <PortName> defines the name specified for the port in your Plant SCADA project.
  - <DeviceName> defines the name of the I/O device.  
For example, parameters set within the section [OMFINS.PrimaryPort1.PrimaryIODev128] only impact

the device connected to the port defined as "PrimaryPort1" and named "PrimaryIODev128."

You can set the following parameters for a specific I/O device:

- Source
- Block
- Delay
- IgnoreNonFatalError

You can set the following parameters for a specific port:

- Source
- Block
- Delay
- IgnoreNonFatalError
- Retry
- Timeout

## Example

The following **Citect.ini** file format is an example of specifying the source parameter differently for different I/O devices communicating with the OMFINS PLC.

Assume that there are two ports in use: PORT1 and PORT2.

- PORT1 has two I/O devices attached: DEV1A and DEV1B.
- PORT2 has two I/O devices attached: DEV2A and DEV2B.

The **Citect.ini** file contains the following entries:

```
[OMFINS]
Source = 1/0/0
[OMFINS.PORT1]
Source =1/0/1
[OMFINS.PORT2]
Source =1/1/0
[OMFINS.PORT1.DEV1A]
Source =0/1/0
[OMFINS.PORT2.DEV2B]
Source =1/1/1
```

The resultant sources for the I/O devices are as follows:

I/O Device	Source	A result of
DEV1A	0/1/0	[OMFINS.PORT1.DEV1A]
DEV1B	1/0/1	[OMFINS.PORT1]
DEV2A	1/1/0	[OMFINS.PORT2]
DEV2B	1/1/1	[OMFINS.PORT2.DEV2B]

## Protocol-specific Parameters

Plant SCADA uses three variations of the OMFINS driver to support the three Omron PLC Series (see [Communicating with Omron PLCs](#)).

To apply a parameter to a specific protocol version, place it under the following section of the Citect.ini file:

[OMFINSx]

Where:

- x = 1 (Series 1)
- x = 2 (Series 2)
- x = 3 (Series 3)

For example, parameters set within the section [OMFINS2] will only impact the device defined to use the OMFINS2 protocol variant.

You can set the following parameters for a specific protocol:

- MaxPending
- PollTime
- WatchTime

See [OMFINS Parameters](#).

## OMFINS Driver Errors

In addition to standard driver errors, the OMFINS driver may use the following driver errors:

Driver Error	Explanation
<b>0x10001</b> (Bad response)	this error means the amount of data received back does not match what was requested (e.g. requested was for two LONGs, but received 7 bytes back instead of 8 bytes).
<b>0x10002</b> (Bad request)	this error indicates that you are trying to write the wrong amount of data for a given type

Driver Error	Explanation
<b>0x10003</b> (Bad tag type)	this error indicates a request cannot be serviced because the tag is not of the native type (e.g. S.MODE must be an INT, but it was configured as a LONG)

## OMFINS Specific Errors

When a hardware error occurs, Plant SCADA generates an alarm, and displays the alarm on the hardware alarm page (in the alarm description of the hardware alarm). To see the error number, you need to have Alarm Category 255 defined with a display format that includes {ErrDesc,15} {ErrPage,10}.

There are a number of generic errors that are common to many protocols. In some cases, only the generic error is available, though often both the generic error and a specific error are given.

The following errors, listed in (hexadecimal) sequence, are specific to this protocol. Plant SCADA displays the error number and description for common protocol-specific errors. Uncommon errors are not contained in the Plant SCADA error database, in which case Plant SCADA will only display the error number.

You may require additional information to enable you to rectify an error. This information should be detailed in the documentation that accompanied the I/O Device (or network). If, after reviewing the documentation, you cannot rectify an error, contact Technical Support for this product.

Messages return a FINS response code. This is two bytes, the main response code and the sub-response code. Response codes are shown in the second table presented. If bit 6 or 7 of the sub-response code is on, the error has occurred in the PC returning the response. If the main response code has bit 7 set, a network relay error has occurred - in this case a further two words will be returned to the driver indicating the location of the error. In the following table response codes are listed in hexadecimal, with xx indicating any value here will return the same generic errors. The two-byte response code will be returned as the Plant SCADA driver error (except for 0001, which will be returned as -1).

Response codes (hex)	Plant SCADA generic error
0000	NOERROR
0001 (returned as -1)	INVALID_RESPONSE
0101, 0102, 0105, 0106, 0201, 0202, 0203, 0304, 0402, 05xx, 20xx	GENERAL_ERROR
0103, 0104, 0204, 0205, 25xx	UNIT_TIMEOUT
0301, 0303, 0302, 22xx, 2301, 2303, 24xx	HARDWARE_ERROR
0401, 10xx, 11xx, 2302	INVALID_COMMAND
21xx	WRITE_PROTECT

Main Code	Sub-Code	Probable Cause	Remedy
00: Normal Completion	00	---	---
01:Local node error	03	Send error in local node was caused by lack of available space in internal buffers	Reduce the load (traffic) on the Ethernet Unit. Check your user applications.
02: Remote node error	01	IP address of remote node not set correctly in IP address table or IP router table	Set IP address of remote node into IP address table and, if inter network transmission is required, into the IP router table.
	02	No node with the specified unit address	Check the remote node's unit address and make sure the correct one is being used in the control data.
	05	Message packet was corrupted by transmission error.	Check the protocol and controller status by reading them with FINS commands. Increase the number of transmit retry attempts.
		Response time-out, response watchdog timer interval too short.	Increase the value for the response monitor time in the control data.
		The transmission frame may be corrupted or the internal reception buffer full.	Read out the error log and correct as required.
03: Communications controller error	01	Error occurred in the communications controller, ERC indicator is lit.	Take corrective action, referring to troubleshooting procedures in this section

	02	CPU error occurred in the PC at the remote node.	Check CPU indicators at the remote node and clear the error in the CPU (refer to the PC's operation manuals)
	04	Unit number setting error	Check that the unit number is specified within range and that the same unit number is not used twice in the same network.
04: Not executable (service not supported)	01	An undefined command has been used.	Check the command code is supported by the Unit to which you are sending it
		A short frame (4 bytes) is being used for the FINS header frame.	Check the FINS header frame length. The Ethernet Unit does not support short headers.
05: Routing error	01	Remote node is not set in the routing tables	Set the remote node in the routing tables
	02	Routing tables aren't registered completely	Set routing tables at the local node, remote node, and any relay nodes.
	03	Routing table error	Set the routing tables correctly
	04	The maximum number of relay nodes (2) was exceeded in the command.	Redesign the network or reconsider the routing table to reduce the number of relay nodes in the command. Communications are possible on three network levels, including the local network.
10: Command format	01	The command is longer	Check the command

error		than the max. permissible length.	format of the command and set it correctly. Verify that broadcast transmissions don't exceed 1,473 bytes.
	02	The command is shorter than minimum permissible length	Check the command format of the command and set it correctly
	03	The designated number of data items differs from the actual number in the command data.	Check the number of items and the data agree
	05	Data for another node on the same network was received from the network	Check the header parameters in the command data, and that the correct command format is being used.
		An attempt was made to send response data for a broadcast address.	
11: Parameter error	00	The parameters in the command data were incorrect or the UDP/TCP socket number was not within the proper range.	Check the parameters, and that the socket number is between 1 and 8.
	01	A correct memory area code has not been used or Expansion Data Memory is not available	Check the command's memory area code and set the appropriate code.
	03	The first word is in an inaccessible area or the bit number is no 00.	Set a first word that is in an accessible area. The bit number needs to be 00 for Ethernet Units.
22: Status error	0F	The same socket service is already in progress at	Use the socket status flag in PC memory to

		the specified socket number.	check that socket service has finished before starting services again.
	10	The specified socket is not open	Open the socket. For TCP sockets, wait until connection is made.
23: Environment error	05	IP address conversion incomplete	Check the IP address and subnet mask in the System Setup are correct.
	07	IP address conversion is set for automatic conversion only.	Check the mode settings in the System Setup. This error will be generated for the READ IP ADDRESS command only.
25: Unit error	03	I/O setting error (The I/O table differs from the actual Unit configuration.)	Either change the actual configuration to match the registered one, or generate the I/O table again.
	05	CPU bus error (An error occurred during data transfer between the CPU and a CPU Bus Unit.)	Check the Units and cable connections and send the ERROR CLEAR command.
	0A	An error occurred during CPI Bus Unit data transfer.	Check the Units and cable connections and send the ERROR CLEAR command.

See 9-3 Troubleshooting via response codes of SYSMAC CV-series Ethernet System Manual pages 109-114 of FINS commands reference (section 8-3)

### Example:

Plant SCADA Error 18501 response code = 0501, (main response = 5, sub response = 1)

Driver Err: 18501 for an incorrect routing table, with relay error flag set

Refers to the error bit preceding the main response code which should be masked, as per the two bits preceding

the sub response code.

Thus if main = 5, sub = 1 then error 10501

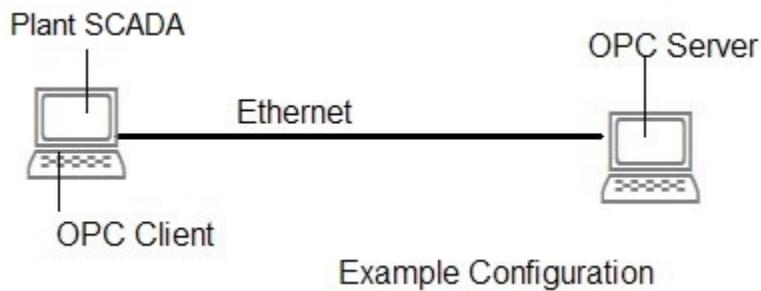
Returned code: [0001][1000][0101][0000][0001] -> 18501

Mask red bits: [1111][0111][1111][0011][1111]

Presented code: [0001][0000][0101][0000][0001] -> 10501

## OPC Driver

The OPC method of communication to OPC devices uses the OPC driver. OPC specifications v1.0a and v2.0 are supported. Using this method you can connect to single or multiple OPC devices, as shown in this example configuration:



The OPC server can also exist on the same computer as the OPC client. Be reminded that the maximum request length (that is, the maximum of data bits that can be read from the I/O device at any one time) for the OPC protocol is 2040.

**Note:** With the release of version 7.20, Plant SCADA supports the retrieval of time-stamped data directly from field devices. This capability is enabled by the Driver Runtime Interface (DRI), a component of Plant SCADA that is used by the OPC driver to directly update time-stamped digital alarms, time-stamped analog alarms and event-based trends. For more information, see the topic *Retrieving Time-stamped Data from Field Devices* in the main Plant SCADA help.

## See Also

- [Preparing the OPC System](#)
- [OPC Devices and Clients](#)
- [Advanced Configuration and Maintenance](#)
- [Troubleshooting](#)

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.
<b>⚠ DANGER</b>	<b>DANGER</b> indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.
<b>⚠ WARNING</b>	<b>WARNING</b> indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.
<b>⚠ CAUTION</b>	<b>CAUTION</b> indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.
<b>NOTICE</b>	NOTICE used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.

## Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

## Before You Begin

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ WARNING****LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

## Preparing the OPC System

This section explains how to prepare your OPC devices for communication with Plant SCADA.

- [Hardware Requirements](#)
- [Software Requirements](#)

### Hardware Requirements

In the OPC architecture, each variety of server is allocated a unique identifier, known as a Class ID. OPC clients use these 128-bit numbers, frequently displayed in the form 6B29FC40-CA47-1067-B31D-00DD010662DA, to access the particular vendor's OPC server. To make it easier for users, these numbers are often referred to by a more manageable string identifier called a ProgID (Program Identifier). These identifiers usually take the form **Vendor.Application** with an optional version number appended; for example, the Plant SCADA OPC Server ProgID is **Plant SCADA.OPC**.

Before the OPC client driver can access a particular OPC server, the server's ProgID needs to be entered into the

Plant SCADA I/O server registry. Where an in-process server or a local server is being used, this should have been done automatically when the server was installed. To access a remote server, refer to the documentation accompanying the server. Frequently, this step is achieved using a registry script (a .REG file) or a small executable.

**Note:**

- Familiarize yourself with the target OPC server(s) before using this driver. This will help you enter the item identifiers for the tag addresses using the correct syntax.
- Using remote servers requires configuring DCOM security. Since configuring security can be complex, you should plan your system settings in advance, and establish a protocol so that the Windows accounts to be used by operators have sufficient privilege. If sufficient privileges are not established, communications errors may result. See [Software Requirements](#) for more information.

## Software Requirements

OPC operates using the Microsoft Distributed COM (DCOM) architecture, which is part of the Windows operating system.

If a distributed OPC system is being accessed (i.e., remote servers), you need to configure DCOM security settings on the remote machines to allow operators access to the servers. To configure these settings, use the program DcomCnfg.exe.

If sufficient privileges are not established, communications errors may result, so take care when configuring your system.

**Note:** You need to confirm with the supplier of the OPC Server you are connecting to that their product is DCOM compliant.

## OPC Devices and Clients

This section provides information required to configure OPC devices and clients within your Plant SCADA project.

- [Device Address Information](#)
- [Communication Settings](#)
- [Data Types](#)
- [Include Project Support](#)
- [Arrays Support](#)
- [Quality Values](#)
- [Timestamp and Quality Support](#)

### Device Address Information

The address to use is the ProgID for the desired server.

This usually takes the form "<Vendor>.<Application>" (e.g. Plant SCADA.OPC) and should be found in the documents accompanying the OPC Server.

## See Also

[Communication Settings](#)

### Communication Settings

To establish communication with a device, configure the associated Boards, Ports and I/O Devices in Plant SCADA Studio's **Topology** activity. The following tables show the recommended settings for communication with an OPC device.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure these settings use the information outlined below.

### Boards

The Boards view in Plant SCADA Studio's **Topology** activity lists all boards used in the Plant SCADA project. Each board record defines a separate board within the project.

Field	Value
Board Name	A unique name (up to 16 alphanumeric character) per server for the computer board being used to connect with the device. This is used by Plant SCADA in the Ports view.
Board Type	The type of board. Enter <b>OPC</b> .
Address	Plant SCADA doesn't need an address for the OPC client. Instead you use the <b>Address</b> box to set the device scan rate on the OPC Server in milliseconds. The value read from the OPC server is typically taken from the OPC server cache. The scan rate value is often used to indicate how often the OPC server polls the device and updates its cache.  Enter <b>0</b> (zero) to use the default of 250ms. Enter any other value to use that value. Valid values range from eight character decimal 0 to 99999999. Up to six character hex values can be used; however, these must be preceded with 0x.  <b>Note:</b> As Plant SCADA supports unsolicited OPC requests, if you set the OPC server to notify Plant SCADA of changes to the value or quality of an OPC device item, make sure the OPC server scan rate is short enough to provide Plant SCADA within an adequate timeframe.
I/O Port	Leave this field blank.

Interrupt	Leave this field blank.
Special Options	<p>OPC server machine name. (Do not enter this name using the traditional dash delimited style. Enter the machine name, for example <b>Cit_Primary</b>.)</p> <p><b>Note:</b> The name entered here is the Windows machine name. For Windows NT and Windows 95 or 98 based OPC servers, you can obtain this from the <b>Identification</b> tab of the OPC server machine's network properties. For other OPC server configurations, refer to the server's documentation.</p> <p>If this field is left blank, the server name is sought in the following manner: a) The in-process (DLL) server on the IO Server machine; b) The local (EXE) server on the I/O server machine; c) The remote server as entered in the I/O server machine's DCOM settings.</p>
Comment	Any useful comment.

## Ports

The Ports view in Plant SCADA Studio's **Topology** activity lists all ports used in the Plant SCADA project. Each port record defines a separate port within the project.

Field	Value
Port Name	A unique name (up to 16 alphanumeric character) for the computer port being used to connect with the device. This is used by Plant SCADA in the Devices dialog box.
Port Number	An integer value, unique to the board as defined in the Board Name field. The Ethernet port needs no identification; however, the Plant SCADA communication database requires a value in this field.
Board Name	The name of the board this port is attached to as defined on the Boards view.
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank.
Special Options	Leave this field blank.

Comment	Any useful comment.
---------	---------------------

## I/O Devices

The I/O Devices view in Plant SCADA Studio's **Topology** activity lists all I/O devices used in the Plant SCADA project. Each record defines a separate I/O device within the project.

Field	Value
Name	A unique name (up to 16 alphanumeric characters) for the I/O device being identified. Each I/O device must have a unique name in the Plant SCADA system.
Number	A unique number for the I/O device (0–16383). Each I/O device must have a unique number in the Plant SCADA system (unless I/O device redundancy is being used).
Address	ProgID for the desired server. These usually take the form Vendor.Application (for example, Plant SCADA.OPC) and should be found in the documents accompanying the OPC Server.
Protocol	Enter "OPC" or "OPC1". See <a href="#">Arrays Support</a> .
Port Name	This field must contain one of the names previously defined in the <b>Port Names</b> field of the Ports view. There must be only one I/O device per port.
Comment	Any useful comment.

## Data Types

OPC servers can provide data in different data formats. An OPC client can request data in a specific format or use the server's type (that is, the type in which the data is stored within the server).

When specifying a data type, consider whether the server will be able to convert data to the requested type.

Use the table below to help you specify a type for a variable tag. It shows mappings from the types used by OPC (VARTYPE) to Plant SCADA data types. If followed, these data types coerce the OPC data Plant SCADA data types.

OPC Type (VARTYPE)	Description	Plant SCADA Data Type
VT_BOOL	Boolean	DIGITAL
VT_UI1	Byte	BYTE
VT_I2	2 byte integer	INT
VT_UI2	2 byte unsigned integer	UINT

VT_I4	4 byte integer	LONG
VT_U14	4 byte unsigned integer	ULONG (requires v7.0)
VT_CY	Currency	STRING
VT_R4	4 byte real	REAL
VT_R8	8 byte real	STRING
VT_DATE	Date	STRING
VT_BSTR	String	STRING
VT_ERROR	Error value	LONG
VT_QUALITY	2 byte integer	INT
VT_TIMESTAMP	4 byte integer	LONG
VT_MILLISECOND	4 byte integer	LONG

## Include Project Support

By default, the OPC driver does not support the definition of variable tags in projects that are included below the main project.

Variable tags defined in a project compiled on a Plant SCADA client needs to match exactly the tags defined in the project compiled on the I/O server. If a tag is added on the client, it needs to also be added on the I/O server. Otherwise client requests may become mismatched and show incorrect data.

## Arrays Support

To associate a Plant SCADA variable tag with an OPC array, follow the OPC array address string with a delimiter like "!", the capital letter "A" (which when combined like this instructs the Plant SCADA OPC driver that the address is for an array), and follow both with the array size as an integer enclosed within square brackets, like this:

<OPCArrayName>!A[<OPCArrayLength>]

---

**Note:** Replace the placeholders <OPCArrayName> and <OPCArrayLength> in the **Address** box with the actual valid values of the OPC Array **Name** and **Length** you want to use, as defined in the OPC server.

---

In the following example with an OPC tag defined on a BACnet OPC server, five register addresses are associated with the Plant SCADA variable tag named Status\_Flags:

- **Variable Tag Name:** Status\_Flags
- **Address:** device(7196).analog-output(1).status!A[5]

You can access individual elements of an OPC array in Plant SCADA by specifying the Plant SCADA tag name and an index number representing the individual element of the array. For example, to refer to the third variable of the array in the above example (Status\_Flags), use this syntax in Plant SCADA:

Variable Tag: Status\_Flags[2]

**Note:** OPC arrays use zero-based indexing. For example, a five register array would contain individual elements indexed as 0, 1, 2, 3, and 4.

The OPC address checking in Plant SCADA always ensures that the !A[<n>] suffix is used, otherwise the tag is not recognized as an OPC array, regardless of whether OPC or OPC1 variant is used. Therefore the format '!A[<n>]' must be followed in a tag address if the tag is defined as an array tag.

For example:

With access path: DB1,DINT0,10!A[10]

Without access path: S7:[OPCFLR]DB1,DINT0,10!A[10]

The OPC1 protocol is used to support Cicode functions TagRead() and TagWrite() for tags with a left square bracket "[" in their address.

However, if there is no left square bracket "[" in the OPC address string, you can use the array length specifier at the end of the address string, and so you should declare OPC as the I/O device protocol used in the I/O device setup procedure.

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

Ensure that the OPC address format used is compliant for the OPC Server with which you are communicating.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## **Array offset support**

The array tag address can include an optional array offset.

The maximum size of an array tag is limited to 256 bytes due to the underlying infrastructure of Plant SCADA. For example, if the tag data type is integer (2 bytes), Plant SCADA only allows the array tag to be configured with 128 elements. However, special array offset syntax can be used to address a sub-array with the particular offset and range within a large array. For example:

<OPCArrayName>\<OFFSET>!A[<OPCArrayLength>]

When Plant SCADA writes to an OPC array, the whole array is read from the driver cache, modified, then written back to the OPC server. However, the driver cache is not updated with the newer data until the server performs its housekeeping tasks and notifies the driver that the array value has been changed. During this time, it is possible for another write request to have been issued with a now out-of-date copy of the original array data from the not-yet-updated driver cache. If subsequently written back to the server, any changes made in the first request of this scenario might be lost inadvertently if they are overwritten by the original duplicated data in the second write back.

To try and minimize this, the OPC driver in Plant SCADA is forced to do a synchronized read before data is modified. However it still has the possibility of losing data if that data is changed by other OPC clients between the time the data is read out and written back. For this reason, it is not recommended that OPC arrays be used in Plant SCADA while other OPC clients might be accessing the same data.

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use OPC arrays in Plant SCADA while other OPC clients have access to the same data.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Quality Values**

The following masks are general bitmasks:

OPC_QUALITY_MASK	0xC0
OPC_STATUS_MASK	0xFC
OPC_LIMIT_MASK	0x03

These are specific quality values:

OPC_QUALITY_BAD	0x00
OPC_QUALITY_UNCERTAIN	0x40
OPC_QUALITY_GOOD	0xC0
OPC_QUALITY_CONFIG_ERROR	0x04
OPC_QUALITY_NOT_CONNECTED	0x08
OPC_QUALITY_DEVICE_FAILURE	0x0C
OPC_QUALITY_SENSOR_FAILURE	0x10
OPC_QUALITY_LAST_KNOWN	0x14
OPC_QUALITY_COMM_FAILURE	0x18
OPC_QUALITY_OUT_OF_SERVICE	0x1C
OPC_QUALITY_LAST_USABLE	0x44
OPC_QUALITY_SENSOR_CAL	0x50
OPC_QUALITY_EGU_EXCEEDED	0x54
OPC_QUALITY_SUB_NORMAL	0x58
OPC_QUALITY_LOCAL_OVERRIDE	0xD8

These can modify any of the above:

OPC_LIMIT_OK	0x00
OPC_LIMIT_LOW	0x01
OPC_LIMIT_HIGH	0x02
OPC_LIMIT_CONST	0x03

## See Also

[Timestamp and Quality Support](#)

### Timestamp and Quality Support

The Plant SCADA OPC driver can use additional variable tags for quality and timestamp values. Quality or timestamp tags are duplicates of a variable tag, with an appended "!Q" or "!T" or "!M" in the address field to define their purpose. Quality tags need to be configured as either INT or DIGITAL data types, whereas Timestamp tags need to be configured as LONG data types.

If you just need to know if an OPC item is good or bad quality (i.e. not uncertain), you should consider using DIGITAL data types for quality tags in combination with [OPC]DigitalQualityUncertainIsBad, rather than using Cicode to compare the value of an INT quality tag against 0xC0.

---

**Note:** The quickest way to create a quality or timestamp tag in Plant SCADA is to locate the relevant variable tag using the Variable Tags dialog box, edit it as required to create a quality or timestamp tag, and save it. This creates a copy of the original variable tag with the required changes saved.

---

#### To create a quality tag:

1. Open the Plant SCADA Studio.
2. In the **Topology** activity, go to **Equipment** tab.
3. In the editor, create a new variable tag, or locate the tag that you want to associate the quality tag with.
4. In the **Name** field, enter an appropriate and unique name for the quality tag, such as "<Tagname>\_Quality".
5. In the **Type** field, choose INT or DIGITAL from the menu.
6. In the **I/O Device** field, select the appropriate I/O device for the tag (if not already displayed).
7. In the **Display Name** box, enter the name of the tag you want to associate with the quality value (if not already displayed), and append "**!Q**" directly to the end of the address (without the quotes and without any spaces).
8. Click **Save**.

---

**Note:** If the exclamation mark character (!) is already defined in the Display Name field (for example, you have already used it to declare an array size "!A[n]"), you either need to include another exclamation mark (e.g. append "!Q" to the display name) after the closing square bracket of the array size declaration, or you can specify quality for arrayTag!A[5] with just arrayTag!Q".

---

#### To create a timestamp tag:

1. Open the Plant SCADA Studio.

2. In the **Topology** activity, go to **Equipment** tab.
3. In the editor, create a new variable tag, or locate the tag that you want to associate the quality tag with.
4. In the **Name** field, enter an appropriate and unique name for the timestamp tag, such as <Tagname>\_sTimestamp (for seconds) or <Tagname>\_msTimestamp (for milliseconds).
5. In the **Type** field, choose **Long** from the menu (if not already selected).
6. In the **I/O Device** field, select the appropriate I/O device for the tag (if not already displayed).
7. In the **Display Name box**, enter the name of the tag you want to associate with a timestamp value (if not already displayed), and append "**!T**" or "**!M**" (as appropriate).

**Note:** If there is already an exclamation mark with other declarations included in the display name, you don't need to include them for timestamping. For example, a timestamp tag monitoring TagName!A[64] can be addressed as TagName!T.

8. Click **Add**.

---

**Note:** Timestamps store the number of seconds since 01/01/1970 when using the "**!T**" element, or the number of milliseconds since midnight using the "**!M**" element. If you want to store a complete time in millisecond accuracy, you need to create two separate timestamp tags, one for seconds and one for milliseconds, each addressed appropriately.

---

## Examples

In the following examples, an OPC device tag defined on a BACnet OPC server is associated with the base Plant SCADA variable tag named **Present\_Value**:

<b>Variable Tag Name</b>	Present_Value
<b>Address</b>	device(7196).analog-output(1).present-value
<b>Plant SCADA Data Type</b>	Data type required by project (for a list of data types, see <a href="#">Data Types</a> ).

### Example 1

Use the following settings to associate a Plant SCADA variable tag with the quality stamp of the OPC register value:

<b>Variable Tag Name</b>	Present_Value_Quality
<b>Address</b>	device(7196).analog-output(1).present-value!Q
<b>OPC Type</b>	VT_QUALITY
<b>Plant SCADA Data Type</b>	INT or DIGITAL

### Example 2

The following tag definition will read the Timestamp value measured in seconds from 1/Jan/1970:

<b>Variable Tag Name</b>	Present_Value_Timestamp
<b>Address</b>	device(7196).analog-output(1).present-value!T

<b>OPC Type</b>	VT_TIMESTAMP
<b>Plant SCADA Data Type</b>	LONG

**Example 3**

The tag definition below will read the Millisecond Timestamp value measured in UTC FileTime from beginning (midnight) of the current day:

<b>Variable Tag Name</b>	Present_Value_Millisecond
<b>Address</b>	device(7196).analog-output(1).present-value!M
<b>OPC Type</b>	VT_MILLISECOND
<b>Plant SCADA Data Type</b>	LONG

The base Plant SCADA tag needs to be defined in Plant SCADA before the timestamp and quality values of the associated OPC Server tag value can be read.

**See Also**

[Quality Values](#)

## Advanced Configuration and Maintenance

This section provides advanced configuration instructions.

- [Status Tags](#)
- [Customizing a Project using Citect.ini Parameters](#)
- [Configuring Redundancy](#)
- [Scan Rates](#)

### Status Tags

You can define status tags in your Citect.ini file to allow the OPC driver to determine the current state of an OPC device. For example, a status tag could be configured to monitor an OPC item that indicates the current operational state of a connected device. If the condition of the status tag is true, the OPC driver can allow the device to come online; otherwise the device may be set to offline.

By default, no status tags are created for the OPC driver. They need to be defined in the project Citect.ini file using the OPC status tag parameters. These parameters determine which OPC items are monitored, the condition values, and how often polling occurs.

A status tag can be defined in the [OPCStatusTags] section of the Citect.INI file to determine the online status of OPC devices. You can also create status tag set for each individual I/O device using the following format:

```
[OPCStatusTags]
<IOServerName>.<IODeviceName>.<parameter>=<value>
```

For example:

```
[OPCStatusTags]
io1.iodev1.tag=ARB3183800_S_2.REQNUM
io1.iodev1.condition= <=10
```

Where "ARB3183800\_S\_2.REQNUM" refers to an OPC item configured in the device.

If you are not regularly reading from a particular device, it is recommended that you enable status units checks. To do this, you need to enable the parameter [IOServer]WatchDogPrimary.Standby units are enabled by default.

## See Also

[OPC Status Tag Parameters](#)

## Customizing a Project using Citect.ini Parameters

You use Citect.ini parameters to fine tune the performance of the Plant SCADA OPC driver and to perform runtime maintenance diagnostics.

- [Common Driver Parameters](#)
- [Driver-specific Parameters](#)
- [Logging Parameters](#)
- [OPC Device-specific Parameters](#)
- [OPC Access Path Parameters](#)
- [OPC Status Tag Parameters](#)



### UNINTENDED EQUIPMENT OPERATION

- Do not under any circumstances change or remove any of the undocumented citect.ini parameters.
- Before deleting sections of the citect.ini file, confirm that no undocumented parameters will be deleted.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

You can customize the way Plant SCADA communicates with the OPC system, and even individual PLCs, by creating or editing the [OPC] section of the Citect.ini file for your project.

There are some common Plant SCADA driver settings, and custom Plant SCADA OPC driver settings.

When Plant SCADA starts at runtime, it reads configuration values from the Citect.ini file stored locally. Therefore, any OPC configuration settings need to be included in the Citect.ini file located on the computer acting as the I/O server to the OPC system.

**Note:** By default, Plant SCADA looks for the ini file in the Plant SCADA project \bin directory. If it can't find it there, it searches the default Windows directory.

The default values for these parameters have been determined by driver development and testing to be optimal for the Plant SCADA OPC driver. Parameters default to a value tested to work in common configurations. Do not

adjust these default values except on the direct advice of AVEVA Global Customer Support.

## Common Driver Parameters

The following table describes the common `citect.ini` driver parameters.

[OPC] Parameter	Allowable Values	Default Value	Description
Block	5 to 256	255	A value (bytes) used by the Plant SCADA I/O server to determine if two or more packets can be blocked into one data request before being sent to the OPC driver. For example, if you set the value to 10, and the I/O Server receives two simultaneous data requests (one for byte 3, and another for byte 8) the two requests are blocked into a single physical data request packet. This single request packet is then sent to the I/O device, saving on bandwidth and processing.
Delay	0 to 300	0	Period (in ms.) to wait between receiving a response and sending the next command to the I/O server.
MaxPending	1 to 1024	250	Maximum number of pending commands that the Plant SCADA OPC driver holds ready for immediate execution.
Polltime	0 to 300	1000	Polling service time (in milliseconds) at which cached data is updated. Recommended to be at least twice that of the value used in the Citect.INI [ALARM]

			ScanTime parameter.
Retry	0 to 8	0	Number of times to retry a command after a timeout.
Timeout	0 to 32000	3000	Milliseconds to wait for a response before considering a write or polling request to be invalid and displaying an error message in Plant SCADA .
Watchtime	0 to 128 (seconds)	15	Frequency that the Plant SCADA OPC driver uses to check the communications link to the OPC system and attempts to bring offline units back online. Also determines the frequency of STATUS_UNIT commands to the driver. See <a href="#">Configuring Redundancy</a> .

## Driver-specific Parameters

The following table describes the driver-specific Citect.ini parameters.

[OPC] Parameter	Allowable Values	Default Value	Description
AddItemAsVtEmpty	0 - Specify data types 1 - Use native data type	1	Determines whether the OPC driver specifies a data type when adding an item, or whether it uses the native data type when adding items.  This parameter is used to select whether the OPC client will specify a type to force the OPC Server to coerce the data, or for the OPC server to provide the type as configured on the server.

ArrayRecordBase	-	-	This parameter has been deprecated.
CacheRead	0 – Value is refreshed and the latest value returned. 1 – Last known value returned.	1	Determines the value returned when an inactive item is read.  See <a href="#">Overriding [OPC]CacheRead</a> for information on how to override the value of this parameter for a particular tag.
DigitalQualityUncertainIsBad	0 or 1	1	Determines how an uncertain quality result is treated when a Digital quality reading is taken.  0 - An uncertain quality will be treated as GOOD when a digital quality reading is taken (normal INT version is unaffected) 1 - an uncertain quality will be treated as BAD when a digital quality reading is taken (normal INT version is unaffected)
FailOnBadData	-	-	The parameter has been deprecated since Citect SCADA v7.30
FailOnUncertain	-	-	The parameter has been deprecated since Citect SCADA v7.30
FillCacheOnStartup	0 - Subscriptions are created in the inactive state and are only activated as read request are received.  1 - Subscriptions are created in the active state.	1	Determines whether subscriptions are created in the active or inactive state initially.
GoOfflineForBadTag	0 – disabled. Do not go offline.  1 - enabled. Go offline.	0	Sets all tags on an OPC device to an OPC_QUALITY status of

			<p>GOOD before Plant SCADA declares the I/O device online.</p> <p>This parameter would only be set on the I/O server where the device is defined as primary, so it can only take over from a standby device definition when all tags and the status tag are valid. This minimizes the chances that com-breaks occur during switchover from standby to primary. It is strongly recommended not to set it on a standby, as it might only be one tag that is invalid.</p>
InhibitActivationOnStandby	0 – A standby server will follow the same rules for subscription activation as the primary server. 1 – A standby server will only active a subscription if it believes the primary server is not active.	1	<p>This parameter allows a standby server to deliberately leave subscriptions in an inactive state when the primary server is online. This is usually used when the primary and standby server are talking to independent OPC servers to reduce the load the standby OPC server would place on field devices.</p> <p>On startup of a standby server, if this parameter is set to 1, items will not be activated immediately. They will only be activated when the unit itself becomes the active unit.</p>
ItemLifeTime	Time (seconds)	5	<p>Determines the time in before a tag is deemed inactive.</p> <p>The global LeaveTagsActive parameter overrides the</p>

			<p>ItemLifeTime parameter to force OPC tags to remain active even though Plant SCADA is no longer requesting them from the driver.</p> <p>The ItemLifeTime parameter can be individually overridden for any OPC device by the Active parameter.</p>
LeaveTagsActive	<p>0 - Do not leave tags active.</p> <p>1 - Leave tags active.</p>	1	<p>Determines whether OPC tags remain active even though Plant SCADA is no longer requesting them from the driver. Normally tags are made inactive 5 seconds after the last time they are requested.</p> <p>Some OPC servers take a long time to return a call to activate/inactivate tags, or doing so slows them down, therefore leaving tags active can sometimes improve performance.</p> <p>The global LeaveTagsActive parameter can be individually overridden by an Active parameter.</p> <p>See ItemLifeTime parameter (above).</p> <p>Note: This parameter minimizes the likelihood of deactivation of tags on the currently active unit only. Tags on units that have become inactive (standby) will still deactivate</p>
NumConcurrentServerInitS	-1 – Unlimited 32767	-1	Represents the maximum number of concurrent initialization and

			shutdown operations that are performed on each configured OPC Server. Higher values, or -1 (unlimited), will incur greater load on the server, while 0 and 1 determine that initialization is serialized into effectively a single thread.
OPCGroupName	Any string (maximum 32 characters)	CitectGroup_<n>	By default, each OPC server is allocated a sequential group name by the OPC driver, i.e. CitectGroup_1, CitectGroup_2, etc. This parameter allows you to replace the default group name with a customized name. You can set this parameter at a device level using <a href="#">OPC Device-specific Parameters</a> .
ReadAfterWrite	0 - No read, rely on the normal polling/OnDataChange mechanism.  1 - Force sync read of item written from OPC server cache after write.  2 - Force sync read of item written from device after write.	0	Determines if a force read should be made, and if it should be from the OPC server cache or from the device.  <b>Note:</b> RefreshAfterWrite causes a refresh of all active items, whereas ReadAfterWrite reads back only the item being written to. You should use one or the other. Where both RefreshAfterWrite and ReadAfterWrite are set to 1, RefreshAfterWrite will take precedence, and ReadAfterWrite will be set to 0.
RefreshAfterAdd	0 - No REFRESH.	1	Determines whether an

	1 - Request a REFRESH from the OPC server cache.  2 - Request a REFRESH from the device.		OPC REFRESH is requested from the OPC server after activating the group.  This parameter can be disabled if slow start up times are experienced.  The server could be automatically sending a REFRESH, in which case another request may be unnecessary.
RefreshAfterWrite	0 – Disable forced REFRESH.  1 - Enable forced REFRESH from OPC server cache.  2 - Enable forced REFRESH from device.	0	Determines whether to force an OPC REFRESH from the OPC Server of ALL active items after Plant SCADA writes to an item.
RefreshBeforeArrayWrite	0 – No refresh command sent  1 – Read command sent to refresh the OPC server cache  2 – Read command sent to refresh the OPC device	0	Enables the driver to send a read command to the OPC server before a write command is sent to the OPC server for an array variant.
ServerOnlineStates	"RUNNING" "FAILED" "NOCONFIG" "SUSPENDED" "TEST"	"RUNNING"	Defines the list of OPC Server states that the driver considers online. The format is a " " separated string, without spaces.  If you experience the OPC Server shutting down on startup, it is recommended that the following states are used: "RUNNING   NOCONFIG   SUSPENDED"
ShutdownWait	Time (milliseconds).	30000	Determines the time the OPC driver waits before shutting down. On shutdown, the driver attempts to close the connection to the OPC

			server. If this takes longer than the time specified by this parameter, the driver shuts down anyway (sometimes with unexpected results). This parameter might be needed if the OPC server does not respond within the default wait time.
StatusWaitPeriod	Time (milliseconds)	5000	Determines the time the OPC driver waits before requesting tag status for an OPC I/O device. On startup, the OPC driver waits for a Unit status from the OPC server before adding tags to the server. If the value of this parameter is exceeded, the tags are added anyway.  The OPC server should respond within a few seconds. The default wait time of 5 seconds should usually be adequate.
SuppressDataNotYetValidError	0 or 1	0	Determines whether "Data Not Yet Valid" error messages are suppressed, or whether they are displayed in the Kernel window and written to the Syslog.dat file. For Plant SCADA to display the message "Error Suppressed", you need V5.50 or above.  0 – Displays "Data Not Yet Valid" errors in the Kernel window and logs them to the Syslog.dat file.  1 - Suppresses "Data Not Yet Valid" errors, which are not displayed in the Kernel window or logged.

			In the Runtime kernel, under View -> I/O Devices, Plant SCADA will display "Error Suppressed" in the Error Message field for the particular device.
UseArrays	-	-	This parameter has been deprecated.
UseAsyncWrites	0 - use synchronous writes. 1 - use asynchronous writes.	0	Determines whether writes are synchronous or asynchronous.  Note: UseAsyncWrites=1 is only valid when [OPC]UseOPC2=1, since OPC1 does not support asynchronous writes.
UseOPC2	0 – OPC2 support disabled (Use OPC 1.0a interface). 1 – OPC2 support enabled (Use OPC DA2 interface).	0	Globally determines whether to use OPC 1.0a or OPC 2.0 interface. Individual overrides can be set using the UseOPC2 parameter to force the use of a particular version OPC interface with a particular I/O device.
UseStatusTags	0 – disabled. 1 – enabled.	0	Determines whether parameters in the [OPCStatusTags] section are active. Individual overrides can be set using the OPC Device-specific Parameters to force the use of a particular status tag parameter with a particular I/O device.
ValidDataWaitPeriod	Time (milliseconds)	10 seconds	Determines the maximum time that the OPC driver will wait for OPC_QUALITY to be GOOD for all tags before declaring that Device online. This minimizes the likelihood of a client

			<p>swapping from a standby to a primary prematurely and suppresses alarm flooding at startup.</p> <p>Set this parameter to 0 to allow units to come online regardless of whether all items for the unit have been verified as good quality.</p> <p><b>Note:</b> FillCacheOnStartup=1 needs to be set when ValidDataWaitPeriod&gt;0</p>
WriteTrueAs1	<p>0 - write true as VARIANT_TRUE(-1).</p> <p>1 - write true as 1.</p>	0	<p>Determines whether a digital value of TRUE is written as "VARIANT_TRUE(-1)" or "1". This parameter should only be required for incorrectly designed OPC servers. When the Plant SCADA data type is digital, VARIANT_TRUE (-1) is used with VT_BOOL variant type to write a value of true. This happens even if [OPC]AddItemAsVtEmpty is set, as OPC servers should be designed to convert the value when they receive it.</p>

## Logging Parameters

The following table describes the OPC logging parameters.

[OPC] Parameter	Description	Allowable Values	Default Value
DebugLevel	<p>The trace level used for log file.</p> <p>See <a href="#">Logging</a> for examples.</p>	<p>ALL or any combination of the following, separated by a pipe character ( ):</p> <p>WARN</p> <p>ERROR</p> <p>TRACE</p>	-

DebugCategory	The categories used for the log file. See <a href="#">Logging</a> for examples.	ALL or any combination of the following, separated by a pipe character ( ): TAG: Tag configuration trace. PROT: Operations related to the OPC protocol interface. CACHE: Operations related to the driver cache. DCB: Front end driver trace. THRD: Thread trace. MISC: Miscellaneous operations. BEND Operations related to the back-end thread of the driver. FEND: Operations related to the front-end thread of the driver.	-
DebugUnits	Controls which units logging is performed for. Multiple units can be specified. Debug messages that are unrelated to a specific unit are logged against the (GLOBAL) unit.	ALL, (GLOBAL) [include the round brackets] or any combination of I/O device names and (GLOBAL) separated by ' ' characters.	All

### OPC Device-specific Parameters

For each OPC I/O device you can specify parameter conditions that override the default global settings. To do this, use the following format in the [OPC] section of the Citect.ini file.

**[OPC]<IOServerName>.<IODeviceName>.<Parameter>=<Value>**

where:

- <IOServerName> = Name of I/O server configured in Plant SCADA .
- <IODeviceName> = Name of I/O device configured for OPC communications.
- <Parameter> = Parameter of the I/O device being overridden.
- <Value> = Value of the overriding parameter being set.

---

**Note:** There are also a number of status tag parameters that you can override global settings for in the

[OPCStatusTags] section of the Citect.ini file. For more information see [OPC Status Tag Parameters](#).

The following device-specific parameters are supported.

[OPC] Parameter	Allowable Values	Default Value	Description
Active	<b>0</b> - deactivate tag. <b>1</b> - leave tag active.	Value of the LeaveTagsActive parameter.	Allows user to individually override the global default LeaveTagsActive parameter to specify whether tags should be left active or inactive for a particular OPC I/O device. Replace the placeholders <IOServerName> and <IODeviceName> in the parameter with the names of the OPC 'I/O Server' and 'I/O Device' you want to override and have remain active. Determines whether individual OPC tags are left active even though Plant SCADA is no longer requesting them from the driver. Normally tags are made inactive 5 seconds from the last time they are requested. See the LeaveTagsActive parameter in <a href="#">Driver-specific Citect.ini Parameters</a> .
Update	0 to 32000 (milliseconds)	Scan rate specified in the <b>Address</b> field of the Boards settings.	Allows the user to individually override the.Default.Update parameter to specify the tag update rate for a particular I/O device. The update rate is the frequency with which the OPC server polls the device for values and updates its own cache. Replace the placeholders <IOServerName> and <IODeviceName> in the

			parameter with the actual names of the OPC I/O server and I/O device you want to override with a specific update rate.  This parameter controls polling performed by the OPC server, not by Plant SCADA. The scope of the parameter is communication between the OPC server and a device, not between the OPC server and Plant SCADA.
UseAsyncWrites	0 - use synchronous writes. 1 - use asynchronous writes.	0	Determines whether writes are synchronous or asynchronous.  <b>Note:</b> UseAsyncWrites=1 is only valid when [OPC]UseOPC2=1, since OPC1 does not support asynchronous writes.
UseOPC2	0 – Individual OPC2 support disabled (Use OPC 1.0a interface). 1 – Individual OPC2 support enabled (Use OPC DA2 interface).	Value of global UseOPC2 parameter	Allows users to individually override the global default UseOPC2 parameter and to specify whether to use OPC 1.0a or OPC 2.0 for a particular I/O device.  Replace the placeholders <IOServerName> and <IODeviceName> in the parameter with the actual names of the OPC I/O server and I/O device you want to override and have with a particular version OPC interface.

## OPC Access Path Parameters

In some instances, an OPC server might have several ways to access a particular item. For instance, a particular server, which communicates by means of modem, might have multiple modems at its disposal. It might be desired that certain items in the OPC server are always accessed using a particular modem, the fastest of the

group, for example. OPC allows you to specify this preference by using the Access Path <OPCAccessPath> value. An access path is optional, provided in addition to the Item ID (the tag address), which is provided as a recommendation to the server, regarding how to access the data.

**Note:** The availability, format, and use of access paths is server-specific. Refer to your OPC server documentation for details relevant to your server.

[OPCAccessPaths]<IOServerName>.<IODeviceName>=<OPCAccessPath>

where:

- <IOServerName> = Name of I/O server configured in Plant SCADA .
- <IODeviceName> = Name of I/O device configured for OPC communications.
- <OPCAccessPath> = Access path to set up communications to the OPC server.

This parameter is not normally required with most OPC servers.

The Plant SCADA OPC client driver supports specification of a single access path for each configured I/O device using the [OPCAccessPaths] section of the Citect.ini file.

To specify the access path for a particular device, add an entry to the ini file in the format <IOServerName>.<IODeviceName>. For example, to specify that the access path COM1: should be used for the I/O device OPC1 on I/O server IOServer1, add the following entry to the Citect.ini file:

[OPCAccessPaths]  
IOServer1.OPC1=COM1:

Multiple entries are also acceptable:

[OPCAccessPaths]  
IOServer1.PLC1=PLC4  
IOServer1.PLC2=TestPLC

## OPC Status Tag Parameters

The OPC status tag parameters determine the online or offline state of an OPC I/O device using the condition of a specified OPC item in the device. For more information, see [Status Tags](#).

When configuring these parameters, be careful not to confuse the OPC item required for status monitoring with a normal Plant SCADA tag.

You should also use a full path to identify an OPC item, as a configured access path in your Citect.ini file will not be known to a device.

**Note:** Use the UseStatusTags parameter to enable or disable support for OPC status tags.

[OPCStatusTags] Parameter	Allowable Values	Default Value	Description
StartWait	time in milliseconds - or - -1 (infinite)	30000 (milliseconds)	On driver startup, this parameter determines the time the OPC driver will wait before requesting status tag data for an OPC I/O device.  The OPC server should

			respond within a few seconds. The default wait time of 30 seconds is usually adequate.
Default.Tag	An OPC item name	STATUS	This parameter specifies the default OPC item to be monitored at runtime to determine the operational status of an associated device.  <b>Note:</b> This parameter implements a global setting for all OPC I/O devices, unless it is specifically overridden using the equivalent <a href="#">OPC Device-specific Parameters</a> .
Default.Update	0 -32000 (milliseconds)	The scan rate from Address field in the Plant SCADABards configuration.	This parameter specifies the default update rate for OPC status tags at runtime. This is the frequency with which the OPC server polls the device for status tag values and then updates its own cache.  This parameter controls polling performed by the OPC server, not by Plant SCADA. The scope of the parameter is communication between the OPC server and a device, not between the OPC server and Plant SCADA.  <b>Note:</b> This parameter implements a global setting for all OPC I/O devices, unless it is specifically overridden using the equivalent <a href="#">OPC Device-specific Parameters</a> .

Default.Condition	<condition> (see table below)	ISGOOD	<p>This parameter is used to determine if the OPC device is online at runtime, by comparing the &lt;Condition&gt; with the value of the default OPC item named in the .Default.Tag parameter.</p> <p>If the specified condition is determined to be true, the device will be considered online.</p> <p>This parameter represents the default condition for all OPC status tags, unless it is individually overridden using the <a href="#">OPC Device-specific Parameters</a>.</p>
-------------------	----------------------------------	--------	--

Allowable values for <condition>	Description
=<Value>	is equal to <Value>
!<Value>	is NOT <Value>
!<Value>	is NOT equal to <Value>
><Value>	is greater than <Value>
>=<Value>	is greater than or equal to <Value>
<<Value>	is less than <Value>
<=<Value>	is less than or equal to <Value>
ISGOOD	the tag Quality is GOOD
ISBAD	the tag Quality is BAD
ISUNCERTAIN	the tag Quality is UNKNOWN

Where:

- <Value> is a placeholder for any valid LONG data type value.

If the specified condition is determined to be true, the device will be considered online.

## Overriding [OPC]CacheRead

You can override the value in [OPC]CacheRead for a particular tag by suffixing the tag name with:

The delimiter "!"

The letter C or D, as follows:

- <Tag Name>!C - The tag value returned will be the cache value at the time of the read.
- <Tag Name>!D - A refresh of the tag value is forced before it is returned.

### Examples:

- Tag1!D - Forces a refresh of Tag1 before the value is returned.
- Tag!C - The last known cache value of Tag2 is returned.

When the tag qualifiers C and D are used with other OPC qualifiers (see [Timestamp and Quality Support](#)), they must appear last.

### Example:

- Tag3QC - Tag3 is associated with a quality tag and the last known value of Tag3 will be returned.

## Configuring Redundancy

Redundancy is handled in Plant SCADA by the Plant SCADA clients. They determine which I/O server to request I/O device data from by allowing the scan rate to be specified per I/O device, and then configuring multiple I/O devices to the same PLC.

A status tag can be defined in the Citect.ini file to determine whether or not to set a unit offline or online depending on the condition of the status tag. If the condition is true the unit is allowed to come online, otherwise the unit is set to offline.

The status tag is polled at the same rate configured for the groups of the corresponding unit. The driver supports STATUS\_UNIT commands on the inactive server, so even though a particular unit is not currently servicing Plant SCADA requests, the state of the status condition will be checked periodically allowing the unit to be taken either offline or online (assuming that [IOServer]WatchDog has not changed from its default of 1).

Both the primary and standby devices will be offline if both have the same status condition and the condition is an error condition. All points in the system for that unit would therefore be #COM.

Status tags can be defined on a per-device or global basis. For more information, see [OPC Status Tag Parameters](#) and [OPC Device-specific Parameters](#).

## Scan Rates

The Plant SCADA OPC driver can define I/O devices with different scan rates. This provides you with the ability to better tune your system, by ordering the tag database into logical groups and defining different scan rates for each group. For example, tags that are being read frequently can be placed in a group that is scanned frequently, and tags that are read infrequently can be placed in a group that is scanned infrequently.

By default, tags are scanned at the default scan rate defined by the ScanRate parameter in the project Citect.INI file. See Citect.INI specific parameters for details.

## Examples

- Set the project scan rate to 1 second (1000 milliseconds):  
[OPC]  
ScanRate=1000
- Set the project scan rate to 1 second (1000 milliseconds), and the inactive server scan adjust rate to a factor of 5:  
[OPC]  
ScanRate=1000
- Set a faster (half second) scan rate for a specific port named "Port1" on the I/O server named "CLServer":  
[CLServer.Port1]  
ScanRate=500
- Set a slower (5 second) scan rate for a specific port named "Port123" on the I/O port named "CLServer":  
[CLServer.Port123]  
ScanRate=5000

## Troubleshooting

Many problems have simple solutions and require only perseverance to solve them. The following topics provide information about the Plant SCADA tools provided to help resolve problems with communication and configuration.

- [Driver Errors](#)
- [Driver Statistics](#)
- [Logging](#)
- [Maintaining the Project Database](#)

### Driver Errors

Plant SCADA OPC driver errors can occur when an OPC device does not respond, is disconnected or offline, or returns an error itself.

- [Common Protocol Errors](#)
- [OPC Protocol Errors](#)

### Common Protocol Errors

Plant SCADA has two kinds of protocol driver errors - generic and specific. Generic errors are hardware errors 0-31, and are common to many protocols. Sometimes only the generic error is available, though often both the generic error and a specific error are given.

Protocol drivers also have their own specific errors, which can be unique and therefore cannot be recognized by the hardware alarm system. The drivers convert specific errors into generic errors that can be identified by the I/O server. For example, when a driver experiences an error, there is often both a protocol-specific error and a corresponding generic error.

When a hardware error occurs, Plant SCADA generates an alarm, and displays the alarm on the hardware alarm page (in the alarm description of the hardware alarm). To see the error number, you need to have Alarm Category 255 defined with a display format that includes {ErrDesc,15} {ErrPage,10}.

If you need more information to resolve an error, refer to the documentation that accompanied the I/O device (or network). If, after reviewing the documentation, you still cannot solve your problem, contact support via the AVEVA Knowledge & Support Center (<https://softwaresupport.aveva.com/>).

## OPC Protocol Errors

The following errors, listed in (hexadecimal) order, are specific to the Plant SCADA OPC protocol:

Error Value (Hex)	Error Definition	Error Description
0XC0040001	GENERIC_SOFTWARE_ERROR	Invalid handle was passed.
0XC0040002	GENERIC_SOFTWARE_ERROR	A duplicate parameter was passed where one is not allowed.
0XC0040004	GENERIC_INVALID_DATA_FORMAT	Server cannot convert between the passed or requested data type and the canonical type.
0xC0040006	GENERIC_ACCESS_VIOLATION	Item's access rights do not allow the operation.
0xC0040007	GENERIC_SOFTWARE_ERROR	Item definition does not exist within the server's address space.
0xC0040008	GENERIC_SOFTWARE_ERROR	Item definition does not conform to the server's syntax.
0xC004000B	GENERIC_INVALID_DATA_FORMAT	Value passed to WRITE was out of range.
0x80020008	GENERIC_SOFTWARE_ERROR	Bad variable type.
0x8002000A	GENERIC_SOFTWARE_ERROR	Out of present range.
0x80020005	GENERIC_SOFTWARE_ERROR	Type mismatch.
0x100	GENERIC_SOFTWARE_ERROR	Could not access variable dbf.
0x101	DRIVER_BAD_DATA	Read of data value bad.
0x102	GENERIC_GENERAL_ERROR	Write of one or more items not completed.
0x103	DRIVER_UNIT_OFFLINE	Could not resolve the server CLASSID.
0x104	GENERIC_SOFTWARE_ERROR	Could not add one or more items to

		the Server
0x80010006	GENERIC_CHANNEL_OFFLINE	Connection terminated
0x80010007	GENERIC_CHANNEL_OFFLINE	Server not available.
0x8001000F	GENERIC_INVALID_RESPONSE	Received data is invalid.
0x80010012	GENERIC_SOFTWARE_ERROR	Call did not execute.
0x80010100	GENERIC_SOFTWARE_ERROR	System call aborted.
0x80010101	GENERIC_SOFTWARE_ERROR	Could not allocate some required resource.
0x80010103	GENERIC_BAD_PARAMETER	Requested interface not registered on server object.
0x80010104	GENERIC_SOFTWARE_ERROR	Could not call server.
0x80010105	GENERIC_CHANNEL_OFFLINE	Server threw an exception.
0x80010108	GENERIC_CHANNEL_OFFLINE	Server has disconnected from its clients.
0x8001011B	GENERIC_CHANNEL_OFFLINE	Access denied.
0x8001011C	GENERIC_CHANNEL_OFFLINE	Remote calls not allowed for this process.

## Driver Statistics

You can use the following driver statistics to help you debug the OPC driver operation.

Statistic	Description
Cached Reads	Number of reads that have been serviced directly from the internal cache of current values.
Cache Misses	Number of reads that were not in the internal cache of current values which resulted in a direct request from the server.
Active Items	Number of items with valid server subscriptions which have been activated to receive data change notification. This may be a subset of the total number of items currently subscribed to.
Total Items	Total number of items with valid server subscriptions. These items might or might not be currently activated to receive data change notifications.

Bad Items	Number of items that were not successfully subscribed to. These items might not exist on the server.
Start ms	Internal driver statistic used for driver debugging by AVEVA Global Customer Support.
Last ms	Internal driver statistic used for driver debugging by AVEVA Global Customer Support.
Difference	Internal driver statistic used for driver debugging by AVEVA Global Customer Support.
ReadItemsReQueued	Internal driver statistic used for driver debugging by AVEVA Global Customer Support.
DCBsInHandlingQ	Internal driver statistic used for driver debugging by AVEVA Global Customer Support.
DataNotYetValid	Internal driver statistic used for driver debugging by AVEVA Global Customer Support.

## Logging

The OPC driver can log combinations of trace levels across different categories. This is achieved by setting the following INI parameters.

### [OPC]DebugLevel

This parameter allows you to define the trace level. The options include:

WARN	Output warning messages.
ERROR	Output error messages.
TRACE	Output trace messages.
DEBUG	Output debug messages.
ALL	All of the above.

### [OPC]DebugCategory

This parameter allows you to enable logging for a particular category of trace. The options include:

TAG	Tag configuration trace
PROT	Operations related to the OPC protocol interface

CACHE	Operations related to the driver cache
DCB	Front end driver trace
MISC	Miscellaneous operations
THRD	Thread trace
BEND	Operations related to the back-end thread of the driver
FEND	Operations related to the front-end thread of the driver
ALL	All of the above

In both cases, you can use ALL or any combination of the available options separated by a pipe character ( | ).

### Example

```
[OPC]
DebugLevel=WARN|ERROR|TRACE
DebugCategory=ALL
```

In most cases, you can set the DebugLevel and DebugCategory parameters to ALL. If this results in a large, cumbersome log file that causes older data to be overwritten, you can consider refining the debug options.

---

**Note:** These parameters do not have a value specified by default. To enable logging for this driver, you need to manually configure these parameters in your INI file.

The events generated by the OPC driver are logged in the following Plant SCADA syslog file:

```
syslog.IOServer.<Cluster name>.<IO Server name>.dat
```

This file is located in the directory specified by the Citect.ini parameter **[CtEdit]Logs**. By default, this location will be:

```
%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Logs
```

---

**Note:** In Citect SCADA v7.0 and earlier, this file was delivered to the Windows system directory (for example, C:\Windows) and its location was not configurable. With the release of Citect SCADA v7.10, the file is delivered to the location specified in the **[CtEdit]Logs** parameter, which is configurable.

---

The size of the syslog file and its rollover process is configured via the **[Debug]SysLogSize** and **[Debug]SysLogArchive** INI parameters. For more information, see the topic *Debug Parameters* in the Plant SCADA documentation.

---

**Note:** If the number of messages being processed for a driver exceeds the value specified by the Plant SCADA parameter **[Kernel]ErrorBuffers**, the pending messages will not be logged. If this occurs, you can increase the value of this parameter so that all the messages generated by the driver are reported. The number of messages that are not logged can be monitored via the Kernel's General window under the item "Lost Errors". If you adjust **[Kernel]ErrorBuffers** and Lost Errors are still occurring, you may need to adjust the level of system logging.

## See Also

[Logging Parameters](#)

## Maintaining the Project Database

Over time with editing, the project database underlying a project can become fragmented and performance can be reduced. During project development you should periodically clear out all duplicated records and any orphaned ones.

Improperly maintained databases can result in communication errors at runtime. A large number of communications errors come from having duplicated or orphaned records in the communications database.

Sometimes orphaned records from a previous I/O server are left behind in the database files. As the forms are indexed on the I/O server name, you can use the scroll bar to quickly navigate to the end of the current range of records for a particular I/O server and examine the last few and next few records to validate that they should be there. If you find extra (unwanted) records, delete them.

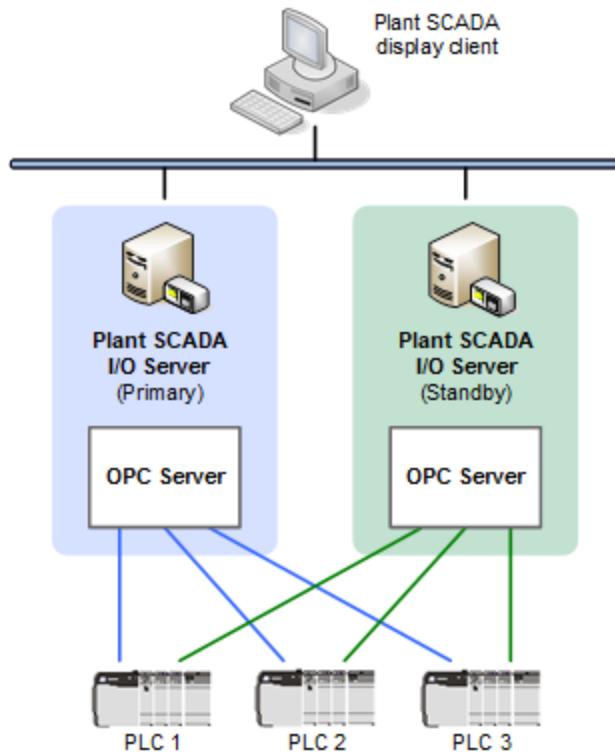
You should also check that there are no duplicated records. Use the record search facility to search for duplicate entries of devices that are causing unexplained communication errors. If you find extra (unwanted) records, delete them.

Once you have deleted orphaned and duplicated records, pack the project. Packing the database removes deleted records, and re-indexes the database. To pack the project databases, use the **Pack** button on the command bar of the **Project** activity in Plant SCADA Studio.

## OPCLX Driver

The OPCLX driver enables Plant SCADA to communicate to ControlLogix system PLCs via the OPC DA protocol and Rockwell software OPC servers.

The OPCLX driver leverages the Ethernet/IP protocol included in the Rockwell software OPC servers, and connects using an industry standard protocol OPC. The OPC specifications v1.0a and v2.0 are supported. Using this method, you can connect to single or multiple [OPC Devices and Clients](#), as shown in this example configuration.



The Rockwell software OPC servers can also exist on the same computer as the Plant SCADA I/O server and OPCLX driver. Remember that the maximum request length (that is, the maximum of data bits that can be read from the I/O device at any one time) for the OPCLX driver is 2040.

**Note:** With the release of version 7.20, Plant SCADA supports the retrieval of time-stamped data directly from field devices. This capability is enabled by the driver runtime interface (DRI), a component of Plant SCADA that is used by the OPCLX driver to update time-stamped digital alarms, time-stamped analog alarms, and event-based trends directly. For more information, see the topic [Retrieving time-stamped data from field devices](#) in the main Plant SCADA help.

## See Also

- [Prepare the OPC System](#)
- [Configure Variable Tags](#)
- [Advanced Configuration and Maintenance](#)
- [Migrate from ABCLX to OPCLX](#)
- [Troubleshooting](#)

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.
<b>⚠ DANGER</b>	<b>DANGER</b> indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.
<b>⚠ WARNING</b>	<b>WARNING</b> indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.
<b>⚠ CAUTION</b>	<b>CAUTION</b> indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.
<b>NOTICE</b>	NOTICE used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.

## Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

## Before You Begin

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ WARNING****LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

## Prepare the OPC System

This section of the help explains how to prepare your Allen Bradley devices to communicate with Plant SCADA through OPC.

## Software Requirements

In the OPC architecture, each server is allocated a unique identifier, known as a class ID. OPC clients use these 128-bit numbers, frequently displayed in the form 6B29FC40-CA47-1067-B31D-00DD010662DA, to access the OPC server of a particular vendor. These numbers are often referred by a manageable string identifier called a Program Identifier (ProgID). These identifiers usually take the format Vendor.Application with an optional version number appended. For example, the Allen Bradley FactoryTalk OPC server ProgID is RSOPC Gateway.

Before the OPCLX client driver can access a particular OPC server, the ProgID of the server needs to be entered into the Plant SCADA I/O server registry. If an in-process server or a local server is used, this is done automatically during the installation of the server. To access a remote server, refer to the documentation accompanying the server. Frequently, this step is achieved using a registry script (a .REG file) or a small

executable.

#### NOTICE

#### LOSS OF DATA

- Appropriately configure DCOM settings to browse the remote OPC servers.
- Assign appropriate privileges to the accounts being used by the operators.

**Failure to follow these instructions can result in equipment damage.**

#### Note:

- Familiarize yourself with the target OPC server before using this driver. This will help you enter the item identifiers for the tag addresses using the correct syntax.
- Using remote servers requires configuring DCOM security. Since configuring security is complex, you can plan your system settings in advance, and establish a protocol so that the Windows accounts to be used by operators have sufficient privilege. If sufficient privileges are not established, there may be communication interruptions found.

## OPC Devices and Clients

This section provides information required to configure Allen Bradley OPC devices and clients within your Plant SCADA project.

- [Device Address](#)
- [Communication Settings](#)
- [Data Types](#)
- [Include Project Support](#)

### Device Address

The address to use is the ProgID for the desired server.

This usually takes the form "<Vendor>.<Application>" and can be found in the documents accompanying the OPC server.

### See Also

[Communication Settings](#)

### Communication Settings

To establish communication with a device, configure the Boards, Ports and I/O Devices in the Plant SCADA Studio's **Topology** activity correctly. If you use the Express Communications Wizard to connect to a device, these will be automatically configured for you. However, if you need to configure them manually, use the settings outlined below.

## Boards

Field	Value
Board Name	A unique name (up to 16 alphanumeric characters) per server for the computer board being used to connect with the device. This is used by Plant SCADA in the Ports dialog box.
Board Type	The type of board. Select OPCLX.
Address	<p>Plant SCADA does not need an address for the OPC client. Instead you use the <b>Address</b> box to set the device scan rate on the OPC server in milliseconds. The value read from the OPC server is typically taken from the OPC server cache. The scan rate value is often used to indicate how often the OPC server polls the device and updates its cache.</p> <p>Enter 0 (zero) to use the default of 250 ms. Enter any other value to use that value. Valid values range from eight character decimal 0 to 99999999. Up to six character hex values can be used; however, these must be preceded with 0x.</p> <p><b>Note:</b> Plant SCADA supports unsolicited OPC requests. So if you set the OPC server to notify Plant SCADA of changes to the value or quality of an OPC device item, make sure that the OPC server scan rate is short enough to provide Plant SCADA within an adequate timeframe.</p>
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	<p>OPC server machine name. (Do not enter this name using the traditional dash delimited style. Enter the machine name, for example Cit_Primary.)</p> <p><b>Note:</b> The name entered here is the Windows machine name. For Windows NT and Windows 95 or 98 based OPC servers, you can obtain this from the Identification tab of the network properties of OPC server machine. For other OPC server configurations, refer to the documentation of the server. If this field is left blank, the server name is sought in the following manner:</p> <ul style="list-style-type: none"> <li>a) The in-process (DLL) server on the I/O server machine;</li> <li>b) The local (EXE) server on the I/O server machine;</li> </ul>

Field	Value
	c) The remote server as entered in the DCOM settings of the I/O server machine.
Comment	Any useful comment.

## Ports

Field	Value
Port Name	A unique name (up to 16 alphanumeric characters) for the computer port being used to connect with the device. This is used by Plant SCADA in the Devices dialog box.
Port Number	An integer value, unique to the board as defined in the Board name field. The Ethernet port needs no identification; however the Plant SCADA communication database requires a value in this field.
Board Name	The name of the board this port is attached to as defined on the Boards dialog box.
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank.
Special Options	Leave this field blank.
Comment	Any useful comment.

## I/O Devices

Field	Value
Name	A unique name (up to 16 alphanumeric characters) for the I/O Device being identified. Each I/O Device must have a unique name in the Plant SCADA system.
Number	A unique number for the I/O Device (0–16383). Each I/O Device must have a unique number in the Plant SCADA system (unless redundancy of the I/O Device is being used).

Field	Value
Address	ProgID for the desired server. These usually take the form Vendor.Application (for example, Allen Bradley FactoryTalk OPC server ProgID is RSOPC.Gateway) and should be found in the documents accompanying the OPC server.
Protocol	Enter OPCLX. See <a href="#">Arrays Support</a> .
Port Name	This field must contain one of the names previously defined in the Port Names field of the Ports dialog box. There must be only one I/O device per port.
Comment	Any useful comment.

## Data Types

OPC servers can provide data in different data formats. An OPC client can request data in a specific format or use the type of the server (that is, the type, in which the data is stored within the server).

When specifying a data type, consider whether the server is able to convert data to the requested type.

Use the table below to help you specify a type for a variable tag. It shows mappings from the types used by OPCLX (VARTYPE) to Plant SCADA data types.

OPCLX type (VARTYPE)	Description	Plant SCADA data type
VT_BOOL	Boolean	DIGITAL
VT_UI1	Byte	BYTE
VT_I2	2 byte integer	INT
VT_UI2	2 byte unsigned integer	UINT
VT_I4	4 byte integer	LONG
VT_U14	4 byte unsigned integer	ULONG (requires v7.0)
VT_CY	Currency	STRING
VT_R4	4 byte real	REAL
VT_R8	8 byte real	STRING
VT_DATE	Date	STRING
VT_BSTR	String	STRING
VT_ERROR	Error Value	LONG

OPCLX type (VARTYPE)	Description	Plant SCADA data type
VT_QUALITY	2 byte integer	INT
VT_TIMESTAMP	4 byte integer	LONG
VT_MILLISECOND	4 byte integer	LONG

## Include Project Support

Variable tags defined in a project compiled on a Plant SCADA client need to match exactly to the tags defined in the project compiled on the I/O server. If a tag is added on the client, it also needs to be added on the I/O server. Otherwise client requests may become mismatched and result in inaccessible tag data.

## Configure Variable Tags

This section provides information that describes how to configure variable tags for OPCLX devices within your Plant SCADA project.

- [Tag Address Configuration](#)
- [Arrays Support](#)
- [Quality Values](#)
- [Time Stamp and Quality Tags Configuration](#)

### Tag Address Configuration

The tags supported by OPCLX are:

- Controller tags
- Program tags.

### Controller Tags

Controller tags contain data which is accessible to the programs within the RSLogix project. When controller tag is configured in OPC gateway, the same can be configured in the Plant SCADA in the following formats depending on the OPC server:

[OPCTopic]TagName

Syntax with the FactoryTalk and RSLinx Classic is the same in case of controller tags.

The table below describes the syntax and example for configuring controller tags:

Address	Plant SCADA type	OPCLX type	Example
[OPCTopic]<item_id>	DIGITAL	VT_BOOL	[OPCLX_test]bool_symbol
[OPCTopic]<item_id>	BYTE	VT_UI1	[OPCLX_test]byte_symbol

Address	Plant SCADA type	OPCLX type	Example
[OPCTopic]<item_id>	INT	VT_I2	[OPCLX_test]int_symbol
[OPCTopic]<item_id>	LONG	VT_I4	[OPCLX_test]long_symbol
[OPCTopic]<item_id>	REAL	VT_R4	[OPCLX_test]real_symbol
[OPCTopic]<item_id>	STRING	VT_ARRAY of VT_UI1	[OPCLX_test]bool_symbol

## Program Tags

Program tags contain data that is used exclusively within a program in RSLogix project. When a program tag is configured in OPC gateway, the same can be configured in Plant SCADA in the following formats depending on the OPC server:

Syntax with the FactoryTalk OPC server is:

::[OPCTopic]Program:MainProgram.<TagName>

Syntax with the RSLinx Classic OPC server is:

[OPCTopic]Program:MainProgram.<TagName>

The table below describes the syntax and example for configuring program tags RSLinx Classic:

Address	Plant SCADA type	OPCLX type	Example
[OPCTopic]<Program:MainProgram>.<item_id>	DIGITAL	VT_BOOL	[OPCLX_test] Program:MainProgram.bool_symbol
[OPCTopic]<Program:MainProgram>.<item_id>	BYTE	VT_UI1	[OPCLX_test] Program:MainProgram.byte_symbol
[OPCTopic]<Program:MainProgram>.<item_id>	INT	VT_I2	[OPCLX_test] Program:MainProgram.int_symbol
[OPCTopic]<Program:MainProgram>.<item_id>	LONG	VT_I4	[OPCLX_test] Program:MainProgram.long_symbol
[OPCTopic]<Program:MainProgram>.<item_id>	REAL	VT_R4	[OPCLX_test] Program:MainProgram.real_symbol
[OPCTopic]<Program:MainProgram>.<item_id>	STRING	VT_ARRAY of VT_UI1	[OPCLX_test] Program:MainProgram.bool_symbol

The table below describes the syntax and examples for configuring program tags FactoryTalk:

Address	Plant SCADA type	OPCLX type	Example
::[OPCTopic]<Program:Ma inProgram>. <item_id>	DIGITAL	VT_BOOL	::[OPCLX_test] Program:MainProgram. bool_symbol
::[OPCTopic]<Program:Ma inProgram>. <item_id>	BYTE	VT_UI1	::[OPCLX_test] Program:MainProgram. byte_symbol
::[OPCTopic]<Program:Ma inProgram>. <item_id>	INT	VT_I2	::[OPCLX_test] Program:MainProgram. int_symbol
::[OPCTopic]<Program:Ma inProgram>. <item_id>	LONG	VT_I4	::[OPCLX_test] Program:MainProgram. long_symbol
::[OPCTopic]<Program:Ma inProgram>. <item_id>	REAL	VT_R4	::[OPCLX_test] Program:MainProgram. real_symbol
::[OPCTopic]<Program:Ma inProgram>. <item_id>	STRING	VT_ARRAY of VT_UI1	::[OPCLX_test] Program:MainProgram. bool_symbol

## Arrays Support

To associate a Plant SCADA variable tag with an OPC array, you can place a delimiter like "!" after OPC array address string. The capital letter "A" which when combined as below, instructs the Plant SCADA OPCLX driver that the address is for an array. Follow both with the array size as an integer enclosed within square brackets, as below:

<OPCTOPIC><OPCArrayName>,L<OPCArrayLength>!A[<OPCArrayLength>]

---

**Note:** Replace the placeholders <OPCTOPIC>, <OPCArrayName>, and <OPCArrayLength> in the Address box with the actual valid values of the OPC Topic, OPC array name, and length you want to use, as defined in the OPC server.

In the following example with an OPCLX tag defined on a OPC server, five register addresses are associated with the Plant SCADA variable tag named Status\_Flags:

- **Variable Tag Name:** Status\_Flags
- **Address:** [OPCLX\_test]status\_flags,L10!A[10]

You can access individual elements of an OPC array in Plant SCADA by specifying the Plant SCADA tag name and an index number representing the individual element of the array. For example, to refer to the third variable of the array in the above example (Status\_Flags), use this syntax in Plant SCADA:

Variable tag: Status\_Flags[2]

---

**Note:** OPC arrays use zero-based indexing. For example, a five register array would contain individual elements

---

indexed as 0, 1, 2, 3, and 4.

The OPCLX address checking in Plant SCADA always checks if the '!A[<n>]' suffix is used; otherwise the tag is not recognized as an OPC array. Therefore the format '!A[<n>]' must be followed in a tag address if the tag is defined as an array tag.

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

Ensure that the OPCLX address format used is compliant for the OPC server with which you are communicating.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## **Example**

When Plant SCADA writes to an OPC array, the whole array is read from the driver cache, modified, then written back to the OPC server. However, the driver cache is not updated with the newer data until the server performs its housekeeping tasks and notifies the driver that the array value has been changed. During this time, it is possible for another write request to have been issued with a now out-of-date copy of the original array data from the not-yet-updated driver cache. Later if you write back to the server, any changes made in the first request of this scenario may be lost inadvertently if they are overwritten by the original duplicated data in the second write-back.

To minimize this, the OPCLX driver in Plant SCADA is forced to do a synchronized read before data is modified. However it still has the possibility of losing data if that data is changed by other OPC clients between the time the data is read out and written back. For this reason, it is not recommended that OPC arrays be used in Plant SCADA while other OPC clients may be accessing the same data.

## **WARNING**

### **UNINTENDED EQUIPMENT OPERATION**

Do not use OPC arrays in Plant SCADA while other OPC clients have access to the same data.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

The table below describes the syntax and examples for configuring a full array accessing tags:

Address	Plant SCADA type	OPCLX type	Example
[OPCTopic]<item_id>, L<size>!A[size]	DIGITAL	array of VT_BOOL	[OPCLX_test]bool_symbol, , L10!A[10]
[OPCTopic]<item_id>, L<size>!A[size]	BYTE	array of VT_UI1	[OPCLX_test]byte_symbol, , L10!A[10]
[OPCTopic]<item_id>, L<size>!A[size]	INT	array of VT_I2	[OPCLX_test]int_symbol, , L10!A[10]

Address	Plant SCADA type	OPCLX type	Example
[OPCTopic]<item_id>, L<size>!A[size]	LONG	array of VT_I4	[OPCLX_test]long_symbol, L10!A[10]
[OPCTopic]<item_id>, L<size>!A[size]	REAL	array of VT_R4	[OPCLX_test]real_symbol, L10!A[10]

The table below describes the syntax and examples for configuring index-based array tags:

Address	Plant SCADA type	OPCLX type	Example
[OPCTopic]<item_id>[index]	DIGITAL	VT_BOOL	[OPCLX_test]bool_symbol[9]
[OPCTopic]<item_id>[index]	BYTE	VT_UI1	[OPCLX_test]bytee_symbol[9]
[OPCTopic]<item_id>[index]	INT	VT_I2	[OPCLX_test]int_symbol[9]
[OPCTopic]<item_id>,[index]	LONG	VT_I4	[OPCLX_test]long_symbol[9]
[OPCTopic]<item_id>[index]	REAL	VT_R4	[OPCLX_test]real_symbol[9]

The table below describes the syntax and examples for configuring full array accessing tags from the specified start index:

Address	Plant SCADA type	OPCLX type	Example
[OPCTopic]<item_id>[index],L<size>!A[size]	DIGITAL	array of VT_BOOL	[OPCLX_test]bool_symbol[1], L10!A[10]
[OPCTopic]<item_id>[index],L<size>!A[size]	BYTE	array of VT_UI1	[OPCLX_test]bytee_symbol[1], L10!A[10]
[OPCTopic]<item_id>[index],L<size>!A[size]	INT	array of VT_I2	[OPCLX_test]int_symbol[1], L10!A[10]
[OPCTopic]<item_id>[index],L<size>!A[size]	LONG	array of VT_I4	[OPCLX_test]long_symbol[1], L10!A[10]
[OPCTopic]<item_id>[index],L<size>!A[size]	REAL	array of VT_R4	[OPCLX_test]symbol[1], L10!A[10]

The table below describes the syntax and examples for configuring structured array tags:

Address	Plant SCADA type	OPCLX type	Example
[OPCTopic]<item_id> [index1].<subindex> [index2],L<size>!A[size]	DIGITAL	array of VT_BOOL	[OPCLX_test]bool_symbol [1].tag[2], L10!A[10]
[OPCTopic]<item_id> [index1].<subindex> [index2],L<size>!A[size]	BYTE	array of VT_UI1	[OPCLX_test]byte_symbol [1].tag[2], L10!A[10]
[OPCTopic]<item_id> [index1].<subindex> [index2],L<size>!A[size]	INT	array of VT_I2	[OPCLX_test]int_symbol[1] .tag[2], L10!A[10]
[OPCTopic]<item_id> [index1].<subindex> [index2],L<size>!A[size]	LONG	array of VT_I4	[OPCLX_test]long_symbol [1].tag[2], L10!A[10]
[OPCTopic]<item_id> [index1].<subindex> [index2],L<size>!A[size]	REAL	array of VT_R4	[OPCLX_test]symbol[1].ta g[2], L10!A[10]

## See Also

[Logging Parameters](#)

## Quality Values

The following masks are general bitmasks:

OPC_QUALITY_MASK	0xC0
OPC_STATUS_MASK	0xFC
OPC_LIMIT_MASK	0x03

These are specific quality values:

OPC_QUALITY_BAD	0x00
QUALITY_UNCERTAIN	0x40
OPC_QUALITY_GOOD	0xC0
OPC_QUALITY_CONFIG_ERROR	0x04
OPC_QUALITY_NOT_CONNECTED	0x08
OPC_QUALITY_DEVICE_FAILURE	0x0C

OPC_QUALITY_SENSOR_FAILURE	0x10
OPC_QUALITY_LAST_KNOWN	0x14
OPC_QUALITY_COMM_FAILURE	0x18
OPC_QUALITY_OUT_OF_SERVICE	0x1C
OPC_QUALITY_LAST_USABLE	0x44
OPC_QUALITY_SENSOR_CAL	0x50
OPC_QUALITY_EGU_EXCEEDED	0x54
OPC_QUALITY_SUB_NORMAL	0x58
OPC_QUALITY_LOCAL_OVERRIDE	0xD8

These can modify any of the above:

OPC_LIMIT_OK	0x00
OPC_LIMIT_LOW	0x01
OPC_LIMIT_HIGH	0x02
OPC_LIMIT_CONST	0x03

## See Also

[Time Stamp and Quality Tags Configuration](#)

### Time Stamp and Quality Tags Configuration

The Plant SCADA OPCLX driver can use additional variable tags for quality and time stamp values. Quality or time stamp tags are duplicates of a variable tag, with an appended "!Q" or "!T" or "!M" in the address field to define their purpose. Quality tags need to be configured as either INT or DIGITAL data types, whereas time stamp tags need to be configured as LONG data types.

Below table shows the example of quality or timestamp tags:

Address	Plant SCADA type	Description
[OPCTopic]<item_id>!Q	INT	The quality reported for the item.
[OPCTopic]<item_id>!T	LONG	The second's component of the timestamp reported for the item (in UTC seconds).
[OPCTopic]<item_id>!M	LONG	The milliseconds component of the timestamp reported for the item.

If you need to know if an OPC item is good or bad quality (that is, not uncertain), you should consider using DIGITAL data types for quality tags in combination with [OPCLX]DigitalQualityUncertainIsBad, rather than using Cicode to compare the value of an INT quality tag against 0xCO

---

**Note:** The quickest way to create a quality or time stamp tag in Plant SCADA is to locate the relevant variable tag using the Variable Tags dialog box. Edit it as required to create a quality or time stamp tag, and save it. This creates a copy of the original variable tag with the required changes saved.

#### To create a quality tag:

1. Go to the **System Model** activity of the Plant SCADA **Studio**.
2. On the menu below the Command Bar, select **Variables**.
3. Add a row to the Grid Editor, or locate and display the equipment to which you want to associate the quality tag.
4. Type the required information in each column, or in the fields in the **Property Grid**.
5. In the **Tag Name** box, enter an appropriate and unique name for the quality tag, such as "<Tagname>\_Quality".
6. In the **Data Type** box, choose INT or DIGITAL from the menu.
7. In the **I/O Device** box, select the appropriate I/O device for the tag (if not already displayed).
8. In the **Address** box, enter the name of the tag you want to associate with the quality value (if not already displayed), and append "!Q" directly to the end of the address (without the quotes and without any spaces).
9. Click **Save**.

---

**Note:** If the exclamation mark character (!) is already defined in the tag address field (for example, you have already used it to declare an array size "!A[n]"), you either need to include another exclamation mark (for example, append "!Q" to the tag address) after the closing square bracket of the array size declaration, or you can specify quality for arrayTag!A[5] with arrayTag!Q".

#### To create a time stamp tag:

1. Go to the **System Model** activity of the Plant SCADA **Studio**.
2. On the menu below the Command Bar, select **Variables**.
3. Add a row to the Grid Editor, or locate and display the equipment to which you want to associate the quality tag.
4. Type the required information in each column, or in the fields in the **Property Grid**.
5. In the **Tag Name** box, enter an appropriate and unique name for the quality tag, such as "<Tagname>\_sTimestamp" (for seconds) or "<Tagname>\_msTimestamp" (for milliseconds).
6. In the **Data Type** box, choose LONG from the menu.
7. In the **I/O Device** box, select the appropriate I/O device for the tag (if not already displayed).
8. In the **Address** box, enter the name of the tag you want to associate with a time stamp value (if not already displayed), and append "!T" or "!M" (as appropriate).

---

**Note:** If there is already an exclamation mark with other declarations included in the tag address, you do not need to include them for timestamping. For example, a time stamp tag monitoring TagName!A[64] can be addressed as TagName!T.

1. Click **Save**.

**Note:** Time stamps store the number of seconds since 01/01/1970 when using the "!T" element, or the number of milliseconds since midnight using the "!M" element. If you want to store a complete time in millisecond accuracy, you need to create two separate time stamp tags, one for seconds and one for milliseconds, each addressed appropriately.

## Examples

In the following examples, an OPC device tag defined on a OPC server is associated with the base Plant SCADA variable tag named Present\_Value:

<b>Variable Tag Name</b>	Present_Value
<b>Address</b>	[OPCLX_test]present_value
<b>Plant SCADA Data Type</b>	Data type required by project (for a list of data types, see <a href="#">Data Types</a> )

### Example 1

Use the following settings to associate a Plant SCADA variable tag with the quality stamp of the OPC register value:

<b>Variable Tag Name</b>	Present_Value
<b>Address</b>	[OPCLX_test]present_value!Q
<b>OPCLX Type</b>	VT_QUALITY
<b>Plant SCADA Data Type</b>	INT or DIGITAL

### Example 2

The following tag definition reads the timestamp value measured in seconds from 1/Jan/1970:

<b>Variable Tag Name</b>	Present_Value
<b>Address</b>	[OPCLX_test]present_value!T
<b>OPCLX Type</b>	VT_TIMESTAMP
<b>Plant SCADA Data Type</b>	LONG

### Example 3

The tag definition below reads the Millisecond time stamp value measured in UTC FileTime from beginning (midnight) of the current day:

<b>Variable Tag Name</b>	Present_Value
<b>Address</b>	[OPCLX_test]present_value!M
<b>OPCLX Type</b>	VT_MILLISECOND
<b>Plant SCADA Data Type</b>	LONG

The base Plant SCADA tag needs to be defined in Plant SCADA before the time stamp and quality values of the associated OPC server tag value can be read.

See also [Data Types](#) and [Quality Values](#).

## Advanced Configuration and Maintenance

This section provides advanced configuration instructions.

- [Status Tags](#)
- [Customize a Project Using Citect.ini Parameters](#)
- [Configure Redundancy](#)
- [Scan Rates](#)
- [Diagnostics](#)

### Status Tags

You can define status tags in your Citect.ini file to allow the OPCLX driver to determine the current state of an OPCLX device. For example, a status tag could be configured to monitor an OPC item that indicates the current operational state of a connected device. If the condition of the status tag is true, the OPCLX driver can allow the device to be online; otherwise the device may be set to offline.

By default, no status tags are created for the OPCLX driver. They need to be defined in the project Citect.ini file using the OPCLX status tag parameters. These parameters determine which OPC items are monitored, the condition values, and how often polling occurs.

A status tag can be defined in the [OPCLXStatusTags] section of the Citect.ini file to determine the online status of OPCLX devices. You can also create status tag set for each individual I/O device using the following format:

```
[OPCLXStatusTags]
<IOServerName>.<IODeviceName>.<parameter>=<value>
```

For example:

```
[OPCLXStatusTags]
io1.iodev1.tag=ARB3183800_S_2.REQNUM
io1.iodev1.condition=<=10
```

Where:

"ARB3183800\_S\_2.REQNUM" refers to an OPC item configured in the device.

If you are not regularly reading from a particular device, it is recommended that you enable status unit checks. To do this, you need to enable the parameter [IOServer]WatchDogPrimary. Standby units are enabled by default.

See also [Time Stamp and Quality Tags Configuration](#).

## Customize a Project Using Citect.ini Parameters

You use Citect.ini parameters to fine-tune the performance of the Plant SCADA OPCLX driver and to perform runtime maintenance diagnostics.

- Common Driver Parameters
- Driver-specific Parameters
- OPCLX Device-specific Parameters
- OPCLX Access Path Parameters
- Status Tag Parameters
- Logging Parameters

**Note:** Support for array parameters in the OPCLX driver has been deprecated in Plant SCADA version 7. To use similar functionality, you should use the procedures described in [Arrays Support](#).

### **WARNING**

#### UNINTENDED EQUIPMENT OPERATION

- Under any circumstances do not change or remove any of the undocumented citect.ini parameters.
- Before deleting sections of the citect.ini file, confirm that no undocumented parameters are deleted.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of technical support personnel for this product regarding undocumented features.

You can customize the way Plant SCADA communicates with the OPC system, and even individual PLCs, by creating or editing the [OPCLX] section of the Citect.ini file for your project.

There are some common Plant SCADA OPCLX driver settings, and custom Plant SCADA OPCLX driver settings. When Plant SCADA starts at runtime, it reads configuration values from the Citect.ini file stored locally. Therefore, any OPCLX configuration settings need to be included in the Citect.ini file located on the computer acting as the I/O server to the OPC system.

**Note:** By default, Plant SCADA looks for the ini file in the Plant SCADA project \bin directory. If it cannot find it there, it searches the default Windows directory.

The default values for these parameters have been determined by driver development and testing to be optimal for the Plant SCADA OPCLX driver. Parameters with a default value are tested to work in common configurations. Do not adjust these default values except on the direct advice of customer support.

### Common Driver Parameters

The following table describes the common citect.ini driver parameters.

[OPCLX] Parameters	Allowable values	Default value	Description
Block	5 to 256	255	A value (bytes) used by

[OPCLX] Parameters	Allowable values	Default value	Description
			the Plant SCADA I/O server to determine if two or more packets can be blocked into one data request before being sent to the OPCLX driver. For example, if you set the value to 10, and the I/O server receives two simultaneous data requests (one for byte 3, and another for byte 8) the two requests are blocked into a single physical data request packet. This single request packet is then sent to the I/O device, saving on bandwidth and processing.
Delay	0 to 300	0	Period (in Milliseconds) to wait between receiving a response and sending the next command to the I/O server.
MaxPending	1 to 1024	250	Maximum number of pending commands that the Plant SCADA OPCLX driver holds ready for immediate execution.
Poltime	0 to 300	1000	Polling service time (in milliseconds) at which cached data is updated. Recommended to be at least twice that of the value used in the Citect.INI [ALARM] ScanTime parameter.
Retry	0 to 8	0	Number of times to retry a command after a timeout.

[OPCLX] Parameters	Allowable values	Default value	Description
Timeout	0 to 32000	3000	Waiting period (in Milliseconds) between a response before considering a write or polling request to be invalid and displaying a detected error message in Plant SCADA.
Watchtime	0 to 128 (seconds)	15	Frequency that the Plant SCADA OPCLX driver uses to check the communications link to the OPC system and attempts to bring offline units back online. Also determines the frequency of STATUS_UNIT commands to the driver. See Configuring Redundancy.

### Driver-specific Parameters

The following table describes the citect.ini parameters that are specific to the OPCLX driver.

[OPCLX] Parameters	Allowable values	Default value	Description
AddItemAsVtEmpty	0 - Specify data types 1 - Use native data type	1	Determines whether the OPCLX driver specifies a data type when adding an item, or whether it uses the native data type when adding items. This parameter is used to select whether the OPC client specifies a type to force the OPC server to coerce the data, or for the OPC server to provide the type as configured on the server.
ArrayRecordBase	Greater than total	30000	Specifies the base of the

[OPCLX] Parameters	Allowable values	Default value	Description
	number of tags and less than (65000 – number of arrays)		array record numbers. Plant SCADA communicates with the OPCLX client driver by the record number of the tag in variable.dbf, so this parameter needs to be greater than the total number of tags in the project.
CacheRead	0 – Value is refreshed and the latest value returned. 1 – Last known value returned.	1	Determines the value returned when an inactive item is read. See <a href="#">Overriding [OPCLX]CacheRead</a> for information on how to override the value of this parameter for a particular tag.
DigitalQualityUncertainIsBad	0 - An uncertain quality is treated as GOOD when a digital quality reading is taken (normal INT version is unaffected)  1 - an uncertain quality is treated as BAD when a digital quality reading is taken (normal INT version is unaffected)	1	Determines how an uncertain quality result is treated when a digital quality reading is taken.
FailOnBadData	0 - If the OPC_QUALITY status for data from a tag is BAD, the driver returns the last known value, and reports no error detected.  1 - If the OPC_QUALITY status for data from a tag is BAD, the driver reports a severity error detected and displays #BAD on all tags of the read block.  2 - If the OPC_QUALITY status for data from a tag	1	Determines whether read requests sent to the OPC client driver return a detected error if one or more of the requested tags are of bad quality data. Bad quality is still a valid value for a tag, and indicated by the OPC_QUALITY status of BAD.

[OPCLX] Parameters	Allowable values	Default value	Description
	is BAD, the driver reports a severity error detected and displays #BAD on all tags of the read block. The start UnitAddress of the read block appears in the kernel window.		
FailOnUncertain	0 - Driver returns the last known tag value and the read succeeds. 1 - Driver returns ERROR_BAD_DATA_VALUE .	0	Determines the value returned when reading data from tags for which the OPC_QUALITY status is UNCERTAIN.
FillCacheOnStartup	0 - Subscriptions are created in the inactive state and are only activated as read request are received. 1 - Subscriptions are created in the active state.	0	Determines whether subscriptions are created in the active or inactive state initially.
GoOfflineForBadTag	0 - disabled. Do not go offline. 1 - enabled. Go offline.	0	Sets all tags on an OPC device to a OPC_QUALITY status of GOOD before Plant SCADA declares the I/O device online.  This parameter would only be set on the I/O server where the device is defined as primary. So it can only take over from a standby device definition when all tags and the status tag are valid. This minimizes the chances that com-breaks occur during switchover from standby to primary. It is recommended not to set it on a standby, as it may only be one tag that is invalid.

[OPCLX] Parameters	Allowable values	Default value	Description
InhibitActivationOnStandby	0 - A standby server follows the same rules for subscription activation as the primary server.  1 - A standby server only activates a subscription if it believes that the primary server is not active.	0	This parameter allows a standby server to leave subscriptions in an inactive state deliberately, when the primary server is online. This is used when the primary and standby servers are talking to independent OPC servers to reduce the load that the standby OPC server would place on field devices.  On startup of a standby server, if this parameter is set to 1, items are not activated immediately. They are only activated when the unit itself becomes the active unit.
ItemLifeTime	Time (seconds)	5	Determines the time in before a tag is deemed inactive.  The global LeaveTagsActive parameter overrides the ItemLifeTime parameter to force OPCLX tags to remain active even though Plant SCADA is no longer requesting them from the driver.  The ItemLifeTime parameter can be individually overridden for any Allen Bradley OPC device by the active parameter.

[OPCLX] Parameters	Allowable values	Default value	Description
LeaveTagsActive	0 - Do not leave tags active. 1 - Leave tags active.	0	<p>Determines whether OPCLX tags remain active even though Plant SCADA is no longer requesting them from the driver. Normally tags are made inactive 5 seconds from the last time they are requested.</p> <p>Some OPC servers take a long time to return a call to activate/inactivate tags, or doing so slows them down. Therefore leaving tags active can improve performance.</p> <p>The global LeaveTagsActive parameter can be individually overridden by an Active parameter. See ItemLifeTime parameter (above).</p> <p><b>Note:</b> This parameter minimizes the likelihood of deactivation of tags on the currently active unit only. Tags on units that have become inactive (standby) still deactivates.</p>
NumConcurrentServerInits	-1 - Unlimited 32767	-1	Represents the maximum number of concurrent initialization and shutdown operations that are performed on each configured OPC server. Higher values, or -1 (unlimited), incur greater load on the server. While 0 and 1 determine that initialization is serialized into effectively a single thread.

[OPCLX] Parameters	Allowable values	Default value	Description
OPCGroupName	Any string (maximum 32 characters)	CitectGroup_<n>	<p>By default, each OPC server is allocated a sequential group name by the OPCLX driver, that is, CitectGroup_1, CitectGroup_2, and so on.</p> <p>This parameter allows you to replace the default group name with a customized name.</p> <p>You can set this parameter at a device level using <a href="#">OPCLX Device-specific Parameters</a>.</p>
ReadAfterWrite	<p>0 - No read, rely on the normal polling/ OnDataChange mechanism.</p> <p>1 - Force sync read of item written from OPC server cache after write.</p> <p>2 - Force sync read of item written from device after write.</p>	0	<p>Determines if a force read should be made, and if it should be from the OPC server cache or from the device.</p> <p><b>Note:</b> RefreshAfterWrite causes a refresh of all active items, whereas ReadAfterWrite reads only the item being written. You should use one or the other. Where both RefreshAfterWrite and ReadAfterWrite are set to 1, RefreshAfterWrite takes precedence, and ReadAfterWrite is set to 0.</p>
RefreshAfterAdd	<p>0 - No REFRESH.</p> <p>1 - Request a REFRESH from the OPC server cache.</p> <p>2 - Request a REFRESH from the device.</p>	1	<p>Determines whether an OPC REFRESH is requested from the OPC server after activating the group.</p> <p>This parameter can be disabled if slow start-up time is experienced. The server could be automatically sending a REFRESH, in which case another request may be</p>

[OPCLX] Parameters	Allowable values	Default value	Description
			unnecessary.
RefreshAfterWrite	0 – Disable forced REFRESH.  1 - Enable forced REFRESH from OPC server cache.  2 - Enable forced REFRESH from device.	0	Determines whether to force an OPC REFRESH from the OPC server of ALL active items after Plant SCADA writes to an item.
RefreshBeforeArrayWrite	0 – No refresh command sent  1 – Read command sent to refresh the OPC server cache  2 – Read command sent to refresh the OPC device	0	Enables the driver to send a read command to the OPC server before a write command is sent to the OPC server for an array variant.
ServerOnlineStates	"RUNNING"  "FAILED"  "NOCONFIG"  "SUSPENDED"  "TEST"	"RUNNING"	Defines the list of OPC server states that the driver considers online. The format is a "   " separated string, without spaces. If you experience the OPC server shutting down on startup, it is recommended that the following states are used: "RUNNING   NOCONFIG   SUSPENDED"
ShutdownWait	Time (milliseconds)	30000	Determines the time the OPCLX driver waits before shutting down. On shutdown, the driver attempts to close the connection to the OPC server. If this takes longer than the time specified by this parameter, the driver shuts down anyway

[OPCLX] Parameters	Allowable values	Default value	Description
			(sometimes with unexpected results). This parameter may be needed if the OPC server does not respond within the default wait time.
StatusWaitPeriod	Time (milliseconds)	50000	Determines the time the OPCLX driver waits before requesting tag status for an OPC I/O device. On startup, the OPCLX driver waits for a unit status from the OPC server before adding tags to the server. If the value of this parameter is exceeded, the tags are added anyway. The OPC server should respond within a few seconds. The default wait time of 5 seconds should usually be adequate.
SuppressData NotYetValidError	0 – Displays "Data Not Yet Valid" errors in the Kernel window and logs them to the Syslog.dat file.  1 - Suppresses "Data Not Yet Valid" errors, which are not displayed in the Kernel window or logged. In the kernel, under View -> I/O Devices, Plant SCADA will display "Error Suppressed" in the Error Message field for the particular device.	0	Determines whether "Data Not Yet Valid" error messages are suppressed, or whether they are displayed in the Kernel window and written to the Syslog.dat file. For Plant SCADA to display the message "Error Suppressed", you need v5.50 or above.

[OPCLX] Parameters	Allowable values	Default value	Description
UseArrays	0 - OPC array support disabled. 1 - OPC array support enabled.	1	Determines whether array support parameters in the [OPCArrays] section are active. See OPC Array Parameters.
UseAsyncWrites	0 - use synchronous writes. 1 - use asynchronous writes.	0	Determines whether writes are synchronous or asynchronous.
UseStatusTags	0 - disabled. 1 - enabled.	0	Determines whether parameters in the [OPCLXStatusTags] section are active. Individual overrides can be set using the <a href="#">OPCLX Device-specific Parameters</a> to force the use of a particular status tag parameter with a particular I/O device.
ValidDataWaitPeriod	Time (milliseconds)	10 seconds	<p>Determines the maximum time that the OPCLX driver waits for OPC_QUALITY to be GOOD for all tags before declaring the device online. This minimizes the likelihood of a client swapping from a standby to a primary prematurely and suppresses alarm flooding at startup. Set this parameter to 0 to allow units to be online regardless of whether all items for the unit have been verified as good quality.</p> <p><b>Note:</b> FillCacheOnStartup=1</p>

[OPCLX] Parameters	Allowable values	Default value	Description
			needs to be set when ValidDataWaitPeriod>0.
WriteTrueAs1	0 - write true as VARIANT_TRUE(-1). 1 - write true as 1.	0	Determines whether a digital value of TRUE is written as "VARIANT_TRUE(-1)" or "1". This parameter should only be required for incorrectly designed OPC servers. When the Plant SCADA data type is digital, VARIANT_TRUE (-1) is used with VT_BOOL variant type to write a value of true. This happens even if [OPCLX]AddItemAsVtEmpty is set, as OPC servers should be designed to convert the value when they receive it.

### OPCLX Device-specific Parameters

The following table describes the citect.ini parameters that are specific to the OPCLX driver.

[OPCLX] Parameters	Allowable values	Default value	Description
Active	0 - deactivate tag. 1 - leave tag active.	Value of LeaveTagsActive parameter.  See <a href="#">Driver-specific Parameters</a> .	Allows you to override the global default value of LeaveTagsActive parameter individually to specify whether tags should be left active or inactive for a particular OPCLX I/O device.  Replace the placeholders <IOServerName> and <IODeviceName> in the

[OPCLX] Parameters	Allowable values	Default value	Description
			<p>parameter with the names of the OPC I/O server and I/O device you want to override and be active.</p> <p>Determines whether individual OPCLX tags are left active even though Plant SCADA is no longer requesting them from the driver. Normally tags are made inactive 5 seconds from the last time they are requested.</p> <p>See the ItemLifeTime parameter in <a href="#">Driver-specific Parameters</a>.</p>
Update	Time (seconds)	5	<p>Determines the time in before a tag is deemed inactive.</p> <p>The global LeaveTagsActive parameter overrides the ItemLifeTime parameter to force OPCLX tags to remain active even though Plant SCADA is no longer requesting them from the driver.</p> <p>The ItemLifeTime parameter can be individually overridden for any Allen Bradley OPC device by the active parameter.</p>
UseAsyncWrites	0 - use synchronous writes. 1 - use asynchronous writes.	0	Determines whether writes are synchronous or asynchronous.

## OPCLX Access Path Parameters

OPCLX does not include access path parameters since OPC topic is addressed as a part of variable tag address.

### Status Tag Parameters

The OPCLX status tag parameters determine the online or offline state of an OPCLX I/O device using the condition of a specified OPC item in the device. For more information, see [Status Tags](#).

When configuring these parameters, be careful not to confuse the OPC item required for status monitoring with a normal Plant SCADA tag.

You should also use a full path to identify an OPC item, as a device does not know a configured access path in your Citect.ini file.

**Note:** Use the UseStatusTags parameter to enable or disable support for OPCLX status tags (see See "Driver-specific Citect.ini Parameters").

[OPCLXStatusTags] Parameters	Allowable values	Default value	Description
StartWait	Time in milliseconds - or - -1 (infinite)	30000 (milliseconds)	On driver startup, this parameter determines the time the OPCLX driver waits before requesting status tag data for an OPC I/O device.  The OPC server should respond within a few seconds. The default wait time of 30 seconds is adequate.
Default.Tag	An OPC item name	STATUS	This parameter specifies the default OPC item to be monitored at runtime to determine the operational status of an associated device.  <b>Note:</b> This parameter implements a global setting for all OPC I/O devices, unless it is overridden using the equivalent <a href="#">OPCLX Device-specific Parameters</a> .
Default.Update	0 -32000 (milliseconds)	The scan rate from <b>Address</b> field in the Plant SCADA Boards settings	This parameter specifies the default update rate for OPCLX status tags at

[OPCLXStatusTags] Parameters	Allowable values	Default value	Description
			<p>runtime. This is the frequency with which the OPC server polls the device for status tag values and then updates its own cache.</p> <p>This parameter controls polling performed by the OPC server, not by Plant SCADA. The scope of the parameter is to communicate between the OPC server and a device, not between the OPC server and Plant SCADA.</p> <p><b>Note:</b> This parameter implements a global setting for all OPC I/O devices, unless it is overridden using the equivalent <a href="#">OPCLX Device-specific Parameters</a>.</p>
Default.Condition	<condition> (see table below)	ISGOOD	<p>This parameter is used to determine if the OPC device is online at runtime, by comparing the &lt;Condition&gt; with the value of the default OPC item named in the.Default.Tag parameter.</p> <p>If the specified condition is determined to be true, the device is considered online.</p> <p>This parameter represents the default condition for all OPCLX status tags, unless it is individually overridden using the <a href="#">OPCLX Device-specific Parameters</a>.</p>

Allowable values for <condition>	Description
=<Value>	Is equal to <Value>
!<Value>	Is NOT <Value>
!=<Value>	Is NOT equal to <Value>
><Value>	Is greater than <Value>
>=<Value>	Is greater than or equal to <Value>
<<Value>	Is less than <Value>
<=<Value>	Is less than or equal to <Value>
ISGOOD	The tag quality is GOOD
ISBAD	The tag quality is BAD
ISUNCERTAIN	The tag quality is UNKNOWN

Where <Value> is a placeholder for any valid LONG data type value.

If the specified condition is determined to be true, the device is considered online.

See also [Status Tags](#).

## Logging Parameters

The following table describes the OPCLX logging parameters.

[OPCLX] Parameter	Description	Allowable Values	Default Value
DebugLevel	The trace level used for log file. See <a href="#">Logging</a> for examples.	ALL or any combination of the following, separated by a pipe character ( ): WARN ERROR TRACE	-
DebugCategory	The categories used for the log file. See <a href="#">Logging</a> for examples.	ALL or any combination of the following, separated by a pipe character ( ): TAG: Tag configuration trace. PROT: Operations related to the OPC protocol interface. CACHE: Operations	-

		<p>related to the driver cache.</p> <p>DCB: Front end driver trace.</p> <p>THRD: Thread trace.</p> <p>MISC: Miscellaneous operations.</p> <p>BEND Operations related to the back-end thread of the driver.</p> <p>FEND: Operations related to the front-end thread of the driver.</p>	
DebugUnits	Controls which units logging is performed for. Multiple units can be specified. Debug messages that are unrelated to a specific unit are logged against the (GLOBAL) unit.	ALL, (GLOBAL) [include the round brackets] or any combination of I/O device names and (GLOBAL) separated by ' ' characters.	All

### Overriding [OPCLX]CacheRead

You can override the value in [OPCLX]CacheRead for a particular tag by suffixing the tag address with:

- The delimiter "!"
- The letter C or D, as follows:

<Tag Name>!C - The tag value returned is the cache value at the time of the read.

<Tag Name>!D - A refresh of the tag value is forced before it is returned.

### Examples:

- Tag1!**D** - Forces a refresh of Tag1 before the value is returned.
- Tag2!**C** - The last known cache value of Tag2 is returned.

When the tag qualifiers C and D are used with other OPCLX qualifiers (see [Time Stamp and Quality Tags Configuration](#)), they must appear last.

For example, with "Tag3!Q!C", Tag3 is associated with a quality tag and the last known value of Tag3 is returned

### Configure Redundancy

Redundancy is handled in Plant SCADA by the Plant SCADA clients. They determine which I/O server to request I/O device data by allowing the scan rate to be specified per I/O device, and then configuring multiple I/O

devices to the same PLC.

A status tag can be defined in the Citect.ini file to determine whether to set a unit offline or online depending on the condition of the status tag. If the condition is true the unit is allowed to come online, otherwise the unit is set to offline.

The status tag is polled at the same rate configured for the groups of the corresponding unit. The driver supports STATUS\_UNIT commands on the inactive server. So even though a particular unit is not currently servicing Plant SCADA requests, the state of the status condition is checked periodically. This allows the unit to be taken either offline or online (assuming that [IOServer]WatchDog has not changed from its default of 1).

Both the primary and standby devices are offline if both have the same status condition and the condition is a detected error condition. All points in the system for that unit would therefore be #COM.

Status Tags can be defined on a per-device or global basis. For more information, see [OPCLX Status Tag Parameters](#) and [OPCLX Device-specific Parameters](#).

## Scan Rates

The Plant SCADA OPCLX driver can define I/O devices with different scan rates. This provides you with the ability to tune your system better, by ordering the tag database into logical groups and defining different scan rates for each group. For example, tags that are read frequently can be placed in a group that is scanned frequently. Tags that are read infrequently can be placed in a group that is scanned infrequently.

By default, tags are scanned at the default scan rate defined by the ScanRate parameter in the project Citect.ini file.

## Examples

- Set the project scan rate to 1 second (1000 milliseconds):  
[OPCLX]  
ScanRate=1000
- Set the project scan rate to 1 second (1000 milliseconds), and the inactive server scan adjust rate to a factor of 5:  
[OPCLX]  
ScanRate=1000
- Set a faster (half second) scan rate for a specific port named "Port1" on the I/O server named "CLServer":  
[CLServer.Port1]  
ScanRate=500
- Set a slower (5 second) scan rate for a specific port named "Port123" on the I/O port named "CLServer":  
[CLServer.Port123]  
ScanRate=5000

## Diagnostics

This feature is supported only by the FactoryTalk OPC server and not by the RSLinx Classic OPC server. FactoryTalk diagnostics collects and provides access to activity, status, warning, and detected error messages generated throughout a FactoryTalk system.

The syntax for configuring the diagnostic tag in Plant SCADA is:

[OPCTOPIC]@<DiagnosticItemName>

For further details of the diagnostic items, refer to the FactoryTalk documentation.

## Migrate from ABCLX to OPCLX

A migration tool can be used to migrate the Plant SCADA projects from ABCLX to the standard OPC formats supported by the OPCLX driver.

This eases upgrading the projects and automating the configuration of the old ABCLX projects to the new format.

- [Migrate a Project from ABCLX to OPCLX](#)
- [Migration of I/O Devices](#)
- [Tag Address Formats](#)

### Migrate a Project from ABCLX to OPCLX

This chapter provides details on how to migrate from a Plant SCADA project configured with the ABCLX driver to the OPCLX driver.

---

**Note:**

- The migration feature is available in all versions of Plant SCADA from v7.20. The migration tool may require a Service Pack and/or Hotfix on versions prior to v7.50.
  - Virtual Port (VP) is obsolete in OPCLX compare to ABCLX.
- 

**To migrate a project from ABCLX to OPCLX:**

1. Open the **Projects** activity in Plant SCADA Studio.
  2. Select the project that you want to migrate.
  3. On the command bar, select **Migration Tool**.  
The **Project Migration** dialog box appears.
  4. In the **Select a project to migrate** drop-down box, select the project that you want to migrate.
  5. Initially the **Migrate ABCLX to OPCLX** check box is disabled. Click **Configure**. The <Project>:<Project Name> dialog box appears.  
For more information on the fields on this dialog box, see the *Field Description and Layout of <Project>:<Project Name> Dialog Box* section below.
  6. In the **OPCServer Type** group box, select the OPC server name (**FactoryTalk** or **RSLinx Classic**).
  7. If you want to migrate all the listed devices to OPCLX, then choose **Select All** check box. Else, select the required I/O devices.
  8. You can change the corresponding **OPC Topic** and **OPC Server**.
- 

**Note:**

- By default the **OPC Server** drop-down list is empty with an edit option to enter the OPC server name manually.
  - To select an OPC server under **OPC Server** drop-down list, in the **OPCServerBrowse** field, click **Browse**. The available OPC servers on the local machine are listed under OPC Server drop-down list. Select the required OPC server from the drop-down list. If no servers are found, then click **edit** and enter OPC server details manually.
-

9. If you want to browse the OPC servers on a remote machine, in the **Machine Name** field, enter the IP address or machine name of the remote server..

---

**Note:**

- The Browse button is disabled until you enter the remote machine name or IP address.
  - If the machine name or IP address is invalid, a pop-up with the message invalid machine name is displayed.
  - If the local machine is unable to establish connection with the specified remote machine, a pop-up with the message **Failed to retrieve OPC servers** is displayed.
  - If the connection is not successful, click edit and provide OPC server details manually.
- 

On successful connection with the remote server, the OPC servers of the remote machine are displayed in the **OPC Server** drop-down list. The first OPC server is selected by default.

10. After updating the required fields, click **OK**.

If you do not want to proceed with the migration process, then click the close button at the top-right corner of the <Project>:<Project Name> dialog box.

---

**Note:** If migration does not happen or some of the items are not migrated properly, then the log file will be updated with the items that are not migrated.

---

## Field Description and Layout of <Project>:<Project Name> Dialog Box

Different fields and their descriptions are given below:

- **OPCServer Type:** FactoryTalk and RSLinx Classic are the supported OPCSERVER/GATEWAY. Based on the selected gateway/OPC server, tags are migrated.
- **OPC Topic:** This is the alias name configured for the device in the Allen Bradley OPC server.
- **OPC SERVER:** This is the Prog IDs of the OPC servers installed on the given machine.

The layout of <Project>:<Project Name> dialog box is as below:

- It lists the devices in the project that you can migrate from ABCLX to OPCLX.
- Different clusters are displayed in separate tabs. Related primary and standby I/O devices are listed under each cluster.
- Each of the primary and standby device names is displayed as <Devicename>\_<Mode>\_<Priority>.
- You can enter OPC topic only for the primary devices. You cannot modify the OPC topic for the standby devices. By default, the I/O device name is displayed as **Topic Name** which can be changed by editing **OPC Topic** field.
- For the standby devices, OPC topic is set to the OPC topic of its corresponding primary device. If you modify the OPC topic in the primary device, the OPC topic for its standby devices is automatically updated.

---

**Note:** Once migration is completed it is recommended to pack the main project and all included projects individually. Packing the database removes deleted records, and re-indexes the database. To pack the project databases, from the **Projects** activity in Plant SCADA Studio, click **Pack**.

---

## See Also

[Migration of I/O Devices](#)

## Migration of I/O Devices

After migrating the project successfully, the following changes can be observed in Boards, Ports, and I/O Devices grid for the migrated SCADA project.

**Note:** You can modify or delete all the configurations manually.

### Boards

The existing ABCLX boards are not modified. For each server, an OPCLX board is created. The board name for the first server is OPCLX\_BOARD1 and that for the second server it is OPCLX\_BOARD2 and so on.

Below table describes different field names for a Board and corresponding changes observed.

Field	Value
Board Type	A new board with type set to OPCLX is created. The earlier boards also exist and you can delete if required.
Address	The value of this field is set to 0 by default. You can manually enter it.
I/O Port	The value of this field is deleted.
Interrupt	The value of this field is deleted.
Special Options	The value of this field is deleted.

### Ports

New ports are created. Number of ports is equal to the number of devices selected in the migration form. By default the port name for the devices is OPCLX1\_PORT1 and that for the second device is OPCLX1\_Port2 and so on. If OPCLX1\_Port1 is created, then Port1 is created for OPCLX1 board. Based on the number of I/O devices selected for that board, equal number of ports are created. Similarly if multiple I/O devices are selected across multiple boards, then multiple ports are created for each board.

Below table describes different field names of the Ports and corresponding changes observed.

Field	Value
Board Name	The board name is the same as the new board of type OPCLX that is created.
Server Name	The server name of the selected device.
Baud Rate	The value of this field is deleted.
Data Bits	The value of this field is deleted.
Stop Bits	The value of this field is deleted.

Field	Value
Parity	The value of this field is deleted.
Special Options	The value of this field is deleted.

## Devices

The table below describes different field names for I/O devices and corresponding changes observed.

Field	Value
Protocol	The protocol name gets changed from ABCLX to OPCLX.
Address	Address field in the I/O devices form gets updated to the corresponding Prog ID of Allen Bradley OPC server that you have entered in the migration tool form for that I/O device.
Port	The port name is the same as the new port that is created.

## See Also

[Tag Address Formats](#)

### Tag Address Formats

After migrating the project successfully, the following changes can be observed in tag addresses.

## Controller Tags

- **Controller Tags**

For the Tag address [OPCTOPIC] will be added to the tag address format.

Example:

ABCLX Tag address => TagAddress\_00

OPCLX Tag address => [OPCTOPIC]TagAddress\_00

- **Controller Array Tags**

If the Tag address contains "{}" braces it will be changed to the [] braces.

Example

ABCLX Tag address => TagAddress\_00{100}

OPCLX Tag address => [OPCTOPIC]TagAddress\_00[100]

- **Controller Array Tags referring to a particular bit in the array**

If the Tag address contains "/" character it will be changed to "." character.

Example

ABCLX Tag address => TagAddress\_00{100}/5

OPCLX Tag address => [OPCTOPIC]TagAddress\_00[100].5

- **Controller sub array tags**

If the Tag address contains "!A[<count>]" then it will be changed to ",L<Count>!A[<count>].

Example

ABCLX Tag address => TagAddress\_00!A[5]

OPCLX Tag address => [OPCTOPIC] TagAddress\_00,L5!A[5]

- **Controller sub array tags**

If the Tag address contains "/<startIndex>!A[<count>]" then it will be changed to "[startIndex],L<Count>!A[<count>].

Example

ABCLX Tag address => TagAddress\_00/10!A[5]

OPCLX Tag address => [OPCTOPIC] TagAddress\_00[10],L5!A[5]

## Program Tags

For FactoryTalk Tag address formats are:

- **Program Tags**

::[OPCTOPIC]PROGRAM:MainProgram.TagName

Example:

ABCLX Tag address => PROGRAM:MainProgram.TagAddress\_00

OPCLX Tag address => ::[OPCTOPIC]PROGRAM:MainProgram.TagAddress\_00

- **Program Array Tags**

::[OPCTOPIC]PROGRAM:MainProgram.TagName

For RSLinx Classic Tagaddress formats are:

- **Program Tags**

[OPCTOPIC]PROGRAM:MainProgram.TagAddress\_00

Example:

ABCLX Tag address => PROGRAM:MainProgram.TagAddress\_00

OPCLX Tag address => [OPCTOPIC]PROGRAM:MainProgram.TagAddress\_00

- **Program Array Tags**

[OPCTOPIC]PROGRAM:MainProgramTagName[ArraySize]

Example:

ABCLX Tag address => PROGRAM:MainProgramTagName{10}

OPCLX Tag address => [OPCTOPIC]PROGRAM:MainProgramTagName[10]

## Troubleshooting

This chapter provides information about the Plant SCADA tools provided to help resolve problems with communication and configuration.

- [Detected Driver Errors](#)
- [Driver Statistics](#)
- [Logging](#)
- [Maintaining the Project Database](#)

### **Detected Driver Errors**

Plant SCADA OPCLX detected driver errors can occur when an OPC device does not respond, is disconnected or offline, or returns an error itself.

This section includes:

- [Detected Common Protocol Errors](#)
- [Detected OPCLX Protocol Errors](#)

The OPCLX driver can log combinations of trace levels across different categories. For more information, see [Logging](#).

### **Detected Common Protocol Errors**

Plant SCADA has two kinds of protocol driver errors - generic and specific. Generic errors are hardware errors 0-31, and are common to many protocols. Sometimes only the generic error is available, though often both the generic error and a specific error are given.

Protocol drivers also have their own specific errors, which can be unique and therefore cannot be recognized by the hardware alarm system. The drivers convert specific errors into generic errors that can be identified by the I/O server. For example, when a driver detects an error, there is often both a protocol-specific error and a corresponding generic error.

When a hardware error occurs, Plant SCADA generates an alarm, and displays the alarm on the hardware alarm page (in the alarm description of the hardware alarm). To see the detected error number, you need to have Alarm Category 255 defined with a display format that includes {ErrDesc,15} {ErrPage,10}.

If you need more information to resolve a detected error, refer to the documentation that accompanied the I/O device (or network). After viewing the documentation, if you still cannot solve your problem, contact technical support for this product.

See also [Detected OPCLX Protocol Errors](#).

### **Detected OPCLX Protocol Errors**

The following detected errors, listed in (hexadecimal) order, are specific to the Plant SCADA OPCLX protocol:

Error value (Hex)	Error definition	Error description
0XC0040001	GENERIC_SOFTWARE_ERROR	Invalid handle was passed.
0XC0040002	GENERIC_SOFTWARE_ERROR	A duplicate parameter was passed where one is not allowed.
0XC0040004	GENERIC_INVALID_DATA_FORMAT	Server cannot convert between the passed or requested data type and the canonical type.
0xC0040006	GENERIC_ACCESS_VIOLATION	Item's access rights do not allow the operation.
0xC0040007	GENERIC_SOFTWARE_ERROR	Item definition does not exist within the server's address space.
0xC0040008	GENERIC_SOFTWARE_ERROR	Item definition does not conform to the server's syntax.
0xC004000B	GENERIC_INVALID_DATA_FORMAT	Value passed to WRITE was out of range.
0x80020008	GENERIC_SOFTWARE_ERROR	Bad variable type.
0x8002000A	GENERIC_SOFTWARE_ERROR	Out of present range.
0x80020005	GENERIC_SOFTWARE_ERROR	Type mismatch.
0x100	GENERIC_SOFTWARE_ERROR	Could not access variable dbf.
0x101	DRIVER_BAD_DATA	Read of data value bad.
0x102	GENERIC_GENERAL_ERROR	Write of one or more items not completed.
0x103	DRIVER_UNIT_OFFLINE	Could not resolve the server CLASSID.
0x104	GENERIC_SOFTWARE_ERROR	Could not add one or more items to the server
0x80010006	GENERIC_CHANNEL_OFFLINE	Connection terminated
0x80010007	GENERIC_CHANNEL_OFFLINE	Server not available.
0x8001000F	GENERIC_INVALID_RESPONSE	Received data is invalid.
0x80010012	GENERIC_SOFTWARE_ERROR	Call did not execute.
0x80010100	GENERIC_SOFTWARE_ERROR	System call aborted.

Error value (Hex)	Error definition	Error description
0x80010101	GENERIC_SOFTWARE_ERROR	Could not allocate some required resource.
0x80010103	GENERIC_BAD_PARAMETER	Requested interface not registered on server object.
0x80010104	GENERIC_SOFTWARE_ERROR	Could not call server.
0x80010105	GENERIC_CHANNEL_OFFLINE	Server threw an exception.
0x80010108	GENERIC_CHANNEL_OFFLINE	Server has disconnected from its clients.
0x8001011B	GENERIC_CHANNEL_OFFLINE	Access denied.
0x8001011C	GENERIC_CHANNEL_OFFLINE	Remote calls not allowed for this process.

See also [Detected Common Protocol Errors](#).

## Driver Statistics

You can use the following driver statistics to help you debug the OPCX driver operation:

Statistics	Description
Cached Reads	Number of reads that have been serviced directly from the internal cache of current values.
Cache Misses	Number of reads that were not in the internal cache of current values which resulted in a direct request from the server.
Active Items	Number of items with valid server subscriptions which have been activated to receive data change notification. This may be a subset of the total number of items that are currently subscribed.
Total Items	Total number of items with valid server subscriptions. These items may or may not be currently activated to receive data change notifications.
Bad Items	Number of items that were not successfully subscribed to. These items may not exist on the server.
Start ms	Internal driver statistic used for driver debugging by Citect support engineers.

Statistics	Description
Last ms	Internal driver statistic used for driver debugging by Citect support engineers.
Difference	Internal driver statistic used for driver debugging by Citect support engineers.
ReadItemsReQueued	Internal driver statistic used for driver debugging by Citect support engineers.
DCBsInHandlingQ	Internal driver statistic used for driver debugging by Citect support engineers.
DataNotYetValid	Internal driver statistic used for driver debugging by Citect support engineers.

## Logging

The OPCLX driver can log combinations of trace levels across different categories. This is achieved by setting the following INI parameters.

### [OPCLX]DebugLevel

This parameter allows you to define the trace level. The options include:

WARN	Output warning messages.
ERROR	Output error messages.
TRACE	Output trace messages.
DEBUG	Output debug messages.
ALL	All of the above.

### [OPCLX]DebugCategory

This parameter allows you to enable logging for a particular category of trace. The options include:

TAG	Tag configuration trace
PROT	Operations related to the OPC protocol interface
CACHE	Operations related to the driver cache
DCB	Front end driver trace
MISC	Miscellaneous operations

THRD	Thread trace
BEND	Operations related to the back-end thread of the driver
FEND	Operations related to the front-end thread of the driver
ALL	All of the above

In both cases, you can use ALL or any combination of the available options separated by a pipe character ( | ).

### Example

```
[OPCLX]
DebugLevel=WARN|ERROR|TRACE
DebugCategory=ALL
```

In most cases, you can set the DebugLevel and DebugCategory parameters to ALL. If this results in a large, cumbersome log file that causes older data to be overwritten, you can consider refining the debug options.

---

**Note:** These parameters do not have a value specified by default. To enable logging for this driver, you need to manually configure these parameters in your INI file.

The events generated by the OPCLX driver are logged in the following Plant SCADA syslog file:

syslog.IOServer.<Cluster name>.<IO Server name>.dat

This file is located in the directory specified by the Citect.ini parameter **[CtEdit]Logs**. By default, this location will be:

%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Logs

If you are running the driver on a version of Plant SCADA prior to v7.20, a file called OPCLX.log will be delivered to the "Logs\Drivers" directory instead.

The size of the syslog file and its rollover process is configured via the **[Debug]SysLogSize** and **[Debug]SysLogArchive** INI parameters. For more information, see the topic *Debug Parameters* in the Plant SCADA documentation.

---

**Note:** If the number of messages being processed for a driver exceeds the value specified by the Plant SCADA parameter **[Kernel]ErrorBuffers**, the pending messages will not be logged. If this occurs, you can increase the value of this parameter so that all the messages generated by the driver are reported. The number of messages that are not logged can be monitored via the Kernel's General window under the item "Lost Errors". If you adjust **[Kernel]ErrorBuffers** and Lost Errors are still occurring, you may need to adjust the level of system logging.

## See Also

[Logging Parameters](#)

## Maintaining the Project Database

Over time with editing, the project database underlying a project can become fragmented and performance can be reduced. During project development, you should periodically clear out all duplicated records and any orphaned ones.

Improperly maintained databases can result in communication errors detected at runtime. Many

communications errors detected come from having duplicated or orphaned records in the communications database.

Sometimes orphaned records from a previous I/O server are left behind in the database files. As the forms are indexed on the I/O server name, you can use the scroll bar to navigate quickly to the end of the current range of records for a particular I/O server. You can examine the last few and next few records to validate that they should be there. If you find extra (unwanted) records, delete them.

You should also check that there are no duplicated records. Use the record search facility to search for duplicate entries of devices that are causing unexplained communication errors detected. If you find extra (unwanted) records, delete them.

Once you have deleted orphaned and duplicated records, pack the project. Packing the database removes deleted records, and re-indexes the database. To pack the project databases, from the **Projects** activity in Plant SCADA Studio, click **Pack**.

## OPCUA Driver

The OPCUA driver enables Plant SCADA to communicate with a system that implements the OPC UA server standard for data access. The server endpoint needs to support OPC.TCP binary protocol.

Citect SCADA 2016 (or later) is required to use the OPCUA Driver.

The driver uses the Unified Automation OPC UA Client SDK v1.7.7.

---

**Note:** If you are installing the OPCUA driver on a legacy version of Plant SCADA (2020 R2 or earlier), you will need to manually install PCS Framework 7.0.0. See [Manually Install PCS Framework](#).

## See Also

[Setting up Device Communications](#)

[Configuring Variable Tags](#)

[Advanced Configuration and Maintenance](#)

[Troubleshooting](#)

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.
<b> DANGER</b>	
<b>DANGER</b> indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.	
<b> WARNING</b>	
<b>WARNING</b> indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.	
<b> CAUTION</b>	
<b>CAUTION</b> indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.	
<b>NOTICE</b>	
<b>NOTICE</b> used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.	

## Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

## Before You Begin

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ WARNING****LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

## Third Party Software

You acknowledge that the Software may contain copyrighted software of third parties which are obtained under a license from such parties. All third party licensors retain all right, title and interest in and to such Third Party Software and all copies thereof, including all copyright and other intellectual property rights. Your use of any

Third Party Software shall be subject to, and You shall comply with, the terms and conditions of this Agreement, and the applicable restrictions and other terms and conditions set forth in any Third Party Software documentation or printed materials, including without limitation an end user license agreement.

Below is a list of third party libraries requiring a special note:

1. MIT License (<https://opensource.org/licenses/MIT>):

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Libraries:

- GalaSoft MVVM Light Toolkit
- libexpatw
- Newtonsoft.Json
- BouncyCastle
- Autofac
- Irony

2. Apache License (<https://opensource.org/licenses/Apache-2.0>).

Libraries:

- Remotion.Linq.dll
- Roboto font
- Owin
- IdentityServer
- Octodiff

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

3. The OPC UA Driver is based on C++ Based OPC UA Client SDK from Unified Automation.

© Unified Automation GmbH - All rights reserved.

The UA Driver contains the following additional third party software:

Third Party Licenses and Open Source License Acknowledgment

- OPC UA ANSI C Stack (Unified Automation Edition)
- OpenSSL
- LibXML2
- WCELIBCEX (used in AnsiC-Stack Platform Layer for WindowsCEonly)

- WCECOMPAT (used in OpenSSL for WindowsCEonly, requires dynamic linking)
- Bouncy Castle Crypto API
- C-ARES async address resolver
- ICU4C Unicode Library (minimum Windows 7)
- JSMN a lightweight JSON parser in C
- MQTT-C a light weight MQTT client library in C
- Eclipse Mosquitto a MQTT C-Client Library

Overview on used third party components:

	AnsiC Stack	OpenSSL	LibXML 2	WCE LIBCEX	WCE COMPAT	Bouncy Castle	C-ARES	ICU4C	JSMN	MQTT-C	Mosquitto
<b>HP C SDK</b>	-	o	-	-	-	-	o	-	o	o	-
<b>ANSI C SDK</b>	m	o (1)	-	o (3)	o (4)	-	-	-	o	o	-
<b>C++ SDK</b>	m	o (1)	o	o (3)	o (4)	-	-	-	o	-	o
<b>.NET SDK</b>	-	o (2)	-	-	-	o (5)	-	-	-	-	-
<b>UaModeler</b>	-	-	-	-	-	-	-	-	-	-	-
<b>UaGateway</b>	m	m	m	-	-	-	-	-	m	-	m
<b>UaExpert</b>	m	m	m	-	-	-	-	-	m	-	m
<b>UaGDS</b>	-	m		-	-	-	m	m	-	-	-

m=mandatory;

o=optional;

-=not used;

(1) recommended on Windows;

(2) used by certificategenerator.exe;

(3) used to compile SDK for WindowsCE only;

(4) used to compile OpenSSL for WindowsCE only, requires dynamic linking of OpenSSL;

## 1. OPC UA ANSI C Stack

This component includes software that was initially developed by the OPC Foundation. Unified Automation acquired a Commercial Source Code License (Buyout Agreement) with the OPC Foundation. Unified Automation has made custom modifications to the original code and its APIs which are kept proprietary. The derivative works as delivered with the SDKs, toolkits and runtime products of Unified Automation now completely falls under the respective license grant as in Section 1 and 2 of this SLA.

### 1.1 Copyright

This software is based in part on the OPC Foundation. Initial version was founded and copyrighted by OPC Foundation, Inc. Copyright (C) 2008, 2014 OPC Foundation, Inc., All Rights Reserved.

### 1.2 License Issues

The OPC UA ANSI C Stack including any custom modifications and enhancements as shipped/delivered by Unified Automation stays under respective license grant as of Section 1 and 2 of this Software License Agreement (SLA) and its source code is confidential. The recipient of this component is NOT bound to any other license the original code may be distributed under i.e. not bound to RCL/RBCL nor any other license.

## 2. OpenSSL crypto and pki

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>).

### 2.1 Copyright

Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com) All rights reserved.

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)

This Windows version of this product includes software written by Tim Hudson (tjh@cryptsoft.com)

### 2.2 License Issues

The OpenSSL toolkit stays under a dual license, i.e. both the conditions of the OpenSSL License and the original SSLeaylicense apply to the toolkit. See below for the actual license texts. Actually both licenses are BSD-style Open Source licenses. In case of any license issues related to OpenSSL please contact openssl-core@openssl.org.

#### Original OpenSSL License:

Copyright (C) 1998-2007 The OpenSSL Project. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- a. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.
- b. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- c. All advertising materials mentioning features or use of this software must display the following acknowledgement:  
"This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)".
- d. The word 'cryptographic' can be left out if the routines from the library being used are not cryptography-related.
- e. If you include any Windows specific code (or a derivative thereof) from the apps directory (application

code) you must include an acknowledgement:

"This product includes software written by Tim Hudson (tjh@cryptsoft.com)".

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

#### **Original SSLeayLicense:**

Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com). All rights reserved.

This package is an SSL implementation written by Eric Young (eay@cryptsoft.com).

The implementation was written so as to conform with Netscape's SSL. This library is free for commercial and non-commercial use as long as the following conditions are adhered to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, SHA, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used. This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package.

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The license and distribution terms for any publicly available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution license [including the GNU Public License].

## **3. LibXML2 -- XML Parser**

This product includes code that was developed for the XML toolkit from the GNOME project (<http://xmlsoft.org/>).

### **3.1 Copyright**

Copyright (C) 1998-2003 Daniel Veillard. All Rights Reserved.

### **3.2 License Issues**

The libxml2 library is released under the MIT License and includes following copyright notice:

Except where otherwise noted in the source code (e.g. the files hash.c, list.c and the trio files, which are covered by a similar license but with different Copyright notices) all the files are:

Copyright (C) 1998-2003 Daniel Veillard. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE DANIEL VEILLARD BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of Daniel Veillard shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from him.

## 4. WCELIBCEX (Windows CE only)

The Windows platform layer of the AnsiC Stack can be configured to run on Windows CE, but than requires WCELIBCEX: The WCELIBCEX is a package of C library extensions for Windows CE operating system. It is a supplement to standard C library available on Windows CE system. This extensions library is needed to fully support OPC UA SDK functionality.

### 4.1 Copyright

Initial version of WCELIBCEX was founded and copyrighted by Taxus SI Ltd., (<http://www.taxussi.com.pl>)

Copyright (c) 2006 Taxus SI Ltd.

Created by Mateusz Loskot(mateusz@loskot.net)

### 4.2 License Issues

The source code of the WCELIBCEX library is licensed under MIT License:

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

The complete license agreement can be found here:

( <http://opensource.org/licenses/mit-license.php> )

## 5. WCECOMPAT (Windows CE only)

The OpenSSL Library can be compiled to run on Windows CE, but therefore requires wcecompat: Windows CE C Runtime Library, a "compatibility" library to fully support OpenSSL functionality. The files of wcecompat are released under the GNU LGPL (Lesser General Public License). By this the OpenSSL Library becomes LGPL and can only be used by linking this library dynamically (do not link static).

### 5.1 Copyright

Copyright (C) 2001-2002 EssemerPty Ltd. All rights reserved.

( <http://www.essemer.com.au/> )

### 5.2 License Issues

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

## 6. Bouncy Castle Crypto API (.NET SDK only)

The Bouncy Castle Crypto APIs for Java consist mainly of the following: A lightweight cryptography API for Java and C# and a provider for the Java Cryptography Extension and the Java Cryptography Architecture. The Bouncy Castle Crypto API license is an adaptation of the MIT X11 License and should be read as such.

### 6.1 Copyright

Copyright (c) 2000 - 2009 The Legion Of The Bouncy Castle (<http://www.bouncycastle.org>)

### 6.2 License Issues

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

The complete license agreement can be found here:

<http://www.bouncycastle.org/licence.html>

## 7. C-ARES - Async Address Resolution

c-ares is a C library for asynchronous DNS requests (including name resolves), C89 compatibility, MIT licensed, builds for and runs on POSIX, Windows, Netware, Android and many more operating systems.

### 7.1 Copyright

Copyright 1998 by the Massachusetts Institute of Technology.

### 7.2 License Issues

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. M.I.T. makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

## 8. ICU4C – Unicode Library

The C and C++ languages and many operating system environments do not provide full support for Unicode and standards-compliant text handling services. Even though some platforms do provide good Unicode text handling services, portable application code can not make use of them. The ICU4C libraries fills in this gap. ICU4C provides an open, flexible, portable foundation for applications to use for their software globalization requirements. ICU4C closely tracks industry standards, including Unicode and CLDR (Common Locale Data Repository). The ICU license is intended to allow ICU to be included both in free software projects and in proprietary or commercial products. Since ICU 58, ICU is covered by the Unicode license which is very similar to the previous ICU license.

### 8.1 Copyright

Copyright © 1991-2022 Unicode, Inc. All rights reserved.

### 8.2 License Issues

#### UNICODE, INC. LICENSE AGREEMENT - DATA FILES AND SOFTWARE

See Terms of Use (<https://www.unicode.org/copyright.html>) for definitions of Unicode Inc.'s Data Files and Software.

**NOTICE TO USER:** Carefully read the following legal agreement. BY DOWNLOADING, INSTALLING, COPYING OR OTHERWISE USING UNICODE INC.'S DATA FILES ("DATA FILES"), AND/OR SOFTWARE ("SOFTWARE"), YOU UNEQUIVOCALLY ACCEPT, AND AGREE TO BE BOUND BY, ALL OF THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU DO NOT AGREE, DO NOT DOWNLOAD, INSTALL, COPY, DISTRIBUTE OR USE THE DATA FILES OR SOFTWARE.

#### COPYRIGHT AND PERMISSION NOTICE

Copyright © 1991-2022 Unicode, Inc. All rights reserved.

Distributed under the Terms of Use in (<https://www.unicode.org/copyright.html>)

Permission is hereby granted, free of charge, to any person obtaining a copy of the Unicode data files and any associated documentation (the "Data Files") or Unicode software and any associated documentation (the "Software") to deal in the Data Files or Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Data Files or Software, and to permit persons to whom the Data Files or Software are furnished to do so, provided that either

- a. this copyright and permission notice appear with all copies of the Data Files or Software, or
- b. this copyright and permission notice appear in associated Documentation.

THE DATA FILES AND SOFTWARE ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE DATA FILES OR SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in these Data Files or Software without prior written authorization of the copyright holder.

## 9 JSMN – JSON parser in C

jsmn (pronounced like 'jasmine') is a minimalistic JSON parser in C. It can be easily integrated into resource-limited or embedded projects. Most JSON parsers offer you a bunch of functions to load JSON data, parse it and extract any value by its name. jsmn proves that checking the correctness of every JSON packet or allocating temporary objects to store parsed JSON fields often is an overkill.

### 9.1 Copyright

Copyright (c) 2010 Serge A. Zaitsev

### 9.2 License Issues

This software is distributed under MIT license, so feel free to integrate it in your commercial products.

Copyright (c) 2010 Serge A. Zaitsev

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## 10. MQTT-C – JSON parser in C

MQTT-C is an MQTT v3.1.1 client written in C. MQTT is a lightweight publisher-subscriber-based messaging protocol that is commonly used in IoT and networking applications where high-latency and low data-rate links are expected. The purpose of MQTT-C is to provide a portable MQTT client, written in C, for embedded systems and PC's alike.

### 10.1 Copyright

Copyright (c) 2018 Liam Bindle

### 10.2 License Issues

LiamBindle/MQTT-C is licensed under the MIT License

Copyright (c) 2018 Liam Bindle

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## 11. MQTT – Eclipse Mosquitto

Eclipse Mosquitto is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 5.0, 3.1.1 and 3.1. Mosquitto is lightweight and is suitable for use on all devices from low power single board computers to full servers. The MQTT protocol provides a lightweight method of carrying out messaging using a publish/subscribe model. This makes it suitable for Internet of Things messaging such as with low power sensors or mobile devices such as phones, embedded computers or microcontrollers. The Mosquitto project also provides a C library for implementing MQTT clients, and the very popular mosquitto\_pub and mosquitto\_sub command line MQTT clients.

Mosquitto is part of the Eclipse Foundation, is an iot.eclipse.org project and is sponsored by cedalo.com.

### 11.1 Copyright

Mosquitto was written by Roger Light (roger@atchoo.org)

### 11.2 License Issues

This project is dual licensed under the Eclipse Public License 2.0 and the Eclipse Distribution License 1.0 as described in the epl-v20 and edl-v10 files (<https://www.eclipse.org/org/documents/edl-v10.php>)

Eclipse Public License - v 2.0 see here for details: (<https://www.eclipse.org/legal/epl-2.0/>)

Eclipse Distribution License - v 1.0

Copyright (c) 2007, Eclipse Foundation, Inc. and its licensors. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- a. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- b. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- c. Neither the name of the Eclipse Foundation, Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,

EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## Appendix B

### 1. Parts of the Licensed Software that are permitted for distribution ("Redistributables"):

- The licensed Software's main and plug-in libraries of the SDK in object code form
- The licensed Software's communication stack in object code form ("C-Stack", "JAVA-Stack", ".NET Stack")
- The licensed Software's administration and configuration tool ("UaAdminDialog, UaConfigTool")
- The licensed Software certificate generation tool ("Opc.Ua.CertificateGenerator.exe")

### 2. Parts of the Licensed Software that are NOT permitted for distribution include, but are not limited to:

- The licensed Software's source code and/or header files of any of the SDK modules
- The licensed Software's binary files together with header files of any of the SDK modules
- The licensed Software's source code and header files of the communication stacks ("C-Stack", "JAVA-Stack", ".NET Stack")
- The licensed Software's source code and header files of the UA Stack's platform layer
- The licensed Software's tool for modelling Addressspace and code generation ("UaModeler")
- The licensed Software's documentation

## Manually Install PCS Framework

You will need to manually install PCS Framework 7.0.0 if you use one of the following versions of Citect SCADA or Plant SCADA:

- Citect SCADA 2016
- Citect SCADA 2018
- Citect SCADA 2018 R2 Update 22 (November 2021)
- Plant SCADA 2020 R2 (RTM).

You need to download, unzip and install "PCS Framework 7.0.0 (x64).exe".

The PCS 7.0.0 installer can be downloaded from **AVEVA Knowledge and Support Center** (located at <https://softwaresupport.aveva.com>). Go to the **Product Hub** and locate the download page for this version of the OPCUA driver.

You will also need to download and install **Operations Control Logger** from this page.

## PCS Framework Prerequisites

You also need to download and install these prerequisites from the Microsoft website. They are required for PCS 7.0.0.

- .NET Framework 4.8 Full

- Microsoft .NET 6 ASP.NET Core Runtime (x64)
- Microsoft .NET 6 Desktop Runtime (x64)
- Visual C++ 2015 - 2022 Redistributable (x86 and x64). Use Universal C Runtime.

## Setting up Device Communications

The information included in the following topics describes how to establish communication with Plant SCADA and OPC UA server using the OPCUA driver.

- [Communication Settings](#)

### Communication Settings

To establish communication with a device, the associated Board, Port and I/O Device need to be correctly configured in Plant SCADA Studio's Topology activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure them, use the settings outlined below.

**Note:** The OPCUA driver only one requires one Board and one Port to be configured.

## Boards

The Boards view in Plant SCADA Studio lists the boards used in the Plant SCADA project. Each board record defines a separate board within the project.

Field	Value
Board Name	This field is user defined (e.g. 'Board1').
Board Type	Enter OPCUA.
Address	Enter zero (0).
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.

## Ports

The Ports view in Plant SCADA Studio lists the ports used in the Plant SCADA project. Each port record defines a separate port within the project.

Field	Value
Port Name	This field is user defined.
Port Number	Leave this field blank.
Board Name	Refers to the board previously defined in Boards (e.g. 'Board1').
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank.
Special Options	Leave this field blank.
Comment	Any useful comment. This field is user defined and not required by the driver.

## I/O Devices

The I/O Devices view in Plant SCADA Studio lists the I/O Devices used in the Plant SCADA project. Each device record defines a device within the project.

Field	Value
Name	This field is user defined.
Number	I/O device number which needs to be unique for each logical device, but identical for redundant devices.
Address	<p>The OPC UA server endpoint URL. The URL can include the host name, IPv4, or IPv6 address. For example:</p> <ul style="list-style-type: none"> <li>• <code>opc.tcp://MyComputer:48010</code> For this URL, the driver will use the dual TCP IPv4/IPv6 protocol stack if it is available, otherwise the IPv4 protocol stack.</li> <li>• <code>opc.tcp://192.168.0.15:48010</code> For this URL, the driver will use IPv4 protocol stack.</li> <li>• <code>opc.tcp://[fe80::20ec:3acb:55d9:a3da]:48010</code> For this URL, the driver will use IPv6 protocol stack.</li> </ul>

Field	Value
Protocol	Enter OPCUA.
Port Name	Refers to the port previously defined in Ports.
Comment	Any useful comment. This field is user defined and not required by the driver.

## Configuring Variable Tags

Variable tags are addressed using the OPC UA NodeId configured in the OPC UA server.

NodeId consists of a namespace index and a unique identifier which can be a numeric value(i=), a string(s=), a globally unique identifier (g=), or an opaque value(b=).

Examples of NodeIds: "ns=2; i=12345" or "ns=3; s=SoftwareRevision"

---

**Note:** .The <Node ID> in a tag address cannot contain the following characters: "\", "!A" and ";!E=".

"\\" and "!A" characters are used for array tag addresses, and ";!E=" characters are used for extension object tag addresses.

The following table demonstrates the variable tag address formats you can use to read and write values.

Address format	OPC UA data type	Plant SCADA data type	Example
<Node ID>	BOOLEAN	DIGITAL	FALSE or TRUE
<Node ID>	SBYTE	INT, LONG	-128 to 127
<Node ID>	BYTE	INT, UINT, LONG, ULONG	0 to 255
<Node ID>	INT16	INT, LONG	-32768 to 32767
<Node ID>	UINT16	UINT, LONG, ULONG	0 to 65535
<Node ID>	INT32	LONG	-2147483648 to +2147483647
<Node ID>	UINT32	ULONG	0 to 4294967295
<Node ID>	INT64	<NOT SUPPORTED>	
<Node ID>	UINT64	<NOT SUPPORTED>	
<Node ID>	FLOAT	REAL	-3.4E38 to 3.4E38
<Node ID>	DOUBLE	REAL	-3.4E38 to 3.4E38
<Node ID>	STRING	STRING	ASCII string (0 to 254 characters)
<Node ID>	DATETIME	STRING	1990-01-01-00:00:00.000

Address format	OPC UA data type	Plant SCADA data type	Example
			to 2089-12-31-23:59:59.999
<Node ID>	GUID	<NOT SUPPORTED>	
<Node ID>	XMLElement	<NOT SUPPORTED>	
<Node ID>	NodeId	<NOT SUPPORTED>	
<Node ID>	Expanded NodeId	<NOT SUPPORTED>	
<Node ID>	Qualified Name	STRING	Update the Qualified Name as string in the tag value
<Node ID>	Localized Text	STRING	Update the text in English Locale in the tag value
<Node ID>	Status Code	LONG	Update the Integer code in the tag value
<Node ID>	ByteString	STRING	Array of Bytes
<Node ID>	Enumeration	LONG	
<Node ID>	Variant	<NOT SUPPORTED>	
<Node ID>	Extension Object	Plant SCADA data type corresponding to the Extension Object element data type.	See the topic <a href="#">Extension Object Support</a> .
<Node ID>	Number  This is a Variant data type which can be FLOAT or DOUBLE.	REAL  Note: Only FLOAT OPC UA data type is supported, and the data type must not change on the server.	
<Node ID>	Integer  This is a Variant data type which can be SBYTE, INT16, INT32 or INT64.	SBYTE: INT, LONG INT16: INT, LONG INT32: LONG  Note: Only SBYTE, INT16 and INT32 OPC UA data types are supported, and the data type must not change on the server.	
<Node ID>	UInteger	BYTE: INT, UINT, LONG,	

Address format	OPC UA data type	Plant SCADA data type	Example
	This is a Variant data type which can be BYTE, UINT16, UINT32 or UINT64.	ULONG UINT16: UINT, LONG, ULONG UINT32: ULONG  Note: Only BYTE, UINT16 and UINT32 OPC UA data types are supported, and the data type must not change on the server.	
<Node ID>	DiagnosticInfo	<NOT SUPPORTED>	

The driver supports conversions between OPC UA data types and Plant SCADA data types. When a configured tag data type cannot be converted into OPC UA data type, the driver marks that tag with quality Bad, Configuration Error.

The following table shows supported conversions between Plant SCADA and OPC UA data types.

Plant SCADA data type	OPC UA data type
DIGITAL	BOOLEAN
INT, UINT	SBYTE, BYTE, INT16, UINT16
LONG, UINT	SBYTE, BYTE, INT16, UINT16, INT32,UINT32
REAL	REAL
STRING	All OPCUA data types supported by the driver, except for BYTE, UINT16, UINT32, UINT64

## See Also

[Array Support](#)

## Array Support

The driver maps the OPCUA arrays to Plant SCADA arrays. The following table demonstrates the variable tag address formats you can use to read array values for supported OPCUA array data types.

Address	OPC UA array data type	Plant SCADA data type
<Node ID>\<OFFSET>!A[SIZE]	BOOLEAN	DIGITAL
<Node ID>\<OFFSET>!A[SIZE]	SBYTE	INT, LONG
<Node ID>\<OFFSET>!A[SIZE]	BYTE	INT, UINT, LONG, ULONG

Address	OPC UA array data type	Plant SCADA data type
<Node ID>\<OFFSET>!A[SIZE]	INT16	INT, LONG
<Node ID>\<OFFSET>!A[SIZE]	UINT16	UINT, LONG, ULONG
<Node ID>\<OFFSET>!A[SIZE]	INT32	LONG
<Node ID>\<OFFSET>!A[SIZE]	UINT32	ULONG
<Node ID>\<OFFSET>!A[SIZE]	FLOAT	REAL

The array tag address should include the array size and an optional array offset.

The maximum size of an array tag is limited to 256 bytes due to the underlying infrastructure of Plant SCADA. For example, if the tag data type is integer (2 bytes), Plant SCADA only allows the array tag to be configured with 128 elements. However, special array offset syntax can be used to address a sub-array with the particular offset and range within a large array.

## Extension Object Support

The OPCUA driver supports OPC UA extension objects which contain a structure (or an array of structures) encoded in a binary format. Each structure member is represented as a separate tag in Plant SCADA.

The Plant SCADA tags are created during tag import. The OPC UA tag import component allows you to browse the supported extension objects and displays the Plant SCADA tags corresponding to the extension object elements.

It is only possible to browse and create tags for the simple OPC UA data types supported by the driver. The simple data type arrays inside the extension objects (BOOLEAN, INT32, and so on) are currently not supported.

When writing to a tag which corresponds to an extension object member, the driver has to update the entire extension object value as the OPC UA protocol does not allow to update the specific extension object element.

Before performing the write request, the driver reads the whole extension object value, updates the specified element and writes the extension object value to the server.

### **WARNING**

#### **LOSS OF CONTROL**

When configuring your system, you need to make sure the extension object value cannot be updated when it is being updated by the driver. Although the driver uses the most up-to-date extension object value when performing a write, if the extension object value is changed by the server at that time (for example by a PLC program), there is a chance the extension object value will be incorrect after the write.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Advanced Configuration and Maintenance

This section provides instructions for advanced configuration and maintenance tasks. It includes the following topics:

- Configuring Redundancy
- Subscription Management
- Security Configuration
- Customizing a Project using Citect.ini Parameters
- Tag Import.

## Configuring Redundancy

The Plant SCADA driver supports multiple I/O server / device redundancy.

The driver represents each OPC UA server as a separate I/O device. In a normal operating state, the driver will communicate to the server via the primary I/O device. If the connection to the server via the primary I/O device is lost, the server will be communicated via a standby I/O device which is connected to the server and has the next highest priority if multiple redundant I/O devices are configured.

## Subscription Management

Plant SCADA's OPCUA driver uses the subscription model described in 5.13.1 of the OPC UA standard to receive value updates from an OPC UA server.

The driver can create up to three OPC UA subscriptions per I/O device with different publishing intervals to provide granular handling of Plant SCADA subscription update rates.

The publishing interval of OPC UA subscriptions can be set by using the ini parameters `Subscription1PublishingInterval`, `Subscription2PublishingInterval`, `Subscription3PublishingInterval` (see [Subscription Parameters](#)).

The default value of `Subscription1PublishingInterval` parameter is set to 400 msec. The default values of `Subscription2PublishingInterval` and `Subscription3PublishingInterval` are set to 0, which means that by default the driver will only create one OPC UA subscription with a publishing interval specified by `Subscription1PublishingInterval` parameter.

The user can change `Subscription2PublishingInterval` and `Subscription3PublishingInterval` parameters to enable OPC UA subscriptions with these publishing intervals.

By default, the driver sets the OPC UA subscription sampling interval to half the length of the associated publishing interval. You can also set the sampling interval to a specific value using the INI parameters `Subscription1SampleRate`, `Subscription2SampleRate` and `Subscription3SampleRate`.

The driver adds monitored items for all I/O Device tags to each of the enabled OPC UA subscriptions. The monitoring mode for all items is set to DISABLED. When the driver gets a subscribe request, it will define the OPC UA subscription in which the item should be enabled based on the Plant SCADA subscription update rate and then it will change the item monitoring mode in that subscription to REPORTING.

If the Plant SCADA subscription update rate changes, the driver will re-evaluate the corresponding OPC UA subscription for the item and change it if required.

## Security Configuration

### NOTICE

When establishing secure connection either via user authentication or message encryption with client and

server certificates, the anonymous connection and non-secure endpoint should be disabled to make sure that unauthorized user cannot connect and perform actions on the OPC UA server.

Also, make sure that the [OPCUA]SecuritySettingsName ini parameter is set to the correct value. The driver log messages will specify the security mode/policy and the authentication mode which the driver will use for a specific I/O device. The driver logs can be found in syslog.IOServer.<Cluster name>.<IO Server name>.dat file.

The OPCUA driver provides the ability to use user credentials when connecting to the device and also to establish secure connection between the driver and the OPC UA server using the client and server X509 certificates. In order to establish a secure connection, the driver needs access to the server certificate and the server needs access to the driver client certificate.

For more information about OPC UA security model, refer to the OPC Unified Architecture Specification, Part 2: Security Model. You can download the document from the OPC Foundation website ([www.opcfoundation.org](http://www.opcfoundation.org)).

The configuration of the secure connection is done via Configurator. After the driver is installed, the Configurator will have a plugin "OPC UA Client Driver".

This plugin allows you to:

- Create security settings.
- Select the client certificate from the Windows certificate store.
- Import the client certificate from a file into the Windows certificate store.
- Issue Plant SCADA OPC UA client certificate, store it in the Windows certificate store and export it to a file.
- Import the server certificate from the file into the Windows certificate store.
- Test connection to the OPC UA Server.

#### To configure the security settings (on a local computer):

1. In the panel on the left side of the Configurator, select OPC UA Client Driver. The SETTINGS page will display. Select the existing settings or enter the settings name to create a new one. If there are no existing settings "Select existing settings" option is disabled.
2. Click the **Next** button. The USER page will display.

On that page, you can enable user authentication and specify the user name and the password which the driver will use to connect to the OPC UA server.

The Configurator stores the user name and the password to the ArchestraA™ Data Store (ADS) service. For the driver to be able to read the user name and the password from ADS, the I/O Server process user account needs to exist in the Citect.User.Drivers Windows group.

The Configurator will add 'Citect.Driver.Users' Windows group if it does not exist and it will also add the currently logged in user to that group. Note that after the user has been added, you need either to restart the computer or to log in / log out so that the driver can read the user name and the password from ADS.

In case when the Plant SCADA I/O process will be configured to run as a service, you will need to select the option 'The driver process will run as a service'. In that case the Configurator will add the Runtime Manager service account to 'Citect.Driver.Users' group.

When user authentication is used, it is recommended that you configure the secure communication with the client and server certificates as well. Otherwise the user name and password will be sent to the server as a plain text.

If you do not do this, it will not be possible to proceed with the configuration after the SECURITY page and an

error message will be displayed.

There are two options to fix this:

- a. Configure a valid security mode on the SECURITY page.
- b. Set the Disable encrypted password check option on the USER page to allow the user name and password to be sent to the server as a plain text.
3. Click the **Next** button. The SECURITY page will display.

Select the message security mode and the security policy.

The security mode specifies the mode of encryption that will be used when messages between the driver and server are sent. There are three options: None, Sign and SignAndEncrypt.

The security policy parameter specifies the endpoint security policy. The supported options are:

- None
- Basic256 and Basic128Rsa15
- Basic256Sha256
- Aes128Sha256RsaOaep
- Aes256Sha256RsaPss.

In the case where the security mode is set to None, the driver will establish a non-secure connection.

4. Click the **Next** button. The CLIENT page will display.

On that page you will need to select the client certificate from the list of available client certificates in the Windows Certificate Store.

If there are no client certificates available, there are two options to add a new one:

- a. Create a new Plant SCADA client certificate.

When this option is selected and you click the "Create" button, the Configurator creates a new OPC UA Client self-signed certificate and adds it to the Windows certificate store.

The newly created client certificate is automatically selected in the list of available certificates.

- b. Import the client certificate from a file.

When this option is selected, you should select the file in which the client certificate is stored. Use the password box if the file is password protected.

The imported client certificate must have a private key and its "Subject Alternative Name" field should be in the format "urn:HostMachineName:Citect.OPCUA.Client.Driver", where the HostMachineName is the host name of the computer running the driver.

In order to install the client certificate on the OPC UA server machine, there is an option on the page to export the client certificate to a file.

The page also provides an option to view the currently selected certificate and to export the client certificate to a file, so it can be installed on the OPC UA server machine.

When you select the Export option, the Configurator will export the client certificate into a file with a public key. If the certificate was issued by a Certificate Authority, then the Certificate Authority certificate will be exported.

5. Click the **Next** button. The SERVER page will display.

On that page you can add the OPC UA server certificate to the Windows certificate store.

If the server certificate with a public key is already in the Windows certificate store, the driver will load it automatically.

There are two options to import the server certificate:

- a. Import a server certificate from a file. When this option is selected, the Configurator will import the server certificate to the Windows Certificate Store.
- b. Import a server certificate directly from the OPC UA server. This option allows to select one of the available server certificates and add it to the Windows Certificate store. You will need to specify the OPC UA server URL and then click the Browse button. After that the Configurator will connect to the server and get certificates for all endpoints exposed by the server. If the operation was successful, the server certificates will be added to the drop-down list of available certificates.

---

**Note:** Importing a server certificate directly from OPC UA server only works for self-signed server certificates.

---

6. Click the **Next** button. The FINISH page will display.

This page informs you that the Configurator is ready to store the configured settings. After you click the Configure button, the settings will be stored in OPCUASettings file in the Plant SCADA Config folder. The Configuration Messages panel will indicate if the configuration was successful. If required, you can use the Previous button to make any changes to your settings before you complete the configuration process or create and configure new settings.

There is also an option to check the connection to the server after the settings have been configured. You will need to specify the OPC UA server URL and then click the Test connection button. If the connection was successful, a message about that will appear in the Configuration Messages panel. Otherwise a message will indicate a specific error.

7. Use device specific SecuritySettingsName ini parameter to apply the settings to a specific I/O device. For example, you can enter:

```
[OPCUA.PORT1_BOARD1.IODev1]  
SecuritySettingsName=MySettings
```

---

**Note:** The security settings are stored in the OPCUASetting.xml file in the Citect SCADA config folder. When upgrading Citect SCADA to a new version, that file has to be manually copied to the new Plant SCADA/Citect SCADA config folder.

---

## Customizing a Project using Citect.ini Parameters

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

- Do not under any circumstances change or remove any of the undocumented citect.ini parameters.
- Before deleting sections of the citect.ini file, confirm that no undocumented parameters will be deleted.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Citect.ini parameters are used to tune the performance of the Plant SCADA OPC UA driver, and to perform runtime maintenance diagnostics.

You can customize the way Plant SCADA communicates with the OPC UA system (and even individual PLCs) by creating or editing the [OPCUA] section of the Citect.ini file for your project.

When Plant SCADA starts at runtime, it reads configuration values from the Citect.ini file that is stored locally.

Therefore, any OPC UA configuration settings must be included in the Citect.ini file located on the computer acting as the I/O server to the OPCUA system.

- [Connection Parameters](#)
- [Subscription Parameters](#)
- [Security Parameters](#)
- [Logging Parameters](#)

**Note:** Any parameters placed under the [OPCUA] section of the Citect.ini file will apply globally to every device using the OPCUA driver. However, you can also set parameters for a specific device, or for every device connected to a specified port. For more information, see [Device-specific Parameters](#).

## Connection Parameters

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

[OPCUA] Parameter	Description	Allowable Values	Default Value
BrowseTimeout	The browse timeout (in milliseconds).  This parameter specifies the amount of time that an OPC UA server browsing window will wait for a browse call to succeed on an OPC UA server. If browsing is not successful in this period of time, it will stop.	1- 1800000	60000 (ms)
ConnectionDelay	The connection delay between reconnection attempts.  If the connection to the server is unsuccessful, by default the driver will immediately try to reconnect. This parameter allows you to specify a delay before the next	0-30000	0 ms

	connection attempt occurs.		
ConnectionTimeout	The connection timeout (in milliseconds). This parameter specifies the time in milliseconds that the driver will wait for a connection with the OPC UA server to be made.	1-1000000	10000 ms
CallTimeout	The timeout in milliseconds for individual service calls. This parameter defines how long the driver will wait for a server response after sending a service call.	1-1000000	10000 ms
ItemsMaximumBlockSize	Maximum number of monitored items the driver can add or read in one request to the OPC UA server.	1-10000	100
SessionTimeout	The session timeout (in milliseconds) for the server connection. If the driver is unable to issue a request within this interval, the server will automatically stop the connection.	1 - 10000000	3600000 ms
StatusItemNodeId	<p>The status item Node ID. If this parameter is specified, the driver will only put the I/O device online if the value of this item is equal or greater than the value specified for the StatusItemValue parameter (see below).</p> <p>For example, if the standard OPC UA ServiceLevel item is used as the status item, these parameters can be set as follows:</p>	-	Not specified.

	StatusItemNodId=i=2267 StatusItemValue=200		
StatusItemValue	The comparison value for the status item specified by the StatusItemNodId parameter (see above).	0 – 1000000	0
UseSamplingMonitoringMode	When a tag is not subscribed in Plant SCADA, the driver sets the OPC UA item monitoring mode to Disabled. This parameter allows you to set it to Sampling mode instead.  This can be useful in case where the OPC UA server does not publish the current item value after the tag is subscribed and the driver changed the item monitoring mode from Disabled to Reporting.	0 – monitoring mode is set to Disabled for unsubscribed tags  1 - monitoring mode is set to Sampling for unsubscribed tags	0
WatchdogTime	The time (in milliseconds) between watchdog checks. This parameter defines how often the driver will perform watchdog read to check the connection to the server.	1- 100000	10000 ms
WatchdogTimeout	The timeout (in milliseconds) for a specific watchdog check.	1- 100000	10000 ms

**Subscription Parameters****UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

[OPCUA ] Parameter	Description	Allowable Values	Default Value
ItemDeactivationDelay	Item deactivation time in seconds.	1 - 600	30 (seconds)
QueueSizePerItem	Maximum number of updates per item queued in the server before publishing.	1 - 10000	10
ReadAfterWrite	<p>Determines if, after writing to an item, the driver performs a synchronous read of the same item.</p> <p>This parameter can be useful in cases where the OPC UA server reports that a write was successful, when in reality the value of the item had not changed. In this case, the tag value on the graphics screen is updated to the written value if [Code]WriteLocal is set to 1 (default behavior). This is not correct.</p> <p>Setting this parameter to 1 will make sure the driver will read the item value after the write is completed, and that the value on the graphics page will be updated to the correct one.</p>	<p>0 - the driver does not perform an item read after write</p> <p>1 - the driver performs a synchronous read of the item after write</p>	0
Subscription1PublishingInterval	Subscription 1 publishing interval in milliseconds.	Any positive value less than Subscription2 PublishingInterval if Subscription2 PublishingInterval is greater than 0.	400 ms
Subscription2PublishingInterval	Subscription 2 publishing	Any positive value less	0 ms

terval	interval in milliseconds.	than Subscription3 PublishingInterval if Subscription3 PublishingInterval is greater than 0.	
Subscription3PublishingIn terval	Subscription 3 publishing interval in milliseconds.	Any positive value less than 86400000.	0 ms
Subscription1SampleRate	Subscription 1 sampling interval in milliseconds.	Any positive value less than 86400000.	0 - this parameter is not used. The sample rate will automatically adjust to half of the associated publishing interval.  You only need to set this parameter if a specific sampling rate is required.
Subscription2SampleRate	Subscription 2 sampling interval in milliseconds.	Any positive value less than 86400000.	0 - this parameter is not used. The sample rate will automatically adjust to half of the associated publishing interval.  You only need to set this parameter if a specific sampling rate is required.
Subscription3SampleRate	Subscription 3 sampling interval in milliseconds.	Any positive value less than 86400000.	0 - this parameter is not used. The sample rate will automatically adjust to half of the associated publishing interval.  You only need to set this parameter if a specific sampling rate is required.

## See Also

[Subscription Management](#)

## Security Parameters



### UNINTENDED EQUIPMENT OPERATION

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

[OPCUA] Parameter	Description	Allowable Values	Default Value
SecuritySettingsName	I/O Device security settings name		Not Specified
SecureChannelLifetime	The time (in milliseconds) after which the SecureChannel between the client and server is automatically renewed. For more information about the SecureChannel Services, please consult the OPC UA specification.	1 - 10000000	3600000 ms

## See Also

[Security Configuration](#)

## Logging Parameters

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

[OPCUA ] Parameter	Description	Allowable Values	Default Value
DebugLevel	The trace level used for the log file.	ALL or any combination of the following, separated by a pipe character ( ):  ERROR WARN TRACE DEBUG See <a href="#">Logging</a> for examples.	-
DebugCategory	The categories used for the log file.	ALL or any combination of the following, separated	-

		<p>by a pipe character ( ):</p> <p>PROT = protocol level debug</p> <p>DCB = front end debug (i.e. I/O server)</p> <p>BEND = back end debug</p> <p>MISC = any other debugging</p> <p>See <a href="#">Logging</a> for examples.</p>	
DebugUnits	Specifies the I/O devices that will have log messages written to a log file.	The names of one or more units separated by a pipe character ( ).	Empty string (log messages sent for every device).

## Device-specific Parameters

### **WARNING**

#### UNINTENDED EQUIPMENT OPERATION

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

Any parameters placed under the [OPCUA] section of the Citect.ini file will apply globally to every device using the OPCUA driver. However, you can also set parameters for a specific device, or for every device connected to a specified port.

To do this, use one of the following section names within the Citect.ini file.

- [<Protocol>.<PortName>]  
where: <Protocol> is OPCUA and <PortName> is the 'Port Name' defined for a specified port on the Ports form.  
For example, any parameters set within the section [OPCUA.PrimaryPort1] will impact every device connected to the port defined as "PrimaryPort1" on the Ports Form.
- [<Protocol>.<PortName>.<DeviceName>]  
where: <Protocol> is OPCUA, <PortName> is the 'Port Name' defined for a specified port on the Ports form and <DeviceName>. For example, any parameters set within the section [OPCUA.PrimaryPort1.PrimaryIODev128] will only impact the device named "PrimaryIODev128".

## Tag Import

You can use Plant SCADA's Import Variables Tags tool to automatically create variable tags that correspond to

OPC UA server items.

### To create variable tags based on OPC UA variables/properties:

1. Open Plant SCADA Studio.
2. In the **Topology** activity, select **I/O Devices** then **Import Variable Tags**.
3. In the Destination panel, select the required I/O Device. The drop-down menu includes a list of all the I/O devices currently configured in your Plant SCADA project. The device you select needs to be configured to use the OPCUA protocol, otherwise an error message will be displayed when you attempt to configure the device connection.
4. In the Source panel, set the **Database Type** to "OPCUA". You can ignore the value of the External Database field.
5. Click on the **Browse...** button to display the OPC UA Browse dialog box.

If the OPC UA server does not allow unsecured or anonymous connections, you will need to configure the security settings via the OPC UA Configurator plugin before browsing or importing tags. You then need to select the security settings name from the drop-down list at the top of the page.

---

**Note:** Select the "Import Item Description" check box to import the OPCUA item description.

6. Click on the expand button (+) to display all the namespaces on the OPC UA Server. Plant SCADA will connect to the OPC UA server to read the names of the namespaces and the folders under each.

There are two options you can use to import the tags:

- a. Create tags for all items under the selected folder node or select the specific items. When using this option, you need to select the namespace or folder in the tree.
  - b. For the second option, you need to click on one of the folder nodes which will display all items under this node. Select which items you want to import. The selected items will be added to the list at the bottom of the page.
7. When you have made your selections, click OK.
  8. On the Import Variable Tags dialog box, click the **Import** button.

If the first tag import option is used, Plant SCADA will connect to the OPC UA server to read the tag name, data type and address (Node ID). If the second import option is used, Plant SCADA will create tags for all selected items.

---

**Note:** The number of items which can be browsed for a particular folder node and the number of tags you can import is limited to 100,000.

The value of the property **Browse Name** will be used as the tag name. If it has a parent folder, it will be pre-fixed with the folder name (separated by '\').

---

**Note:** If the name of the tag is greater than 78 characters then it will be truncated to 78 characters. A warning message will be logged in the tag import log file.

9. On Tag Import Results dialog box, click **Yes** to display the log file.

## Troubleshooting

The following topics provide information about the Plant SCADA tools available to diagnose and resolve any unexpected behavior within the runtime system.

- [OPC UA Server Connection](#)

- [Driver Errors](#)
- [Driver Statistics](#)
- [Logging](#)

## OPC UA Server Connection

If secure connection is not required, only the OPC UA server URL needs to be specified in the I/O Device address form to establish the connection. But if the user authentication or message encryption is required, the security settings will need to be configured by using the "OPC UA Client Driver" Configurator plugin.

The plugin provides an option to check if the connection can be established after the settings have been configured.

If the connection cannot be established, the specific error will help to identify the cause of the problem.

Points to consider include:

- The user credentials are stored in ArchestraA™ Data Store and the ArchestraA™ Data Store service must be running on the machine for the plug-in to store the data and for the driver to read it. The ArchestraA™ Data Store service should be installed and running after the driver installation.
- When the user credentials are configured for the first time, the plug-in adds the currently logged-in user to the Citect.User.Drivers Windows group so that the I/O Server running under the same user account can read the data. When a user is added to that group for the first time, you will need either to restart the computer or to log in / log out.
- The client and server certificates must be present in the Windows certificate store. The client certificate must have a private key and its "Subject Alternative Name" field should be in the format "urn:HostMachineName:Citect.OPCUA.Client.Driver", where the HostMachineName is the host name of the computer running the driver.

The driver logs can also help to find out why the driver cannot connect to the server.

When full driver logging is enabled by setting the ini parameters ([OPCUA]DebugLevel=ALL and [OPCUA]DebugCategory=ALL), the driver log will contain a number of messages or errors describing the connection attempt for each I/O Device.

## Driver Errors

Plant SCADA has two kinds of protocol driver errors: generic and specific.

- Generic errors are hardware errors 0-31, which are common to many protocols.
- Specific errors are unique to a particular driver.

When a driver specific error occurs, the hardware alarm system will not recognize it. The driver converts the specific error into a generic error that can be identified by the I/O server.

This means a driver error may be represented by a specific error, and a corresponding generic error.

When a hardware error occurs, Plant SCADA generates an alarm and displays it on the hardware alarm page. To see the error number, you need to have Alarm Category 255 defined with a display format that includes {ErrDesc,15} {ErrPage,10}.

Refer to the Plant SCADA Help for more information on generic hardware errors.

If you need more information to resolve an error, refer to the documentation that accompanied the I/O device (or network). If, after reviewing the documentation, you still cannot solve your problem, contact Technical Support for this product.

## Driver Statistics

You can use the following driver statistic counters to help you debug the OPCUA driver. These statistics can be viewed within the Plant SCADA Kernel using the Page Driver command and then by pressing the 'v' key to select verbose mode.

See the topic Runtime > Debug Runtime > The Kernel > Kernel Commands > Page Driver in the main Plant SCADA help for more information.

Statistic	Description
# tags subscribed	Total number of subscribed tags for all I/O Devices.
# total updates	Total number of tag value updates received from the OPC UA server for all I/O Devices.
# updates per second	Total number of tag value updates per second received from the OPC UA server for all I/O Devices.
# writes	Total number of write requests processed by the driver for all I/O Devices.
# writes failed	Total number of failed write requests for all I/O Devices.
# configured units	Number of configured I/O Devices.
# units online	Number of I/O Devices which are online.

## Logging

The OPCUA driver can log combinations of trace levels across different categories. This is achieved by setting the following INI parameters.

### [OPCUA]DebugLevel

This parameter allows you to define the trace level. The options include:

WARN	Output warning messages.
ERROR	Output error messages.
TRACE	Output trace messages.
DEBUG	Output debug messages.

ALL	All of the above.
-----	-------------------

## [OPCUA]DebugCategory

This parameter allows you to enable logging for a particular category of trace. The options include:

PROT	Operations related to the OPC protocol interface
DCB	Front end driver trace
BEND	Operations related to the back-end thread of the driver
MIISC	Miscellaneous operations
ALL	All of the above

In both cases, you can use ALL or any combination of the available options separated by a pipe character ( | ).

### Example

```
[OPCUA]
DebugLevel=WARNING|ERROR|TRACE
DebugCategory=ALL
```

In most cases, you can set the DebugLevel and DebugCategory parameters to ALL. If this results in a large, cumbersome log file that causes older data to be overwritten, you can consider refining the debug options.

---

**Note:** These parameters do not have a value specified by default. To enable logging for this driver, you need to manually configure these parameters in your INI file.

The events generated by the OPCUA driver are logged in the following Plant SCADA syslog file:

```
syslog.IOServer.<Cluster name>.<IO Server name>.dat
```

This file is located in the directory specified by the Citect.ini parameter **[CtEdit]Logs**. By default, this location will be:

```
%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Logs
```

The size of the syslog file and its rollover process is configured via the **[Debug]SysLogSize** and **[Debug]SysLogArchive** INI parameters. For more information, see the topic *Debug Parameters* in the Plant SCADA documentation.

## See Also

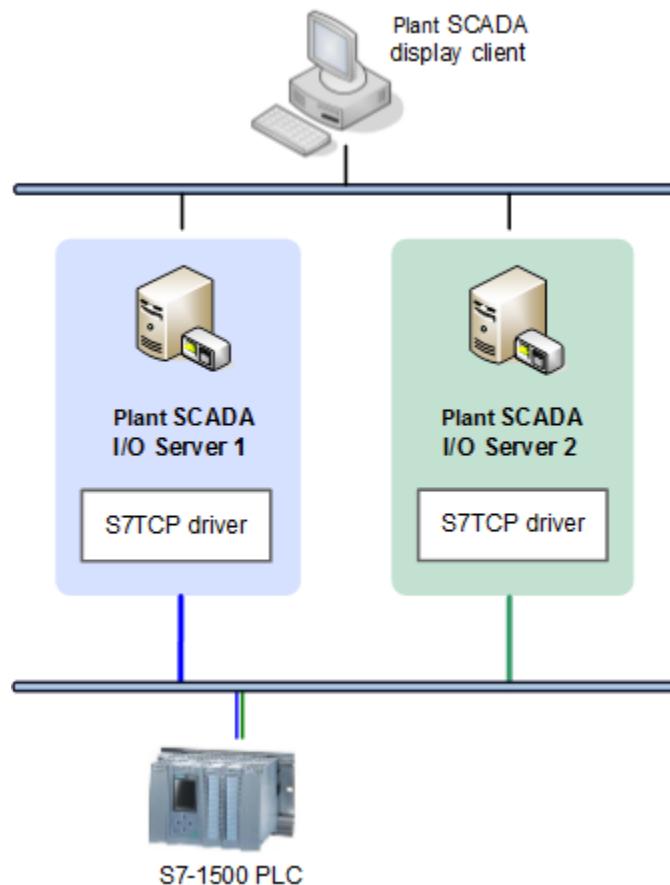
[Logging Parameters](#)

## S7TCP Driver

The S7TCP Driver enables Plant SCADA to communicate with Siemens S7-PLC series via an Ethernet connection. The driver provides access to a Siemens S7-PLC through standard Ethernet network interface card in the computer.

The driver supports symbolic tag names of PLC items.

The following diagram demonstrates a simple system configuration, with Plant SCADA connected Siemens S7-PLC via the S7TCP driver hosted on a single pair of redundant I/O servers.



**Note:** The S7TCP driver requires Citect SCADA version 7.50 Service Pack 1 (Patch 9) or a later release to operate successfully.

## See Also

- [Supported Devices](#)
- [Setting up Device Communications](#)
- [Configuring Variable Tags](#)
- [Advanced Configuration and Maintenance](#)

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.
<b>⚠ DANGER</b>	<b>DANGER</b> indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.
<b>⚠ WARNING</b>	<b>WARNING</b> indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.
<b>⚠ CAUTION</b>	<b>CAUTION</b> indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.
<b>NOTICE</b>	NOTICE used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.

## Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

## Before You Begin

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ WARNING****LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

## Supported Devices

The S7TCP driver allows Plant SCADA to communicate with the Siemens S7-PLC family of controllers.

- The driver supports S7-1500 firmware up to v3.1.0.
- The driver supports S7-1200 firmware between v4.0 and v4.5.

---

**Note:** The driver does not support secure communication to a PLC.

---

## See Also

[Setting up Device Communications](#)

## Setting up Device Communications

The information included in the following topics describes how to establish communication with Plant SCADA and S7-PLC devices using the S7TCP driver.

### See Also

[Device Address](#)

[Communication Settings](#)

### Device Address

The address to use for an I/O device using the S7TCP driver is the IP address of the Ethernet interface on the S7-PLC.

### See Also

[Communication Settings](#)

### Communication Settings

To establish communication with a device, the associated **Board**, **Port** and **I/O Device** need to be correctly configured in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure them, use the settings outlined below.

### Boards

The Boards view in Plant SCADA Studio's Topology activity lists all boards used in the Plant SCADA project.

Field	Value
Board Name	This field is user defined (e.g. 'Board1').
Board Type	Enter S7TCP
Address	Enter zero (0)
I/O Port	Leave this field blank
Interrupt	Leave this field blank
Special Options	Leave this field blank
Comment	Any useful comment. This field is user defined and not required by the driver.

## Ports

The Ports view in Plant SCADA Studio's Topology activity lists the ports used in the Plant SCADA project.

Field	Value
Port Number	This field is user defined
Baud Rate	Leave this field blank
Board Name	Refers to the board previously defined in the Board settings
Data Bits	Leave this field blank
Stop Bits	Leave this field blank
Parity	Leave this field blank
Special Options	Leave this field blank
Comment	Any useful comment. This field is user defined and not required by the driver.

## I/O Devices

The I/O Devices view in Plant SCADA Studio's Topology activity lists I/O devices used in the Plant SCADA project.

Field	Value
Name	This field is user defined
Number	I/O device number which needs to be unique for each logical device, but identical for redundant devices.
Address	The IP address of the remote S7-PLC
Protocol	Enter S7TCP.
Port Name	Refers to the port previously defined in the Port settings.
Comment	Any useful comment. This field is user defined and not required by the driver.
Min Update Rate	Defines the period in seconds at which the driver polls the PLC for tag value updates. You may select any predefined value from the drop-down list, or enter your own in the format of HH:MM:SS. If you do not enter a value, the default value of 0 seconds will be used in which case the polling period will be equal to

Field	Value
	1 second.

## Configuring Variable Tags

Variable tags are addressed using the symbolic tag names configured for Siemens S7-PLC device. The following table demonstrates the variable tag address formats you can use to read and write values

Address Format	S7-1500 Data Type	S7-1200 Data Type	Plant SCADA Type	Value Range
<Symbolic Tag Name>	BOOL	BOOL	DIGITAL	FALSE or TRUE
<Symbolic Tag Name>	BYTE	BYTE	INT, UINT, LONG, ULONG	0 to 255
<Symbolic Tag Name>	WORD	WORD	UINT, LONG, ULONG	0 to 65535
<Symbolic Tag Name>	DWORD	DWORD	ULONG	0 to 4294967295
<Symbolic Tag Name>	SINT	SINT	INT, LONG	128 to 127
<Symbolic Tag Name>	USINT	USINT	INT, UINT, LONG, ULONG	0 to 255
<Symbolic Tag Name>	INT	INT	INT, LONG	-32768 to 32767
<Symbolic Tag Name>	UINT	UINT	UINT, LONG, ULONG	0 to 65535
<Symbolic Tag Name>	DINT	DINT	LONG	-2147483648 to +2147483647
<Symbolic Tag Name>	UDINT	UDINT	ULONG	0 to 4294967295
<Symbolic Tag Name>	REAL	REAL	REAL	-3.4E38 to 3.4E38
<Symbolic Tag Name>	CHAR	CHAR, STRING	CHAR, STRING	ASCII character
<Symbolic Tag Name>	WCHAR	WCHAR	STRING	Unicode character represented as ASCII character in Plant

Address Format	S7-1500 Data Type	S7-1200 Data Type	Plant SCADA Type	Value Range
				SCADA <b>Note:</b> Driver supports multi-byte strings only when it is possible to convert a multi-byte character to the ASCII.
<Symbolic Tag Name>	WSTRING	WSTRING	STRING	Unicode string (0-254 characters) represented as ASCII string in Plant SCADA <b>Note:</b> Driver supports multi-byte strings only when it is possible to convert a multi-byte character to the ASCII.
<Symbolic Tag Name>	S5TIME	N/A	LONG	0 to 9990000 (milliseconds)
<Symbolic Tag Name> -	TIME	TIME	LONG	-2147483648 to +2147483647 (milliseconds)
<Symbolic Tag Name>	DATE	DATE	LONG	0 to 65378 (days)
<Symbolic Tag Name>	TIME_OF_DAY (TOD)	TIME_OF_DAY (TOD)	ULONG	0 to 86399999 (milliseconds)
<Symbolic Tag Name>	DATE_AND_TIME	N/A	STRING	1990-01-01-00:00:00.000 to 2089-12-31-23:59:59.999
<Symbolic Tag Name>	LTIME	N/A	STRING	-9223372036854775808 to +9223372036854775807 (nanoseconds)
<Symbolic Tag Name>	LTOD (LTIME_OF_DAY)	N/A	STRING	0 to 8639999999999 (nanoseconds)

Address Format	S7-1500 Data Type	S7-1200 Data Type	Plant SCADA Type	Value Range
<Symbolic Tag Name>	LDT (DATE_AND_LTIME)	N/A	STRING	0 to 9254908036854775 808 (nanoseconds)

When configuring tags to access PLC arrays and structures, a separate tag needs to be created for each array or structure element.

The driver supports conversions between S7-PLC data types and Plant SCADA data types. When a configured tag data type cannot be converted into PLC data type, the driver marks that tag with quality Bad, Configuration Error. The following table shows supported conversions between Plant SCADA and S7-PLC data types.

Plant SCADA Data Type	S7-PLC Data Type
DIGITAL	BOOL
INT, UINT	BYTE, WORD, SINT, USINT, INT,UINT
LONG, UINT	BYTE, WORD, DWORD, SINT, USINT, INT,UINT, DINT, UDINT
REAL	REAL
STRING	All S7-PLC data types supported by the driver, except for BYTE, USINT, WORD, UINT, DATE

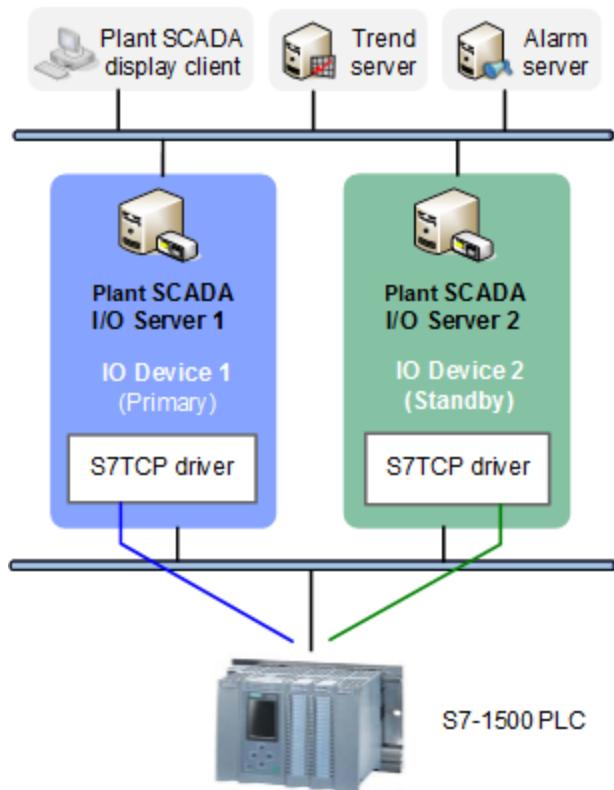
## Advanced Configuration and Maintenance

This section provides instructions for advanced configuration and maintenance tasks.

- [Configuring Redundancy](#)
- [Customizing a Project Using Citect.ini Parameters](#)
- [Tag Import](#)

### Configuring Redundancy

The S7TCP driver supports multiple I/O server / device redundancy. The driver represents each S7-PLC as a separate I/O device. In normal operating state, the driver will poll the PLC via the primary I/O device. If the connection to the PLC via the primary I/O device is lost, the PLC will be polled via a standby I/O device which is connected to the PLC and has the highest priority in case when multiple redundant I/O devices are configured.



**Note:** The redundancy switchover will only work properly if Plant SCADA periodically checks the communication connection to the primary and standby I/O devices. Plant SCADA automatically checks the primary I/O device, but this process can be disabled for the standby I/O device if the I/O server parameter [IOServer]WatchDog is set to 0 (disabled). By default, this parameter enables Watch Dog functionality for a standby I/O device. It should not be changed.

### Customizing a Project using Citect.ini Parameters

#### **WARNING**

##### UNINTENDED EQUIPMENT OPERATION

- Do not under any circumstances change or remove any of the undocumented citect.ini parameters.
- Before deleting sections of the citect.ini file, confirm that no undocumented parameters will be deleted.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

Citect.ini parameters are used to tune the performance of the Plant SCADA S7TCP driver, and to perform runtime maintenance diagnostics.

You can customize the way Plant SCADA communicates with the S7TCP system (and even individual PLCs) by creating or editing the [S7TCP] section of the Citect.ini file for your project.

When Plant SCADA starts at runtime, it reads configuration values from the Citect.ini file that is stored locally.

Therefore, any S7TCP configuration settings must be included in the Citect.ini file located on the computer acting as the I/O server to the S7TCP system.

## See Also

[S7TCP Driver Parameters](#)

[Logging Parameters](#)

[Using Device or Port-specific Parameters](#)

## S7TCP Driver Parameters

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

[S7TCP] Parameter	Description	Allowable Values	Default Value
ConnectionTimeout	This parameter specifies the time in milliseconds that the driver will wait for a connection with a PLC to be made.	0-100000	20000
HeartBeatTag	This parameter specifies the name of the item that gets updated in the PLC at a rate determined by the HeartBeatInterval parameter.  Some system items are already programmed in the PLC and can be used to avoid changes to the PLC program.  For example, "db_test.Test_Uint32" has value updates every 5ms.	PLC item name	-
HeartBeatInterval	This parameter defines the period the driver will use to check the status of the item specified by the	1 - 2147483647 (seconds)	60

[S7TCP] Parameter	Description	Allowable Values	Default Value
	<p>HeartBeatTag parameter. If the item value has not changed during the period specified, the driver will put the I/O device offline.</p> <p><b>Note:</b> The driver checks the HeartBeatTag value every Minimum Update rate specified by either the "Min Update Rate" field in the I/O Device configuration settings (which equals 1 sec by default), or by the MinUpdateRate parameter.</p>		
MinUpdateRate	<p>This parameter defines the period in milliseconds at which the driver polls the PLC for tag value updates. Note that by default the polling period is specified by the "Min Update Rate" field in the I/O Device configuration settings which equals to 1 sec by default. The ini parameter allows to specify the minimum update rate in milliseconds.</p> <p>If this parameter is specified and is not equal to 0, the driver will use it to define the polling rate. Otherwise the "Min Update Rate" field in the I/O Device settings will be used.</p>	0-100000	0
ReplyTimeout	This parameter specifies the time in milliseconds that the driver will wait for a response from the	0-100000	5000

[S7TCP] Parameter	Description	Allowable Values	Default Value
	PLC		
StandbyDbCheckInterval	<p>This parameter defines the period in seconds at which the driver polls the PLC to check for database changes in the PLC.</p> <p>If this parameter is specified, the driver will use it to define the database check interval only for standby devices to reduce the network traffic. In this case the primary device will be polled every 3 seconds.</p> <p>If the primary device is down and the secondary device is active, the secondary device will be polled every 3 seconds.</p>	1-120 (seconds)	3

## Logging Parameters

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

- Do not under any circumstances change or remove any of the undocumented citect.ini parameters.
- Before deleting sections of the citect.ini file, confirm that no undocumented parameters will be deleted.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

[S7TCP] Parameter	Description	Allowable Values
DebugLevel	The trace level used for the log file.	ALL or any combination of the following, separated by a pipe character ( ) : ERROR WARN TRACE

		DEBUG See note below. See <a href="#">Logging</a> for examples.
DebugCategory	The categories used for the log file.	ALL or any combination of the following, separated by a pipe character ( ): PROT = protocol level debug DCB = front end debug (i.e. I/O server) MISC = any other debugging ALL = all of the above See <a href="#">Logging</a> for examples.

**Note:** Setting DebugLevel to DEBUG or ALL will cause the driver to log a large amount of debugging information (for example, sent and received protocol messages in binary format). This information is typically unnecessary for driver diagnostics. This may impact performance and result in multiple wraparounds that write over data.

It is recommended that you use the following DebugLevel and DebugCategory settings:

- [S7TCP]DebugLevel=ERROR|WARN|TRACE
- [S7TCP]DebugCategory=ALL

### Using Device or Port-specific Parameters

#### **WARNING**

##### **UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

Any parameters placed under the [S7TCP] section of the Citect.ini file will apply globally to every device using the S7TCP driver. However, you can also set parameters for a specific device, or for every device connected to a specified port.

To do this, use one of the following section names within the Citect.ini file.

- [<Protocol>.<PortName>]

where:

<Protocol> is S7TCP

<PortName> is the 'Port Name' defined for a specified port in the Ports grid.

For example, any parameters set within the section [S7TCP.PrimaryPort1] will impact every device connected to the port defined as "PrimaryPort1" in the Ports grid.

- [<Protocol>.<PortName>.<DeviceName>]

where:

<Protocol> is S7TCP

<PortName> is the name defined for the specified port in the Ports grid.

<DeviceName> is the name defined for the specified device on the I/O Devices grid.

For example, any parameters set within the section [S7TCP.PrimaryPort1.PrimaryIODev128] will only impact the device named "PrimaryIODev128".

The following list includes the parameters you can set at a device or port-specific level.

- ConnectionTimeout
- ReplyTimeout

## See Also

[S7TCP Driver Parameters](#)

## Tag Import

You can use Plant SCADA's Import Variables Tags tool to automatically create variable tags that correspond to S7 1200/1500 PLC items.

### To create variable tags based on S7 tag properties:

1. Open Plant SCADA Studio.
2. In the **Topology** activity select **I/O Devices**, then **Import Tags**.
3. In the Destination panel, select the required I/O Device. The drop-down menu includes a list of all the I/O devices currently configured in your Plant SCADA project. The device you select needs to be configured to use the S7TCP protocol, otherwise an error message will appear when you attempt to configure the device connection.
4. In the Source panel, set the **Database type** to "S7TCP". You can ignore the value of the **External database** field.
5. Click **Browse...** to display the S7TCP Browse dialog box. To select the tags for import:
  - a. Click **Browse...** to display all the tags corresponding to S7 PLC. Plant SCADA will connect to the S7 PLC to read the tag database.
  - b. Select the items you want to import. The selected items will be added to the list at the bottom of the page.
  - c. If you want to prefix the I/O device name to the tag names, select the check box **I/O Device name as a prefix to tag name**.
  - d. When you have made your selections, click **OK**.
6. Click **Import**. Plant SCADA will create tags for all selected items.

---

#### Note:

- The number of items which can be browsed for a PLC, and the number of tags you can import, is limited to 100,000.
  - If the name of the tag is greater than 78 characters, then it will be truncated to 78 characters. A warning message will be logged in the tag import log file.
-

7. On **Tag Import Results** dialog box, click **Yes** to display the log file.

## Troubleshooting

The following topics provide information about the Plant SCADA tools available to diagnose and resolve any unexpected behavior within the runtime system.

- [Driver Errors](#)
- [Driver Statistics](#)
- [Logging](#)

### Driver Errors

Plant SCADA has two kinds of protocol driver errors: generic and specific.

- **Generic errors** are hardware errors 0-31, which are common to many protocols.
- **Specific errors** are unique to a particular driver.

When a driver specific error occurs, the hardware alarm system will not recognize it. The driver converts the specific error into a generic error that can be identified by the I/O server.

This means a driver error may be represented by a specific error, and a corresponding generic error.

When a hardware error occurs, Plant SCADA generates an alarm and displays it on the hardware alarm page. To see the error number, you need to have Alarm Category 255 defined with a display format that includes {ErrDesc,15} {ErrPage,10}.

Refer to the Plant SCADA Help for more information on generic hardware errors.

If you need more information to resolve an error, refer to the documentation that accompanied the I/O device (or network). If, after reviewing the documentation, you still cannot solve your problem, contact Technical Support for this product.

### Driver Statistics

You can use the following driver statistic counters to help you debug the S7TCPdriver. These statistics can be viewed within the Plant SCADA Kernel using the Page Driver command and then by pressing the 'v' key to select verbose mode.

See the topic *Using Plant SCADA > Monitoring and Debugging Runtime > Using the Kernel > Kernel Commands > Page Driver* in the main Plant SCADA help for more information.

Statistic	Description
# tags subscribed	Total number of subscribed tags for all I/O Devices
# plc tags resolved	Total number of resolved PLC tags for all I/O Devices
# plc tags polled	Total number of polled PLC tags for all I/O Devices
# plc reads	Total number of PLC read requests sent by the driver

Statistic	Description
	for all I/O Devices
# writes	Total number of write requests processed by the driver for all I/O Devices
# writes failed	Total number of failed write requests for all I/O Devices
# configured units	Number of configured I/O Devices
# units online	Number of I/O Devices which are online

## Logging

The S7TCP driver can log combinations of trace levels across different categories. This is achieved by setting the following INI parameters.

### [S7TCP]DebugLevel

This parameter allows you to define the trace level. The options include:

WARN	Output warning messages.
ERROR	Output error messages.
TRACE	Output trace messages.
DEBUG	Output debug messages.
ALL	All of the above.

You can use a combination of these options separated by a pipe character ( | ).

#### CAUTION

Setting [S7TCP]DebugLevel to DEBUG or ALL will cause the driver to log a large amount of debugging information (for example, sent and received protocol messages in binary format). This information is typically unnecessary for driver diagnostics and may result in multiple wraparounds that write over data.

#### Example

```
[S7TCP]
DebugLevel=WARNING|ERROR|TRACE
```

### [S7TCP]DebugCategory

This parameter allows you to enable logging for a particular category of trace. The options include:

PROT	Operations related to the OPC protocol interface
DCB	Front end driver trace
MIsc	Miscellaneous operations
ALL	All of the above

You can use ALL or a combination of these options separated by a pipe character ( | ).

#### Example

```
[S7TCP]
DebugLevel=WARNING|ERROR|TRACE
DebugCategory=ALL
```

---

**Note:** These parameters do not have a value specified by default. To enable logging for this driver, you need to manually configure these parameters in your INI file.

The events generated by the S7TCP driver are logged in the following Plant SCADA syslog file:

```
syslog.IOServer.<Cluster name>.<IO Server name>.dat
```

This file is located in the directory specified by the Citect.ini parameter **[CtEdit]Logs**. By default, this location will be:

```
%PROGRAMDATA%\AVEVA Plant SCADA <VersionNumber>\Logs
```

The size of the syslog file and its rollover process is configured via the **[Debug]SysLogSize** and **[Debug]SysLogArchive** INI parameters. For more information, see the topic *Debug Parameters* in the Plant SCADA documentation.

---

**Note:** If the number of messages being processed for a driver exceeds the value specified by the Plant SCADA parameter **[Kernel]ErrorBuffers**, the pending messages will not be logged. If this occurs, you can increase the value of this parameter so that all the messages generated by the driver are reported. The number of messages that are not logged can be monitored via the Kernel's General window under the item "Lost Errors". If you adjust **[Kernel]ErrorBuffers** and Lost Errors are still occurring, you may need to adjust the level of system logging.

## See Also

[Logging Parameters](#)

## SBUS Driver

The SBUS driver supports SERIAL communication with the [SAIA PCD PLCs](#). The protocol can use three different transmission methods:

- 1) Parity changing method
- 2) Break mode transmission
- 3) Data mode transmission

The transmission method is determined by the **[SBUS]SendBreak** parameter. Refer to [SBUS Protocol Parameters](#) for information on how to set the transmission method.

The maximum request length for the SBUS protocol is 16 bits.

## Hints and Tips

To setup the proper communication modes for SBUS with PCDs, you may need to use the utilities provided by SAIA to alter the firmware settings of the PCD devices. Refer to the topic titled [Software Setup](#) for software and firmware version required for SBUS Data Mode.

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

#### DANGER

**DANGER** indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.

#### WARNING

**WARNING** indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.

#### CAUTION

**CAUTION** indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.

#### NOTICE

NOTICE used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.

## Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

## Before You Begin

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

### **WARNING**

#### **LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

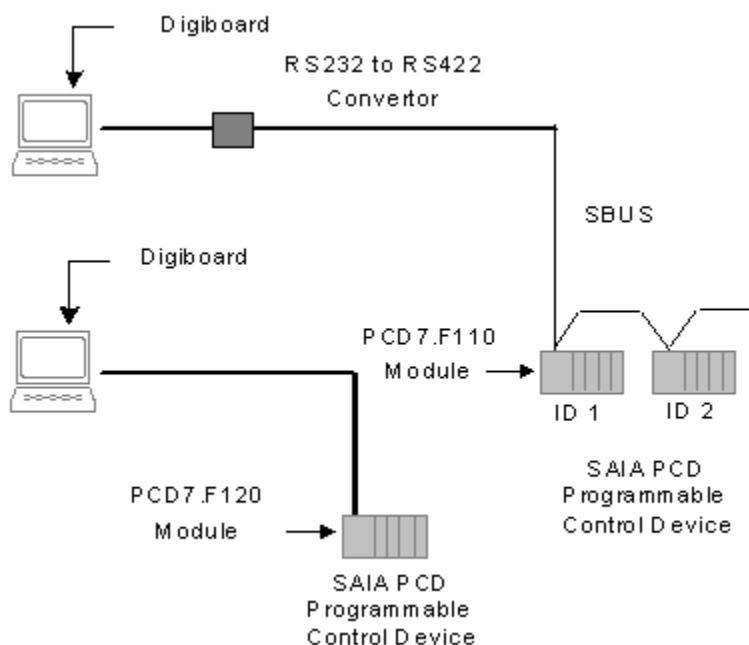
**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

## SAIA PCD PLCs

The SBUS Driver supports SERIAL communication to the Programmable Control Device (PCD) family of PLCs, manufactured by SAIA. Using this method, you can connect to single PCDs or to multiple PCDs as in the following

diagram:



**Note:** To setup the proper communication modes for SBUS with PCDs, you may need to use the utilities provided by SAIA to alter the firmware settings of the PCD devices. For details, see [Software Setup](#).

## Device Address

The I/O Device Address for a PCD is the station address (1-90) configured on the PCD, for example:

- **1** specifies PCD address **1**.
- **49** specifies PCD address **49**.

See [Communications Settings](#) for more information about this setup.

## Hardware Setup

### PCD Family of PLCs

The Saia PCD (Programmable Control Device) is a family of PLCs which are programmed with the same instruction set. All PLCs in the family have 8192 flags, 1520 timer counters and 4096 registers.

- PCD2 - This is the economy PLC which allows up to 128 I/O points, has up to 160kb of memory and up to 4 serial interfaces.
- PCD4 - This is the midrange PLC which allows up to 512 I/O points, has up to 428kb of memory and up to 4 serial interfaces.
- PCD6 - This is the deluxe PLC which allows up to 5120 I/O points, has up to 1MB of memory and up to 28 serial interfaces. This PLC also allows multi processors.

## PCDs Setup

Set your PCD to the communication setting you want. The default hardware settings for the PCD are as follows:

Setting	Value
Baud Rate	9600
Data Bits	8
Stop Bits	1
Parity	0 (See note below.)

These settings are recommendations only. If you use the Communications Express Wizard, these default settings are configured in your project automatically, though your hardware may support other values. If you do set the baud rate, data bits, stop bits or parity to another value, you must manually set the new value(s) in your project.

**Note:** This protocol can use two different transmission methods:

- 1) Parity changing method
- 2) Break method of transmission.

The Plant SCADA SBUS driver uses as default the parity changing method. Run the SAIA PCD Programming Utilities to set the the transmission method.

## PCD6 Hardware Configuration Example

A basic configuration of the PCD6 is as follows:

PCD6-M540Single processor module with 4 serial interfaces.  
PGU, RS232 connection for the programming unit.  
Port 1, RS422 or RS485 for point to point or SAIA bus connection.  
Port 2, RS232 connection for a modem.  
Port 3, 20mA current loop.

PCD7-R210Memory module with 64k RAM and real time clock, or ...

PCD7-R220Memory module with 256k RAM and real time clock.

PCD6-N100D4Power supply unit.

PCD6-C100Rack unit (card cage).

SAIA bus cableRS422 cable for connection to the PC.

PGU cableRS232 cable for the programming interface.

## Software Setup

Each PCD will need to be set up with the right communication configuration. SAIA provides configuration programs (such as PG3 or PG4) to configure the desired protocol, baud rate, timeout, and so on.

The **Run** indicator displayed on the front panel of the PCD indicates whether there is a program loaded into the CPU of the unit. A running program is not required for the unit to function, however, any specific register (such as the LCD display register) may not be accessible without the use of a proper configuration program.

To select SBUS - Data Mode, you need to have the following firmware and software that supports it:

SAIA Program	Firmware Version
PCD1 Firmware	From V 002
PCD2 Firmware	From V 005
PCD4.xx5 Firmware	From V 00D
PCD6.M3 Firmware	From V 001
PG3	From V 2.1
PG4	From V 1.4
SCCOMM-DLL 32 bit	From V \$114
S-Bus Analyser	From V \$ 007

## Wiring Diagram

The PCD Family of PLCs support the following serial communications:

- RS422 or RS485 with PCD7.F110 plug in module.
- RS232 with PCD7.F120 plug in module.
- 20mA current loop with PCD7.F130 plug in module

The serial cable is connected to Interface number 1 on the PLC (pins 10-19). The actual connections depend on the type of serial link. Refer to the PCD2 Series Hardware Manual.

## Communication Settings

To establish communication with a device, the associated Board, Port and I/O Device need to be correctly configured in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. However, if you need to manually configure them, use the settings outlined below.

## Boards

Typically, you would use a serial board or COM port for this communication.

### Board Type

Enter COMx.

### Address

Enter 0.

### I/O Port

Leave this field blank.

### Interrupt

Leave this field blank.

#### Special Options

Leave this field blank.

## Ports

#### Port Number

This value should match the COM port number. This number is defined in the Ports section of the Control Panel.

#### Baud Rate

This value must match the setting of the PCD - 9600 is the default.

#### Data Bits

Enter 8.

#### Stop Bits

Enter 1.

#### Parity

This protocol can use two different transmission methods: 1) a Parity changing method or 2) a Break method of transmission. The Plant SCADA SBUS driver uses as default the parity changing method. Run the SAIA PCD Programming Utilities to set the PLC the transmission method.

#### Special Options

You may want to use the Special Options for the COMx Driver if you are using a Modem (or similar) and want the driver to perform differently.

## I/O Devices

#### I/O Device Address

The station address (1-90) configured on the PCD, for example:

- **1** specifies PCD address **1**.
- **49** specifies PCD address **49**.

#### I/O Device Protocol

Enter SBUS.

## Data Types

Data Types	Address Format	Range ( <i>n</i> )	Plant SCADA Data Type
Flag	F <i>n</i>	0 to 8191	DIGITAL
Output	O <i>n</i>	0 to 5120	DIGITAL
Inputs	I <i>n</i>	0 to 5120	DIGITAL

Data Types	Address Format	Range (n)	Plant SCADA Data Type
Status	S $n$	0 to 7	INTEGER
Version	Version	-	STRING
Display	D[.n ]	-	LONG
Display	D	-	REAL
Register	R $n$ [.n ]	0 to 4095	LONG
Register	R $n$	0 to 4095	FLOAT
Counter	C $n$ [.n ]	0 to 1599	LONG
Timer	T $n$ [.n ]	0 to 1599	LONG
RealTime Clock	K[.n ]	-	LONG

**EXAMPLES:**

Data Type	INTEGER
Address	S6
Comment	Status 6
Data Type	DIGITAL
Address	I107
Comment	Running feedback indicator

## SBUS Driver Parameters

The SBUS driver supports the capability to apply different initialisation parameter values to specific I/O Devices. For further information, see [SBUS Port Specific Parameters](#).

**UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

---

**Note:** Always seek the advice of Technical Support for this product regarding undocumented features.

## [SBUS]Block

A value (bytes) used by the I/O Server to determine if two or more packets can be blocked into one data request before being sent to the I/O Device. For example, if you set the value to 10, and the I/O Server receives two simultaneous data requests - one for byte 3, and another for byte 8 - the two requests will be blocked into a single physical data request packet. This single request packet is then sent to the I/O Device, saving on bandwidth and processing.

**Allowable Values** 5 to 256

**Default Value** 8

## [SBUS]Delay

The period (in milliseconds) to wait between receiving a response and sending the next command.

**Allowable Values** 0 to 300 (milliseconds)

**Default Value** 0

## [SBUS]DisableGmtOffset

Forces the driver to use UTC time for reads and writes to the "K" time register.

**Allowable Values** 0 or 1 (disables the GmtOffset)

**Default Value** 0

## [SBUS]FailOnBadData

Determine whether or not to force the display of good data within a block read which contains bad tags.

**Allowable Values** 0 or 1

0 = Ignore any bad tags in the block reads and just show 0 for those tags.

1 = Display reads that include a bad tag in the block with #BAD.

**Default Value** 1

## [SBUS]GmtOffset

Specifies an offset from GMT in hours to use for reads and writes to the "K" time register.

**Allowable Values**

Local time zone difference on the computer:

-12 to +14 (hours)

**Default Value** 0

## [SBUS]MaxPending

The maximum number of pending commands that the driver holds ready for immediate execution.

**Allowable Values** 1 to 32

**Default Value** 2

## [SBUS]PollTime

The interrupt or polling service time (in milliseconds). Setting the polling time to 0 puts the driver in interrupt mode.

**Allowable Values** 0 to 300 (milliseconds)

**Default Value** 0

## [SBUS]Retry

The number of times to retry a command after a timeout.

**Allowable Values** 0 to 8

**Default Value** 4

## [SBUS]Timeout

Specifies how many milliseconds to wait for a response before displaying an error message.

---

**Note:** Only set this value if the time taken for the return of a data request is greater than the default of 1second. In this situation, set the value to a time period greater than the time taken for the return of a data request.

**Allowable Values** 0 to 32000 (milliseconds)

**Default Value** 1000

## [SBUS]WatchTime

The frequency (in seconds) that the driver uses to check the communications link to the I/O Device.

**Allowable Values** 0 to 128 (seconds)

**Default Value** 30

## [SBUS]SendBreak

This protocol can use three different transmission methods:

- Parity changing method
- Break mode transmission
- Data mode transmission.

Omission of a setting for this parameter will result in Parity changing method being implemented by default.

**Allowable Values**

0 = Parity changing method

1 = break mode

2 = data mode

**Default Value** 0

---

**Note:** This driver level parameter can be over-ridden on an individual port basis with the use of the SBUS.<PortName> parameter. See [SBUS Port Specific Parameters](#).

## Port-specific Parameters

To set parameters for a particular port, you have to create a new section in the Citect.INI file. Label it with the driver name followed by the full stop character (.) and the name of the particular port you want to specify the parameter setting for.

### [SBUS.<PortName>]SendBreak

This applies the SendBreak parameter setting to the specified port.

**Allowable Values** 0 = Parity changing method

1 = break mode

2 = data mode

**Default Value** 0

---

**Note:** This port level parameter will over-ride the driver wide setting made with the use of the parameter.

---

**Example:**

If two ports named PORT1 and PORT2 are being used, and we want to set the SendBreak parameter for PORT2 only, the Citect.INI would contain the following parameters:

[SBUS.PORT2] SendBreak = 1

See also [SBUS Protocol Parameters](#).

## SBUS Driver Specific Errors

When a hardware error occurs, Plant SCADA generates an alarm, and displays the alarm on the hardware alarm page (in the alarm description of the hardware alarm). To see the error number, make sure you have Alarm Category 255 defined with a display format that includes {ErrDesc,15} {ErrPage,10}.

There are a number of generic errors that are common to all protocols. In some cases, only the generic error is available, though often both the generic error and a specific error are given.

The following errors are specific to this protocol. Plant SCADA displays the error number and description for common protocol-specific errors. Uncommon errors are not contained in the Plant SCADA error database, in which case Plant SCADA will only display the error number.

The error below is common to all three modes of transmission used by the SBUS protocol (parity, break, data).

- **28h GENERIC\_GENERAL\_ERROR**

Negative acknowledgement

The following errors only apply when the protocol is set to data mode of transmission.

- **29h GENERIC\_GENERAL\_ERROR**

Negative acknowledgement because of password

- **30h GENERIC\_GENERAL\_ERROR**

Negative acknowledgement because the port is configured with the reduced protocol

- **31h GENERIC\_GENERAL\_ERROR**

Negative acknowledgement because PGU is active (involve from full to reduced protocol)

You may require additional information to enable you to rectify an error. This information should be detailed in the documentation that accompanied the I/O Device (or network). If, after reviewing all documentation, you cannot rectify an error, contact Technical Support.

## SNMP II Driver

The SNMP II driver enables Ethernet-based communication between Plant SCADA and a Simple Network Management Protocol (SNMP II) system.

The maximum request length for the SNMP II protocol is 255 bytes (2040 bits) for a string. Strings use one SNMP request per string tag, while numeric data can be grouped into one SNMP request.

### About SNMP II

SNMP II is a request-and-response protocol. An SNMP manager sends a request to an agent. The agent replies with a response that indicates if the operation was performed successfully or if an error is detected.

Each SNMP operation has its own type of message. Each of these messages is used by a management system to request that an operation be performed on managed variables maintained by an SNMP agent. There are three request and response operations: **Get**, **GetNext**, and **Set**.

A fourth operation, **Trap**, is an unsolicited message sent by an agent; it therefore does not have a corresponding request message. Trap messages can be generated for changes such as host system startup, shutdown, or password violation. Trap destinations can be configured by a user, but the occurrences which generate a trap message are internally defined by the SNMP agent.

The SNMP II protocol specifies the behavior of the Get, GetNext, Set, and Trap operations, and defines the format of the SNMP messages exchanged by managers and agents.

---

**Note:**

- The Plant SCADA driver does not support the GetNext request.
  - SNMP II driver supports SNMP GETS and PUTS (read and write), trap receiving, forwarding and trap generation. Each device runs in a separate NT thread for maximum speed.
  - If an SNMP device only supports a fixed IP reply address, then Plant SCADA I/O Server redundancy will not be supported for that device.
  - SNMP II driver has partial support for SNMP protocol version V2C and currently it supports only the SNMP GETS and PUTS (read and write) operation. The other operations will be supported in the future release version of this driver.
- 

## Safety Information

### Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

Symbol	Description
 or 	The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.
	This is the safety alert symbol. It is used to alert you to personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.
<b> DANGER</b>	
<b>DANGER</b> indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.	
<b> WARNING</b>	
<b>WARNING</b> indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.	
<b> CAUTION</b>	
<b>CAUTION</b> indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.	
<b>NOTICE</b>	
<b>NOTICE</b> used without a safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property or equipment damage.	

## Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by AVEVA for any consequences arising out of the use of this material.

## Before You Begin

Plant SCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to Plant SCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your Plant SCADA project before implementing it in an operational setting. Observe the following:

**⚠ WARNING****UNINTENDED EQUIPMENT OPERATION**

Do not use Plant SCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ WARNING****LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- Each implementation of a control system created using Plant SCADA must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

1. For additional information, refer to NEMA ICS 1.1 (latest edition) "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control", and to NEMA ICS 7.1 (latest edition) "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

## Preparing the SNMP II System

This section explains how to prepare your SNMP II-based devices for communication with Plant SCADA.

This section includes the following topics:

- [Hardware Requirements](#)
- [Installing the SNMP Service](#).

Before installing or configuring your SNMP II driver, you might want to read the [Setup Tips and Techniques](#) section for information on how to avoid problems arising from incorrect setup or installation.

### Hardware Requirements

This section describes how to set up the device that Plant SCADA will query via SNMP.

**To set up the device:**

1. Connect the device to the TCP/IP network for a PC, a network card needs to be installed.
2. If the device is a PC running Windows 2000/7/8/XP, or Windows Server 2003/2008/2012, the TCP/IP protocol needs to be installed and configured, as well as the SNMP service. For details, see [Installing the SNMP Service](#). Refer to the help in the Windows Control Panel for more information.
3. Reinstall any installed Service Pack after installing the TCP/IP protocol or SNMP service.
4. Configure the device with an IP address. Refer to the device's hardware documentation for details; or if the device is a PC, refer to the online help in Control Panel.

**Installing the SNMP Service****To install the SNMP service on Windows:**

1. Go to Control Panel and select **Programs**, then **Turn Windows features on or off**.
2. In the Windows Features dialog box, select **Simple Network Management Protocol (SNMP)** and click **OK**. Installation will commence. You will be notified when it is complete.

**To check that the SNMP service is running:**

1. Go to Control Panel, and select **System and Security**, then **Administrative Tools**.
2. Double-click on **Services**.
3. Scroll down the list of services and locate **SNMP Service** and **SNMP Trap**.
4. Check that the Status column for SNMP Service is "Running". If not, highlight the SNMP Service row and click **Play** on the toolbar to start the service.

**Setup Tips and Techniques**

- This section describes techniques to consider when setting up your system.
- Before the first writing to a variable, at least one read has to be made so that the driver can get the SNMP data type of that variable.
- Trap data types can only be used on IP networks. Trap data is treated as global and stored in a queue in this driver; i.e., trap variables could be defined under any I/O device and any two trap variables with the same address (such as T0) will have the same value. A read will get the variable value from the first elements of the queue and a write will delete that first element from the queue. The queue length is limited by available memory.
- The default value of TF is 127.0.0.1; that is, the local PC address.
- The GET-NEXT SNMP functionality is not supported in the driver.
- The target device has to support large PDU packets if PDUGroupOK is high. Some devices might become inoperable if this value is too large.
- When an error is detected in reading one item in a group, all items are flagged as errors. To establish which item is causing the error, set PDUGroupOK = 0, so that the driver sends a separate SNMP packet for each item, and set ReturnError = 0, so that the driver does not return an error to the I/O Server. Then the item which is causing the problem will be displayed with the value "Bad" if it is a string, or the value 2989 (the

decimal equivalent of the hex 0xBAD) if it is numeric. Once the problem item has been located, always remember to set ReturnError = 1, so that further errors generate #COM.

- The fields of a Variable Tag correspond to the entries in the SNMPVARS.DBF file defined for the port the device is on. Note that the **Address** field is a combination of the INDEX and SNMPTYPE entries in the SNMPVARS.DBF. You do not have to adjust these parameters in Plant SCADA or Excel; the utility allows you to do this.

---

**Note:** From version 2.7.20.0, the SNMPII driver includes new functionality that prevents a device from going online if duplicate indexes exist. This functionality requires every index to be unique at all times. Unique indexes can be achieved by creating multiple ports and splitting the SNMPVARS.DBF into multiple databases (one for each respective port).

---

## Configuration Overview

Simple Network Management Protocol (SNMP) is a network management standard widely used in TCP/IP networks to monitor and control computers, routers, printers, and other devices connected to a network.

SNMP uses distributed architecture consisting of managers and agents. A manager is an SNMP application that generates queries to SNMP agent applications. An agent is an SNMP application that responds to queries from SNMP manager applications. The SNMP agent is responsible for retrieving and updating local management information based on the requests of the SNMP manager. The agent also notifies registered managers when a significant event or a trap occurs.

The SNMP manager is realized in Plant SCADA through the SNMPII driver. On computers running Windows 2000, the SNMP agent is implemented by the SNMP service (SNMP.exe). Run the SNMP services on the machine that the I/O server is running on.

The Plant SCADA driver uses the Microsoft SNMP services and thus supports SNMPv1 and SNMPv2C.

Every SNMP-manageable device has a set of "objects" that can be read or modified via SNMP. This set of managed objects is known as a Management Information Base (MIB).

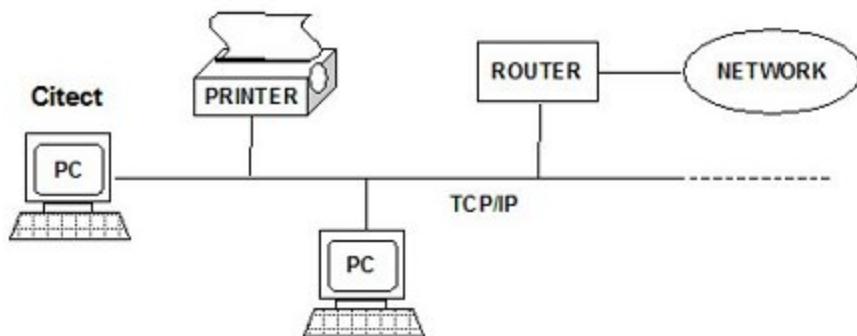
---

**Note:**

- The Plant SCADA driver does not support the GetNext request.
- SNMPII driver supports SNMP GETS and PUTS (read and write), trap receiving, forwarding, and trap generation. Each device runs in a separate NT thread for maximum speed.
- If an SNMP device only supports a fixed IP reply address, Plant SCADA I/O server redundancy will not be supported for that device.
- SNMPII driver has partial support for SNMP protocol version V2C and currently it supports only the SNMP GETS and PUTS (read and write) operation. The other operations will be supported in the future release version of this driver.

---

Each device to be monitored or controlled by Plant SCADA has to be connected to a TCP/IP network as illustrated in the following diagram:

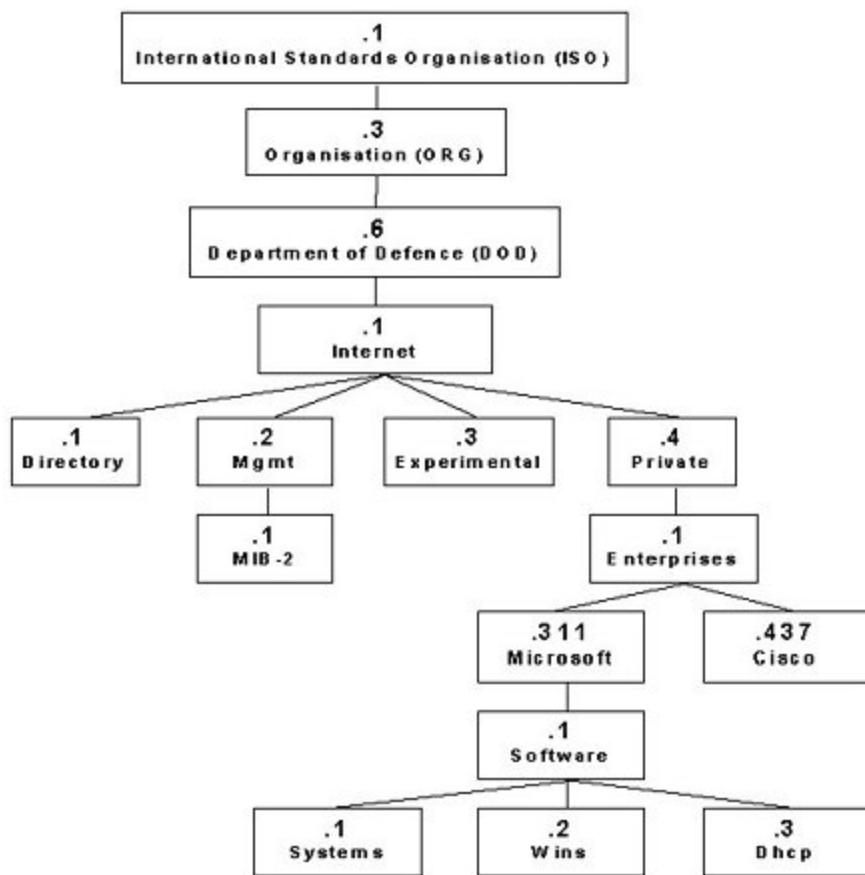


All SNMP-manageable devices that reside on a TCP/IP network are required to support a minimum standard set of managed objects, collectively known as MIB-II. Industry vendors can define additional MIBs that allow unique hardware or software services developed by the vendor to be monitored and managed by an SNMP server.

Every object in a MIB is identified by a universally unique label referred to as an object-identifier (OID). The identifier includes the object's type (such as counter, string, gauge or address), the object's access level (such as read/write), size restrictions and range information. (For more information about how OIDs work in Plant SCADA, see [Tag-based Driver Considerations](#).)

The object name space is implemented as a multi-part, hierarchical, naming scheme. A hierarchical naming scheme can be viewed as an inverted tree with the branches pointing downward. Each point where a new branch is added is referred to as a node. This OID is internationally accepted and allows developers and vendors to create new components and resources and assign a unique OID to each new component or resource.

The OID naming scheme is governed by the Internet Engineering Task Force (IETF). The IETF grants authority for parts of the name space to individual organizations such as Microsoft. For example, Microsoft has the authority to assign the OIDs that can be derived from branching downward from the node in the MIB name tree that starts at .1.3.6.1.4.1.311, as illustrated here:



SNMP programs use the OID to identify the objects on each device that can be managed by using SNMP. For example, when Plant SCADA needs information about managed objects from a computer on the network, the SNMP II driver sends a message over the network that requests information about the object as identified by the OID. The computer that receives the message can use the OID to retrieve information from the specific object on the computer and send the information back to Plant SCADA.

The OID in the hierarchy is written as a sequence of sub-identifiers beginning at the root and ending at the object. Sub-identifiers are separated with a period. For example, the OID for "sysDescr", which is a textual description of the SNMP manageable device, is **.1.3.6.1.2.1.1.1.0**.

## SNMP Redundancy

Redundancy (in the Plant SCADA context) is a function of the support provided in the SNMP device you are talking to. Some SNMP devices have a fixed IP address of who can talk to them. In this case, only one Plant SCADA I/O server can talk to them and redundancy is impossible.

However, most SNMP devices allow any device to talk to them provided their SNMP community name (like a password) is valid. In this case, Plant SCADA redundancy will work.

For an SNMP device to support trap redundancy (that is, to send traps to multiple places) the device has to support allowing two or more trap IP addresses. Each trap has to be sent to the primary and standby I/O server. If the SNMP device only supports one trap IP address, the standby I/O server cannot receive traps if the primary server is down.

## SNMP Enterprise Numbers

The SNMP MIB tree has parts that are universal and reflect various RFC standards; that is, each SNMP-enabled device supports the common lot of these functions; for example, sysDescr or .1.3.6.1.2.1.1.1.0 under mib-2-system returns the name of the device.

When you want to return data specific to your device, use the iso.org.dod.internet.private.enterprises path or under .1.3.6.1.4.1.x, where x is the unique number assigned to an organization or piece of equipment.

Microsoft's number is 311; that is, anything under .1.3.6.1.4.1.311.y is a Microsoft MIB item. Plant SCADA's number is 6216.

These enterprise numbers are needed in a global WAN so that every unique piece of equipment in the world can be identified. In a private network, you can essentially use any enterprise number you want. You can obtain new numbers from <http://www.iana.org/cgi-bin/enterprise.pl>.

You can use this number when generating user-defined traps. The number .1.3.6.1.4.1.6216.1.x is reserved by Plant SCADA for internal use and testing. Users are free to use .1.3.6.1.4.1.6216.2.y , .1.3.6.1.4.1.6216.3.y, and so on.

## SNMPII Driver Recommended Settings

This section includes the required [Device Address](#) and recommended settings for communication with an SNMPII device.

These are the settings implemented by the Express Communications Wizard on the Boards, Ports, and I/O Devices dialog boxes. To manually configure these settings, see [Communication Settings](#).

See also [Data Types](#) and [SNMPII Protocol-specific Parameters](#).

### Device Address

Use a valid IP address. The format is:

**aaa.bbb.ccc.ddd** (community string).

The community string has to be correct to access the remote agent.

The IP address can also be a device name known by the IP network (the network must have name resolving services running. The 'ping' program can be used to check that names are resolved to the correct IP address). The common community string used is *public*.

### Communication Settings

To establish communication with a device, the associated Board, Port and I/O Device need to be configured in Plant SCADA Studio's **Topology** activity.

If you use the Express Communications Wizard to connect to a device, these settings will be automatically configured for you. If you need to manually configure them, you should use the values outlined below.

## Boards

Field	Value
Board Name	This field is user defined and is not used by this driver.
Board Type	The type of board. Enter <b>SNMPII</b> .
Address	Enter <b>0</b> .
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Opt	Leave this field blank.
Comment	This field is user defined and is not used by this driver.

## Ports

Field	Value
Port Name	This field is user defined and is not used by the driver.
Port Number	Needs to be unique but is not used in this driver.
Board Name	Refers to the board previously defined.
Baud Rate	Leave this field blank.
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank.
Special Options	The .DBF file used to store additional tag data, for example, SNMPVARS.DBF. If there is no SNMPVARS.DBF in the project directory, the port will not initialize.
Comment	This field is user-defined and is not used by this driver.

**Note:** From version 2.7.20.0, the SNMPII driver includes new functionality that prevents a device from going online if duplicate indexes exist. This functionality requires every index to be unique at all times. Unique indexes can be achieved by creating multiple ports and splitting the SNMPVARS.DBF into multiple databases (one for each respective port).

## I/O Devices

Field	Value
I/O Device Name	This field is user defined, and is not used by this driver.
I/O Device Number	Has to be unique, but is not used by this driver.
I/O Device Address	<p>Use a valid IP address; format is: 'aaa.bbb.ccc.ddd community _string'. The community string has to be correct to enable access to the remote agent.</p> <p>The IP address can also be a device name known by the IP network (the network must have name resolving services running). The 'ping' program can be used to check that names are resolved to the correct IP address. The common community_string used is public; for example, "192.168.1.34 public"</p> <p>If you added your own community "PlantSCADA" into the SNMP service, the address would be "192.168.1.34 PlantSCADA".</p>
I/O Device Protocol	Enter <b>SNMPII</b> .
Port Name	Refers to the port previously defined.
Comment	This field is user-defined and is not used by this driver.

## Data Types

All managed objects defined under SNMP are based on the three ASN.1 UNIVERSAL types:

- **INTEGER** - A signed whole number with no specific size limit, although most SNMP implementations define INTEGERS to be signed or unsigned 32-bit numbers.
- **OCTET STRING** - A sequence of octets, where each octet is defined as an 8-bit byte. The sequence may be printable ASCII characters or arbitrary binary data.
- **OBJECT IDENTIFIER** - Stores the location of a variable (managed object) within an MIB. The location is stored as an array of 16-bit unsigned integers called sub-identifiers.

The following table describes the data types, address format, Plant SCADA data type, and applicable SNMP data type.

Data Types	Address Format	Plant SCADA Data Type	Applicable SNMP Data Type
Alphanumeric (display string) objects	DAn	STRING	ASN_OCTETSTRING, ASN_BITS, ASN_OPAQUE, ASN_IPADDRESS, ASN_OBJECTIDENTIFIER
Alphanumeric (binary	BAn	STRING	ASN_OCTETSTRING,

string) objects			ASN_BITS, ASN_OPAQUE
Numeric objects (treat as long in Plant SCADA )	Nn	LONG	ASN_INTEGER, ASN_INTEGER32, ASN_UNSIGNED32, ASN_COUNTER64, ASN_COUNTER32, ASN_GAUGE32, ASN_TIMECLIDKS
Numeric objects (treat as real in Plant SCADA )	Rn	REAL	ASN_INTEGER, ASN_INTEGER32, ASN_UNSIGNED32, ASN_COUNTER64, ASN_COUNTER32, ASN_GAUGE32, ASN_TIMECLIDKS
Trap objects (null terminated string)	Ta	STRING	Trap object
Trap objects (binary string)	BTa	STRING	Trap object (only for ASN_OCTETSTRING, ASN_BITS, ASN_OPAQUE type value in varbinding field).
Trap forwarding (write only)	TF	STRING	IP address to forward a copy of the last trap to. One IP address is required per call to TF. TF can be called multiple times.
Trap Next (read and write)	TN	LONG	TN as read is the traps in queue, 0 for nothing. A TN write removes the current entry and gets the next.
Multiple Trap Variable	TNVB	LONG	A read of TNVB returns the number of varbinds in the trap, a write to TNVB controls which varbinds data is read or written to.

Where:

n	0 to 65535
a	0 to 6 (T0 to T6 are read and write variables)

For string types, **DA** stands for null-terminated string and **BA** for binary string.

A null-terminated string is a string with 0x00 as an end flag, but in a binary string a 0x00 may be valid content.

For example, <0x41><0x42><0x43><0x00><0x61><0x62><0x63>:

- If defined as **DA**, Plant SCADA will display it as 'ABC' and the content from <0x61> will be truncated.
- If defined as **BA**, Plant SCADA will display it as '41424300616263'.

This also applies to those ASN\_OCTETSTRING type fields in a trap object, with **DA** replaced by **T**, and **BA** replaced by **BT**.

Whether a tag is readable/writable depends on how the SNMP object to which it corresponds is defined in the appropriate .MIB file.

For the type **DA**, **BA**, **N** and **R**, the n represents the value of the INDEX field for a tag's entry in the SNMPVARS.DBF file.

You should not directly modify the tag database.

For each tag, you can specify the Plant SCADA data type you require; however you have to verify that the selection you make is appropriate for the corresponding SNMP object.

Tags with a Plant SCADA data type of STRING are limited to a maximum length of 255 characters. Tags with a required numeric range outside the range 147483648 to 2147483647 can be assigned a Plant SCADA data type of REAL.

## Numeric Data Grouping Considerations

Plant SCADA allows you to group enquiries together to enable greater SNMP efficiency. The user has to decide which items need to be grouped together. It is recommended no more than 20 items are grouped together. This means that if any one item is read, the whole group will be read.

Groups should be separated in units of 100; for example, N0,N2,N3, then N100,N101, and so on.

### SNMP II Protocol-specific Parameters

#### **WARNING**

##### **UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage..**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

This section describes the SNMP II protocol-specific parameters. The parameter values have been set to the usual value for this protocol. You cannot change these values except on the advice of Technical Support for this product.

**Note:** If you plan on using Include projects and your communications settings are not in the main project, you need to set the [driver-specific parameter ComSettingsInInclude](#) to 1.

Parameter	Description
[SNMPII]Block	This value is not used by the driver.
[SNMPII]Delay	The period (in milliseconds) to wait between receiving a response and sending the next command. <b>Allowable Values:</b> 0 to 300 (milliseconds) <b>Default Value:</b> 0
[SNMPII]MaxPending	The maximum number of pending commands that the driver holds ready for immediate execution. <b>Allowable Values:</b> 1 to 256 <b>Default Value:</b> 2 You need to adjust MaxPending so that at least 2 DCBs are available per unit on a port; for example, if you have 10 devices on a port, Maxpending should have a minimum value of 20, or else when one unit is offline it can cause significant slowing of the responses from other units.
[SNMPII]PollTime	The interrupt or polling service time (in milliseconds). Setting the polling time to 0 puts the driver in interrupt mode. <b>Allowable Values:</b> 500 to 2000 (milliseconds) <b>Default Value:</b> 1000
[SNMPII]Retry	Not used by the driver. See the <a href="#">driver-specific parameter</a> [SNMPII]SnmpRetry.
[SNMPII]Timeout	Not used by the driver. See the <a href="#">driver-specific parameter</a> [SNMPII]SnmpTimeout.
[SNMPII]WatchTime	The period (in seconds) that the driver uses to check the communications link to the I/O Device. <b>Allowable Values:</b> 0 to 128 (seconds) <b>Default Value:</b> 30 (seconds)

## Driver-Specific Parameters

 **WARNING**

**UNINTENDED EQUIPMENT OPERATION**

Do not change these protocol parameters, except on the advice of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage..**

**Note:** Always seek the advice of Technical Support personnel for this product regarding undocumented features.

This section describes the SNMPII driver-specific parameters.

Parameter	Description
[SNMPII]MaxTimeouts	<p>When SNMP API detected timeout errors exceed this value, DRIVER_UNIT_OFFLINE will be returned by the driver, otherwise DRIVER_TIMEOUT will be returned.</p> <p><b>Allowable Values:</b> 1 to 3 (requests to SNMP API)</p> <p><b>Default Value:</b> 1</p>
[SNMPII]SnmpTimeout	<p>Period in milliseconds to consider an SNMP timeout when using the SNMP API. Timeout is internal to the SNMP API.</p> <p><b>Allowable Values:</b> 200 to 6000 (ms)</p> <p><b>Default Value:</b> 1000</p> <p>You can override the values set for this parameter for a particular device. This would appear in the Citect.INI file as follows:</p> <pre>[SNMPII] SnmpTimeOut=1200 Port1.IODev1.SnmpTimeOut=3300</pre> <p>where:</p> <p>Port1 = the port the specified device is connected to IODev1 = the name of the device</p> <p>The above configuration means all devices use an SnmpTimeOut value of 1200mS, except for the device IODev1 on Port1, which will timeout after 3300mS.</p>
[SNMPII]SnmpRetry	<p>Retry count to apply to the SnmpTimeout period before a true timeout is indicated. This then is used by the MaxTimeouts parameter. The count is internal to the SNMP API.</p> <p><b>Allowable Values:</b> 1 to 5</p> <p><b>Default Value:</b> 1</p> <p>You can override the values set for this parameter for a particular device. This would appear in the Citect.INI file as follows:</p> <pre>[SNMPII] SnmpRetry=2 Port1.IODev2.SnmpRetry=3</pre> <p>where:</p>

	<p>Port1 = the port the specified device is connected to IODev2 = the name of the device The above configuration means all devices use an SnmpRetry value of 2, except for the device IODev2 on Port1, which will retry three times.</p>
[SNMPII]PDUGroupOK	<p><b>Allowable Values:</b> 0 to disable grouping, or &gt;1 to enable grouping. This specifies the maximum number of SNMP requests which will be grouped into the one packet to the remote SNMP agent. The recommended value is 20, and it cannot exceed 98. <b>Default Value:</b> 20 When PDUGroupOK is 0 or 1, separate SNMP packets need to be generated for each item asked by the I/O server. When non-zero, items in one request from IOServer will be requested in one packet to target device. For more information on setting the PDUGroupOK value, see <a href="#">Setup Tips and Techniques</a>.</p>
[SNMPII]ExtraDebug	<p>Extra debug statements for diagnostic purposes. Debug flags need to be enabled. <b>Allowable Values:</b> 1 or 0 <b>Default Value:</b> 0</p>
[SNMPII]ReturnError	<p>Used with PDUGroupOK to isolate the variable in a group of tags that is causing an error. When this parameter is set to 1, the driver returns an error when an SNMP packet is returned with an error. When set to 0 (and PDUGroupOK=0), the driver returns either the string "Bad" or the number 2898 (0xBAD). This allows you to see the specific bad variable on your screen. <b>Allowable Values:</b> 1 or 0 <b>Default Value:</b> 1</p>
[SNMPII]UseV2C	<p>When this parameter is set to 0, it uses SNMP v1.0 and when set to 1, it uses SNMP V2C to communicate with the device. <b>Allowable Values:</b> 1 or 0 <b>Default Value:</b> 0 <b>Note:</b> To configure the parameters at device level, see <a href="#">I/O Device-specific Parameters</a>.</p>
[SNMPII]x.y.TestOIDStr (where x is portname, and y is the unitname)	<p>The SNMPII driver tests the status of each Plant SCADA I/O device by transmitting a request to read the "sysDescr" (.1.3.6.1.2.1.1.1.0) object in the device. TestOIDStr allows alternative OID to be used for</p>

	testing the status of each Plant SCADA I/O device. An OID needs to end with ".0". If the I/O device is online, it responds by transmitting the requested data. If no response is received, the Plant SCADA I/O device is considered to be offline. <b>Default Value:</b> Test OID string (x.x.x.x.x.0)
ChangePorts2000 or EnableChangePorts	Default 0. Only change if you are on a Win2000/XP/Win2003 Server machine and you want to change the default SNMP port. <b>Default value:</b> 0 or 1
x.y.Port (where x is portname, and y is the unitname)	Allows the default SNMP port (the default is 161) to be changed. Your SNMP agent needs to be able to listen on the new port number. <b>Default value:</b> Any
ComSettingsInInclude	Default 0, this needs to be set to 1 if the SNMPVARS.DBF information is going to be in Communication Settings in Include projects. <b>Default value:</b> 0 or 1

The ReturnError parameter is likely the only parameter you'll use; the other parameters relate to specialized setups.

When PDUGroupOK is 0 or 1, separate SNMP packets are generated for each item asked by the I/O server. When non-zero, items in one request from the I/O server are requested in one packet to target device.

**Note:**

- 1) The target device needs to be able to support large PDU packets if PDUGroupOK is high. Some devices may exhibit unexpected behaviour if this value is too large.
- 2) When an error occurs in reading one item in a group, all items are flagged as errors. To establish which item is causing the error, set PDUGroupOK=1 so that the driver sends a separate SNMP packet for each item, and set ReturnError =0 so that the driver does not return an error to the I/O server. Then the item which is causing the problem will be displayed with the value "Bad" if it a string, or with the value 2989 (the decimal equivalent of the hex 0xBAD) if numeric. After locating the problem item, remember to set ReturnError back to 1 so that further errors generate #COM.

## I/O Device-specific Parameters

This section describes the I/O device-specific parameters.

Parameter	Description
[IODevice]UseV2C	When this parameter is set to 0, it uses SNMP v1.0 and when set to 1, it uses SNMP V2C to communicate with the device. <b>Allowable Values:</b> 1 or 0

	<b>Default Value:</b> 0  <b>Note:</b> The <b>IODevice</b> refers to the I/O device name configured in the project.
--	--

## SNMP II Blocking

You can configure the SNMP II driver for blocking. You can only block longs and reals (N and R data-types). For every SNMP II I/O device a snmpvar.dbf file is created. In that file an index is defined for each SNMP tag. This index is used by Plant SCADA for blocking. That index in the snmpvar.dbf is also used in the tag address in the variable.dbf, with a prefix for the different data-types (N for long, R for real, DA for string).

You can implement blocking for the SNMP II driver by manually manipulating the index numbers in **snmpvar.dbf** file. If you create consecutive numbers in a data type, Plant SCADA can block these together in one request. This index number has to be unique for each tag (per I/O device); for example: N100, N101, N102, N103 will be a block of four LONGs, and R200, R201, R203 will be a block of three REALs.

The SNMP II driver by default only blocks 20 items (for longs and reals this equals 640 bits). This value is set via the [SNMP]PDUGroupOK parameter and the BIT\_BLOCK value in protdir.dbf for the SNMP II protocol. The target device has to support large PDU packets if PDUGroupOK is high. Some devices may crash if this value is too large. The [SNMP II]Block parameter is set to 256 by default.

In the Kernel use the Probe command to view the blocking behavior. There you will see all requests from the client to the I/O server and the size of these requests. The address numbers there are derived from the index numbers in the snmpvar.dbf files.

## Debug Messages

The SNMP driver generates the following debug messages:

### Data Read

```
Mon Jun 05 11:08:50 2015 02:31:28.986 StartTransmit Length 0
Mon Jun 05 11:08:50 2015 02:31:28.986 Transmit Chk 00000000 Length 0
Mon Jun 05 11:08:50 2015 02:31:28.989 Read Request Addr:2 020E75F8 1 Length 0
Mon Jun 05 11:08:50 2015 02:31:28.990 Request Finish. Thread Will Suspend. Addr:2 Length 0
```

### Data Write

```
Mon Jun 05 14:19:49 2015 00:23:04.493 Write Request Addr:56 020E75F8 1 Length 0
Mon Jun 05 14:19:49 2015 00:23:04.495 Request Finish. Thread Will Suspend. Addr:56 Length 0
```

In these debug messages, Addr:2 or Addr:56 means the variable address (index in the SNMPVARS.DBF file) is 2 or 56. The following hexadecimal string is the pointer of the corresponding DCB and the following 1 is the returned value of the ResumeThread() function.

## Driver-generated Statistics

The SNMP driver generates the following statistics:

Number	Label	Description
0	Rd Req	Number of read requests from Plant SCADA but excludes trap requests.
1	Rd OK	Number of successful read requests.
2	Wrt Req	Number of write request from Plant SCADA but excludes trap writes.
3	Wrt OK	Number of successful write requests.
4	Sent Packets	Number of packets sent to SNMP agent.
5	PDUGroupOK	Value of PDUGroupOK.
6	Att traps	Number of received traps.
7	Traps in queue	Number of traps in the queue.
8	Traps Rd	Number of trap read requests.
9	Traps Wrt	Number of trap write requests.

When PDU grouping occurs, the packet count is less than the number of desired reads and writes. The difference indicates how well the "group" feature is working.

## SNMP Traps

SNMP traps differ from other SNMP messages in that they are unsolicited messages sent by an SNMP agent when a predefined event has occurred. For this reason, the SNMPII driver in Plant SCADA handles traps differently than other SNMP variables.

Because there is no way of knowing in advance how many traps will be received by Plant SCADA, it is impossible to set up variable tags ready for each trap. Instead, incoming traps are queued and read one at a time from the queue.

A set of seven tags, addressed as T0, T1, ...T6, is used to read each trap.

### Structure of a Trap

The table below shows the components of a trap PDU and the special tags used to manipulate them.

Component	Special tag
enterprise	T0

agent-addr	T1
generic-trap	T2
specific-trap	T3
time-stamp	T4
variable-bindings	T5, T6, TNVB and SubfieldNumber

- **variable-bindings** is a list of VarBinds, each consisting of **name** (see [T5](#)) and **value** (see [T6](#)).

**Note:** Most SNMP traps only contain one record; however, it is possible to get a trap with many records. A mechanism exists using the "subfield number" to iterate and read all these trap records.

## Special Tags for Reading and Creating Traps

The table below shows the tag address and the field of the tag object assigned to the tag.

Tag Address	Field of trap object assigned to Tag
T0	Enterprise (device sending the trap), for example .1.3.6.1.4.1.311.1.1.3.1.1
T1	IP address; for example 223.125.1.17
T2	Generic type. Available options are: <b>0</b> - cold start <b>1</b> - warm start <b>2</b> - link down <b>3</b> - link up <b>4</b> - authentication unsuccessful <b>5</b> - egpNeighbor loss <b>6</b> - enterprise-specific trap
T3	Specific type (i.e., a user-defined trap in the enterprise MIB). For example, 3.
T4	Time stamp; for example, 6336403.
T5	Accesses the OID for the SNMP VarBind which holds the selected subfield of the current trap. The variable 'SubfieldNumber' (see below) determines which subfield is accessed; e.g., .1.3.6.1.2.1.1.5.0 Note that T5 contains a value. It can be a dummy value if the SNMP variable doesn't exist. This value is only of use if your MIB contains access to the trap

	variable.
T6	Accesses the data in subfields of the current trap. The variable 'SubfieldNumber' determines which subfield is accessed. When writing, you have to provide a value and a type. The type has to be "INTEGER", "COUNTER", "GAUGE", "TIMETICK", "ADDRESS", or "STRING". Strings should be less than 80 characters long.
TF	Trap forwarding IP address and community name. When writing, the trap community name parameter is optional; for example, 196.168.1.34 trap.
TN (TrapNext)	Read - Returns the number of traps in the queue. Write - Deletes the current trap from the queue. The next trap in the queue becomes the current trap.
TNVB	Read - Returns the number of VarBinds in the VarBindList of the current trap. Write - Update SubfieldNumber.
SubfieldNumber	This is not a real tag: it is a software variable. SubfieldNumber can be updated by writing to TNVB. SubfieldNumber is set to zero whenever TN is written.

## Reading Queued Traps

You can read queued traps.

### To read queued traps:

1. Read tag TN to see if it is > 0.
2. If TN is 0, there are no traps in the queue. Exit the process.
3. Read the required data from the current trap by using tags Tx (where x is any value from 0-5).
4. Log the values of the Tx or store them in other variables.
5. Read tag TNVB to see if there are any subfields.
6. If the number returned is greater than 0, read each subfield in turn by writing the subfield number to TNVB, and then reading the subfield OID and value from T5 and T6.

---

**Note:** It is not necessary to write to TNVB (SubfieldNumber) before reading the first subfield because the default behavior of T5 and T6 reads is to return the value of the first sub-field.

---

7. Write to TN to delete the current trap and move the traps in the queue forward.
8. Return to step 1.

## Trap Forwarding

If a trap needs to be forwarded to another system, Cicode has to be written to write to the "TF" tag before deleting the trap (using "TN"); for example, TF - "203.168.34.23". Multiple TF's are allowed so that several destinations can be used. If no second argument is given, the default of "trap" is used to send the trap.

Default value of TF is **127.0.0.1**.

## Generating Traps

Traps can be created by writing suitable data to T0-T6, then forwarded using the "TF" as before. This enables non-SNMP devices to generate traps via Plant SCADA .

### To create a trap:

1. Create an empty trap by writing to one of the Tx tags (T0 - T6). From that point on, all tag reads and writes are assumed to be for the newly created trap.
2. Update any other fields, as required.
3. Write to TF to specify a forwarding address and send the trap.
4. Repeat the above one or two steps as many times as required.
5. When writing to subfields you cannot skip over uninitialized subfields. For example, the sequence 0,1,2,0,2 is OK, but 0,2 is not. This is so that there are never uninitialized VarBinds in the VarBindList.
6. Write to TN to erase the generated trap and regain access to the queue.

The empty trap has these default values:

T0 (Enterprise)	.1.2.3.6.1.311.1.1
T1 (IP address)	Local IP address
T2 (GenType)	6
T3 (SpecType)	0
T4 (TimeStamp)	Time since driver was started, in 100th of seconds.
T5 (OID)	.1.3.6.1.2.4.2.0
T6 (Value and type)	0 INTEGER

## Queue Management

You can access to only one trap at a time. This is normally the trap at the head of the queue. To move the next trap you have to delete the current trap by writing to tag TN.

If you have created a trap, the trap at the head of the queue is inaccessible until the created trap is erased by a write to TN.

## Notes

It is possible for many tags to refer to the same Address tags, as tags on different I/O device names talk to different SNMP agents. Also the **I/O Device** field could be changed to refer to any other I/O device that has been set up for SNMP communications.

If you insert a trap, you need to create valid entries in a project MIB file for third-party SNMP viewers to know more about the trap.

## Troubleshooting

This chapter provides information about the Plant SCADA tools provided to help resolve problems with communication and configuration.

### Driver Errors

When a hardware error is detected, Plant SCADA generates an alarm and displays the alarm on the hardware alarm page (in the alarm description of the hardware alarm). To see the error number, you need to have Alarm Category 255 defined with a display format that includes {ErrDesc,15} {ErrPage,10}.

Several generic errors are common to many protocols. In some cases only the generic error is available, though often both the generic error and a specific error are given.

The following errors, listed in (hexadecimal) sequence, are specific to this protocol. Plant SCADA displays the error number and description for common protocol-specific errors. Uncommon errors are not contained in the Plant SCADA error database, in which case Plant SCADA only displays the error number.

You might need more information to rectify an error. This information should be in the documentation that accompanied the I/O device (or network). If, after reviewing the documentation, you cannot rectify an error, contact Technical Support for this product.

Error	Description
0x101 SNMP_ERRORSTATUS_TOOBIG	The response too large for agent to handle.
0x102 SNMP_ERRORSTATUS_NOSUCHNAME	Requested OID (1) does not exist in supported MIB view or (2) is not accessible or (3) a write operation was attempted on a read only MIB variable.
0x103 SNMP_ERRORSTATUS_BADVALUE	Bad MIB variable data type specified in variable bindings.
0x104 SNMP_ERRORSTATUS_READONLY	Attempt made to write to read-only field.
0x105 SNMP_ERRORSTATUS_GENERR	An unknown error.
0x108 SNMP_DISCONNECT	Cables have been disconnected.
0x180 GENERIC_NO_MEMORY	No memory could be allocated.
0x181 GENERIC_GENERAL_ERROR	Could not convert an OID string into a valid OID.

0x182 ERR_SNMP_TRAP_LISTEN	Could not register to receive SNMP traps.
0x183 ERR_SNMP_TRAP_QUEUE	Could not create SNMP trap queue.
0x184 ERR_SNMP_TRAP_VARBIND_NUM	Trying to access a varbind number that does not exist.

**Note:** Other errors are either standard Plant SCADA driver errors (below 0x100) or SNMP API errors. Subtract 0x100 from the value and search [www.microsoft.com](http://www.microsoft.com) for "SNMP error codes".

## Kernel Diagnostics

The Plant SCADA Kernel window diagnostics framework (known as the Kernel) is the primary gateway into the internal workings of Plant SCADA at runtime, and is provided for diagnostics and debugging purposes only.

**Note:** Use the Kernel window carefully: once you are in the Kernel, you can execute any Cicode function with no privilege restrictions. You (or anyone using the Kernel) has total control of Plant SCADA (and subsequently your plant and equipment).

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

Do not use the Kernel for normal Plant SCADA operation. The Kernel is only for diagnostics and debugging purposes.

Configure your security so that only approved personnel can view or use the Kernel.

Do not view or use the Kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

You need to configure your Plant SCADA project to provide run-time access to the Kernel diagnostics window. You can do one or both of the following:

- Enable the Kernel on Plant SCADA start-up.
- Add the Kernel menu item to the runtime Plant SCADA Control menu.

The Plant SCADA Kernel window is a diagnostics framework that can display several different diagnostic windows, each displaying an active view into the workings of the Plant SCADA runtime system. Each window is displayed when selected from the Plant SCADA Kernel View menu. For details, refer to the Plant SCADA User Guide.

The key diagnostic windows included with the Kernel for testing I/O device communications are the I/O Devices window and the Main window:

- The [Kernel I/O Devices window](#) displays the current status of devices defined in the I/O Device database.
- The [Kernel Main window](#) displays the diagnostic messages a line at a time indicating the current Plant SCADA startup operation and status. When running, the Main diagnostics window continues to report changes in the status of the I/O Devices.

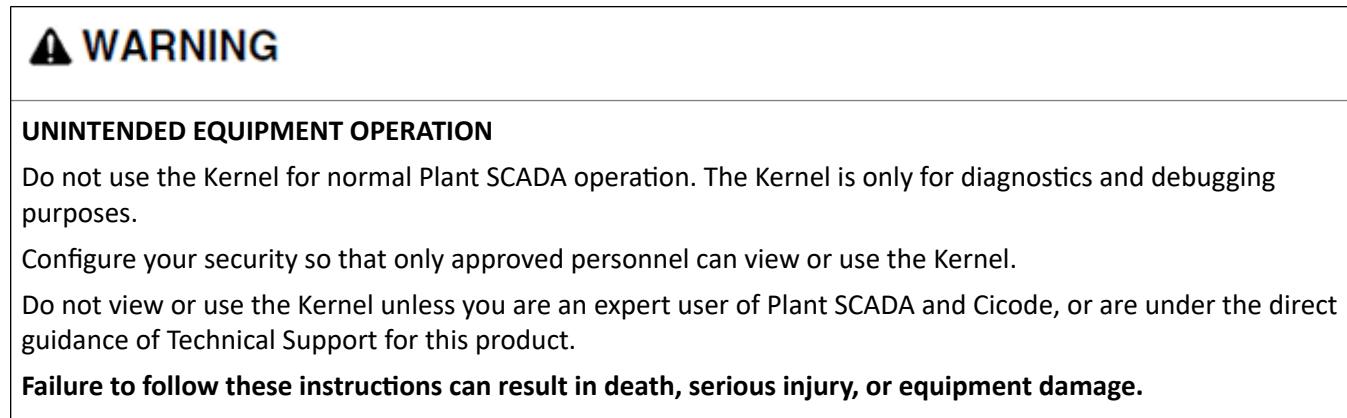
You can also use the Plant SCADA Kernel command Page Driver to call up a drivers diagnostic window that displays information about each driver in Plant SCADA . The statistics it presents include read requests, physical

reads, digital reads per second, register reads, cache reads, error count, timeouts, and so on.

This window is only displayed if the Plant SCADA computer is configured as an I/O server with physical I/O devices attached.

## Kernel I/O Devices Window

The I/O Devices diagnostics window displays the current status of all devices defined in the I/O Device database. In this window, use the **Page Up** or **Page Down** keys to browse through the available devices. Note that the below image is a generic representation of this window; your details might differ.



Check that communications with the I/O device are enabled and running by verifying the **Server Status** and **Client Status** fields. When the I/O device is online, these both display "RUNNING".

The following table contains a description of the fields displayed for each I/O device (or Unit) in the I/O Devices diagnostics window.

### Unit Statistics Fields

Field	Description
Unit	The name of the I/O device defined in the project (with the I/O Devices dialog box).
IO Server	The name of the I/O server that is servicing this I/O Device.
Comment	A description of the I/O device defined in the project (with the I/O Devices dialog box).
Unit No	The I/O device number defined in the project (with the I/O Devices dialog box).
PLC Number	Physical I/O device address defined in the project (with the I/O Devices dialog box).
Port Name	Communication port to which the I/O device is connected.
Protocol	Protocol used for communication with the I/O device.

Server Status and Client Status	<p>Status of the I/O device. The Server Status is only valid if the computer is an I/O server and it is servicing this I/O device. The Client Status field is valid for clients only, and indicates the status of the I/O device that is attached to the I/O server.</p> <p>The I/O device status can be one of the following:</p> <p><b>RUNNING</b> Indicates that the communication link with the I/O device is good.</p> <p><b>STANDBY</b> Indicates that the communication link with the I/O device is good, but communication with that I/O device is currently being performed by another port. This port is in standby mode.</p> <p><b>STARTING</b> Indicates that the server is currently establishing a communication link (with the I/O device).</p> <p><b>STOPPING</b> Indicates that the server is currently relinquishing control of the communication link (with the I/O device).</p> <p><b>OFFLINE</b> Indicates that the server cannot establish a communication channel with the I/O device. If a standby port or server is available, Plant SCADA tries to communicate to the I/O device using that port.</p> <p><b>REMOTE</b> Indicates that the status of the I/O device is OK, but it is not currently connected. Also indicates if it is a scheduled device.</p>
Primary	Indicates if the I/O device is in primary mode; Yes = Primary, No = Standby. If the I/O device is in primary mode, the server starts a communication channel with the I/O device as soon as the server is activated. If an I/O device is in standby mode, the I/O device remains inactive when the server starts (until a primary I/O device becomes inoperative).  If a unit is configured as the primary, it will be the preferred unit to service Plant SCADA requests. If the unit is configured as the standby, Plant SCADA requests will only be directed to this unit if the primary is offline.
Client Using	Name of the I/O Server that this client is using. This allows you to identify which I/O server is currently processing Plant SCADA requests.
Generic Error	Last generic error code returned by the driver. Because protocol drivers have their own special errors, they cannot be recognized by the I/O Server. The drivers convert their special errors into generic

	errors that can be identified by the server.
Error Handle	Error handle assigned by the I/O Server to each error. This handle is not used by Plant SCADA (at this time).
Driver Error	Driver-specific error code. Each driver has its own special error codes. Refer to the driver specific errors (for the particular protocol) for an explanation of each of the error codes.
Error Message	Error message associated with the generic error code.
Error Count	Total number of errors from the I/O Device.
Restarts	Number of times the server has tried to establish a connection with the I/O Device. This number is normally 1, because the server establishes a connection at startup. If this field displays a number greater than 1, this indicates there has been, or currently is, a problem with the communication channel or the device.
Response Times	<p>Time taken by the driver to process read and write requests (i.e. the time taken to process a single read or write operation to the I/O Device). This time depends only on the physical response time of the I/O Device, because no queue waiting time is included. This field reflects any tuning of the communication channel (e.g. doubling the baud rate should half the response time). The average, minimum, and maximum times are displayed.</p> <p><b>Note:</b> One I/O Device with a slow response can slow down your entire system. For example, if you have an I/O Device with a response of 2000ms, any pages in your system that use data from that device, will have a minimum update time of 2000ms.</p>
Cached	Indicates if the I/O Device data is cached.
Cache Timeout	If the I/O Device is cached, this field displays the cache timeout value. Data is held in the cache for this timeout period before being discarded and re-read from the I/O Device. Only read data is cached.
Blocking Constant	Current blocking constant value for this I/O Device, as specified in the protocol.

## Kernel Main Window

Maximize the Kernel Main window to view the diagnostic messages. This enables you to see the Plant SCADA start-up procedures.

**Note:** The Kernel Main diagnostic window has no scroll bar. New messages are added to the bottom of the message list; as the window fills, the oldest message (at the top of the window list) scrolls out of view. Maximize the Plant SCADA Kernel and the Main windows to gain the maximum view of available messages, alternatively, you can view the same information using Syslog.dat file.

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

Do not use the Kernel for normal Plant SCADA operation. The Kernel is only for diagnostics and debugging purposes.

Configure your security so that only approved personnel can view or use the Kernel.

Do not view or use the Kernel unless you are an expert user of Plant SCADA and Cicode, or are under the direct guidance of Technical Support for this product.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

After your I/O devices are properly configured, use the Main window to check that all of your I/O devices come online correctly when Plant SCADA started. First the ports will be initialized, then the I/O Device will be brought online. If there is a problem, Plant SCADA displays a message such as "PLC not responding", "I/O Device Offline", or similar.

Some I/O devices might take two attempts to come online. If so, Plant SCADA waits (usually 30 seconds) and tries again. If your I/O device does not come online after the second attempt, check your configuration (at both ends) and the network in between.

If an element repeatedly becomes inoperative at startup, you should investigate the problem. Common factors that may cause startup errors include:

- Incorrect computer setup: usually solved by running the Computer Setup Wizard.
- Networking errors or bad hardware: cables wrongly wired or unplugged, or equipment not powered up, or inoperative or improperly configured equipment .
- Communication errors: usually a configuration mistake like wrong protocol settings, wrong port, and so on.

#### **To enable the Kernel diagnostics window on Plant SCADA start-up:**

1. In the Citect Explorer menu, select **View | Configuration File**. This launches Windows Notepad and loads and displays the Citect.ini file for manual editing.
2. Scroll down to the "[Debug]" category, and edit the section to include the following parameter:  
`[DEBUG  
Kernel=1]`
3. **Save** and **Close** Notepad.

This parameter causes the Plant SCADA Kernel window diagnostics framework to appear automatically when Plant SCADA starts up. Swap to it to Plant SCADA runtime processes.

### To add the Kernel menu item to the runtime Control menu:

1. In Plant SCADA Studio's **Projects** activity, launch the Computer Setup Wizard and select Custom mode.
2. Click **Next** until you reach the **Security Setup - Control Menu** page.
3. Check the **Kernel on menu option** and click **Next** until finished.

At runtime, you can display the Kernel window by selecting Kernel from the **Control** menu (top-left corner). If a Title Bar is not displayed in your runtime project, you can access the Control menu by pressing **Alt + Spacebar** (if the Alt-Space enabled option is ticked on the Security Setup - Keyboard page).

You should clear these options before placing the system into service to minimize the likelihood of accidental or unauthorized use of the Kernel in the delivered system.

### Tag-based Driver Considerations

The Plant SCADA SNMII driver uses OIDs (object identifiers) to uniquely identify tag addresses. These OIDs are generated by Plant SCADA during compilation.

You should know how OIDs are implemented in a tag-based driver to minimize the possibility of tag mismatches, particularly if you are using network distributed systems. For example, all Plant SCADA servers and clients need to have exactly the same variable database on them, otherwise a client could make a request for a tag with an OID that does not exist on the I/O Server, or refers to a different tag. For this reason, care needs to be taken when editing a variables database.

The following information highlights activities you should pay special consideration to due to the potential impact they might have on OIDs in your system.

For details on OIDs, see Tech Note TN5868 (*How Do Object IDs (OIDs) Work?*) on the AVEVA Knowledge & Support Center (<https://softwaresupport.aveva.com/>). You will need to register to search this site.

### Project IDs

OIDs are generated using a Project ID, and the position (that is, record number) in the associated variable.dbf file. Therefore, when using control clients, standby servers and so on, check that a project restored to a PC has a consistent Project ID in each machine included in a system.

This should be checked when you restore a Plant SCADA project to a PC, because if the Project ID is in use by another project on that computer, Plant SCADA will generate a new Project ID number.

A project's ID number can be determined by selecting a project in Plant SCADA Studio's **Projects** activity and viewing its properties. The Project ID is shown on the **General** tab. It is also stored in the project backup file (.ctz).

---

**Note:** Check that each Include project has the same Project ID on each computer using the same Include project. Check this after restoring a project to a computer.

---

### Variable Databases

If you have different variable.dbf files across your plant, you might have OID mismatches.

To avoid this, check that after changing your variable.dbf on your primary server that you reset the OID (in citect.ini file set [OID]Reset=1) and do a pack and full compile (that is, ensure that incremental compile is turned off).

Then copy the variable.dbf file to the standby server and any Control Clients using the project. This sets the OID on your primary server to the same value as the OID used on your display client and/or standby server.

**Note:** If you are using Include projects, pack your variable.dbf before resetting the OIDs and doing a full compile of your project. To pack the project databases, use the **Pack** button on the command bar of the **Project** activity in Plant SCADA Studio.

---



**AVEVA Group plc**  
High Cross  
Madingley Road  
Cambridge  
CB3 0HB  
UK

Tel +44 (0)1223 556655

**[www.aveva.com](http://www.aveva.com)**

To find your local AVEVA office, visit **[www.aveva.com/offices](http://www.aveva.com/offices)**

AVEVA believes the information in this publication is correct as of its publication date. As part of continued product development, such information is subject to change without prior notice and is related to the current software release. AVEVA is not responsible for any inadvertent errors. All product names mentioned are the trademarks of their respective holders.